Identity Based Authenticated Key Agreement Protocols from Pairings*

Liqun Chen[1] and Caroline Kudla[2]**

[1] Hewlett-Packard Laboratories, Bristol, UK
liqun.chen@hp.com
[2] Information Security Group
Royal Holloway, University of London, UK
c.j.kudla@rhul.ac.uk

27 May 2004

Abstract. We investigate a number of issues related to identity based authenticated key agreement protocols using the Weil or Tate pairings. These issues include how to make protocols efficient; how to avoid key escrow by a Trust Authority (TA) who issues identity based private keys for users, and how to allow users to use different Trusted Authorities. We describe a few authenticated key agreement (AK) protocols and AK with key confirmation (AKC) protocols which are modified from Smart's AK protocol [Sm02]. We study the security of these protocols heuristically and using provable security methods. In addition, we prove that our AK protocol is immune to key compromise impersonation attacks, and we also show that our second protocol has the TA forward secrecy property (which we define to mean that the compromise of the TA's private key will not compromise previously established session keys). We also show that this TA forward secrecy property implies that the protocol has the perfect forward secrecy property.

1 Introduction

Key establishment is a process whereby two (or more) entities can establish a shared secret key (session key). There are two different approaches to key establishment between two entities. In one scenario, one entity generates a session key and securely transmits it to the other entity. This is known as enveloping or key transport. More commonly, both entities contribute information from which a joint secret key is derived. This is known as key agreement. All the protocols discussed in this paper are of this latter form.

A key agreement protocol is said to provide implicit key authentication (of entity B to entity A) if A is assured that no other entity besides B can possibly ascertain the value of the secret key. A key agreement protocol that provides mutual implicit key authentication is called an authenticated key agreement

* This work is a revision of an earlier version [CK02].

** This author is funded by Hewlett-Packard Laboratories.

protocol (or AK protocol). A key agreement protocol provides key confirmation (of B to A) if A is assured that B in fact possesses the secret key. A protocol that provides mutual key authentication as well as mutual key confirmation is called an authenticated key agreement with key confirmation protocol (or an AKC protocol).

It is desirable for AK and AKC protocols to possess the following security attributes:

Known-key security: Each run of the protocol should result in a unique secret session key. The compromise of one session key should not compromise other session keys.

Forward secrecy: If long-term private keys of one or more of the entities are compromised, the secrecy of previously established session keys should not be affected. We say that a system has partial forward secrecy if the compromise some but not all of the entities' long-term keys can be corrupted without compromising previously established session keys, and we say that a system has perfect forward secrecy if the long-term keys of all the entities involved may be corrupted without compromising any session key previously established by these entities. There is a further (perhaps stronger) notion of forward secrecy in identity-based systems, which we call *TA forward secrecy*, which certainly implies perfect forward secrecy. This is the idea that the TA's long-term private key may be corrupted (and hence all users' long-term private keys) without compromising the security of session keys previously established by any users.

Key-compromise impersonation resilience: Compromising an entity A's long-term private key will allow an adversary to impersonate A, but it should not enable the adversary to impersonate other entities to A.

Unknown key-share resilience: An entity A should not be able to be coerced into sharing a key with any entity C when in fact A thinks that she is sharing the key with another entity B.

Key control: Neither entity should be able to force the session key to be a preselected value.

The first key agreement protocol based on asymmetric cryptography was the Diffie-Hellman protocol [DH76]. It is a fundamental technique providing unauthenticated key agreement using exponentiation. Its security is based on the intractability of the Diffie-Hellman problem. Many key agreement protocols are based on the ideas of Diffie and Hellman, and such protocols can be described in groups such as the multiplicative groups $\mathbb{Z}_p^*$ (p a prime), $\mathbb{F}_{2^m}$, or the group of points on an elliptic curve over a finite field.

There have been many attempts to add authentication (and key confirmation) to the Diffie-Hellman protocol. One of the well-known authenticated key agreement (AK) protocols in the Diffie-Hellman family is the MQV protocol due to Menezes, Qu and Vanstone [MQV95]. This is a two-pass AK protocol. Law et al [LMQSV98] later presented a three-pass AKC protocol. It seems that the MQV protocol provides mutual implicit key authentication, and has the following security attributes: known-key security, forward secrecy, key-compromise impersonation and key control. As pointed out by Kaliski in [K01], this AK protocol is vulnerable to an unknown key share attack, although the later three-pass AKC protocol seems resistant to this attack.

In 1984, Shamir [Sh84] proposed the idea of using an identity-based asymmetric key pair where an arbitrary string (typically an identity string) can be used as a user's public key. A trusted authority (TA) is required to derive private keys from arbitrary public keys, and also publishes public information required for all encryption, decryption, signature and verification algorithms in the system. Systems of this nature are referred to as identity-based. Shamir gave a practical identity-based signature scheme but left as an open question the problem of finding an efficient identity-based encryption scheme.

An authenticated key establishment protocol is called identity-based if in the protocol, users use an identity based asymmetric key pair instead of a traditional public/private key pair for authentication and determination of the established key.

A few identity-based key agreement protocols have been developed based on Diffie-Hellman and using Shamir's key set up idea. For instance, Okamoto [Ok86] presented an identity-based scheme and Tanaka and Okamoto slightly modify this in [TO91]. Girault and Paillés [GP90] developed an identity-based system, which can be used for non-interactive key agreement. Another non-interactive identity-based key agreement scheme is also described in Annex B of ISO/IEC 11770-3 [ISO11770].

In 2000, Sakai et al. [SOK00] introduced a non-interactive identity-based key agreement scheme based on pairings on elliptic curves. Around the same time, Joux [Jo00] also used pairing techniques to present a (non-identity-based) tripartite key agreement protocol. Then in 2001 the first feasible solutions for identity-based encryption were published. One of them is Boneh and Franklin's identity-based encryption scheme [BF01], which uses the same key set-up as Sakai et al. Shortly after that, a few identity-based key agreement protocols (as well as signature schemes) based on pairing techniques were developed. Smart, by combining the ideas from [BF01,MQV95,LMQSV98] and [Jo00], proposed an identity-based authenticated key agreement protocol (ID-AK) and an identity-based authenticated key agreement protocol with key confirmation (ID-AKC) in [Sm02], although the security properties of these protocols have not been formally proven.

The contributions of this paper are as follows:

- To introduce an ID-AK protocol more efficient than Smart's.
- To modify Smart's protocol and our more efficient AK protocol to include the TA forward secrecy property which also prevents the TA from being able to passively escrow users' session keys.
- To introduce an AK protocol (also modified from Smart's) which allows users to choose different TAs.
- A discussion of the security properties of these protocols using heuristic arguments as well as formal provable security methods.

This paper is a corrected version of [CK02]. In the previous version we claimed that our authenticated key agreement protocols were secure in the model of Blake-Wilson et al. [BJM97], omitting the fact that this is only true provided the adversary is restricted in a certain way (by disallowing reveal queries). This was pointed out to us by Zhaohui Cheng [C03]. This version of the paper aims to correct this error.

2 Technical Background

2.1 Pairings

Pairings have recently had a number of positive applications in cryptography, for instance, identity-based encryption [BF01], identity-based key agreement [Jo00,Sm02,SOK00], identity-based signatures [He02,Pa02,SOK00], and short signatures [BLS01].

Let G_1 and G_2 denote two groups of prime order q, where G_1, with an additive notation, denotes a subgroup of the group of points on an elliptic curve; and G_2, with a multiplicative notation, denotes a subgroup of the multiplicative group of a finite field.

A pairing is a computable bilinear map between these two groups. Two pairings have been studied for cryptographic use. They are the Weil pairing [MOV93,Si94] and a modified version [Ve01,BF01], and the Tate pairing [FMR99, Ga01,GHSo02]. For the purposes of this paper, we let $\hat{e}$ denote a general bilinear map, i.e., $\hat{e} : G_1 \times G_1 \to G_2$, which can be either a modified Weil pairing or a Tate pairing, and which has the following three properties:

- Bilinear: If $P, P_1, P_2, Q, Q_1, Q_2 \in G_1$ and $a \in \mathbb{Z}_q^*$, then $\hat{e}(P_1 + P_2, Q) = \hat{e}(P_1, Q).\hat{e}(P_2, Q)$, and $\hat{e}(P, Q_1 + Q_2) = \hat{e}(P, Q_1).\hat{e}(P, Q_2)$.
- Non-degenerate: There exists a $P \in G_1$ such that $\hat{e}(P, P) \neq 1$.
- Computable: If $P, Q \in G_1$, one can compute $\hat{e}(P, Q)$ in polynomial time.

2.2 Some Mathematical Structures and Problems

A Diffie-Hellman (DH) tuple in G_1 is a tuple $(P, xP, yP, zP) \in G_1^4$ for some x, y, z chosen at random from $\mathbb{Z}_q$ satisfying $z = xy \bmod q$.

Computational Diffie-Hellman (CDH) problem: Given the first three elements in a DH tuple, compute the remaining element.

CDH assumption: There exists no algorithm running in expected polynomial time, which can solve the CDH problem with non-negligible probability.

Decision Diffie-Hellman (DDH) problem: Given a tuple $(P, xP, yP, zP) \in G_1^4$ for some x, y, z chosen at random from $\mathbb{Z}_q$, decide if it is a valid DH tuple. This can be solved in polynomial time by verifying the equation $\hat{e}(xP, yP) = \hat{e}(P, zP)$.

Bilinear Diffie-Hellman (BDH) problem: Let P be a generator of G_1. The BDH problem in $G_1, G_2, \hat{e}$ is given $(P, xP, yP, zP) \in G_1^4$ for some x, y, z chosen at random from $\mathbb{Z}_q$, compute $W = \hat{e}(P, P)^{xyz} \in G_2$.

BDH assumption: There exists no algorithm running in expected polynomial time, which can solve the BDH problem in $G_1, G_2, \hat{e}$ with non-negligible probability.

The security properties of our authenticated key agreement protocols are based on the CDH and BDH assumptions.

2.3 Security Model

In a later part of this paper, we will analyze the security of our protocols using a security model which was initially proposed by Bellare and Rogaway in [BR93] and later extended to the public key setting by Blake-Wilson et al in [BJM97]. Other papers also make use of suitably modified versions of this security model, including Al-Riyami and Paterson in [AP02], where they analyze the security of a tripartite authenticated key agreement protocol that is also based on pairing techniques. We adapt the model to the identity-based setting and summarize the adapted model as follows.

The model includes a set U of participants, each participant is modelled by an oracle, e.g., $\Pi^n_{I,J}$ would model a participant I carrying out a protocol session in the belief that it is communicating with another participant J for the nth time (i.e. the nth run of the protocol between I and J). Oracles keep transcripts which record messages they have sent or received as a result of queries they have answered.

Each participant has a pair of identity-based (ID-based) long-term asymmetric keys, where the public key is created using the participant's identifier and the private key is computed and issued by a TA. We assume there is a setup algorithm *Setup* which produces a description of the groups G_1 and G_2 and the bilinear map $\hat{e}$, assigns random tapes and oracles as necessary, and distributes a long-term master key to the TA.

The model also includes an adversary, E, who is neither a participant nor a TA. E is modelled by a probabilistic polynomial time Turing Machine and has access to all the participants' oracles as well as the random oracles in the game. E can relay, modify, delay, interleave or delete messages. E is called the benign adversary if she simply passes messages to and fro between participants. We note that all communications go through the adversary. Participant oracles only respond to queries by the adversary and do not communicate directly amongst themselves.

The adversary at any time can create new participants by making the following query:

Create: This allows E to set up a new participant for any identity ID of her choice. A new oracle is created to model this participant. The participant's public key is derived from ID, and the private key is obtained from the TA. E is given the public key of the participant.

In addition it is assumed that E is allowed to make the following types of queries of existing oracles as defined in [BJM97]:

Send: this allows E to send a message of her choice to an oracle, say $\Pi^n_{I,J}$, in which case participant I assumes the message has been sent by participant J. E may also make a special Send query λ to an oracle $\Pi^n_{I,J}$ which instructs I to initiate a protocol run with J. An oracle is an *initiator oracle* if the first message it has received is a λ. If an oracle did not receive a message λ as its first message, then it is a *responder oracle*.

Reveal: this allows E to ask a particular oracle to reveal the session key (if any) it currently holds to E.

Corrupt: this allows E to ask a particular oracle to reveal its long-term private key.

An oracle exists in one of the following several possible states:

Accepted: an oracle has accepted if it decides to accept, holding a session key, after receipt of properly formulated messages.

Rejected: an oracle has rejected if it decides not to establish a session key and to abort the protocol.

State *: an oracle is in state * if it has not made any decision to accept or reject.

Opened: an oracle is opened if it has answered a reveal query.

Corrupted: an oracle is corrupted if it has answered a corrupt query.

If two oracles, say $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$, have received (via the adversary) properly formatted messages exclusively generated by the other oracle, and exactly one oracle is an initiator oracle, having received the special Send query λ, we say that these two oracles have had a matching conversation (see [BR93] for a formal definition).

At some point in her attack, E may choose one of the oracles, say $\Pi^n_{I,J}$, to ask a single Test query. This oracle must have accepted, be unopened and neither I nor J can have been corrupted. Furthermore, there must be no opened oracle $\Pi^t_{J,I}$ with which it has had a matching conversation. To answer the query, the oracle flips a fair coin $b \in \{0,1\}$, and returns the session key held by $\Pi^n_{I,J}$ if $b = 0$, or else a random key sampled from $\{0,1\}^k$ if $b = 1$.

To attack a protocol, E does an experiment with the set of oracles generated by a challenger. During the experiment E may ask a polynomially bounded number of queries, including one Test query. At the end of E's attack, she must output a bit b' as her guess for b. E's advantage, denoted $advantage^E(k)$, is the probability that E can distinguish the session key held by the queried oracle from a random string, and is defined as:

$$Advantage^E(k) = |\Pr[b' = b] - 1/2|$$

Definitions of secure AK and AKC protocols are as follows:

Definition 1. *[BJM97] A protocol is a secure AK protocol if:*

1. *In the presence of the benign adversary on $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$, both oracles always accept holding the same session key, and this key is distributed uniformly at random on $\{0,1\}^k$; and if for every adversary E:*

2. *If uncorrupted oracles $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$ have matching conversations then both oracles accept and hold the same session key;*
3. *$Advantage^E(k)$ is negligible.*

A function $f(k)$ is negligible if for every $c > 0$ there exists $k_c > 0$ such that $f(k) < k^{-c}$ for all $k > k_c$. A function is non-negligible if it is not a negligible function.

Definition 2. *[BJM97] A protocol is a secure AKC protocol if:*

1. *In the presence of the benign adversary on $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$, both oracles always accept holding the same session key, and this key is distributed uniformly at random on $\{0,1\}^k$; and if for every adversary E:*
2. *If uncorrupted oracles $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$ have matching conversations then both oracles accept and hold the same session key;*
3. *The probability of $No-Matching^E(k)$ is negligible;*
4. *$Advantage^E(k)$ is negligible.*

In the third condition, $No-Matching^E(k)$ denotes the event that, when E attacks the protocol, there exists an oracle $\Pi^n_{I,J}$, which accepted where neither I nor J are corrupted, but there is no oracle $\Pi^t_{J,I}$ which has engaged in a matching conversation with $\Pi^n_{I,J}$. This condition says that essentially the only way for any adversary to get an uncorrupted entity to accept in a run of the protocol with any other uncorrupted entity is by relaying communications like a wire.

Our proofs of security assume that a participant I does not enter into protocol runs with itself, although we can remove this assumption provided that the BDH assumption is still valid against all algorithms that take as input the values (P, xP, yP, yP) instead of the values (P, xP, yP, zP).

2.4 Security Properties of the Model

We recall the security attributes which were mentioned in Section 1 and which are commonly required of AK and AKC protocols. We consider which of these attributes are implied by the above definitions of a secure AK or AKC protocol.

The property of known-key security is implied by the definitions. It follows by the following two properties of the model: (i) the adversary E is allowed to make Reveal queries to any oracle except for $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$ to obtain any session keys except for the key shared between $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$, called K_{IJ}, and (ii) after knowing all the other keys, her ability to distinguish between K_{IJ} and a random number is still negligible. Therefore, the knowledge of any other session keys does not help E to deduce any information about K_{IJ}.

The definitions also imply the unknown key-share resilience property. To show this we give a small sketch of a proof by contradiction as follows:

Suppose Π is a secure AK or AKC protocol and suppose that Π is susceptible to the unknown key-share attack. Then E has a non-negligible probability of making an oracle $\Pi^n_{I,J}$ accept holding a key K where I believes that there has been a matching conversation with $\Pi^t_{J,I}$ and K is shared with J, but K is in fact shared with some other oracle $\Pi^v_{X,Y}$ (usually $Y = I$ here). By the definitions of the security model described in Section 2.1, E can make a Reveal query to $\Pi^v_{X,Y}$ to obtain K because it is neither $\Pi^n_{I,J}$ nor $\Pi^t_{J,I}$. E can then choose oracle $\Pi^n_{I,J}$ to answer the test query. $\Pi^n_{I,J}$ will answer the test query (because both $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$ are unopened and both I and J are uncorrupted) and E will win the game. $Advantage_E(k)$ would therefore be non-negligible, contradicting the definitions.

We note that the definitions do not imply the key compromise impersonation or the forward secrecy properties. This is because the model does not allow the adversary to make queries of corrupted oracles and therefore does not model these types of attack. We prove these properties of our protocols separately later in the paper using adapted models.

3 Smart's ID-based AK Protocol

To describe the protocol, we use the notation, $Mi : A \rightarrow B : M$, to state that in the ith message flow, entity A sends a message M to entity B. This notation will be used throughout the paper.

Smart's ID-AK protocol [Sm02] involves three entities: two users Alice and Bob who wish to establish a shared secret session key, and a TA from whom they each acquire their own private key.

To provide a private key generation service, the TA uses a master public/private key pair. The public key is $(P, P_s = sP \in G_1)$ where P is a generator of G_1 and the private key is $s \in \mathbb{Z}_q$. When a user registers with the TA, the TA issues a private key $S = sQ$ for the user, where $Q = H_1(ID) \in G_1$, H_1 is a hash function, $H_1 : \{0,1\}^* \rightarrow G_1$, and ID is the user's identifier string. Note that this kind of identity based asymmetric key setup has been used in a number of ID-based encryption and signature schemes using pairings [BF01,He02,Pa02,SOK00].

Suppose that the TA issues the following private keys for Alice and Bob respectively: $S_A = sQ_A$ where $Q_A = H_1(\text{Alice's } ID)$, and $S_B = sQ_B$ where $Q_B = H_1(\text{Bob's } ID)$.

Alice and Bob each randomly choose an ephemeral private key, $a, b \in \mathbb{Z}^*_q$, and compute the values of the corresponding ephemeral public keys, $T_A = aP$ and $T_B = bP$. They then exchange the ephemeral public keys as follows:

Protocol 1.

$M1 : \text{Alice} \rightarrow \text{Bob} : T_A$

$M2 : \text{Bob} \rightarrow \text{Alice} : T_B$

Alice then computes $K_{AB} = \hat{e}(S_A, T_B) \cdot \hat{e}(aQ_B, P_s)$, and Bob computes $K_{BA} = \hat{e}(S_B, T_A) \cdot \hat{e}(bQ_A, P_s)$. If Alice and Bob follow the protocol, they will compute the same shared secret:

$$K = K_{AB} = K_{BA} = \hat{e}(bQ_A + aQ_B, P_s).$$

This shared secret value K is suitable to be used to derive a shared session key [GHS02]. We then use a key derivation function $H_2 : G_2 \rightarrow \{0,1\}^k$ to generate the shared session key $FK = H_2(K)$. This function would generally be a secure hash function, but we will model both H_1 and H_2 as random oracles.

Smart informally argues that this protocol has the following security properties: mutual implicit key authentication, known key security, partial forward secrecy (see discussion in the next section), imperfect key control (see discussion in the next section), key-compromise impersonation, and unknown key-share resilience. However no formal proofs are given for any of these claims.

We are now concerned about the following three issues:

Efficiency: In Protocol 1 (Smart's protocol), each participant has to generate a random number, perform two elliptic curve point multiplications, and compute two pairings (which are the most expensive operations in the protocol). In the next section, we will introduce a more efficient protocol, which seems to offer the same security properties as Protocol 1.

Key escrow: As mentioned above, this protocol only has partial forward secrecy, and not perfect forward secrecy, which means the compromise of both of the long-term private keys used in a key agreement protocol compromises the session key established in that protocol. This also implies that the TA (who can compute all the private keys in the system) is able to passively escrow session keys established between users. This property may not acceptable for some applications. Although users in an ID-based system must trust the TA to generate their private keys and not to impersonate them, they may still want to conduct their own communications without the TA being able to passively escrow their session keys as well.
This security feature holds in the identity-based key agreement protocols using Shamir's key set up [Ok86,TO91,GP90,ISO11770]. In Section 5, we will describe a modification for Smart's protocol that prevents the TA from being able to compute the established key.

Single TA: In this protocol, both Alice and Bob register with a single TA. This may be suitable for certain applications where Alice and Bob belong to the same domain. However in reality, Alice and Bob may have registered with different TAs and may still want to be able to share a key. Existing identity-based key agreement protocols do not seem to provide this capability. In Section 6, we will introduce an extended protocol to support such a requirement.

In each of the following three sections, we give a modification of Smart's protocol. Each focuses on one of the above three issues.

4 A More Efficient AK Protocol

We describe our first modification of Protocol 1. Alice and Bob each randomly choose an ephemeral private key, $a, b \in \mathbb{Z}_q^*$, and compute the values of the corresponding ephemeral public keys, $W_A = aQ_A$ and $W_B = bQ_B$. They then exchange the public keys as follows:

Protocol 2.

$M1$: Alice $\rightarrow$ Bob : W_A

$M2$: Bob $\rightarrow$ Alice : W_B

At the conclusion of the protocol Alice computes $K_{AB} = \hat{e}(S_A, W_B + aQ_B)$, and Bob computes $K_{BA} = \hat{e}(W_A + bQ_A, S_B)$. If Alice and Bob follow the protocol, they will compute the same shared secret:

$$K = K_{AB} = K_{BA} = \hat{e}(Q_A, Q_B)^{s(a+b)}.$$

Their shared secret session key is then $FK = H_2(K)$.

The above protocol has a very similar construction to Protocol 1. However, it is more efficient. Protocol 1 requires each party to perform two elliptic curve point multiplications and two evaluations of the pairing. Protocol 2 requires each party to perform two elliptic curve point multiplications, one elliptic curve point addition and only a single evaluation of the pairing.

With the description of the security model in Section 2.2, we now state:

Theorem 1. *Protocol 2 is a secure AK protocol, assuming that the adversary does not make any Reveal queries, the BDH problem (for the pair of groups G_1 and G_2) is hard and provided that H_1 and H_2 are random oracles.*

Proof. Conditions 1 and 2 of Definition 1 follow from the assumption that the two oracles follow the protocol and in both cases have matching conversations. Therefore both oracles accept (since they both receive correctly formatted messages from the other oracle) holding the same key FK (since $K_{AB} = K_{BA}$ by the bilinearity of the pairing and the matching conversation). Since H_2 is a random oracle, FK is distributed uniformly at random on $\{0,1\}^k$.

Condition 3. We assume that there exists an adversary E who can win the above game with non-negligible advantage $\eta(k)$ in time $\tau(k)$, making at most T_H queries to the H_2 random oracle and T_C Create queries.

We now construct from E an algorithm F which solves the BDH problem with non-negligible probability. Given input of, as described in Section 2, the two groups G_1, G_2, the bilinear map $\hat{e}$, a generator of P of G_1, and a triple of elements $xP, yP, zP \in G_1$ with $x, y, z \in \mathbb{Z}_q^*$ where q is the prime order of G_1 and G_2, F's task is to compute and output the value $g^{xyz} \in G_2$ where $g = \hat{e}(P, P)$.

F chooses distinct random values I and J from $\{1, ..., T_C\}$, and a value $l \in \{1, ..., T_H\}$. F simulates the Setup algorithm and sets the TA's master key to be xP. F will also simulate all oracles required during the game.

F then starts E, and answers all E's queries as follows. F maintains two random oracles H_1 and H_2. E can query the H_2 oracle directly at any time, but the H_1 oracle is only used to answer Create queries (described later), and so is never queried directly by E.

Random Oracle queries: F simulates the random oracle H_1 by keeping a list of tuples $\langle ID_i, Q_i \rangle$ which is called the H_1-List. When the H_1 oracle is queried (via a Create query) with an input $ID_i \in \{0,1\}^*$, F responds as follows. If ID_i is already on the H_1-List in the tuple $\langle ID_i, Q_i \rangle$, then F outputs Q_i. Otherwise:

1. If ID_i is the Jth distinct H_1 query, then the oracle outputs $Q_= yP$ and adds the tuple $\langle ID_i, Q_i \rangle$ to the H_1 list.

2. Otherwise F selects a random $r_i \in \mathbb{Z}_q^*$ and outputs $Q_i = r_i P$, and then adds the tuple $\langle ID_i, Q_i \rangle$ to the H_1 list.

F simulates the H_2 oracle in the same way, keeping an H_2 list, but always answers distinct queries randomly.

Create queries: We require that E never queries the H_1 oracle directly, but always indirectly through the Create query. F simulates the create query on input ID_i by querying the H_1 oracle on ID_i for the appropriate public key $Q_i = r_i P$, and in addition sets up a new oracle with public key Q_i and private key $S_i = r_i x P$. The public key Q_i is given to E. We call the oracle set up by the ith distinct Create query the ith participant. In particular, the Jth participant, which we shall call participant J, will have public key $Q_J = yP$. We note that F is unable to compute the private key for this oracle.

Corrupt queries: F answers Corrupt queries as specified by a normal oracle, i.e., revealing the long-term private key of the related participant, except that if E asks I or J a Corrupt query, F gives up.

Send queries: F answers all Send queries as specified for a normal oracle, i.e., for the first Send query to an oracle, F takes a random value in $\mathbb{Z}_q^*$ to form its contribution, except that if E asks $\Pi^n_{I,J}$ for any n its first Send query, F chooses a random $s_n \in \mathbb{Z}_q^*$ and answers $s_n r_I z P = s_n z Q_I$. We only specify the response to the first Send query to an oracle since this is the only Send query to which the oracle will respond by outputting its ephemeral contribution to the session key in our protocol. If the oracle is an initiator oracle, the first Send query will be a special initiating query, and if it is a responder oracle, the first Send query will contain an ephemeral input to $\Pi^n_{I,J}$ (purportedly from oracle J).

Reveal queries: F does not need to answer Reveal queries since we do not allow the adversary to make Reveal queries.

Test query: At some point in the simulation, E will ask a Test query of some oracle. If E does not choose one of the oracles $\Pi^n_{I,J}$ for some n to ask the Test query, then F aborts. However if E does pick $\Pi^n_{I,J}$ for the Test query, then $\Pi^n_{I,J}$ must have accepted, and I and J must be uncorrupted. Assuming that $\Pi^n_{I,J}$ obtained some value jQ_J as its input prior to accepting, the oracle should hold a session key of the form $H_2(\hat{e}(xQ_I, s_n z Q_J + jQ_J))$. However F cannot compute this key, so F cannot correctly simulate the Test query. Instead, F simply outputs a random value.

If F does not abort at some stage during the simulation and E does not detect F's inconsistency in answering the Test query, then E's probability of success is still $\eta(k)$ as in a real attack. However for E to distinguish the session key from a random value with non-negligible probability $\eta(k)$, E must have queried the H_2 oracle on the value $H_2(\hat{e}(xQ_I, s_n z Q_J + jQ_J))$ with some non-negligible probability $\eta'(k)$.

If F does not abort at some point during the attack but E is able to detect F's inconsistency in answering the Test query, then E's behavior becomes undefined after this point. In particular, E may not terminate, so we assume that F terminates E's attack if it lasts longer that time $\tau(k)$. However for E to be able to detect this inconsistency, E must have queried H_2 on the value

$H_2(\hat{e}(xQ_I, s_n zQ_J + jQ_J))$. Up until this point, E's view of the simulation is indistinguishable from a normal attack, so the number of H_2 queries made by E is still bounded by T_H at this point.

At the end of E's attack, if E has not made at least l queries to the H_2 oracle, then F aborts. Otherwise F picks E's lth query (on some value h) to the H_2 oracle, which F guesses to be the value $\hat{e}(xQ_I, s_n zQ_J + jQ_J) = \hat{e}(xP, yP)^{r_I(s_n z + j)} = \delta\hat{e}(P, P)^{xyz\gamma}$, with $\delta = \hat{e}(r_I xP, jQ_J)$ and $\gamma = r_I s_n$. F outputs $(h/\delta)^{1/\gamma}$ as its guess for the value $\hat{e}(p, p)^{xyz}$.

So the probability that F did not abort at some stage and produces the correct output is at least

$$\frac{\eta'(k)}{T_C^2 T_H}$$

which is non-negligible. □

In Blake-Wilson et al. [BJM97], the proof of security of their AK protocol also assumes that the adversary makes no Reveal queries. They also point out that no protocol in which the initiator oracle and responder oracle generate the shared key in the same way (ie. the protocol is role symmetric) can be secure according to Definition 1. Since our protocol is role symmetric, it is certainly not secure by Definition 1 without further restricting the adversary in some way.

To see a further example of why Protocol 2 is not secure according to Definition 1 unless we disallow the adversary to make Reveal queries, we illustrate a simple "attack" in the model that the adversary could use to win the game.

In the example we refer to the adversary as E, and assume A and B wish to share a key.

A sends aQ_A (intended for B), but E forwards $aQ_A + cQ_A$ to B for some $c \in_R \mathbb{Z}_q^*$

B sends bQ_B (intended for A), but E forwards $bQ_B + cQ_B$ to A.

A's transcript contains aQ_A and $bQ_B + cQ_B$ and A computes her key as $K_A = \hat{e}(sQ_A, aQ_B + bQ_B + cQ_B)$.

B's transcript contains bQ_B and $aQ_A + cQ_A$. This is different to A's transcript, so A and B have not had a matching conversation, but B computes his key as $K_B = \hat{e}(bQ_A + aQ_A + cQ_A, sQ_B) = K_A$.

Now A and B share the same key, but have not had matching conversations. E cannot compute this key, but E can ask to reveal B's key, and then choose oracle A as her test oracle. This is a valid choice, since neither A or B are corrupted, A is not revealed, and no oracle with which A had a matching conversation has been revealed (since there exists no such oracle). E then trivially wins the game since she learned the exact value of A's key by revealing B's key.

The problem is really that it is possible for two oracles to generate the same session key without having had a matching conversation. We do not consider this attack to be particularly useful to an adversary in real life, but it does illustrate one of the reasons we restrict the adversary to not being able to make Reveal queries. In fact several other protocols are vulnerable to such a type of "attack", such as Protocol 3 in [BJM97], Smart's protocol [Sm02], and certain protocols in [AP02].

Since we assume that the adversary cannot make reveal queries, our proof of security does not guarantee the Known-key security property. However we can heuristically argue that each protocol run produces a different session key, and it seems difficult to deduce any information about future (new) session keys from knowledge of past session keys under the BDH assumption.

4.1 Additional Properties of the Protocol

Since the security model which we used to prove Theorem 1 does not cover some known active attacks, we now discuss some of the security properties related to these attacks, using heuristic arguments to support our claims.

1. Partial forward secrecy. We consider the following three separate parts of this property:
   (1) Compromise of long-term secret keys S_A or S_B does not seem to lead to the compromise of past communications. But compromising both S_A and S_B does lead to the compromise of past communications. This is because K can be computed as $K = \hat{e}(S_A, W_B) \cdot \hat{e}(W_A, S_B)$, which requires knowledge of the ephemeral exchanges and the long term private keys, but not the ephemeral secret values. So the protocol does not offer perfect forward secrecy.
   (2) Compromise of the TA's master key s does lead to the compromise of past communications, because $K = \hat{e}(Q_A, W_B)^s \hat{e}(W_A, Q_B)^s$. This means the protocol does not offer TA forward secrecy.
   (3) Compromise of one or both of the ephemeral private keys, a and b, reveals neither the long-term secret keys, S_A, S_B and s, nor the shared secret session key FK.
   In the following section, we will modify Protocols 1 and 2 in order to provide TA forward secrecy. This means that compromising the TA's master key (which also means compromising the long-term secret keys of all users) does not lead to the compromise of past communications.
2. Imperfect key control. Protocol 2 does not have the full key control attribute since Bob can select his ephemeral key after having received Alice's ephemeral key. Bob can force l bits of the shared secret key to have a nominated value by evaluating K for roughly 2^l different choices of b. As is noted in [MWW98], the responder in a protocol almost always has an unfair advantage in controlling the value of the established session key. This can be avoided by the use of commitments, although this seems to intrinsically require an extra round of communication.
3. Unknown key-share resilience. This property follows from Theorem 1 above based on the discussion in Section 2.3.
4. Key-compromise impersonation. When an adversary knows Alice's long-term private key, S_A, the adversary is not able to impersonate other entities, say Bob, to Alice in a successful protocol run. We establish this in the following result.

Theorem 2. *For any participants Alice and Bob, no polynomial time adversary who knows Alice's private key but not Bob's private key can impersonate Bob* *to Alice in Protocol 2, under the BDH assumption, assuming the adversary does not make any reveal queries, and provided that H_1 and H_2 are random oracles.*

Proof. First of all, to prove this theorem, we adapt the model in Section 2.3 to allow an adversary E to make a Test query to any oracle $\Pi^n_{I,J}$ where I (but not J) has been corrupted. We also assume that, although the adversary may know the long term key of the test oracle, we do not allow the adversary control over the outputs of this oracle (otherwise the adversary can trivially win the game).

Now the proof follows exactly the proof of Theorem 1 except for the minor adjustment to allow the adversary to corrupt participant I at any stage. We notice that the challenger F can compute the private key of I, so can answer this query, and revealing this private key to the adversary makes no probabilistic difference to the outcome of the game. So the rest of the proof follows that of Theorem 1 exactly as before. □

5 Modification of Protocols 1 and 2 without Key Escrow

Note that in systems using identity-based cryptography (IBC), we cannot escape the possibility of a TA impersonating any user in the system because the TA is always able to do so. In PKI we have the same problem in fact. A CA (certification authority) can generate a key pair, and (falsely) certify that the public key belongs to a user A. The CA can then impersonate A to any other user B. In both IBC and PKI we therefore have to assume that the trusted authority (TA or CA) will not impersonate users.

However a property that we may require from our identity-based key agreement protocol is that, if two users are actually communicating with each other (that is, no user is being actively impersonated by the TA), then the TA cannot passively derive (and therefore escrow) the established session key. This is mainly a privacy issue since users must trust the TA with their long-term keys, but may wish to be able to escape from the escrow environment (assuming no active attacks by the TA) for communications they wish to keep confidential even from the TA.

TA forward secrecy means that if the TA's master key is compromised, this should not compromise the previously established session keys. We notice that this property also implies that the TA is unable to passively deduce information about session keys established between users.

Recall that TA forward secrecy also implies the property of perfect forward secrecy, another property we would like from our protocols. However the converse is not necessarily true: perfect forward secrecy does not necessarily imply TA forward secrecy since the TA knows not only all users' long-term private keys, but also s, the TA's long-term master secret.

So TA forward secrecy seems like a good property for identity-based key agreement protocols, and guarantees both the perfect forward secrecy property and resistance to session key escrow by the TA.

To provide TA forward secrecy, the ephemeral keys (perhaps combined with the long-term keys) are used in such a way that the result cannot be computed by the TA (or by anyone else who knows only the long-term secret keys). The most well-known method of achieving this is for the users to calculate a Diffie-Hellman shared key from their ephemeral contributions. We now introduce protocols modified from Protocols 1 and 2, which have the above desired properties.

Protocol 1':

There is a simple way to introduce a Diffie-Hellman shared key in Protocol 1 by changing the key derivation function H_2 in Protocol 1 to $H_2' : G_2 \times G_1 \rightarrow \{0,1\}^k$, and the shared session key becomes

$$FK = H_2'(K, abP).$$

where K is established as in Protocol 1.

We will refer to this protocol as Protocol 1'.

In this case, if an adversary compromises both the users' long-term private keys, S_A and S_B, at some point in the future, the adversary is not able to compromise communications in the past, because the adversary can calculate K but not abP. It is obvious that this modification also prevents the TA from being able to compute the session key in passive attacks.

Protocol 2':

Note that this exact modification cannot be used in Protocol 2 because in Protocol 2, Alice and Bob exchange aQ_A and bQ_B, which are not Diffie-Hellman contributions. If avoidance of key escrow is required, we suggest the following modification.

Alice and Bob exchange aQ_A, aP and bQ_B, bP. They then compute K as in Protocol 2, and finally compute the shared secret key as $FK = H_2'(K, abP)$. We will refer to this protocol as Protocol 2'.

Compared with Protocol 1', Protocol 2' is more computationally efficient, in particular in pairing computations since only a single pairing is required (as opposed to two), but less efficient on the message bandwidth since two points (as opposed to only one) need to be distributed by each user.

Theorem 3. *Protocol 2'*

i) is a secure AK protocol, assuming that the adversary does not make any Reveal queries, the BDH problem (for the pair of groups G_1 and G_2) is hard and provided that H_1 and H_2' are random oracles, and

ii) has the TA forward secrecy property, assuming that the CDH problem (for group G_1) is hard, and provided that H_2' is a random oracle.

Proof. (sketch) i) The fact that Protocol 2' is a secure AK protocol by Definition 1 provided that the adversary does not make any reveal queries follows directly from Theorem 1 (with slight modifications to the parsing of inputs to the H_2 oracle) since the properties proved for Protocol 2 follow directly for Protocol 2'.

ii) We note that when proving the TA forward secrecy property, we need to slightly modify the game between the adversary and the challenger. E will be given the TA's master secret in the setup procedure, and will no longer need to make corrupt queries since E can now compute all private keys for herself. However we will allow E to once again make reveal queries. We also now require that when E chooses an oracle $\Pi^n_{I,J}$ for her test query, this oracle must indeed have had a matching conversation with some other oracle $\Pi^t_{J,I}$. We prove that Protocol 2' has TA forward secrecy as follows:

The proof follows along similar lines to the proof of Theorem 1. We assume that there exists an adversary E who can win the above game with non-negligible advantage $\eta(k)$ in time $\tau(k)$, making at most T_H queries to the H_2 random oracle and T_C Create queries.

We now construct from E an algorithm F which solves the CDH problem with non-negligible probability. Given input of the group G_1, a generator of P of G_1, and the elements $aP, bP \in G_1$ with $a, b \in \mathbb{Z}_q^*$ where q is the prime order of G_1, F's task is to compute and output the value $abP \in G_1$.

F's operation is as before in the proof of Theorem 1, picking the values I, J and l and simulating the running of the key generation algorithm, but in this case, the TA's public key is sP where s is known, and E is given the master secret s.

F answers E's queries as follows:

F simulates the H_1 oracle as before except that now $H_1(ID_J)$ is computed in the same way as for any other input, so J's public key is $Q_J = H_1(ID_J) = r_J P$ for some $r_j \in_R \mathbb{Z}_q^*$. The H_2 and Create queries are as before.

We now allow E to ask Reveal queries, and F answers by outputting the appropriate key, except if E asks $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ a Reveal query, then F gives up.

If E asks $\Pi^n_{I,J}$ its first Send query, F answers $s_n aQ_A = s_n r_A aP, s_n aP$ and if E asks $\Pi^t_{J,I}$ its first Send query, F answers $s'_t bQ_B = s'_t r_B bP, s'_t bP$.

Test query: If E does not choose one of the oracles $\Pi^n_{I,J}$ for some n to ask the Test query, then F aborts. However if E does pick $\Pi^n_{I,J}$ for the Test query, then $\Pi^n_{I,J}$ must have accepted after having had a matching conversation with an oracle $\Pi^t_{J,I}$ for some t. Therefore $\Pi^n_{I,J}$ obtained the value $s'_t bQ_B = s'_t r_B bP, s'_t bP$ as its input prior to accepting. The oracle should hold a session key of the form $H_2(\hat{e}(sQ_I, s_n aQ_J + s'_t bQ_J), s_n s'_t abP)$. However F cannot compute this key, so F cannot correctly simulate the Test query. Instead, F simply outputs a random value.

At the end of E's attack, if E has not made at least l queries to the H_2 oracle, then F aborts. Otherwise F picks E's lth query (on some pair of values h, h') to the H_2 oracle, and F outputs $h'/s_n s'_t$ as its guess for the value abP.

As before, the probability that F did not abort at some stage and produces the correct output is at least

$$\frac{\eta'(k)}{T_C^2 T_H}$$

which is non-negligible. □

6 A Pairing Based AK Protocol with Separate TAs

Suppose that there are two trusted authorities, say TA_1 and TA_2, which have public/private key pairs $(P, s_1P \in G_1, s_1 \in \mathbb{Z}_q^*)$ and $(P, s_2P \in G_1, s_2 \in \mathbb{Z}_q^*)$ respectively, where P, G_1 and G_2 are globally agreed, e.g., recommended by an international standards body.

Suppose also that Alice registers with TA_1 and gets her private key $S_A = s_1Q_A$ where $Q_A = H_1(\text{Alice's ID})$, and Bob registers with TA_2 and gets his private key $S_B = s_2Q_B$ where $Q_B = H_1(\text{Bob's ID})$.

Protocol 1 can be modified as follows. Alice and Bob each randomly choose an ephemeral private key, $a, b \in \mathbb{Z}_q^*$, and compute the values of the corresponding ephemeral public keys, $T_A = aP$ and $T_B = bP$. They then exchange the ephemeral public keys as follows:

Protocol 3:

$M1 : \text{Alice} \rightarrow \text{Bob} : T_A$

$M2 : \text{Bob} \rightarrow \text{Alice} : T_B$

Alice then computes $K_{AB} = \hat{e}(S_A, T_B) \cdot \hat{e}(Q_B, as_2P)$, and Bob computes $K_{BA} = \hat{e}(S_B, T_A) \cdot \hat{e}(Q_A, bs_1P)$. If Alice and Bob follow the protocol, they will compute the same shared secret: $K = K_{AB} = K_{BA} = \hat{e}(bS_A + aS_B, P)$. Their shared secret session key is then $FK = H_2(K)$, which does not have the TA forward secrecy property (if the two TAs collude), or $FK = H_2'(K, abP)$, which has the TA forward secrecy property.

Efficiency: Each party is required to compute two elliptic curve point multiplications and two evaluations of the pairing, and one extra elliptic curve point multiplication if the TA forward secrecy property is required.

Security: This protocol has the same security properties as Protocol 1, except for the second part of TA forward secrecy. For the standard version, the compromise of either of the TA's master keys s_1 or s_2 does not lead to the compromise of communications in the past. But knowing both s_1 and s_2 will allow anyone to compute the session key via $K = \hat{e}(Q_B, T_A)^{s_2} \cdot \hat{e}(Q_A, T_B)^{s_1}$. This implies that the two TAs must work together (or collude) in order to determine any secret session keys. For the version with TA forward secrecy, even the compromise of both s_1 and s_2 (or the collusion of both TAs) does not compromise the shared session key. Based on Definition 1 described in Section 2, we have the following theorem.

Theorem 4. *Protocol 3 is a secure AK protocol, assuming that the adversary does not make any Reveal queries, the BDH problem (for the groups G_1 and G_2) is hard and provided that H_1 and H_2 are random oracles.*

Proof. We assume in this proof that when the adversary picks an oracle $\Pi^n_{I,J}$ for its test query that participant I's secret key is generated by TA_1, and participant J's long term key is generated by TA_2. This assumption may be omitted, but it simplifies the proof somewhat. The proof of this theorem now follows that of Theorem 1 except for the following minor changes.

When F simulates the running of the key generation algorithm O, there are now 2 TAs, so F chooses TA_1's public key as s_1xP and TA_2's public key as s_2xP, and all other participants' keys are computed as in the proof of Theorem 1. During E's attack, F answers all of E's queries as it does in the proof of Theorem 1, except that when E asks $\Pi^n_{I,J}$ its first Send query, F answers s_nzP (and not s_nr_IzP as before).

The rest of the proof follows the proof of Theorem 1 almost exactly (with slight changes to the values of δ and γ). We leave the details to the reader. □

In the protocol presented, each user gets their long term private key from a single chosen TA. Therefore users have to trust the TAs not to impersonate any entity using their key generation services. If users do not want an individual TA to have too much power, each user can use multiple TAs instead of a single one. [CHSS02,CHMSS02] have recently proposed a number of techniques for using multiple TAs in ID-based cryptography, where the TAs do not have to share secrets with each other, and users are able to flexibly choose a set of TAs for each application.

7 Key Confirmation Process

This section describes how to add key confirmation to Protocol 2 to form an AKC protocol (the same can be done for Protocol 3). We derive an AKC protocol from an AK protocol by adding the MACs of the flow number, identities and the ephemeral public keys.

The following is a general protocol extended from Protocol 2. In particular, W_A, W_B and K are computed as in Protocol 2. Here, MACs are used to provide key confirmation; and H_2 and H_3 are two independent key derivation functions, $FK = H_2(K)$ and $FK' = H_3(K)$.

Protocol 4:

$M1$: Alice $\rightarrow$ Bob : W_A

$M2$: Bob $\rightarrow$ Alice : $W_B, \text{MAC}_{FK'}(2, ID_A, ID_B, W_A, W_B)$

$M3$: Alice $\rightarrow$ Bob : $\text{MAC}_{FK'}(3, ID_A, ID_B, W_A, W_B)$

If the protocol succeeds, Alice and Bob share the session key, FK.

Smart also adds key confirmation to his AK protocol to form an AKC protocol [Sm02], although it is slightly different to the method described here. It also explicitly includes the session key material before the derivation function is applied inside the MAC, which is not generally considered as secure as simply including the ephemeral public keys.

The method used here is well known and is identical to that used to add key confirmation to the MQV AK protocol as described in [LMQSV98]. This in turn followed the key confirmation method used by Blake-Wilson et al in [BJM97].

By using Definition 2 of a secure AKC protocol in Section 2 and the concept of a secure MAC, taken from [BJM97], we have the following theorem.

Theorem 5. *Protocol 4 is a secure AKC protocol, assuming that the adversary does not make any reveal queries, the BDH problem (for the pair of groups* G_1

and G_2) is hard, the MAC is secure and provided that H_1, H_2 and H_3 are random oracles.

The proof of this theorem follows similar lines to the proof of Theorem 9 in [BJM97], as well as our proof of Theorem 1, and we leave the details to the reader.

8 Conclusions

We have investigated some security issues related to identity based authenticated key agreement, and proposed a few new protocols modified from a previous protocol to efficiently achieve certain security properties. We have then used techniques from provable security to analyze the security properties of our new protocols.

Interesting further work would be to find an ID-based AK protocol which is provably secure in our model without having to place additional restrictions on the adversary. It would also be interesting to develop a more comprehensive model of security for key agreement protocols in which a definition of security would automatically imply all the security properties deemed necessary in this paper.

Acknowledgements

We thank Kenneth Paterson for invaluable comments on this latest version and Zhaohui Cheng for pointing out the error in the original version of the paper.

References

[AP02] S. Al-Riyami and K.G. Paterson, Tripartite authenticated key agreement protocols from pairings. In *K.G. Paterson (ed), Proc. IMA Conference on Cryptography and Coding*, LNCS Vol. 2898, pages 332-359, Springer-Verlag, Berlin, 2003.

[BF01] D. Boneh and M. Franklin, Identity-based encryption from the Weil pairing. In *Advances in Cryptology - CRYPTO '01*, LNCS 2139, pages 213-229, Springer-Verlag, 2001.

[BJM97] 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*, LNCS 1355, pages 30-45, Springer-Verlag, 1997.

[BLS01] D. Boneh, B. Lynn, and H. Shacham, Short signatures from the Weil pairing. In *Advances in Cryptology - ASIACRYPT '01*, LNCS Vol.2248, pp. 514-532, Springer-Verlag, 2001.

[BR93] M. Bellare and P. Rogaway, Entity authentication and key distribution. In *Advances in Cryptology - CRYPTO '93*, LNCS 773, pages 232-249, Springer-Verlag, 1994. Full version available at `http://www-cse.ucsd.edu/users/mihir`.

[C03] Z. Cheng, Private communication, December 2003.

[CHSS02] L. Chen, K. Harrison, N.P. Smart, and D. Soldera, Applications of multiple trust authorities in pairing based cryptosystems. In *Proceedings of Infrastructure Security Conference 2002*, LNCS 2437, pages 260-275, Springer-Verlag, 2002.

[CHMSS02] L. Chen, K. Harrison, A. Moss, N.P. Smart, and D. Soldera, Certification of public keys within an identity based system. In *Proceedings of Information Security Conference 2002*, LNCS 2433, pages 322-333, Springer-Verlag, 2002.

[CK02] L. Chen and C. Kudla, Identity based key agreement protocols from pairings. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop*, pages 219-233, IEEE Computer Society Press, June 2003.

[DH76] W. Diffie and M.E. Hellman, New directions in cryptography. In *IEEE Transactions on Information Theory*, 22:644-654, 1976.

[FMR99] G. Frey, M. Mller and H. Rck, The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. In *IEEE Transactions on Information Theory*, 45(5):1717-1719, 1999.

[Ga01] S. Galbraith, Supersingular curves in cryptography. In *Advances in Cryptology - Asiacrypt' 01*, LNCS 2248, pages 495-513, Springer-Verlag, 2001.

[GHS02] S.D. Galbraith, H.J. Hopkins and I.E. Shparlinski, Secure Bilinear Diffie-Hellman Bits. In the *Cryptology ePrint Archive*, Report 2002/155, 2002.

[GHSo02] S.D. Galbraith, K. Harrison and D. Soldera, Implementing the Tate Pairing. In *ANTS 2002*, pages 324-337, 2002.

[GP90] M. Girault and J.C. Paillés. An identity-based scheme providing zero-knowledge authentication and authenticated key exchange. In *Proceeedings of ESORICS '90*, pages 173-184, 1990.

[He02] F. Hess, Efficient identity based signature schemes based on pairings. In *Proceedings of the 9th Workshop on Selected Areas in Cryptography, SAC 2002*, LNCS 2595, pages 310-324, Springer-Verlag, 2003.

[ISO11770] ISO/IEC 11770-3, Information technology - Security Techniques - Key management - Part 3: Mechanisms using asymmetric techniques. International Organization for Standardization, Geneva, Switzerland, 1999 (first edition).

[Jo00] A. Joux, A one round protocol for tripartite Diffie-Hellman. In *Proceedings of Algorithmic Number Theory Symposium, ANTS-IV*, LNCS 1838, pages 385-394, Springer-Verlag, 2000.

[K01] B.S. Kaliski Jr, An unknown key-share attack on the MQV key agreement protocol. In *ACM transactions on Information and System Security*, 4(3):275-288, August 2001.

[LMQSV98] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, An efficient protocol for authenticated key agreement. Technical Report CORR 98-05, 1998. Available at `citeseer.nj.nec.com/law98efficient`.

[MOV93] A. Menezes, T. Okamoto and S. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field. In *IEEE Transactions on Information Theory*, 39:1639-1646, 1993.

[MQV95] A. Menezes, M. Qu and S. Vanstone, Some new key agreement protocols providing mutual implicit authentication. In *Proceedings of the Second Workshop on Selected Areas in Cryptography, SAC '95*, pages 22-32, 1995.

[MWW98] C. Mitchell, M. Ward and P. Wilson, Key control in key agreement protocols. In *Electronics Letters*, 34:980-981, 1998.

[Ok86] E. Okamoto, Proposal for identity-based key distribution system. In *Electronics Letters*, 22:1283-1284, 1986.

[Pa02] K.G. Paterson, ID-based signatures from pairings on elliptic curves. In *Electronics Letters*, Vol. 38 (18) (2002), 1025-1026. Available at `http://eprint.iacr.org/2002/004/`.

[Sh84] A. Shamir, Identity-based cryptosystems and signature schemes. In *Advances in Cryptology - CRYPTO '84*, LNCS 196, pages 47-53, Springer-Verlag, 1984.

[Si94] J.H. Silverman, *Advanced topics in the arithmetic of elliptic curves*, Graduate Texts in Math., vol. 151, Springer-Verlag, Berlin and New York, 1994.

[Sm02] N. Smart, An identity based authenticated key agreement protocol based on the Weil pairing. In *Electronics Letters*, 38:630-632, 2002.

[SOK00] R. Sakai, K. Ohgishi and M. Kasahara, Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security (SCIS2000)*, Okinawa, Japan, 2000.

[TO91] K. Tanaka and E. Okamoto, Key distribution system for mail systems using ID-related information directory. In *Computers and Security*, 10:25-33, 1991.

[Ve01] E. Verheul, Evidence that XTR is more secure than supersingular elliptic curve systems. In *Advances in Cryptology - EUROCRYPT 2001*, LNCS 2045, pages 195-210, Springer-Verlag, 2001.