Identity Based Authenticated Key Agreement Protocols from Pairings

Liqun Chen and Caroline Kudla[1]

Hewlett-Packard Laboratories,
Filton Road, Stoke Gifford, Bristol BS34 8QZ, United Kingdom.
{liqun.chen, caroline.kudla}@hp.com

28th November 2002

Abstract. We investigate a number of issues related to identity based authenticated key agreement protocols in the Diffie-Hellman family enabled by the Weil or Tate pairings. These issues include how to make protocols efficient; to avoid key escrow by a Trust Authority (TA) who issues identity based private keys for users, and to allow users to use different TAs. We describe a few authenticated key agreement (AK) protocols and AK with key confirmation (AKC) protocols by modifying Smart's AK protocol [Sm02]. We discuss the security of these protocols heuristically and give formal proofs of security for our AK and AKC protocols (using a security model based on the model defined in [BJM97]). We also prove that our AK protocol has the key compromise impersonation property. 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), and we note that this also implies that it 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 form.

A key agreement protocol is said to provide *implicit key authentication* (of B to 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* protocol (or AK protocol). A key agreement protocol provides *key confirmation* (of B to A) if A is assured that B 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:

1. *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.
2. *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 of one (or more 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

[1] Associated with Royal Holloway, University of London. {c.j.kudla@rhul.ac.uk}

corrupted (and hence all users' long-term private keys) without compromising the security of session keys previously established by any users.

3. *Key-compromise impersonation*: The compromise of 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.
4. *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.
5. *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 and the discrete logarithm problem. Many key agreement protocols are based on the ideas of Diffie-Hellman, and such protocols can be described in any group in which the discrete logarithm is hard and exponentiation is efficient. Such groups include the multiplicative groups $\mathbf{Z}^*_p$ (p a prime), 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 by Menezes, Qu and Vanstone [MQV95]. This is a two-pass AK protocol. It is heuristically argued 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 Burton and Kaliski in [BK01], this AK protocol is vulnerable to an unknown key share attack. To prevent this attack, Law *et al* [LMQSV] present a three-pass AKC protocol.

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

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. The TA also publishes public information required for all encryption, decryption, signature and verification algorithms in the system. This is referred to as identity-based cryptography (IBC). Shamir gave a practical ID-based signature scheme but left as an open question the problem of finding an efficient ID-based encryption scheme.

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

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 is based on pairing on elliptic curves. Shortly after that, a few feasible identity-based key agreement protocols (as well as signature schemes) based on pairing techniques were developed. Smart, by combining the ideas from [BF01], [MQV95] 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].

The contributions of this paper are as follows:

1. Introducing an ID-AK protocol more efficient than Smart's.
2. Modifying Smart's and our proposed AK protocol to include the TA forward secrecy property and to avoid TAs being able to access user's communications.

3. Introducing an AK protocol (also modified from Smart's) to allow users to choose different TAs.
4. Discussions on the security properties of these protocols and using formal security models and methods to prove them.

2 Technical Backgrounds

2.1 Pairing Technique Concepts

Pairing has recently had a number of positive applications in cryptography, for instances, identity-based encryption [BF01], identity-based signatures [He02, Pa02, and SOK00], key agreement [Jo00, Sm02, and 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 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 Weil pairing ref to [MOV93, Si94] and a modified version ref to [BF01], and Tate pairing ref to [FMR99, Ga01]. For the purposes of this paper, we let $\hat{e}$ denote a general bilinear map, i.e., $\hat{e}$: $G_1 \times G_1 \rightarrow G_2$, which can be either a modified Weil pairing or a Tate pairing.

A Diffie-Hellman (DH) tuple in G_1 is $(P, xP, yP, zP) \in G_1$ for some $x, y, z \in \mathbf{Z}^*_q$ satisfying $z = xy \bmod q$.

Computational Diffie-Hellman (CDH) problem: Given any three elements from the four 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 $P, xP, yP, zP \in G_1$, decide if it is a valid DH tuple. This can be solved in polynomial time by verifying $\hat{e}(xP, yP) = \hat{e}(P, zP)$.

Bilinear Diffie-Hellman (BDH) problem: Let P be a generator of G_1. The BDH problem in $\langle G_1, G_2, \hat{e}\rangle$ is that given (P, xP, yP, zP) for some x, y, z $\in \mathbf{Z}^*_q$, 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 $\langle G_1, G_2, \hat{e}\rangle$ with non-negligible probability.

Security of our authenticated key agreement protocols described in this paper is based on the CDH and BDH assumptions.

2.2 Security Model

In the later part of this paper, we will prove security of the protocols proposed in this paper using a security model, which was proposed by Bellare and Rogaway in [BR95] and used in a number of other papers such as Blake-Wilson et al in [BJM97] and Al-Riyami and Paterson in [AP02]. We now highlight the model as follows.

The model includes a set $U = \{1, \ldots, T_1(k)\}$ of protocol participants and a set $V = \{1, \ldots, T_4(k)\}$ of TAs, where k is a security parameter, $T_1(k)$ is a polynomial bound on the number of participants in k for some polynomial function T_1 and $T_4(k)$ is a polynomial bound on the number of TAs in k for some polynomial function T_4. These participants are modelled by oracles, e.g., $\Pi^n_{I,J}$, which simulates 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 that keep track of messages they have sent or received, and of queries they have answered.

Each participant has a pair of 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 key generation algorithm O which produces a description of groups G_1 and G_2 and the bilinear map $\hat{e}$, assigns random tapes and oracles as necessary, and

distributes long-term master keys to TAs and ID-based long-term public/private keys to participants.

The model also includes an adversary, *E*, who is neither a participant nor a TA. *E* is a probabilistic polynomial time Turing Machine and she has access to the participants' oracles. *E* can relay, modify, delay, interleave or delete messages. For any pair of oracles $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$, *E* is called the benign adversary on these two oracles if she simply passes messages to and fro between participants, *I* and *J*. It is assumed that *E* is allowed to make the following three types of queries of the oracles, as defined in [BJM97]:

- *Send:* this allows *E* to send a message of her choice to an oracle, say $\Pi^n_{I,J}$, or to initiate a protocol run between two participants, *I* and *J*.
- *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 it's long-term private key.

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

- *Accepted*: an oracle is accepted if it decides to accept, holding a session key, after receipt of properly formulated messages.
- *Rejected*: an oracle is rejected if it decides to reject holding a session key.
- *: an oracle is * 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 both oracles, say $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$, have received (perhaps via the adversary) messages exclusively generated by the other oracle, we say that these two oracles have had a matching conversation (see [BR93] for a formal definition).

For attacking a protocol, *E* does an experiment with a set of selected oracles. During the experiment *E* asks a polynomially bounded number of queries (including Send, Reveal and Corrupt) to the oracles and finally makes a Test query to a chosen oracle. The oracle, say $\Pi^n_{I,J}$, to be chosen for answering the Test query must have accepted, be unopened and be uncorrupted. Furthermore, there must be no opened or corrupted oracle $\Pi^t_{J,I}$ with which it has had a matching conversation. To answer the query, the oracle flips a fair coin $b \leftarrow \{0, 1\}$, and returns the session key if $b = 0$, or else a random key sampled from $\{0, 1\}^k$ if $b = 1$. Then *E* has to guess *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 it is defined as:

$$Advantage^E(k) = |Pr[\text{guess correct}] - ½|.$$

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, FK, 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, FK;*
3. *$advantage^E(k)$ is negligible.*

A function $\varepsilon(k)$ is negligible if for every $c > 0$ there exists $k_c > 0$ such that $\varepsilon(k) < k^{-c}$ 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, FK, 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, FK;*
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 the protocol is run against adversary E, there exists an oracle $\Pi^n_{I,J}$, which accepted, but there is no oracle $\Pi^t_{J,I}$ which has engaged in a matching conversation to $\Pi^n_{I,J}$. 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.

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 involves three entities: two users Alice and Bob who wish to establish a shared secret session key, and a TA from whom they each require their own private key.

To provide a private key generation service, the TA uses a 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 $s \in \mathbf{Z}^*_q$. When a user registers with TA, the TA issues a private key $S = sQ$ for the user, where $Q = H_1(\text{ID}) \in G_1$, 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 with paring, e.g., [BF01, He02, Pa02, SOK00].

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

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

Protocol 1.

M1: Alice $\rightarrow$ Bob: T_A
M2: Bob $\rightarrow$ Alice: T_B

At the conclusion of the protocol Alice computes $K_{AB} = \hat{e}(S_A, T_B)\hat{e}(aQ_B, P_s)$, and Bob computes $K_{BA} = \hat{e}(S_B, T_A)\hat{e}(bQ_A, P_s)$. If Alice and Bob follow the protocol, they will compute the same shared secret: $K = K_{AB} = K_{BA} = \hat{e}(bQ_A + aQ_B, P_s)$. Their shared secret session key is then $FK = H_2(K)$, where H_2 is a key derivation function. This function will typically be a random oracle, or secure hash function. It is important to make use of these key derivation functions since an attacker might otherwise be able to gain partial information about the session key even though the underlying problem is hard.

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.*

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. In the next section, we will introduce a more efficient protocol, which offers the same security properties as Protocol 1.

Key escrow. As mentioned above, this protocol allows the TA to escrow the session key shared between Alice and Bob. This property may not acceptable for some applications. Although one main property of ID-based systems is that the TA generates private keys for users, some users may still want to conduct their own communications without the TA

eavesdropping. This security feature holds in the identity-based key agreement protocols using Shamir's key set up, ref to [Ok86, TO91, GP90, and ISO11770]. In Section 5, we will describe a solution for Smart's protocol that avoids the TA being able to deduce the established key.

Single TA. In this protocol, both Alice and Bob register with a single TA. This is suitable for the situation in which Alice and Bob belong to the same community. In some other real life applications, Alice and Bob may personally make use of different TAs but they both trust these two TAs to provide a key generation service properly. This security feature does not hold in the existing identity-based key agreement protocols using Shamir's key set up. In Section 6, we will introduce an extended protocol to support such a requirement.

In the following three sections, we give three modifications 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, $b \in \mathbf{Z}^*_q$, 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 $FK = H_2(K)$.

Efficiency. The above protocol has a very similar construct 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. This protocol requires each party to perform two elliptic curve point multiplications, one elliptic curve point addition and one 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 BDH problem (for the pair of groups G_1 and G_2) is hard and provided that H_2 is a random oracle.*

Proof: Condition 1 follows from the assumption that the two oracles follow the protocol and E is benign. In this case, 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 2 follows from the fact that if the two oracles are uncorrupted, then they cannot be impersonated, and if they have had matching conversations then each has received properly formatted messages from the other. So they will both accept holding the same session key FK where FK has the same properties as for Condition 1.

Condition 3. Consider an arbitrary adversary E, and suppose, by the way of contradiction, that $advantage^E(k)$ is non-negligible. Suppose that there exists an oracle $\Pi^n_{I,J}$ - after having a matching conversation to another oracle $\Pi^t_{J,I}$ (both I, J have not been corrupted), $\Pi^n_{I,J}$ holds the session key with the form $H_2(\hat{e}(Q_I, Q_J)^{s(i+j)})$ for i chosen at random by $\Pi^n_{I,J}$ and j chosen at random by $\Pi^t_{J,I}$. We say that E succeeds (against $\Pi^n_{I,J}$) if at the end of E's experiment, E picks $\Pi^n_{I,J}$ to ask a *Test* query and outputs the correct bit guess. Thus,

$$Pr[E \text{ succeeds}] = \tfrac{1}{2} + \eta(k),$$

for some non-negligible $\eta(k)$ by assumption. Now call A_k the event that H_2 has been queried on $\hat{e}(Q_I, Q_J)^{s(i+j)}$ by E or some oracle other than $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$. Then

$$Pr[E \text{ succeeds}] = Pr[E \text{ succeeds}|A_k]Pr[A_k] + Pr[E \text{ succeeds}|\bar{A}_k]Pr[\bar{A}_k].$$

Since H_2 is a random oracle, and $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$ remain unopened by definition, $Pr[E \text{ succeeds}|\bar{A}_k] = ½$. Thus

$$½ + \eta(k) \leq Pr[E \text{ succeeds}|A_k]Pr[A_k] + ½,$$

so that $Pr[A_k] \geq \eta(k)$. We conclude that given E picks some $\Pi^n_{I,J}$ for which there exists some $\Pi^t_{J,I}$ that has had a matching conversation to $\Pi^n_{I,J}$, then the probability that H_2 has previously been queried on $\hat{e}(Q_I, Q_J)^{s(i+j)}$ by E or some oracle other than $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ is non-negligible.

Therefore we use E to construct an algorithm F which solves the BDH problem with non-negligible probability.

F's task: Given input of, as described in Section 2, the two groups G_1, G_2, the bilinear map $\hat{e}$, a generator of G_1, P, and a triple of elements $xP, yP, zP \in G_1$ with $x, y, z \in \mathbf{Z}^*_q$ where q is the prime order of G_1 and G_2, F's task is to compute and output the value g^{xyz} where $g = \hat{e}(P, P)$.

F's operation: F picks $I, J \in_R U$ (the probability of picking a particular pair is $1/T_1(k)^2$), n, $t \in_R \{1, \ldots, T_2(k)\}$ (the probability of picking a particular session is $1/T_2(k)^2$), and $l \in_R \{1, \ldots, T_3(k)\}$ (the probability of picking a particular value is $1/T_3(k)$), where $T_2(k)$ denotes polynomial bounds in the security parameter, k, on the number of sessions an oracle may enter into with another oracle, for some polynomial function T_2; and $T_3(k)$ denotes polynomial bounds in the security parameter, k, on the number of distinct H_2 queries made by E and its oracles for some polynomial function T_3. F guesses that E will select $\Pi^n_{I,J}$ to ask its *Test* query after $\Pi^t_{J,I}$ has had a matching conversation to $\Pi^n_{I,J}$, and also guesses that the lth distinct H_2 call made during the experiment will be on $\hat{e}(P, P)^{xyz}$.

F simulates the running of the key generation algorithm O by choosing xP as TA's public key, sP, choosing all participants' long-term public keys randomly and computing the corresponding long-term private keys, e.g., for participant M, the public key is $Q_M = r_M P$ where $r_M \in_R \mathbf{Z}^*_q$, and the private key is $S_M = r_M xP$, but with the exception of I and J's keys. As public values for I and J, F chooses yP as I's public key, Q_I, and $(i + j)^{-1}zP$ as J's public key, Q_J, where $i, j \in_R \mathbf{Z}^*_q$. F then starts E.

During the period of E's attacking experiment, F answers E's queries in the following ways:

1. *Hash* query. F answers all H_2 oracle queries at random, just like a real random oracle would.
2. *Corrupt* query. 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.
3. *Reveal* query. F answers *Reveal* queries as specified by a normal oracle, i.e., revealing the session key (if any) the oracle currently holds, except that if E asks $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ a *Reveal* query, then F gives up.
4. *Send* query. F answers all *Send* queries as specified by a normal oracle, i.e., taking a random sample to form its challenge, except that if E asks $\Pi^n_{I,J}$ *Send* query, F answers iyP and if E asks $\Pi^t_{J,I}$ *Send* query, F answers $j(i + j)^{-1}zP$.

There are the following possible results from the experiment involving F and E:

1. E does not make its queries in such a way that $\Pi^n_{I,J}$ has a matching conversation to $\Pi^t_{J,I}$, then F gives up.
2. E and its oracles do not make l distinct H_2 oracle calls before E asks its *Test* query, then F gives up.

3. E does make its queries in this way, then $\Pi^n_{I,J}$ will accept (holding the key formed as $H_2(\hat{e}(Q_I, Q_J)^{s(i+j)}) = H_2(\hat{e}(yP, (i+j)^{-1}zP)^{x(i+j)}) = H_2(\hat{e}(P, P)^{xyz})$, although F doesn't know $\hat{e}(Q_I, Q_J)^{s(i+j)}$, and so can't actually compute this key).
4. If the previous item happens and the lth distinct H_2 call is made (say on h), then F stops and outputs h as its guess at $\hat{e}(P, P)^{xyz}$.

It is easy to observe that if the lth distinct H_2 query made by E or its oracles is on $\hat{e}(P, P)^{xyz}$, then F certainly wins its experiment. Therefore, the probability that F outputs the correct value $\hat{e}(P, P)^{xyz}$ is:

$$Pr[A_k]/(T_1(k)^2T_2(k)^2T_3(k)) \geq \eta(k)/(T_1(k)^2T_2(k)^2T_3(k)),$$

which is non-negligible. This contradicts the BDH assumption. We conclude that $\eta(k)$ is negligible, and thus that $advantage^E(k)$ must be negligible. □

Since the security model which we used to prove Theorem 1 doesn't cover some known active attacks, we now heuristically discuss some of the security properties related to these attacks.

1. *Known key security.* In the proof of Theorem 1 above, E is allowed to make *Reveal* queries to any oracle except for $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$. That does not help her to obtain the key shared between $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$. This captures the notion of known key security. Actually Protocol 2 has this property because each run of the protocol produces a different session key, therefore knowledge of past session keys does not allow deduction of future session keys.
2. *Partial forward secrecy.* We consider the following three separate parts of this property:
   (1) Compromise of long-term secret keys, either S_A or S_B, at some point in the future does not lead to the compromise of communications in the past. But compromising of both of them at some point in the future leads to the compromise of communications in the past, because $K = \hat{e}(S_A, W_B)\,\hat{e}(W_A, S_B)$. So the protocol does not offer perfect forward secrecy.
   (2) Compromising of the TA's master key s leads to the compromise of communications in the past, 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) Compromising of one or both of the ephemeral private keys, a and b, reveals none of 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, i.e., compromising the TA's master key (which also means compromising the two long-term secret keys of the users) does not lead to the compromise of communications in the past.
3. *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 it 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 intrinsically requires an extra round.
4. *Unknown key-share resilience.* It is not easy to give a formal proof of whether the protocol possesses the unknown key-share resilience attribute or not. It seems to be difficult for an adversary to replace Alice's or Bob's public key with their own one because every user's public key is a hash function output with their identity string as input. To give a formal proof of this security feature is an interesting open problem.

5. *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.

   Theorem 2. *An adversary, who is polynomial time, cannot impersonate Bob to Alice in Protocol 2 with knowing Alice's private key but not Bob's private key, under the assumption of BDH problem and provided that H_2 is a random oracle.*

   We prove this theorem in Appendix A. Since the protocol is symmetric, the adversary cannot impersonate Alice to Bob either, if the adversary obtains Bob's private key but not Alice's.

5 Modification of Protocols 1 and 2 without Key Escrow

There are two properties missing from Protocols 1 and 2 that we may require. These are TA forward secrecy (and therefore also perfect forward secrecy), and the fact that we may not want the TA to be able to escrow session keys established by two users in the protocol.

Note that in identity-based cryptography (IBC) systems 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 derive (or therefore escrow) the established session key. This is mainly a privacy issue since users may 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 is another security issue. If at any stage the TA's key is compromised, this should not compromise the previously established session keys. This is known as TA perfect forward secrecy, and in Protocols 1 and 2, this does not hold.

Recall that the concept of perfect forward secrecy captures the idea that the corruption of both users' long-term private keys does not compromise their previously established session keys. Note that if the protocol has the property of TA forward secrecy, then it has perfect forward secrecy since the TA knows all users' long-term private keys. The converse is not necessarily true since the TA knows not only all users' long-term private keys, but also s, the TA's long-term master secret.

Ideally, we would like a key agreement protocol in which the long-term keys are used for authentication, but the ephemeral keys are used in a way that cannot be known 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 seem to be a simple way to escape the key escrow using the above protocol 1 and Diffie-Hellman contributions. The key derivation function H_2 in Protocol 1 could be changed to

$$H_2' : G_2 \times G_1 \rightarrow \{0, 1\}^k,$$

and the shared secret key becomes $FK = H_2'(K, abP)$. 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 access the session key.

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.

Let Alice and Bob exchange aQ_A, aP and bQ_B, bP. Then they compute K as the same as in Protocol 2. They 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 efficient on computation, in particular on pairing computations since only a single pairing is required, 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′ is a secure AK protocol, and has the TA forward secrecy property, assuming that BDH problem (for the pair of groups G_1 and G_2) is hard, the CDH problem (for group G_1) is hard, and provided that H_2' is a random oracle.*

Proof (sketch): The fact that Protocol 2′ is a secure AK protocol by Definition 1 follows directly from Theorem 1 since the properties proved for Protocol 2 follow directly into Protocol 2′.

We prove that Protocol 2′ has TA forward secrecy as follows:

The proof follows along similar lines to the proof of Theorem 1. Consider an arbitrary adversary E as before, and suppose, by the way of contradiction, that $advantage^E(k)$ is non-negligible. E is as before except that when E asks the test query of oracle $\Pi^n_{I,J}$, E knows the TA's secret key s, so both oracles are in fact corrupted. However oracle $\Pi^n_{I,J}$ must be unopened and have had a matching conversation with another unopened oracle $\Pi^t_{J,I}$. Note that since the oracles had matching conversations, their ephemeral data was relayed without modification, so they should both have accepted holding a session key of the form $H_2(\hat{e}(Q_I, Q_J)^{s(i+j)}, ijP)$ for i chosen at random by $\Pi^n_{I,J}$ and j chosen at random by $\Pi^t_{J,I}$.

Since H_2' is a random oracle and the probability that E succeeds is non-negligible, the probability that H_2' has previously been queried on $[\hat{e}(Q_I, Q_J)^{s(i+j)}, ijP]$ by E or some oracle other than $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ is non-negligible.

Therefore we use E to construct an algorithm F which solves the CDH problem with non-negligible probability.

F's task: Given input of the two elements aP, $bP \in G_1$ with $a,b \in \mathbf{Z}^*_q$, F's task is to compute and output the value abP.

F's operation is as before, picking a pair of participants $I, J \in_R U$, simulating the running of the key generation algorithm O and then starting E with the TA's secret key s.

During the period of E's attacking experiment, F answers E's queries as a real oracle except that if E asks $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ a *Reveal* query, then F gives up and if E asks $\Pi^n_{I,J}$ *Send* query, F answers aP and if E asks $\Pi^t_{J,I}$ *Send* query, F answers bP.

E makes its queries as before, and if F is not forced to give up, then F stops and outputs the second portion of the lth distinct H_2' oracle calls as its guess at abP. As in the proof of Theorem 1, we call A_k the event that H_2' has been queried on $[\hat{e}(Q_I, Q_J)^{s(i+j)}, ijP]$ by E or some oracle other than $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$, and then the probability that F outputs the correct value is

$$Pr[A_k]/(T_1(k)^2T_2(k)^2T_3(k)) \geq \eta(k)/(T_1(k)^2T_2(k)^2T_3(k)),$$

which is non-negligible, contradicting the CDH assumption. □

As was mentioned before, the TA forward secrecy property implies perfect forward secrecy, so our Protocol 2′ also has perfect forward secrecy. This security property holds in Protocol 1′ as well. We also note that the TA cannot deduce the session keys established by oracles without actively attacking the protocol. So the TA cannot escrow the session keys established by users in the system.

6 A Pairing Based AK Protocol with Separate TAs

Suppose that there are two trusted authorities, say TA_1 and TA_2; they each have a public/private key pair: (P, $s_1P \in G_1$, $s_1 \in \mathbf{Z}^*_q$) and (P, $s_2P \in G_1$, $s_2 \in \mathbf{Z}^*_q$), where P and G_1 are globally agreed, e.g., recommended by an international standard body.

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

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

Protocol 3.

M1: Alice → Bob: T_A
M2: Bob → Alice: T_B

At the conclusion of the protocol Alice computes $K_{AB} = \hat{e}(S_A, T_B)\hat{e}(Q_B, as_2P)$, and Bob computes $K_{BA} = \hat{e}(S_B, T_A)\hat{e}(Q_A, bs_1P)$. 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 TA forward secrecy and allows key escrow (if two TAs collude), or $FK = H_2'(K, abP)$, which has TA forward secrecy and does not allow key escrow.

Efficiency. For the key escrow version, each party is required to compute two elliptic curve point multiplications and two evaluations of the pairing. For no key escrow, each party has to compute one extra elliptic curve point multiplication.

Security. This protocol has the same security properties as Protocol 1, except for the second part of TA forward secrecy. For the key escrow 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 $K = \hat{e}(Q_B, T_A)^{s_2}\hat{e}(Q_A, T_B)^{s_1}$. This implies that the two TAs must work together (or collude) in order to determine any secret session keys. For the no key escrow version, 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 BDH problem (for the pair of groups G_1 and G_2) is hard and provided that H_2 is a random oracle.*

We prove this theorem in Appendix B.

7 Key Confirmation Process

This section describes the AKC variant of Protocol 2 (the same can be done for Protocol 3). Based on [SK00], an AKC protocol can be derived 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. 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 $FK' = H_3(K)$. These two functions could be different since they are often with different inputs from different groups and may be required to produce outputs of different forms.

Protocol 4.

M1: Alice $\rightarrow$ Bob: W_A
M2: Bob $\rightarrow$ Alice: W_B, $\text{MAC}_{FK'}(2, \text{ID}_A, \text{ID}_B, W_A, W_B)$
M3: Alice $\rightarrow$ Bob: $\text{MAC}_{FK'}(3, \text{ID}_A, \text{ID}_B, W_A, W_B)$

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 [LMQSV]. 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 BDH problem (for the pair of groups G_1 and G_2) is hard, the MAC is secure and provided that H_2 and H_3 are random oracles.*

We prove this theorem in Appendix C.

8 Conclusions

We have investigated some security issues related to identity based authenticated key agreement, and proposed a few new protocols modified from previous protocols to efficiently achieve certain security properties. We have then used formal methods to prove that our new protocols do indeed possess these security properties.

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

Acknowledgements

We thank Sattam Al-Riyami, Alex Dent, Steven Galbraith, Keith Harrison, Antonio Lain, Wenbo Mao, Brain Monahan, Kenneth Paterson, Nigel Smart and David Soldera for useful comments and discussions.

References

[AP02] Al-Riyami S. and Paterson K. G.. Tripartite authenticated key agreement protocols from pairings. Cryptology ePrint Archive, Report 2002/035, available at http://eprint.iacr.org/2002/035/.

[BF01] Boneh, D. 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] Blake-Wilson S., Johnson D. and Menezes A. 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.

[BK01] Burton S. and J.R. Kaliski. An unknown key-share attack on the MQV key agreement protocol. *ACM transactions on Information and System Security*, **4**(3):275-288, August 2001.

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

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

[BR95] Bellare M. 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*, pages 57-66, ACM, 1995.

[CHSS02] Chen L., 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] Chen L., 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.

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

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

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

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

[He02] Hess F.. Efficient identity based signature schemes based on pairings. To appear in *Proceedings of the Ninth Annual Workshop on Selected Areas in Cryptography*.

[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).

[JN01] Joux A. and K. Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003, available at http://eprint.iacr.org/2001/003/.

[Jo00] Joux A.. 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.

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

[MQV95] Menezes A., 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, Ottawa, May 18-19), pages 22-32.

[MWW98] Mitchell C., M. Ward and P. Wilson. Key control in key agreement protocols. *Electronics Letters*, **34**:980-981, 1998.
[Ok86] Okamoto E.. Proposal for identity-based key distribution system. *Electronics Letters*, **22**:1283-1284, 1986.
[Pa02] Paterson K.G.. ID-based signatures from pairings on elliptic curves. Cryptology ePrint Archive, Report 2002/004, available at http://eprint.iacr.org/2002/004/.
[Sh84] Shamir A.. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology - CRYPTO '84*, LNCS 196, pages 47-53, Springer-Verlag, 1984.
[Si94] Silverman J.H.. Advanced topics in the arithmetic of elliptic curves. GTM 151, ISBN 0-387-94325-0, Springer-Verlag, 1994.
[SK00] Song B. and K. Kim. Two-pass authenticated key agreement protocol with key confirmation. In *Progress in Cryptology - INDOCRYPT 2000*, LNCS 1977, pages 237-249, Springer-Verlag, 2000.
[Sm02] Smart N.P.. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, **38**:630-632, 2002.
[SOK00] Sakai R., K. Ohgishi and M. Kasahara. Cryptosystems based on pairing. In *the 2000 Symposium on Cryptography and Information Security (SCIS2000)*, Okinawa, Japan, Jan. 26–28, 2000.
[TO91] Tanaka K. and E. Okamoto. Key distribution system for mail systems using ID-related information directory. *Computers & Security*, **10**:25-33, 1991.

Appendix A: Proof of Theorem 2

Theorem 2. *An adversary, who is a polynomial time, cannot impersonate Bob to Alice in Protocol 2 with knowing Alice's private key but not Bob's private key, under the assumption of BDH problem and provided that H_2 is a random oracle.*

Proof: First of all, to prove this theorem, we allow an adversary E to make a *Test* query to an oracle, which has been corrupted.

Consider an arbitrary adversary E, and suppose, by the way of contradiction, that the probability of E successfully impersonating J to I is non-negligible, given that E has compromised I's long term private key, sQ_I. Suppose that there exists an oracle $\Pi^n_{I,J}$ (which has been corrupted) - after having a matching conversation to E, which impersonates another oracle $\Pi^t_{J,I}$ (which has not been corrupted) , $\Pi^n_{I,J}$ holds the session key with the form $H_2(\hat{e}(Q_I, Q_J)^{s(i+j)})$ for i chosen at random by $\Pi^n_{I,J}$ and j chosen at random by E. We say that E succeeds (impersonate $\Pi^t_{J,I}$ to $\Pi^n_{I,J}$) if at the end of E's experiment, E picks $\Pi^n_{I,J}$ to ask a *Test* query and outputs the correct bit guess. Thus,

$$Pr[E \text{ succeeds}] = \tfrac{1}{2} + \eta(k),$$

for some non-negligible $\eta(k)$ by assumption. Now call A_k the event that H_2 has been queried on $\hat{e}(Q_I, Q_J)^{s(i+j)}$ by E or some oracle other than $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$. Then

$$Pr[E \text{ succeeds}] = Pr[E \text{ succeeds}|A_k]Pr[A_k] + Pr[E \text{ succeeds}|\bar{A}_k]Pr[\bar{A}_k].$$

Since H_2 is a random oracle, and $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$ remain unopened, $Pr[E \text{ succeeds}|\bar{A}_k] = \tfrac{1}{2}$. Thus

$$\tfrac{1}{2} + \eta(k) \leq Pr[E \text{ succeeds}|A_k]Pr[A_k] + \tfrac{1}{2},$$

so that $Pr[A_k] \geq \eta(k)$. We conclude that given E picks some $\Pi^n_{I,J}$ for which there exists some $\Pi^t_{j,i}$ that has been impersonated by E to have had a matching conversation to $\Pi^n_{I,J}$, then the probability that H_2 has previously been queried on $\hat{e}(Q_I, Q_J)^{s(i+j)}$ by E or some oracle other than $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ is non-negligible.

Therefore we use E to construct an algorithm F which solves the BDH problem with non-negligible probability.

F's task: The same as in the proof of Theorem 1, F's task is given a triple of elements xP, $yP, zP \in G_1$ with $x, y, z \in \mathbf{Z}^*_q$ compute the value $g^{xyz} \in G_2$ where $g = \hat{e}(P, P)$.

F's operation: F picks $I, J \in_R U$, $n, t \in_R \{1, \ldots, T_2(k)\}$, and $l \in_R \{1, \ldots, T_3(k)\}$, guessing that E will select $\Pi^n_{I,J}$ to ask its *Test* query after E has impersonated $\Pi^t_{J,I}$ and has had a matching conversation to $\Pi^n_{I,J}$, and also guessing that the lth distinct H_2 call made during the experiment will include $\hat{e}(P, P)^{xyz}$.

F simulates the running of the key generation algorithm O by choosing xP as TA's public key, sP, choosing all participants' long-term public keys randomly and computing the corresponding long-term private keys, e.g., for participant M, the public key is $Q_M = r_M P$ where $r_M \in_R \mathbf{Z}^*_q$, and the private key is $S_M = r_M xP$, but with the exception of J's keys. As public value for J, F chooses yP as J's public key, Q_J,. F then starts E.

During the period of E's attacking experiment, F answers E's queries in the following ways:

1. *Hash* query. F answers all H_2 oracle queries at random, just like a real random oracle would.
2. *Corrupt* query. 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 J a *Corrupt* query, F gives up. Note that when E asks for I a *Corrupt* query, F reveals $r_I xP$.
3. *Reveal* query. F answers *Reveal* queries as specified by a normal oracle, i.e., revealing the session key (if any) the oracle currently holds, except that if E asks $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ a *Reveal* query, then F gives up.
4. *Send* query. F answers all *Send* queries as specified by a normal oracle, i.e., taking a random sample to form its challenge, except that if E asks $\Pi^n_{I,J}$ *Send* query, F ask her for an input, then E provides jyP, where $j \in_R \mathbf{Z}^*_q$, and if E asks $\Pi^t_{J,I}$ *Send* query, F answers $(1/r_I)zP$.

There are the following possible results from the experiment involving F and E:

1. E does not make its queries in such a way that E has impersonated $\Pi^n_{I,J}$ to have a matching conversation to $\Pi^t_{J,I}$, then F gives up.
2. E and its oracles do not make l distinct H_2 oracle calls before E asks its *Test* query, then F gives up.
3. E does make its queries in this way, then $\Pi^n_{I,J}$ will accept (holding the key formed as $H_2(\hat{e}(Q_I, Q_J)^{s(i+j)}) = H_2(\hat{e}(sQ_I, jQ_J)\hat{e}(sQ_I, iQ_J)) = H_2(\hat{e}(r_I xP, jyP)\hat{e}(r_I xP, (1/r_I)zyP)) = H_2(\hat{e}(r_I xP, jyP)\hat{e}(P, P)^{xyz})$, although F doesn't know $\hat{e}(Q_I, Q_J)^{s(i+j)}$, and so can't actually compute this key). By the way F knows $\hat{e}(r_I xP, jyP)$.
4. If the previous item happens and the lth distinct H_2 call is made (say on h), then F stops and outputs $h/\hat{e}(r_I xP, jyP)$ as its guess at $\hat{e}(P, P)^{xyz}$.

It is easy to observe that if the lth distinct H_2 query made by E or its oracles is on $\hat{e}(r_I xP, jyP)\hat{e}(P, P)^{xyz}$, then F certainly wins its experiment. Therefore, the probability that F outputs the correct value $\hat{e}(P, P)^{xyz}$ is:

$$Pr[A_k]/(T_1(k)^2 T_2(k)^2 T_3(k)) \geq \eta(k)/(T_1(k)^2 T_2(k)^2 T_3(k)),$$

which is non-negligible. This contradicts the BDH assumption. We conclude that $\eta(k)$ is negligible, and thus that the probability of E successfully impersonating $\Pi^t_{J,I}$ to $\Pi^n_{I,J}$ must be negligible.

Appendix B: Proof of Theorem 4

Theorem 4. *Protocol 3 is a secure AK protocol, assuming that BDH problem (for the pair of groups G_1 and G_2) is hard and provided that H_2 is a random oracle.*

Proof: As discussed in the proof of Theorem 1 described in Section 4, conditions 1 and 2 follow immediately from the assumption that H_2 is a random oracle.

Condition 3. Consider an arbitrary adversary E, and suppose, by the way of contradiction, that $advantage^E(k)$ is non-negligible. Suppose that there exists an oracle $\Pi^n_{I,J}$ - after having a matching conversation to another oracle $\Pi^t_{J,I}$ (both I, J have not been corrupted), $\Pi^n_{I,J}$ holds the session key with the form $H(\hat{e}(jS_I + iS_J, P))$ for i chosen at random by $\Pi^n_{I,J}$ and j chosen at random by $\Pi^t_{J,I}$. We say that E succeeds (against $\Pi^n_{I,J}$) if at the end of E's experiment, E picks $\Pi^n_{I,J}$ to ask a *Test* query and outputs the correct bit guess.

From the same argument made in the proof of Theorem 1, we have that given E picks some $\Pi^n_{I,J}$ for which there exists some $\Pi^t_{J,I}$ that has had a matching conversation to $\Pi^n_{I,J}$, then the probability that H_2 has previously been queried on $\hat{e}(jS_I + iS_J, P)$ by E or some oracle other than $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ is non-negligible.

Therefore we use E to construct an algorithm F which solves the BDH problem with non-negligible probability.

F's task: The same as in the proof of Theorem 1, F's task is given a triple of elements xP, yP, $zP \in G_1$ with $x, y, z \in \mathbf{Z}^*_q$ compute the value $g^{xyz} \in G_2$ where $g = \hat{e}(P, P)$.

F's operation: F picks $I, J \in_R U$, $n, t \in_R \{1, \ldots, T_2(k)\}$, and $l \in_R \{1, \ldots, T_3(k)\}$, guessing that E will select $\Pi^n_{I,J}$ to ask its *Test* query after $\Pi^t_{J,I}$ has had a matching conversation to $\Pi^n_{I,J}$, and also guessing that the lth distinct H_2 call made during the experiment will be on $\hat{e}(P, P)^{xyz}$.

F simulates the running of the key generation algorithm O by choosing xP as both TA_1 and TA_2's public key, i.e., $s_1P = s_2P = xP$, choosing all participants' long-term public keys randomly and computing the corresponding long-term private keys, e.g., for participant M, the public key is $Q_M = r_MP$ and the private key is $S_M = r_MxP$, but with the exception of I and J's keys. As public values for I and J, F chooses yP as I's public key, Q_I, and zP as J's public key, Q_J. F then starts E.

During the period of E's attacking experiment, F answers E's *Hash* queries, *Corrupt* queries and *Reveal* queries in the same way as it does in the proof of Theorem 1. Only for Send query, F answers all *Send* queries as specified by a normal oracle, i.e., taking a random sample to form its challenge, except that if E asks $\Pi^n_{I,J}$ *Send* query, F answers $(1/2)yP$ and if E asks $\Pi^t_{J,I}$ *Send* query, F answers $(1/2)zP$.

Again, the same as in the proof of Theorem 1, if E does not make its queries in such a way that $\Pi^n_{I,J}$ has a matching conversation to $\Pi^t_{J,I}$, and if E and its oracles do not make l distinct H_2 oracle calls before E asks its *Test* query, F gives up. Otherwise, $\Pi^n_{I,J}$ will accept (holding the key formed as $H_2(\hat{e}(jQ_I + iQ_J, sP)) = H_2(\hat{e}((1/2)yzP + (1/2)yzP, xP)) = H_2(\hat{e}(P, P)^{xyz})$, and F will outputs the lth distinct H_2 call as its guess at $\hat{e}(P, P)^{xyz}$.

Therefore, if the probability that H_2 has previously been queried on $\hat{e}(jS_I + iS_J, P)$ by E or some oracle other than $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ is non-negligible, the probability F output the correct value $\hat{e}(P, P)^{xyz}$ is also non-negligible. This contradicts the BDH assumption. We conclude that $advantage^E(k)$ must be negligible.

It is obvious that so far we have also proved the security of Protocol 1, because the only difference between Protocol 1 and Protocol 3 is in Protocol 1 two users make use of the same TA and in Protocol 3 two users make use of different TAs. In the above description of proof of Theorem 3, we let F to choose $s_1P = s_2P = xP$, therefore to prove the security of Protocol 1, we can simply allow F to choose $sP = xP$.

Appendix C: Proof of Theorem 5

Definition 3 [BJM97]: A MAC is a secure MAC if for every polynomially bounded adversary E of the MAC, the function $\varepsilon(k)$ defined by

$$\varepsilon(k) = Pr[k' \leftarrow \{0,1\}^k; (m,a) \leftarrow E : (m,a) = MAC_{k'}(m)]$$

is negligible.

This means that the probability of a polynomially bounded adversary E being able to forge a valid MAC on any message m is negligible.

Theorem 5. *Protocol 4 is a secure AKC protocol, assuming that BDH problem (for the pair of groups G_1 and G_2) is hard, the MAC is secure and provided that H_2 and H_3 are random oracles.*

Proof (sketch): Again the proof is by contradiction. The proof follows both the proofs of Theorem 1 described in Section 4 this paper and the proof of Theorem 8 in [BJM97], so we reference these and leave most of the details to the reader.

Conditions 1 and 2 follow immediately from the assumption that H_2 is a random oracle. Condition 4 follows from Theorem 1 and the fact that H_2 is a random oracle. It remains to prove that Condition 3 holds.

Condition 3. Consider an arbitrary adversary E, and suppose, by the way of contradiction, that $Pr[No\text{-}Matching^E(k)]$ is non-negligible. We say that E succeeds (*against* oracle $\Pi^n_{I,J}$) in her experiment if there exists an oracle $\Pi^n_{I,J}$ which has accepted but no oracle $\Pi^t_{J,I}$ has had a matching conversation to $\Pi^n_{I,J}$. Thus,

$$Pr[E \text{ succeeds}] = \eta(k),$$

for some non-negligible $\eta(k)$ by assumption. Now call A_k the event that H_3 has been queried on $\hat{e}(Q_I, Q_J)^{s(i+j)}$ by E or some oracle other than $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$.

Case 1: Suppose that $Pr[A_k] = \eta_1(k)$ is non-negligible. In this case we can construct from E an algorithm F which can solve the BDH problem with non-negligible probability. This is done in the same manner as is described in the proof of Theorem 1 described in Section 4, so the details are left to the reader.

Case 2: We let $\eta_2(k)$ be the probability that E succeeds against at least one initiator oracle, and $\eta_3(k)$ be the probability that E succeeds against at least one responder oracle but no initiator oracles. So we have

$$\eta(k) = \eta_2(k) + \eta_3(k).$$

Now either $\eta_2(k)$ or $\eta_3(k)$ are non-negligible, and in either case, we can construct from adversary E an adversary F of the MAC. These two sub-cases follow the same form as in [BJM97] with some minor adjustments. We consider the case where $\eta_2(k)$ is non-negligible, and the case where $\eta_3(k)$ is non-negligible is similar and we leave the details to the reader.

So suppose that $\eta_2(k)$ is non-negligible. We now construct from E an adversary F of the MAC.

F's task: F has access to a MACing oracle that computes MACs under a key FK'' which was chosen at random from $\{0,1\}^k$. F's task is to compute a valid MAC (m,a) where m was not queried of its oracle.

F's operation: F picks $I, J \in_R U$, $n \in_R \{1, \ldots, T_2(k)\}$. F guesses that E will succeed against initiator oracle $\Pi^n_{I,J}$.

F simulates the running of the key generation algorithm O and chooses all participants' long-term public keys randomly and computing the corresponding long-term private keys and forms the directory *public-info*. F then starts E.

During the period of E's attacking experiment, F answers E's queries in the following ways:

1. *Hash* query. F answers all H_2 and H_3 oracle queries at random, just like a real random oracle would except for the case when H_3 is queried on $K = \hat{e}(jQ_I + iQ_J, sP)$ (which F can in fact compute). If H_3 is queried on K by E or an oracle other than $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ then F gives up. Such queries by oracles $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ are specified below.
2. *Corrupt* query. 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.

3. *Reveal* query. F answers *Reveal* queries as specified by a normal oracle, i.e., revealing the session key (if any) the oracle currently holds, except that if E asks $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$ a *Reveal* query, then F gives up.
4. *Send* query. F answers all *Send* queries as specified by a normal oracle, except that instead of calculating $FK' = H_3(K)$ and using this key to MAC messages, F calls its own MACing oracle (on FK'') to compute its response. F also calls its MACing oracle to decide whether or not such oracles should accept.

There are the following possible results from the experiment involving F and E:

1. If E does not invoke $\Pi^n_{I,J}$ as an initiator oracle then F gives up.
2. If E does invoke $\Pi^n_{I,J}$ as an initiator oracle at time t_0, F makes iQ_I as $\Pi^n_{I,J}$'s response. If $\Pi^n_{I,J}$ does not at some later time receive a message of the form (m,a) where $m = (2, \mathrm{ID}_I, \mathrm{ID}_J, iQ_I, jQ_J)$ for some jQ_J then F gives up. Note that for $\Pi^n_{I,J}$ to accept (and therefore for E to succeed), it must receive such a message.
3. If E does invoke $\Pi^n_{I,J}$ as an initiator oracle at time t_0, and $\Pi^n_{I,J}$ receives at some later time a message of the correct form (m,a) and F has previously queried its MACing oracle on m then F gives up. Otherwise F stops and outputs (m,a) as its guess at a valid forgery.

Analysis: If E succeeds against initiator oracle $\Pi^n_{J,I}$, then F outputs a valid MAC forgery and wins the experiment, provided E or some other oracle (except $\Pi^n_{I,J}$ or $\Pi^t_{J,I}$) has not called H_3 on K, and provided F has not previously calculated the message flow that makes $\Pi^n_{I,J}$ accept.

By Case 1, the probability that H_3 has been queried on K is negligible.

The probability that F has called its MACing oracle to produce the appropriate flow is also negligible. This is because F could only have called the MAC on this message on behalf of a responder $\Pi^t_{J,I}$ which received iQ_I as its own first flow, or on behalf of an initiator $\Pi^u_{I,J}$ with $u \neq n$ which also chose iQ_I and needs to know whether or not to accept.

The probability that responder $\Pi^t_{J,I}$ made the call before t_0 is negligible since i is chosen at random, and if the call was made after t_0 then $\Pi^n_{I,J}$ and $\Pi^t_{J,I}$ had a matching conversation.

The probability that the call was made by $\Pi^u_{I,J}$ is also negligible since in this event, $\Pi^n_{I,J}$ and $\Pi^u_{I,J}$ independently chose the same i.

Therefore, the probability F outputs a valid MAC (m,a) is:

$$\eta_2(k)/(T_1(k)^2 T_2(k)) - \lambda(\mathrm{k}),$$

for some negligible $\lambda(\mathrm{k})$, which is non-negligible. This contradicts the assumption of a secure MAC. Thus $\eta_2(k)$ is negligible.

Similarly, it can be shown that $\eta_3(k)$ is also negligible, so we conclude that $\eta(k)$ is negligible, and thus that $Pr[\textit{No-Matching}^E(k)]$ must be negligible.