Int. J. Inf. Secur. (2011) 10:201–212
DOI 10.1007/s10207-011-0136-3

REGULAR CONTRIBUTION

Integrating identity-based and certificate-based authenticated key exchange protocols

Berkant Ustaoğlu

Published online: 28 May 2011

Abstract Key establishment is becoming a widely deployed cryptographic primitive. As such, there has been extensive research on designing algorithms that produce shared secret keys. These protocols require parties to either hold certificates or rely on identity (ID)-based primitives to achieve authentication. Chain and cross certifications allow users trusting different certification authorities to interact. Similarly, there are methods to extend ID-based solutions across multiple key generation centers (KGC). However, there has been no dedicated work on interoperability between the two settings. A straightforward solution would require each user to maintain certificates and ID-based static keys to accommodate all peers. The cost of maintaining many secret keys; matching keys with protocols; and preventing undesired interference would arguably make such a solution impractical. In this work, we offer an alternative where a user needs to keep a single static key pair and can subsequently engage in a session key establishment with peers holding certificates or identity-based keys. Thus, the proposed solution has none of disadvantages of maintaining multiple static private keys.

Keywords Authenticated key establishment · Certificate-based protocols · ID-based protocols · Shared static state · ID-PKI integration

1 Motivation

Authenticated key exchange (AKE), along with public key encryption and digital signatures, is a basic cryptographic primitive used to establish authenticated and confidential communication channel between two users. It allows Alice and Bob to agree on a shared secret over a public channel. Diffie and Hellman [10] proposed the first key agreement protocol, which became known as the ephemeral Diffie–Hellman protocol. There are many certificate-based modifications of the original protocol that provide additional properties, e.g., [1,16–18,24]. For authentication, these protocols need certificates: Each party is required to obtain a certificate binding a public key to its identity and at the same time verify validity of peer certificates. Certificate management is a non-trivial procedure that requires extra care as flawed management can result in severe weaknesses.

Shamir [22] proposed the alternative idea of ID-based primitives, where the public key is the identity. The corresponding private key is generated by a designated key generation center. Thus, ID-based primitives dispense with certificate management and the need to assert binding between users and public keys.

Early ID-based AKE (ID-AKE) protocols were proposed by Okamoto and Tanaka [20] and Günther [13]. In the former protocol, the key generation relies on an RSA [21]-based technique to issue static private keys; the latter uses ElGamal [11] type algorithms to create static private keys. In both cases, the actual key establishment relies on the Diffie–Hellman [10] type shared secret computation. As many early key establishment schemes, these protocols do not meet modern security goals. Furthermore, before engaging in a protocol run, a party must possess a static private key, without which the party cannot prepare outgoing messages. Consequently, the application of these protocols and their variants is limited.

More recently, Smart [23] used pairings to devise an identity-based key agreement protocol; the idea was further developed by Chen and Kudla [8]. Identity-based protocols were also proposed in [3,9,19,25–27] but are not all secure;

B. Ustaoğlu (✉)
NTT Information Sharing Platform Laboratories,
Tokyo 180-8585, Japan
e-mail: bustaoglu@cryptolounge.net

further discussion is available in [2,7,9]. Security arguments for these protocols are in models that are weaker in comparison with latest certificate-based models. Huang and Cao [14] showed that there is no inherent difficulty to devise ID-AKE protocols that are resilient to test session ephemeral leakage, something not considered in previous ID-AKE protocols. It follows that ID-based and certificate-based AKE protocols can provide similar level of security assurances.

ID-based primitives are not as widely deployed as certificate-based primitives. On the one hand, there is not much incentive for a company or a single party to use the services of an external KGC to generate static private keys as it implies external entities possess the static private keys. It is therefore unlikely that there will be a few widely accepted KGC centers to distribute private keys. On the other hand, within a single company, it is reasonable to deploy ID-based solutions to achieve confidentiality and dispense of certificate management and possibly provide assistance in identification of external entities. There are also methods to extend ID-based solutions across multiple KGC domains [8], thus allowing members of different KGC domains to communicate with each other. Interoperability between certificate- and ID-based primitives is less studied.

Boyd et al. [3] proposed an authenticated key agreement protocol relying on key encapsulation mechanisms (KEMs). The proposal is generic in the sense that the KEMs can be identity based. Therefore, parties could establish a shared secret even if one uses certificate-based and the other ID-based primitive. However, to provide further security properties, the protocol needs to be augmented. For example, forward secrecy is guaranteed only if the parties essentially run an ephemeral Diffie–Hellman protocol along with the basic protocol; it is unclear whether ephemeral leakage resilience can be achieved.

We propose a collection of protocols that allow two parties to establish a session key under the following conditions: (i) each party has a single static key pair dedicated to key establishment; (ii) the key pairs can be both ID based or both certificate based or combination of the two; (iii) at the onset, the protocol initiator need not know what type of key the other party has; and (iv) the protocol initiator also need not know the exact identity of the other party, only a destination to send messages. These conditions imply that a static key is used for more than one protocol. Sharing state between different protocols is in general not a sound cryptographic practice as discussed by Kelsey, Schneier, and Wagner [15]. Even if static keys are shared only among key agreement protocols, security is not necessarily guaranteed as exposed in [6]. Moreover, it is hard, if not impossible, to prevent a party from using the same public key in different protocols, especially if these protocols provide the same functionality. For this reason, our design specifically allows sharing static keys. The strength of this relaxation is perhaps best described with the following scenario: Bob has a certificate (an ID-based primitive works in the same way) and wants to establish a session key with an entity that belongs to a company. It suffices for Bob to be able to send the message to the company (essentially a destination address) and await the response. When responding the company representative, Alice can supply a public key along with a certificate that binds the identity "Alice" to a public key or inform that "Alice" is part of an ID-based primitive; and if desired can delegate verification of Bob's credential to a dedicated entity within the company while keeping the session key secret. At the same time, the request send by Bob and the subsequent session computations do not depend in any way on what static public keys Alice supplies in the response. This is in contrast with Boyd et al. [3], where the initiator needs to know a priori the peer's static public key.

2 Notation and outline

We introduce the notation in use, but do not present the formal definitions that are widely available.

- $\in_R$ means selected uniformly at random;
- $\mathbb{G} = \langle P \rangle$ is an additive group of order q; the function "log" denotes the discrete logarithm base P;
- $\hat{\mathsf{e}} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$ is a non-degenerate pairing;
- $\mathbb{G}_t = \langle g \rangle = \langle \hat{\mathsf{e}}(P, P) \rangle$; $g = \hat{\mathsf{e}}(P, P)$ is the multiplicative group that is the range of $\hat{\mathsf{e}}$;
- the gBDH assumption holds if no polynomially bounded algorithm can produce $g^{\log U \log V \log W}$, given (U, V, W) and an oracle that distinguishes tuples $(x_1P, x_2P, x_3P, g^{x_1x_2x_2})$ from (x_1P, x_2P, x_3P, g^y), where $y \in_R \mathrm{Z}_q^*$;
- $\mathrm{H}_{id} : \{0, 1\}^* \to \mathbb{G}$ is a random oracle mapping binary strings to $\mathbb{G}$;
- $\mathrm{F}_* : \mathbb{G} \to [1, q]$ is a random oracle[1] mapping $\mathbb{G}$ elements to integers in the interval $[1, q]$;
- $\mathrm{H} : \{0, 1\}^* \to \{0, 1\}^\gamma$ is a random oracle mapping binary strings to binary strings of length γ, where γ is a security parameter;
- Key generation center (KGC) has static public key $Z = zP \in \mathbb{G}$ and private key $z \in \mathrm{Z}_q^*$;
- a certification authority (CA) is responsible for creating certificates;
- users are denoted as $\mathcal{U}_A, \mathcal{U}_B \ldots$; user $\mathcal{U}_A$ has an associated identifier $\hat{A}$;
- certificate $\hat{A}$ belongs to $\mathcal{U}_A$ and binds identifier $\hat{A}$ to a public key A; the static private key corresponding to A is $a = \log(A)$;

[1] For F_* the random oracle can be substituted with an efficiently computable function. However, in that case, protocol analysis requires more technical details, and hence, we opt to use a random oracle.

- for identity string $\hat{\mathcal{A}}$, the corresponding public key is $\mathtt{ID}_A = \mathtt{H}_{id}(\hat{\mathcal{A}})$; private key is $\mathtt{id}_A = z\mathtt{ID}_A$;
- $\mathcal{U}_A$'s ephemeral public–private key pair is $(X_A, x_A) \in (\mathbb{G}, \mathtt{Z}_q^*)$;
- $\tilde{\mathcal{U}_A}$ a possibly anonymous destination address that can be used to reach $\mathcal{U}_A$;
- CP is a set of parties with certificate-based public keys, each $\mathcal{U}_A$ from the set CP has a certificate $\hat{A}$ that binds $\mathcal{U}_A$'s identity to a static public key A;
- IP is a set of parties with identity-based public keys, each $\mathcal{U}_A$ from the set IP has identity $\hat{\mathcal{A}}$ and associated static public key $\mathtt{ID}_A = \mathtt{H}_{id}(\hat{\mathcal{A}})$;
- $\mathcal{I}$ and $\mathcal{R}$ fixed strings denoting the initiator and responder roles, respectively, in a session.

Outline Next section motivates and describes the security model that incorporates protocols between users with ID-based private keys and users with certificates. We then proceed to describe a set of AKE protocols between different type of users. Afterward, we provide the security argument and conclude our discussion.

3 Model

3.1 Model motivations

A typical AKE protocol usually identifies a party with a probabilistic Turing machine that is assigned a certificate or an ID-based key pair depending on the protocol. For example, the CK01 model [4] assigns a certificate to a Turing machine (called party) and initiates protocols with two identities. The CK02 model [5] improves over CK01 by allowing a party to initiate a protocol without knowledge of its peer identity. Recently, Chatterjee, Menezes, and Ustaoğlu [6] showed that Bob's static information use influences security of session keys at Alice should Alice and Bob engage in sessions. In particular, if Bob uses the same static key for more than one key agreement protocol, then session keys at Alice could become insecure. Thus, it is important to consider the implications of sharing static information where peers have either certificates or use ID-based static keys and engage in different protocols: either purely certificate- or ID-based or mixture of these.

In our model, we identify a party $\mathcal{U}_A$ with a probabilistic Turing machine. The set of all parties is divided into two: CP and IP; for description of parties in CP and IP see the previous section. We focus on Diffie–Hellman type protocols that exchange random ephemeral keys in $\mathbb{G}$ with the first messages.

A certificate authority (CA) issues a certificate $\hat{A}$ that binds the public key A to an identifier $\hat{\mathcal{A}}$. We require that the identifier is part of the certificate. Note that in practice, nothing prevents A from being used in multiple certificates. A user could potentially obtain multiple certificates with different identifiers but with the same public key. Such practice is not cryptographically sound, see for example [15], but hard to prevent. Moreover, parties that want to maintain a small number of static private key to reduce the likelihood of static key leakage may adopt such approach without any malicious intent. For a key A, it is the user's responsibility to create the corresponding static private key (say by first selecting the static private key and then computing the corresponding static public key). For simplicity in our model, a party in CP has one certificate.

In practice, there are multiple CA authorities. We abstract by assuming a single CA that performs static public key validation and issues certificates binding any valid public key with any identifier provided by the party. Thus, we leave out issues of chain certification and allow an adversary to obtain certificates without knowledge of the corresponding private key. This is a widely used approach in key establishment.

The "equivalent" of cross-certification in the ID-based setting is domain extension. The feasibility of this idea was demonstrated in [8], and therefore, we assume a single KGC center where ID-based protocols utilize domain extension techniques wherever necessary. We assume that the adversary can obtain a static private key for an identity of its choice but is limited to identities that are not used by honest users. For example, the adversary may collaborate with a malicious KGC to request the key for the string "Alice". However, in the actual protocol, any honest user could and should use "Alice" concatenated with a string that identifies the KGC that issued the static key. Thus, our adversary can obtain private keys only for identifying strings that are not bound to honest parties.

In our proposal, we explicitly consider four protocol types: between parties that use ID-based algorithms, parties that use certificates, and the mixture of the two.[2] As said, [6] demonstrated that using certificates and corresponding keys in distinct AKE protocols may lead to security failures. There is no a priori reason why in the ID-based or integrated scenario such vulnerabilities would not be present. Therefore, it is important to consider simultaneously all protocols that access the same static key pair in case reduced number of those keys is to be maintained.

Note that while the adversary cannot obtain the private key corresponding to $\mathtt{H}_{id}$("Alice") if the string belongs to a non-malicious entity, it is feasible for the adversary to obtain a *certificate* that binds "Alice" to any group element and in particular to $\mathtt{H}_{id}$("Alice"). This is a subtle point that we take into account in our model. While the security of our proposed

[2] Accounting for whether the session initiator has a certificate or not.

protocols is not affected, it is plausible to believe that there are protocols that can be affected.

3.2 Model description

Setup In the previous section, we introduced the parties, the CA, and the KGC. We next proceed to describe sessions and the adversary.

Session creation A party $\mathcal{U}_A$ can be activated via an incoming message[3] to create a session. The incoming message has one of the following forms: (i) $(\Pi_{\mathtt{i}}, \mathcal{U}_A, \mathcal{U}_B)$ or (ii) $(\Pi_{\mathtt{i}}, \mathcal{U}_A, \mathcal{U}_B, X_B)$, where $\Pi_{\mathtt{i}}$ identifies which protocol is activated. If activated with $(\Pi_{\mathtt{i}}, \mathcal{U}_A, \mathcal{U}_B)$, then $\mathcal{U}_A$ is the session *initiator*; otherwise, the role of $\mathcal{U}_A$ is *responder*. Each of $\mathcal{U}_A$ and $\mathcal{U}_B$ is a certificate or an identifier or an address that carries sufficient information to direct the message to the right destination.

Session initiator If $\mathcal{U}_A$ is the session initiator, then $\mathcal{U}_A$ creates a separate session state where session-specific short-lived data are stored and prepares a reply that includes an ephemeral public key X_A. The session is labeled *active* and identified via a (temporary and incomplete) session identifier $\mathtt{s} = (\Pi_{\mathtt{i}}, \mathcal{U}_A, \mathcal{U}_B, \mathcal{I}, X_A)$. The outgoing message is $(\Pi_{\mathtt{i}}, \mathcal{U}_B, \mathcal{U}_A, X_A)$.

Session responder If $\mathcal{U}_A$ is the session responder, then $\mathcal{U}_A$ creates a separate session state and prepares a reply that includes an ephemeral public key X_A. The session identifier is $\mathtt{s} = (\Pi_{\mathtt{i}}, \mathcal{U}_A, \mathcal{U}_B, \mathcal{R}, X_B, X_A)$ and the outgoing message is $(\Pi_{\mathtt{i}}, \mathcal{U}_B, \mathcal{U}_A, X_B, X_A)$.

Session update A party $\mathcal{U}_A$ can be activated to update a session via an incoming message of the form $(\Pi_{\mathtt{i}}, \mathcal{U}_A, \mathcal{U}_B, X_A, X_B)$. Upon receipt of this message, $\mathcal{U}_A$ checks that $\mathcal{U}_A$ owns an active session with identifier $\mathtt{s} = (\Pi_{\mathtt{i}}, \mathcal{U}_A, \mathcal{U}_B, \mathcal{I}, X_A)$; except with negligible probability, $\mathcal{U}_A$ can own at most one such session. If no such session exists, then the message is rejected, otherwise $\mathcal{U}_A$ updates $\mathtt{s}$ to $(\Pi_{\mathtt{i}}, \mathcal{U}_A, \mathcal{U}_B, \mathcal{I}, X_B, X_A)$. For two-pass protocols as proposed in the next session, the above description suffices. The model can be extended to accommodate different number of exchanged messages. When the protocol specifies that no further messages will be received, the session completes and accepts a session key. Note that along the run of the protocol, the information $\mathcal{U}_A$ in the session identifier can be updated from a destination address to either a certificate or identity. Such change can occur once.

Aborted sessions A protocol may require parties to perform some checks on incoming messages, e.g., the protocols proposed here require public key validation. If a party is activated to create a session with an incoming message that does not meet the protocol specifications, then that message is rejected and no session is created. If a party is activated to update an active session with an incoming message that does not meet the protocol specifications, then the party deletes all information specific to that session (including the session state and the session key if it has been computed) and *aborts* the session. Abortion occurs before the session identifier is updated. At any given time, a session is in exactly one of the following states: active, completed, and aborted.

Matching sessions Since ephemeral public keys are selected at random on a per-session basis, session identifiers are unique except with negligible probability. For a session $(\Pi_{\mathtt{i}}, \mathcal{U}_A, \mathcal{U}_B, *, *, *)$, we say $\mathcal{U}_A$ is the session *owner* and $\mathcal{U}_B$ the session *peer*; together $\mathcal{U}_A$ and $\mathcal{U}_B$ are referred to as the *communicating parties*. Let $\mathtt{s} = (\Pi_{\mathtt{i}}, \mathcal{U}_A, \mathcal{U}_B, \mathtt{r}_A, \mathtt{comm}_A)$ be a session owned by $\mathcal{U}_A$, where $\mathtt{r}_A \in \{\mathcal{I}, \mathcal{R}\}$. A session $\mathtt{s}^* = (\Pi_{\mathtt{j}}, \mathcal{U}_C, \mathcal{U}_D, \mathtt{r}_C, \mathtt{comm}_C)$, where $\mathtt{r}_C \in \{\mathcal{I}, \mathcal{R}\}$, is said to be *matching* to $\mathtt{s}$ if $\Pi_{\mathtt{i}} = \Pi_{\mathtt{j}}, \mathcal{U}_C = \mathcal{U}_B, \mathcal{U}_A = \mathcal{U}_D$, $\mathtt{r}_A \neq \mathtt{r}_C$, and either $\mathtt{comm}_C$ is a substring of $\mathtt{comm}_A$ or vice versa. The equality of two parties implies either same identities or certificates, or a destination address and either a certificate or an identity that can be reached at that destination address. It can be seen that the session $\mathtt{s}$, except with negligible probability, can have more than one matching session if and only if $\mathtt{comm}_A$ has exactly one component, i.e., is comprised of a single outgoing message.

Adversary The adversary $\mathcal{M}$ is a probabilistic Turing machine and controls *all* communications. Parties submit outgoing messages to $\mathcal{M}$, who makes decisions about their delivery. The adversary presents parties with incoming messages via *Send*(message), thereby controlling the activation of parties. The adversary does not have immediate access to a party's private information; however, in order to capture possible leakage of private information, $\mathcal{M}$ is allowed to make the following queries:

- *RevealStaticKey*($\mathcal{U}_A$): $\mathcal{M}$ obtains $\mathcal{U}_A$'s static private key.
- *RevealMasterKey*(): $\mathcal{M}$ obtains the master secret key used by the KGC to generate private keys. Consequently, $\mathcal{M}$ can obtain private keys for all identities used by all parties.
- *RevealEphemeralKey*($\mathtt{s}$): $\mathcal{M}$ obtains the ephemeral private key held by session $\mathtt{s}$. We will henceforth assume that $\mathcal{M}$ issues this query only to sessions that hold an ephemeral private key.

[3] We assume all messages are represented as binary strings.

- *RevealSessionKey*($\mathtt{s}$): If $\mathtt{s}$ has completed, then $\mathcal{M}$ obtains the session key held by $\mathtt{s}$. We will henceforth assume that $\mathcal{M}$ issues this query only to sessions that have completed.
- *RevealEphemeralPublicKey*($\mathcal{U}_A$): $\mathcal{M}$ obtains the ephemeral public key that $\mathcal{U}_A$ will use the next time a session is created within $\mathcal{U}_A$.
- *EstablishID*($\hat{\mathcal{M}}$): This query allows $\mathcal{M}$ to register an identifier $\hat{\mathcal{M}}$ and obtain the corresponding private key from KGC.
- *EstablishCert*($\hat{\mathcal{M}}$,M): This query allows $\mathcal{M}$ to obtain a valid CA certificate that binds identifier $\hat{\mathcal{M}}$ to the public key M.

Identities and certificates that were established by $\mathcal{M}$ using *EstablishID* and *EstablishCert* are called *corrupted* or *adversary controlled*; otherwise, they are said to be *honest*. This adversary interaction with KGC and CA permits the modeling of malicious insiders.

Remark A session with identifier $(\Pi_{\mathtt{i}}, \ldots)$ will be called $\Pi_{\mathtt{i}}$ session.

Adversary goal To capture indistinguishability, $\mathcal{M}$ is allowed to make a special query *Test*($\mathtt{s}$) to a 'fresh' session $\mathtt{s}$. In response, $\mathcal{M}$ is given with equal probability either the session key held by $\mathtt{s}$ or a random key. If $\mathcal{M}$ guesses correctly whether the key is random or not, then the adversary is said to be successful and meet its goal. Note that $\mathcal{M}$ can continue interacting with the parties after issuing the *Test* query, but must ensure that the test session remains fresh throughout $\mathcal{M}$'s experiment. The adversary can obtain a static private key corresponding to an identity either by explicitly querying for it or by obtaining the KGC master private key. In the following definition, we assume that *RevealMasterKey* implies that *RevealStaticKey* has been issued against all identities but not certificates.

Definition 1 **(Π-fresh)** Let $\mathtt{s}$ be the identifier of a completed Π-session, owned by an honest party $\mathcal{U}_A$ with peer $\mathcal{U}_B$, who is also honest. Let $\mathtt{s}^*$ be the identifier of the matching session of $\mathtt{s}$, if the matching session exists. Define $\mathtt{s}$ to be Π-*fresh* if none of the following conditions hold:

1. $\mathcal{M}$ issued *RevealSessionKey*($\mathtt{s}$) or *RevealSessionKey*($\mathtt{s}^*$) (if $\mathtt{s}^*$ exists).
2. $\mathtt{s}^*$ exists and $\mathcal{M}$ issued one of the following:
   (a) Both *RevealStaticKey*($\mathcal{U}_A$) and *RevealEphemeralKey*($\mathtt{s}$).
   (b) Both *RevealStaticKey*($\mathcal{U}_B$) and *RevealEphemeralKey*($\mathtt{s}^*$).
3. $\mathtt{s}^*$ does not exist and $\mathcal{M}$ issued one of the following:
   (a) Both *RevealStaticKey*($\mathcal{U}_A$) and *RevealEphemeralKey*($\mathtt{s}$).
   (b) *RevealStaticKey*($\mathcal{U}_B$).

Definition 2 Let Π_1,Π_2…,Π_{d} be a collection of d distinct key agreement protocols. The protocol collection is said to be *secure* in the shared model if the following conditions hold:

1. For any $i \in [1, d]$ if two honest parties complete matching $\Pi_{\mathtt{i}}$-sessions, then, except with negligible probability, they both compute the same session key.
2. For any $i \in [1, d]$, no polynomially bounded adversary $\mathcal{M}$ can distinguish the session key of a fresh $\Pi_{\mathtt{i}}$-session from a randomly chosen session key, with probability greater than $\frac{1}{2}$ plus a negligible fraction.

Remark In the session identifiers, $\tilde{\mathcal{U}_A}$ can be updated only once. This is to prevent potential attacks where the adversary updates the identity of dishonest party to an honest party and thus distinguish the session key of a "fresh" session from a randomly chosen session key. Similarly, sessions are aborted before updating the session identifier to prevent potential attacks where the adversary forces a "matching" session to become non-matching via modification of incoming messages.

Remark The model is extension of the combined model presented in [6], which in turn is based on the model presented in [17]. Here, we contribute IC and CP, thus allowing protocols with ID-based primitives running along with traditional certificate primitives. Observe also that via *RevealEphemeralPublicKey* query, the adversary can obtain the next ephemeral public key of a party and then decide which protocol to initiate at the party. Thus, the model covers the scenario described in the introduction where Bob does not know in advance which protocol will be executed yet reveals the ephemeral public key Bob will use in the session.

4 Protocol descriptions

We outline the protocol descriptions and then provide design motivations.

4.1 ID-ID variant

The protocol described in this section was first described in Fujioka, Suzuki, and Ustaoğlu [12]. In our description, $\mathcal{U}_A$ uses identity string $\hat{\mathcal{A}}$, with static key pair $(\mathtt{id}_A, \mathtt{ID}_A)$ and is the session initiator; similarly, $\mathcal{U}_B$ uses identity string $\hat{\mathcal{B}}$, has static key pair $(\mathtt{id}_B, \mathtt{ID}_B)$, and is the session responder. The two-pass protocol variant is denoted by $\Pi_{\mathtt{ii}}$.

$\mathcal{U}_A$ pre-computations: $\mathcal{U}_A$ chooses at random an ephemeral private key $x_A \in_R \mathtt{Z}_q$, computes the ephemeral public key $X_A = x_A P$, and stores the pair (X_A, x_A) in a temporary memory.
$\mathcal{U}_B$ pre-computations: $\mathcal{U}_B$ chooses at random an ephemeral private key $x_B \in_R \mathtt{Z}_q$, computes the ephemeral public key $X_B = x_B P$, and stores the pair (X_B, x_B) in a temporary memory.
$\mathcal{U}_A \xrightarrow{\mathcal{M}} \mathcal{U}_B$: $(\Pi_{\mathtt{ii}}, \hat{\mathcal{A}}, \tilde{\mathcal{U}_B}, X_A)$.
$\mathcal{U}_B \xrightarrow{\mathcal{M}} \mathcal{U}_A$: $(\Pi_{\mathtt{ii}}, \hat{\mathcal{A}}, \hat{\mathcal{B}}, X_A, X_B)$.
$\mathcal{U}_A$ computations: $\mathcal{U}_A$ verifies $X_B \in \mathbb{G}$; computes the shared secrets

$$\sigma_1 = \hat{\mathtt{e}}(\mathtt{id}_A + x_A Z, \mathtt{ID}_B + X_B),$$
$$\sigma_2 = \hat{\mathtt{e}}(\mathtt{id}_A, \mathtt{ID}_B),$$
$$\sigma_{\text{ee}} = x_A X_B;$$

the session key $\kappa = \mathtt{H}(\sigma_1, \sigma_2, \sigma_3, \hat{\mathcal{A}}, \hat{\mathcal{B}}, X_A, X_B, \Pi_{\mathtt{ii}})$; and completes by deleting (X_A, x).
$\mathcal{U}_B$ computations: $\mathcal{U}_B$ verifies $X_A \in \mathbb{G}$; computes the shared secrets

$$\sigma_1 = \hat{\mathtt{e}}(\mathtt{ID}_A + X_A, \mathtt{id}_B + x_B Z),$$
$$\sigma_2 = \hat{\mathtt{e}}(\mathtt{ID}_A, \mathtt{id}_B),$$
$$\sigma_{\text{ee}} = x_B X_A;$$

the session key $\kappa = \mathtt{H}(\sigma_1, \sigma_2, \sigma_{\text{ee}}, \hat{\mathcal{A}}, \hat{\mathcal{B}}, X_A, X_B, \Pi_{\mathtt{ii}})$; and completes by deleting (X_B, x_B).

Parties compute shared secrets

$$\sigma_1 = g^{z(\log(\mathtt{ID}_A)+x_A)(\log(\mathtt{ID}_B)+x_B)},$$
$$\sigma_2 = g^{z\log(\mathtt{ID}_A)\log(\mathtt{ID}_B)},$$
$$\sigma_{\text{ee}} = x_A x_B P,$$

and therefore, compute the same session key κ.

4.2 Cert-Cert variant

The protocol described in this section was first described in Ustaoğlu [24]. In our description, $\mathcal{U}_A$ uses certificate $\hat{A}$ with static key pair (a, A) and is the session initiator; similarly, $\mathcal{U}_B$ uses certificate $\hat{B}$ with static key pair (b, B) and is the session responder. The two-pass protocol variant is denoted by $\Pi_{\mathtt{cc}}$.

$\mathcal{U}_A$ pre-computations: $\mathcal{U}_A$ chooses at random an ephemeral private key $x_A \in_R \mathtt{Z}_q$, computes the ephemeral public key $X_A = x_A P$, and stores the pair (X_A, x_A) in a temporary memory.
$\mathcal{U}_B$ pre-computations: $\mathcal{U}_B$ chooses at random an ephemeral private key $x_B \in_R \mathtt{Z}_q$, computes the ephemeral public key $X_B = x_B P$, and stores the pair (X_B, x_B) in a temporary memory.
$\mathcal{U}_A \xrightarrow{\mathcal{M}} \mathcal{U}_B$: $(\Pi_{\mathtt{cc}}, \hat{A}, \tilde{\mathcal{U}_B}, X_A)$.
$\mathcal{U}_B \xrightarrow{\mathcal{M}} \mathcal{U}_A$: $(\Pi_{\mathtt{cc}}, \hat{A}, \hat{B}, X_A, X_B)$.
$\mathcal{U}_A$ computations: $\mathcal{U}_A$ verifies $X_B \in \mathbb{G}$; computes the shared secrets

$$\sigma_{\text{A}} = (x_A + a)(X_B + \mathrm{F}_{X_B} B),$$
$$\sigma_{\text{B}} = (x_A + \mathrm{F}_{X_A} A)(X_B + B);$$

the session key $\kappa = \mathtt{H}(\sigma_{\text{A}}, \sigma_{\text{B}}, \hat{A}, \hat{B}, X_A, X_B, \Pi_{\mathtt{cc}})$; and completes by deleting (X_A, x_A).
$\mathcal{U}_B$ computations: $\mathcal{U}_B$ verifies $X_A \in \mathbb{G}$; computes the shared secrets

$$\sigma_{\text{A}} = (x_B + b)(X_A + \mathrm{F}_{X_A} A),$$
$$\sigma_{\text{B}} = (x_B + \mathrm{F}_{X_B} B)(X_A + A);$$

the session key $\kappa = \mathtt{H}(\sigma_{\text{A}}, \sigma_{\text{B}}, \hat{A}, \hat{B}, X_A, X_B, \Pi_{\mathtt{cc}})$; and completes by deleting (X_B, x_B).

Parties compute shared secrets

$$\sigma_{\text{A}} = (x_A + a)(x_B + \mathrm{F}_{X_B} b)P,$$
$$\sigma_{\text{B}} = (x_A + \mathrm{F}_{X_A} a)(x_B + b)P,$$

and therefore, compute the same session key κ.

4.3 Cert-ID combinations

4.3.1 Cert-ID variant

In the description, $\mathcal{U}_A$ uses certificate $\hat{A}$ with static key pair (a, A) and is the session initiator; $\mathcal{U}_B$ uses identity $\hat{\mathcal{B}}$, has static key pair $(\mathtt{id}_B, \mathtt{ID}_B)$ and is the session responder. The two-pass protocol variant is denoted by $\Pi_{\mathtt{ci}}$.

$\mathcal{U}_A$ pre-computations: $\mathcal{U}_A$ chooses at random an ephemeral private key $x_A \in_R \mathtt{Z}_q$, computes the ephemeral public key $X_A = x_A P$, and stores the pair (X_A, x_A) in a temporary memory.
$\mathcal{U}_B$ pre-computations: $\mathcal{U}_B$ chooses at random an ephemeral private key $x_B \in_R \mathtt{Z}_q$, computes the ephemeral public key $X_B = x_B P$, and stores the pair (X_B, x_B) in a temporary memory.
$\mathcal{U}_A \xrightarrow{\mathcal{M}} \mathcal{U}_B$: $(\Pi_{\mathtt{ci}}, \hat{A}, \tilde{\mathcal{U}_B}, X_A)$.
$\mathcal{U}_B \xrightarrow{\mathcal{M}} \mathcal{U}_A$: $(\Pi_{\mathtt{ci}}, \hat{A}, \hat{\mathcal{B}}, X_A, X_B)$.
$\mathcal{U}_A$ computations: $\mathcal{U}_A$ verifies $X_B \in \mathbb{G}$; computes the shared secrets

$$\sigma_1 = \hat{\mathtt{e}}(\mathtt{ID}_B + X_B, (a + x_A)Z),$$
$$\sigma_2 = \hat{\mathtt{e}}(\mathtt{ID}_B, aZ),$$
$$\sigma_{\text{se}} = aX_B,$$
$$\sigma_{\text{ee}} = x_A X_B;$$

the session key $\kappa = \mathtt{H}(\sigma_1, \sigma_2, \sigma_{\text{se}}, \sigma_{\text{ee}}, \hat{A}, \hat{\mathcal{B}}, X_A, X_B, \Pi_{\mathtt{ci}})$; and completes by deleting (X_A, x_A).

$\mathcal{U}_B$ computations: $\mathcal{U}_B$ verifies $X_A \in \mathbb{G}$; computes the shared secrets

$$\begin{aligned}\sigma_1 &= \hat{\mathtt{e}}(\mathtt{id}_B + x_B Z, A + X_A),\\ \sigma_2 &= \hat{\mathtt{e}}(\mathtt{id}_B, A),\\ \sigma_{se} &= x_B A,\\ \sigma_{ee} &= x_B X_A;\end{aligned}$$

the session key $\kappa = \mathtt{H}(\sigma_1, \sigma_2, \sigma_{se}, \sigma_{ee}, \hat{A}, \hat{\mathcal{B}}, X_A, X_B, \Pi_{\mathtt{ci}})$; and completes by deleting (X_B, x_B).

Parties compute shared secrets

$$\begin{aligned}\sigma_1 &= g^{(a+x_A)z(\log(\mathtt{ID}_B)+x_B)},\\ \sigma_2 &= g^{az\log(\mathtt{ID}_B)},\\ \sigma_{se} &= ax_B P,\\ \sigma_{ee} &= x_A x_B P,\end{aligned}$$

and therefore, compute the same session key κ.

4.3.2 ID-Cert variant

In the description, $\mathcal{U}_A$ uses identity $\hat{\mathcal{A}}$ with static key pair $(\mathtt{id}_A, \mathtt{ID}_A)$ and is the session initiator; $\mathcal{U}_B$ uses certificate $\hat{B}$ with static key pair (b, B) and is the session responder. The two-pass protocol variant is denoted by $\Pi_{\mathtt{ic}}$.

The protocol computations are symmetrical to the Cert-ID variant with the computations of $\mathcal{U}_A$ and $\mathcal{U}_B$ reversed. Omitting details, parties compute shared secrets

$$\begin{aligned}\sigma_1 &= g^{z(\log(\mathtt{ID}_A)+x_A)(b+x_B)},\\ \sigma_2 &= g^{z\log(\mathtt{ID}_A)b},\\ \sigma_{se} &= x_A bP,\\ \sigma_{ee} &= x_A x_B P,\end{aligned}$$

and therefore, compute the same session key κ.

4.4 Design motivation

We concentrate on the design considerations for the ID-Cert protocol variants.

The shared secret σ_{ee} ensures that the protocol inherits the original Diffie–Hellman properties. On the one hand, in the case of ID-AKE, it may be desirable to allow the KGC to be able to generate past session keys. Within a single entity, such property has value since it is a problem of trust that concerns a single company. On the other hand, an external party may target a specific division and be unwilling to place its trust in other divisions. We believe that this approach is more sensible and hence include σ_{ee} in the key derivation function which is a standard measure to prevent the KGC center from computing past session keys. Should it be desired, it is possible to consider protocol variants that do allow KGC to recover past session keys. Such variant could either dispense with σ_{ee} or modify σ_{ee} in a way that allows KGC to compute the value σ_{ee}, e.g., $\sigma_{ee} = g^{zx_Ax_B}$.

Protocols $\Pi_{\mathtt{ci}}$ and $\Pi_{\mathtt{ic}}$ are symmetrical and we concentrate on $\Pi_{\mathtt{ci}}$, where the initiator provides a certificate. Next, we will informally argue that if the initiator $\mathcal{U}_A$ believes that the initiator's static or initiator's ephemeral private key has not been leaked, then any peer $\mathcal{U}_B$ that can compute the session key must know both $\mathtt{id}_B$ and x_B. Similarly, if the responder is assured that $\mathtt{id}_B$ or x_B has not been exposed, then the purported initiator $\mathcal{U}_A$ must know both a and x_A.

Suppose at least one of x_A and a is known *only*[4] to the initiator, and hence $x_A + a$ is known only to initiator. On the one hand, if x_A is private to the initiator, to compute the session key any entity needs σ_{ee} and hence x_B. Computing the session key also requires σ_1 in this case, since $x_A + a$ is private to the initiator computing σ_1 implies knowledge of $z(\log(\mathtt{ID}_B) + x_B)P$. With x_B and $z(\log(\mathtt{ID}_B) + x_B)P$, anyone can derive $\mathtt{id}_B = z\log(\mathtt{ID}_B)P$. Thus, in this case, the initiator is assured that his peer has both x_B and $\mathtt{id}_B$. On the other hand, if a is private to the initiator to compute the session key, any entity needs σ_{se} and hence x_B. The previous reasoning applies to deduce that the entity that computes the session key can obtain both x_B and $\mathtt{id}_B$. Consequently, if the KGC center does not behave maliciously and z has not been leaked the initiator holding a certificate is assured about the identity of its peer.

Suppose now at least one of x_B and $\mathtt{id}_B$ is known *only* to the responder, and hence $g^{z(\log(\mathtt{ID}_B)+x_B))}$ is available only to the responder. On the one hand, if x_B is private to the responder, to compute the session key, any entity needs σ_{se} and σ_{ee}, and hence x_A and a. On the other hand, if $\mathtt{id}_B$ is private to the responder and since $\log(\mathtt{ID}_B)$ is hard to compute, an entity that can compute all shared secrets can also compute $g^{x_A z\log(\mathtt{ID}_B)}$ and $g^{az\log(\mathtt{ID}_B)}$; σ_{se} is crucial to extract these two values. If z is not used with malicious intent, the responder is assured that the peer has x_A and a.

The above reasoning shows that σ_{se} prevents KGC from impersonating parties with certificates to parties with identity-based private keys. Such impersonation does not arise in purely id-based protocols, where the KGC can derive any static private key and therefore can impersonate any user. Thus, key compromise impersonation (KCI) attacks with respect to KGC are not considered. In the mixed variant, however, such attacks can be incorporated in the model. We believe that there is a close relation between key escrow and KCI attacks: if KGC can recover past keys, the KGC can trivially impersonate users with certificates. While it is worth considering protocol variants that allow key escrow in the purely ID-based setting, in the mixed ID-based and certificate-based setting key escrow is less desirable.

[4] In other words x_A or a has not been compromised.

5 Security

Theorem 1 *If* $\mathtt{F}_{(\cdot)}$, $\mathtt{H}_{id}$, *and* $\mathtt{H}$ *are random oracles and the gBDH assumption holds in* $(\mathbb{G}, \mathbb{G}_t)$, *then protocols* $\Pi_{\mathtt{ii}}$, $\Pi_{\mathtt{cc}}$, $\Pi_{\mathtt{ci}}$, *and* $\Pi_{\mathtt{ic}}$ *are secure in the combined model.*

Argument To argue the validity of Theorem 1, we have to verify the conditions of Definition 2. Condition 1 is straightforward; we verify Condition 2. A successful adversary $\mathcal{M}$ is an adversary that distinguishes the session key of a fresh session from a randomly chosen session key. The event that $\mathcal{M}$ is successful is denoted by M. Let γ denote the security parameter and assume by contradiction there exists an adversary $\mathcal{M}$ that can distinguish the session key of a fresh session with probability

$$\frac{1}{2} + p(\gamma) \leq Pr(M), \tag{1}$$

where $p(\gamma)$ is non-negligible function in γ. Let the test session be

1. $\mathtt{s}^t = (\Pi_{\mathtt{t}}, \mathcal{U}_A, \mathcal{U}_B, \mathcal{I}, X_A, X_B)$ or
2. $\mathtt{s}^t = (\Pi_{\mathtt{t}}, \mathcal{U}_A, \mathcal{U}_B, \mathcal{R}, X_B, X_A)$,

where $\Pi_{\mathtt{t}}$ is one of $\Pi_{\mathtt{ii}}$, $\Pi_{\mathtt{cc}}$, $\Pi_{\mathtt{ci}}$, or $\Pi_{\mathtt{ic}}$, and depending on $\Pi_{\mathtt{t}}, \mathcal{U}_A$ and $\mathcal{U}_B$ are certificates or identities. Let H be the event that $\mathcal{M}$ queries the random oracle $\mathtt{H}$ with

1. $(\sigma_{\mathrm{t}}, \mathcal{U}_A, \mathcal{U}_B, X_A, X_B, \Pi_{\mathtt{t}})$ in the former
2. and with $(\sigma_{\mathrm{t}}, \mathcal{U}_B, \mathcal{U}_A, X_B, X_A, \Pi_{\mathtt{t}})$ in the latter case.

In the query, σ_{t} is the collection of shared secrets used in protocol $\Pi_{\mathtt{t}}$, and $\mathcal{U}_A$ and $\mathcal{U}_B$ are as defined by $\Pi_{\mathtt{t}}$. Let $\overline{H}$ be the complement of H. We have that

$$Pr(M) = Pr(M \wedge H) + Pr(M \wedge \overline{H}). \tag{2}$$

The key derivation function includes the complete session identifiers, and therefore, non-matching sessions have different inputs to the key derivation function. Since $\mathtt{H}$ is a random oracle, $\mathcal{M}$ cannot obtain any information about the test session key from session keys of non-matching sessions. In event $M \wedge \overline{H}$, the adversary does not issue *RevealSessionKey* against the test session and its matching session and hence $Pr(M \wedge \overline{H}) \leq \frac{1}{2}$. Combined with Eqs. 1 and 2, it follows that

$$p(\gamma) \leq Pr(M \wedge H). \tag{3}$$

In the remainder of this argument, M^* denotes $M \wedge H$.

Let $\mathcal{M}$ succeeds in an environment with $n_i + n_c$ parties, where n_i parties obtain their static keys from a KGC and the remainder obtain certificates from a CA. Let s be an upper bound on the number of sessions activated within each party.

The following conventions will be used in the security argument. The BDDH oracle on input

$$(xP, yP, zP, g^r)$$

returns the bit 1 if $\hat{\mathrm{e}}(P, P)^{xyz} = g^r$ and the bit 0 otherwise. A decisional Diffie–Hellman oracle for $\mathbb{G}$ can be simulated using standard pairing techniques; we assume a DDH oracle is available to the gBDH solver $\mathcal{S}$. Also,

$$\xi : \mathbb{G} \times \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$$

is a random function known only to $\mathcal{S}$ such that

$$\xi(X_1, X_2, X_3) = \xi(X_i, X_j, X_k)$$

for any permutation ijk of 123. Similarly,

$$\mu : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$$

is a random function known only to $\mathcal{S}$, such that

$$\mu(X_1, X_2) = \mu(X_2, X_1).$$

The algorithm $\mathcal{S}$ uses ξ and μ to represent BDH$(\cdot, \cdot, \cdot)$ and CDH$(\cdot, \cdot)$, respectively, in situations when $\mathcal{S}$ does not possess the discrete logarithms of the elements involved. Except with negligible probability of guessing ξ and μ, the adversary will not detect their usage.

Let $\mathtt{s}^t$ denote the test session and $\mathtt{s}^m$ its matching if it exists. We consider the following complementary events.

1. Event $M^* \wedge ev_1$: $\mathtt{s}^t$ has a matching session $\mathtt{s}^m$, and $\mathcal{M}$ queries the ephemeral private keys of neither $\mathtt{s}^t$ nor $\mathtt{s}^m$.
2. Event $M^* \wedge ev_2$: $\mathcal{M}$ does not query for $\mathtt{s}^t$'s ephemeral private key and does not query for the static private key of the test session peer.
3. Event $M^* \wedge ev_3$: $\mathtt{s}^t$ has a matching session $\mathtt{s}^m$ and $\mathcal{M}$ queries neither the static private key of the test session owner nor the ephemeral secret of $\mathtt{s}^m$.
4. Event $M^* \wedge ev_4$: $\mathcal{M}$ does not query for the static private keys of the test session peers.

If any of the peers uses an identity-based static key and $\mathcal{M}$ does not query for the corresponding static private key, then $\mathcal{M}$ also does not query for KGC's master static private key. Note also that Events $M^* \wedge ev_2$ and $M^* \wedge ev_4$ include the case where $\mathtt{s}^m$ does not exist. It is straightforward to verify that

$$Pr(M^*) = \sum_{i=1}^{4} Pr(M^* \wedge ev_i) \tag{4}$$

The goal of $\mathcal{S}$ is to produce a solution to a BDH challenge (U, V, W).

Remark If the argument is restricted only to the Cert-ID variant, then using the protocol notation the embedding of the BDH challenge corresponds to:

Event $M^* \wedge ev_1$ $x_A = u$ and $x_B = v$;
Event $M^* \wedge ev_2$ $x_A = u$, $\log \mathtt{id}_B = v$ and $z = w$
Event $M^* \wedge ev_3$ $a = u$ and $x_B = v$;
Event $M^* \wedge ev_4$ $a = u$, $\log \mathtt{id}_B = v$ and $z = w$.

5.1 Event $M^* \wedge ev_1$

For Event $M^* \wedge ev_1$, $\mathcal{S}$ begins by establishing a set CP of n_c parties and a set IP with n_i parties. In addition, $\mathcal{S}$ selects a master static private key z and corresponding static public key Z. Parties in CP are given static key pairs $(a, A = aP)$; parties in IP are given static key pairs $(\mathtt{id}_A = z\mathtt{ID}_A, \mathtt{ID}_A)$. Technically, for this event, the adversary can be allowed to set party identifiers. For each honest party $\mathcal{U}_A$, $\mathcal{S}$ maintains a list of at most s ephemeral key pairs, and two markers – a party marker and an adversary marker. The markers initially point to the first entry of the list. Whenever $\mathcal{U}_A$ is activated to create a new session, $\mathcal{S}$ checks if the party marker points to an empty entry. If so, then $\mathcal{S}$ selects a new ephemeral key pair on behalf of $\mathcal{U}_A$ as described in the pre-computation steps of the protocols. If the list entry is not empty, then $\mathcal{S}$ uses the ephemeral key pair in that list entry for the newly created session. In either case, the party marker is updated to point to the next list entry, and the adversary marker is also advanced if it points to an earlier entry. If $\mathcal{M}$ issues an *RevealEphemeralPublicKey* query, then $\mathcal{S}$ selects a new ephemeral key pair on behalf of $\mathcal{U}_A$ as described in the pre-computation steps of the protocols. $\mathcal{S}$ stores the key pair in the entry pointed to by the adversary marker, returns the public key as the query response, and advances the adversary marker.

In addition to the above steps, $\mathcal{S}$ randomly selects two parties $\mathcal{U}_C, \mathcal{U}_D$ and two integers $i, j \in_R [1, s]$ subject to the condition that $(\mathcal{U}_C, i) \neq (\mathcal{U}_D, j)$. $\mathcal{S}$ selects ephemeral key pairs on behalf of honest parties as described above, with the following exceptions. The ith ephemeral public key selected on behalf of $\mathcal{U}_C$ is chosen to be U, and the jth ephemeral private key selected on behalf of $\mathcal{U}_D$ is V. The sessions with outgoing ephemeral public keys U and V will be denoted by $\mathtt{s}^u$ and $\mathtt{s}^v$, respectively; $\mathcal{S}$ does not possess the corresponding ephemeral private keys.

In the simulation, $\mathcal{S}$ responds to all adversary queries faithfully except when session $\mathtt{s}^u$ or $\mathtt{s}^v$ is activated. If any of those session is activated, $\mathcal{S}$ deviates by setting the shared secrets via the functions ξ and μ. If $\mathcal{M}$ queries for the ephemeral private key of either $\mathtt{s}^u$ or $\mathtt{s}^v$, then $\mathcal{S}$ aborts with failure. In addition, $\mathcal{S}$ aborts if the test session and its matching session are not $\mathtt{s}^u$ and $\mathtt{s}^v$. If a key derivation $\mathtt{H}$-query involving $\mathtt{s}^u$ and $\mathtt{s}^v$ is issued, and in that query the shared secrets are not equal to ξ and μ, then $\mathcal{S}$ uses the decisional BDH and DH oracles to provide consistent $\mathtt{H}$ query responses. That is, if DBDH and DDH both return 1, then $\mathcal{S}$ responds with $\mathtt{H}$ value that is equal to the $\mathtt{H}$ query using ξ and μ. Possibly, ξ and μ have not yet been defined, but $\mathcal{S}$ is successful in solving the BDH challenge when both decisional oracles return bit 1 for the $\mathtt{s}^t$ or $\mathtt{s}^m$ query.

With probability at least $\frac{1}{(s(n_i+n_c))^2}$, $\mathcal{S}$ guesses correctly the test session and its matching session activation as sessions $\mathtt{s}^u$ and $\mathtt{s}^v$. Suppose this is indeed the case and so $\mathcal{S}$ does not abort during the test session query. Under Event $M^* \wedge ev_2$, $\mathcal{M}$ does query for the ephemeral secrets of $\mathtt{s}^t$ and $\mathtt{s}^m$, and therefore, $\mathcal{S}$ does not abort at any *RevealEphemeralKey* query. Furthermore, in Event M^*, $\mathcal{M}$ queries $\mathtt{H}$ with shared secrets that satisfy the decisional oracles. When such query is made, $\mathcal{S}$ who detects it during the $\mathtt{H}$ query response terminates the adversary and returns $\hat{\mathsf{e}}(\sigma_{\text{ee}}, W)$ if $\Pi_{\mathtt{t}}$ is one of $\Pi_{\mathtt{ii}}$, $\Pi_{\mathtt{ic}}$, or $\Pi_{\mathtt{ci}}$. If $\Pi_{\mathtt{t}}$ is $\Pi_{\mathtt{cc}}$, then

$$\sigma_{\text{ee}} = \text{CDH}(U, V) = \sigma_{\text{A}} - a\left(X_B + \text{F}_{X_B} B\right) - \text{F}_{X_B} b X_A,$$

where $\{U, V\} = \{X_A, X_B\}$, and $\mathcal{U}_A$ and $\mathcal{U}_B$ are session peers, and so $\mathcal{S}$ is also successful. The success probability of $\mathcal{S}$ in Event $M^* \wedge ev_1$ is bounded by

$$Pr(S) \geq \frac{1}{(s(n_i + n_c))^2} Pr(M^* \wedge ev_1) = \frac{p_1}{(s(n_i + n_c))^2}, \tag{5}$$

where $p_1 = Pr(M^* \wedge ev_1)$.

5.2 Event $M^* \wedge ev_2$

For Event $M^* \wedge ev_2$, $\mathcal{S}$ begins by establishing a set CP of n_c parties and a set IP with n_i parties. Furthermore, $\mathcal{S}$ selects two parties $\mathcal{U}_C$ and $\mathcal{U}_D$ at random from the set $CP \cup IP$ and an integer $i \in_R [1, s]$.

If $\mathcal{U}_D \in IP$, then $\mathcal{S}$ sets $Z = W$ and $\mathtt{ID}_D = V$. Note that this implies $\mathtt{H}_{id}(\hat{\mathcal{D}}) = \mathtt{ID}_D$, and hence, in this and subsequent events, $\mathcal{S}$ selects identifiers for parties. For other $\mathtt{H}_{id}$ values, $\mathcal{S}$ defines $\mathtt{H}_{id}(\hat{\mathcal{A}}) = aP = \mathtt{ID}_A$, where a is a random integer selected by $\mathcal{S}$ from the set $[1, q]$. The static private key corresponding to $\mathtt{ID}_A$ is $\mathtt{id}_A = aW$. Except with negligible probability, $\mathcal{M}$ cannot distinguish the simulation of $\mathtt{H}_{id}$ from a true random oracle. If $\mathcal{U}_D \in CP$, then $\mathcal{S}$ sets $D = V$ and the remaining parties are set up as in Event $M^* \wedge ev_1$. Note that $\mathcal{S}$ does not possess the static private key of $\mathcal{U}_D$.

As in Event $M^* \wedge ev_1$, for each honest party, $\mathcal{S}$ maintains a list of s ephemeral key pairs and the two markers. $\mathcal{S}$ deviates from the protocol specifications by setting the ith session ephemeral public key of $\mathcal{U}_C$ equal U; $\mathcal{S}$ does not possess the corresponding ephemeral private key. This session will be denoted by $\mathtt{s}^u$.

In the simulation, $\mathcal{S}$ responds to all adversary queries faithfully except when session $\mathtt{s}^u$ is activated or if $\mathcal{U}_D$ is activated. In either case, $\mathcal{S}$ deviates by setting the shared secrets via the functions ξ and μ. If $\mathcal{M}$ queries for the ephemeral private key of $\mathtt{s}^u$, then $\mathcal{S}$ aborts with failure; $\mathcal{S}$ also aborts with failure if $\mathcal{M}$ queries for the static private key of $\mathcal{U}_D$. Furthermore, if $\mathcal{U}_D \in IP$, then $\mathcal{S}$ also aborts if $\mathcal{M}$ queries for the master static private key.

If the test session $\mathtt{s}^t$ is not $\mathtt{s}^u$ with peer $\mathcal{U}_D$, then $\mathcal{S}$ aborts. If a $\mathtt{H}$ query involving $\mathtt{s}^u$ is issued and in that query the shared secrets are not equal to ξ and μ, then similar to Event $M^* \wedge ev_1$ $\mathcal{S}$ uses the decisional oracles to provide consistent query responses.

With probability at least $\frac{1}{s(n_i+n_c)^2}$, $\mathcal{S}$ guesses correctly the test session and its peer. Suppose this is indeed the case and so $\mathcal{S}$ does not abort during the test session query. Under Event $M^* \wedge ev_2$, $\mathcal{M}$ does query for the ephemeral secrets of $\mathtt{s}^u$ and the static private key of the test session peer $\mathcal{U}_D$. If $\mathcal{U}_D \in IP$, then $\mathcal{M}$ does not query for the master static private key of the key generation center. Therefore, $\mathcal{S}$ does not abort with failure during the simulation. Furthermore, in Event M^*, $\mathcal{M}$ queries $\mathtt{H}$ with shared secrets that satisfy the decisional oracles. When that $\mathtt{H}$ query is made $\mathcal{S}$ terminates $\mathcal{M}$ and

if $\mathtt{s}^t$ is $(\Pi_{\mathtt{ii}}, \hat{\mathcal{A}}, \hat{\mathcal{B}}, \mathcal{I}, X_A, X_B)$ or $(\Pi_{\mathtt{ii}}, \hat{\mathcal{A}}, \hat{\mathcal{B}}, \mathcal{R}, X_B, X_A)$
$\mathcal{S}$ returns

$$\mathrm{BDH}(U, V, W) = \frac{\sigma_1}{\sigma_2 \hat{\mathrm{e}}\,(\sigma_{\mathrm{ee}} + \log(\mathtt{ID}_A) X_B, W)};$$

if $\mathtt{s}^t$ is $(\Pi_{\mathtt{cc}}, \hat{A}, \hat{B}, \mathcal{I}, X_A, X_B)$ or $(\Pi_{\mathtt{cc}}, \hat{A}, \hat{B}, \mathcal{R}, X_B, X_A)$
$\mathcal{S}$ obtains

$$\mathrm{CDH}(U, V) = \frac{(\sigma_{\mathrm{A}} - a(X_B + \mathrm{F}_{X_B} B))}{\mathrm{F}_{X_B} - 1} - \frac{(\sigma_{\mathrm{B}} - \mathrm{F}_{X_A} a(X_B + B))}{\mathrm{F}_{X_B} - 1},$$

and returns

$$\mathrm{BDH}(U, V, W) = \hat{\mathrm{e}}\,(\mathrm{CDH}(U, V), W);$$

if $\mathtt{s}^t$ is $(\Pi_{\mathtt{ic}}, \hat{\mathcal{A}}, \hat{B}, \mathcal{I}, X_A, X_B)$ or $(\Pi_{\mathtt{ci}}, \hat{\mathcal{A}}, \hat{B}, \mathcal{R}, X_B, X_A)$
$\mathcal{S}$ returns

$$\mathrm{BDH}(U, V, W) = \hat{\mathrm{e}}(\sigma_{\mathrm{se}}, W);$$

if $\mathtt{s}^t$ is $(\Pi_{\mathtt{ic}}, \hat{A}, \hat{\mathcal{B}}, \mathcal{R}, X_B, X_A)$ or $(\Pi_{\mathtt{ci}}, \hat{A}, \hat{\mathcal{B}}, \mathcal{I}, X_A, X_B)$
$\mathcal{S}$ returns

$$\mathrm{BDH}(U, V, W) = \frac{\sigma_1}{\sigma_2 \left(\hat{\mathrm{e}}(\sigma_{\mathrm{se}} + \sigma_{\mathrm{ee}}, W)\right)}.$$

In Event $M^* \wedge ev_2$ the success probability of $\mathcal{S}$ is bounded by

$$Pr(S) \geq \frac{1}{s(n_i + n_c)^2} Pr(M^* \wedge ev_2) = \frac{p_2}{s(n_i + n_c)^2}, \quad (6)$$

where $p_2 = Pr(M^* \wedge ev_2)$.

5.3 Event $M^* \wedge ev_3$

The setup and simulation in Event $M^* \wedge ev_3$ is the same as for Event $M^* \wedge ev_2$, except the abortion conditions for the test session. More precisely, if the test session $\mathtt{s}^t$ is not owned by $\mathcal{U}_D$ or is not matching to $\mathtt{s}^u$, then $\mathcal{S}$ aborts.

With probability at least $\frac{1}{s(n_i+n_c)^2}$, $\mathcal{S}$ guesses correctly the test session owner and the session matching to the test session. Suppose this is indeed the case and so $\mathcal{S}$ does not abort during the test session query. Under Event $M^* \wedge ev_3$, $\mathcal{M}$ does query for the ephemeral secrets of $\mathtt{s}^u$ and the static private key of the test session owner $\mathcal{U}_D$. If $\mathcal{U}_D \in IP$, then $\mathcal{M}$ does not query for the master static private key of the key generation center. Furthermore, in Event $M^* \wedge ev_3$, $\mathcal{M}$ queries $\mathtt{H}$ with shared secrets that satisfy the decisional oracles. When such query is made, $\mathcal{S}$ who detects it during the $\mathtt{H}$ query response terminates the adversary and

if $\mathtt{s}^t$ is $(\Pi_{\mathtt{ii}}, \hat{\mathcal{A}}, \hat{\mathcal{B}}, \mathcal{I}, X_A, X_B)$ or $(\Pi_{\mathtt{ii}}, \hat{\mathcal{A}}, \hat{\mathcal{B}}, \mathcal{R}, X_B, X_A)$
$\mathcal{S}$ returns

$$\mathrm{BDH}(U, V, W) = \frac{\sigma_1}{\sigma_2 \hat{\mathrm{e}}\,(\sigma_{\mathrm{ee}} + \log(\mathtt{ID}_B) X_A, W)};$$

if $\mathtt{s}^t$ is $(\Pi_{\mathtt{cc}}, \hat{A}, \hat{B}, \mathcal{I}, X_A, X_B)$ or $(\Pi_{\mathtt{cc}}, \hat{A}, \hat{B}, \mathcal{R}, X_A, X_B)$
$\mathcal{S}$ obtains

$$\mathrm{CDH}(U, V) = \frac{(\sigma_{\mathrm{B}} - b(X_A + \mathrm{F}_{X_A} A))}{\mathrm{F}_{X_A} - 1} - \frac{(\sigma_{\mathrm{A}} - \mathrm{F}_{X_B} b(X_A + A))}{\mathrm{F}_{X_A} - 1},$$

and returns

$$\mathrm{BDH}(U, V, W) = \hat{\mathrm{e}}\,(\mathrm{CDH}(U, V), W);$$

if $\mathtt{s}^t$ is $(\Pi_{\mathtt{ic}}, \hat{\mathcal{A}}, \hat{B}, \mathcal{I}, X_A, X_B)$ or $(\Pi_{\mathtt{ci}}, \hat{\mathcal{A}}, \hat{B}, \mathcal{R}, X_B, X_A)$
$\mathcal{S}$ returns

$$\mathrm{BDH}(U, V, W) = \frac{\sigma_1}{\sigma_2 \left(\hat{\mathrm{e}}(\sigma_{\mathrm{se}} + \sigma_{\mathrm{ee}}, W)\right)};$$

if $\mathtt{s}^t$ is $(\Pi_{\mathtt{ic}}, \hat{A}, \hat{\mathcal{B}}, \mathcal{R}, X_B, X_A)$ or $(\Pi_{\mathtt{ci}}, \hat{A}, \hat{\mathcal{B}}, \mathcal{I}, X_A, X_B)$
$\mathcal{S}$ returns

$$\mathrm{BDH}(U, V, W) = \hat{\mathrm{e}}(\sigma_{\mathrm{se}}, W).$$

In Event $M^* \wedge ev_3$ the success probability of $\mathcal{S}$ is bounded by

$$Pr(S) \geq \frac{1}{s(n_i + n_c)^2} Pr(M^* \wedge ev_3) = \frac{p_3}{s(n_i + n_c)^2}, \quad (7)$$

where $p_3 = Pr(M^* \wedge ev_3)$.

5.4 Event $M^* \wedge ev_4$

For Event $M^* \wedge ev_4$, the setup of parties is the same as for Event $M^* \wedge ev_2$. In this case, $\mathcal{S}$ selects two parties $\mathcal{U}_C$ and $\mathcal{U}_D$ and sets their static public keys as U and V, respectively. If either party comes from the set IC, then $\mathcal{S}$ sets static private keys of parties in IC as in Event $M^* \wedge ev_2$, otherwise as in Event $M^* \wedge ev_1$. For parties $\mathcal{U}_C$ and $\mathcal{U}_D$, $\mathcal{S}$ does not possess the corresponding static private keys and hence aborts if $\mathcal{M}$ queries for them. In case either party is in IC, then $\mathcal{S}$ also aborts if $\mathcal{M}$ queries for the master static private key of the key generation center. Lastly, $\mathcal{S}$ aborts if $\mathcal{M}$ selects a test session with peers that are not $\mathcal{U}_C$ and $\mathcal{U}_D$.

With probability at least $\frac{1}{(n_i+n_c)^2}$, the test session communicating partners are $\mathcal{U}_C$ and $\mathcal{U}_D$. Suppose this is indeed the case and so $\mathcal{S}$ does not abort during the test session query. Under Event $M^* \wedge ev_4$, $\mathcal{M}$ does not query for the static private key of the test session peers or the master static private key if any of the peers is in IC. In Event M^*, $\mathcal{M}$ queries $\mathtt{H}$ with shared secrets that satisfy the decisional oracles. When such query is made, $\mathcal{S}$ who detects it during the $\mathtt{H}$ query response terminates the adversary and

if $\mathtt{s}^t$ is $(\Pi_{\mathtt{ii}}, \hat{\mathcal{A}}, \hat{\mathcal{B}}, \mathcal{I}, X_A, X_B)$ or $(\Pi_{\mathtt{ii}}, \hat{\mathcal{A}}, \hat{\mathcal{B}}, \mathcal{R}, X_B, X_A)$ $\mathcal{S}$ returns

$\mathrm{BDH}(U, V, W) = \sigma_2$;

if $\mathtt{s}^t$ is $(\Pi_{\mathtt{cc}}, \hat{A}, \hat{B}, \mathcal{I}, X_A, X_B)$ or $(\Pi_{\mathtt{cc}}, \hat{A}, \hat{B}, \mathcal{R}, X_B, X_A)$ $\mathcal{S}$ obtains

$$\mathrm{CDH}(U, V) = \frac{\mathrm{F}_{X_A}(\sigma_{\mathrm{A}} - x_A(X_B + \mathrm{F}_{X_B}B))}{\mathrm{F}_{X_A}\mathrm{F}_{X_B} - 1} - \frac{(\sigma_{\mathrm{B}} - x_A(X_B + B))}{\mathrm{F}_{X_A}\mathrm{F}_{X_B} - 1},$$

and returns

$\mathrm{BDH}(U, V, W) = \hat{\mathrm{e}}\,(\mathrm{BDH}(U, V), W)$;

otherwise $\mathcal{S}$ returns

$\mathrm{BDH}(U, V, W) = \sigma_2$

In Event $M^* \wedge ev_4$, the probability of success is bounded by

$$Pr(S) \geq \frac{1}{(n_i + n_c)^2} Pr(M^* \wedge ev_4) = \frac{p_4}{(n_i + n_c)^2}, \quad (8)$$

where $p_4 = Pr(M^* \wedge ev_4)$.

5.5 Analysis

Combining Eqs. 5–8, the success probability of $\mathcal{S}$ is bound by

$$Pr(S) \geq \frac{1}{(n_i + n_c)^2} \max\left(\frac{p_1}{s^2}, \frac{p_2}{s}, \frac{p_3}{s}, p_4\right),$$

which is non-negligible if $\mathcal{M}$ is successful with non-negligible probability. Furthermore, the actions of $\mathcal{S}$ in response to $\mathcal{M}$ queries are executed in polynomial time. Therefore, $\mathcal{S}$ is a polynomially bounded algorithm that solves a BDH instance with non-negligible probability contradicting the assumption in the theorem. □

5.6 Reflections

In the simulations of $M^* \wedge ev_4$, it is assumed that $\mathcal{U}_C$ and $\mathcal{U}_D$ are distinct parties. More precisely, if the test session owner and peer are the same party, then $\mathcal{S}$ may fail as $\mathcal{M}$ may produce $\mathrm{CDH}(U, U)$ or $\mathrm{CDH}(V, V)$ instead of $\mathrm{CDH}(U, V)$ or produce $\mathrm{BDH}(U, U, W)$ or $\mathrm{BDH}(V, V, W)$. The case $\mathcal{U}_C = \mathcal{U}_D$ can be encompassed by a reduction from the square variants of the CDH and the BDH problems, where some of the element U or V or W are the same group element. To do so, $\mathcal{S}$'s actions are modified as follows: given $U = uP$, $\mathcal{S}$ selects $v, w \in_R [1, q-1]$ and computes $V = vU$ and $W = wU$. Given $\mathrm{BDH}(U, V, W)$, v and w it is straightforward to compute $\mathrm{BDH}(U, U, U)$. Therefore, in the event that the test session's initiator and responder are the same party can be used to solve a square variant of the BDH problem.

5.7 Identifier selection

In the security model, we briefly mentioned selection of identifiers and said that in Events $M^* \wedge ev_2$, $M^* \wedge ev_3$ and $M^* \wedge ev_4$ $\mathcal{S}$ selects party identifiers. It is possible to allow $\mathcal{M}$ to set party identifiers via the first *Send* query to the party. In that case, $\mathcal{S}$ must guess which $\mathtt{H}_{id}$ queries to match with U and V, equivalently, guess which of $\mathcal{M}$'s $\mathtt{H}_{id}$ queries will be used as the identifiers for $\mathcal{U}_C$ and $\mathcal{U}_C$. Consequently, the reduction tightness decreases. For simplicity sake, we allow $\mathcal{S}$ to set honest party identifiers.

6 Concluding remarks

In this work, we motivated and described a collection of algorithms that allow users to establish secure session keys independent from what type of static public key they use. In particular, we proposed a method whereby a user who has a certificate can establish a session key with a user that is part of an ID-based infrastructure. The security of the established session is not violated even if the same identities and certificates are also used in purely ID- and purely certificate-based settings. As such, our algorithms are interesting since they do not require users to manage additional static private keys.

Both pure variants of our protocols $\Pi_{\mathtt{ii}}$ and $\Pi_{\mathtt{cc}}$ have been shown to be efficient protocols. The mixed variants could potentially be optimized to reduce the number of group exponentiations, while interesting such modifications will imply more complicated security arguments and possibly loss of tightness. Since pairing computations are more expensive than group exponentiations, we kept extra group exponentiations but obtained a protocol with more intuitive security. It is worth noting that the pairing computations are the same across protocols and so potential implementations are easier to devise.

Lastly, we observe that in the mixed protocol variant, it is of interest to consider whether a malicious KGC can impersonate honest clients that hold certificates to members of the KGC domain. This idea has a natural extension in the

ID-based domain extension. In particular, if an ID-based protocol is executed between two users that belong to different KGC centers and standard domain extension techniques are used, then it is worth to consider whether any KGC center can impersonate users belonging to other domains to users within its own domain.

References

1. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Darnell, M. (ed.) 6th IMA International Conference, vol. 1355 of LNCS, pp. 30–45. Springer, Berlin (1997)
2. Boyd, C., Choo, K.-K.R.: Security of two-party identity-based key agreement. In: Dawson, E., Vaudenay, S. (eds.) Progress in Cryptology—Mycrypt 2005, vol. 3715 of LNCS, pp. 229–243. Springer, Berlin (2005)
3. Boyd, C., Cliff, Y., González Nieto, J.M., Paterson, K.G.: Efficient one-round key exchange in the standard model. In: Mu, Y., Susilo, W., Seberry, J. (eds.) Information Security and Privacy—ACISP 2008, vol. 5107 of LNCS, pp. 69–83. Springer, Berlin (2008)
4. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) Advances in Cryptology—EUROCRYPT 2001, vol. 2045 of LNCS, pp. 453–474. Springer, Berlin (2001)
5. Canetti, R., Krawczyk, H.: Security analysis of IKE's signature-based key-exchange protocol. In: Yung, M. (ed.) Advances in Cryptology—CRYPTO 2002, vol. 2442 of LNCS, pp. 143–161. Springer, Berlin (2002)
6. Chatterjee, S., Menezes, A., Ustaoğlu, B.: Reusing static keys in key agreement protocols. In: Roy, B., Sendrier, N. (eds.) Progress in Cryptology—INDOCRYPT 2009, vol. 5922 of LNCS, pp. 39–56. Springer, Berlin (2009)
7. Chen, L., Cheng, Z., Smart, N.P.: Identity-based key agreement protocols from pairings. Int. J. Inf. Security **6**(4), 213–241 (2007)
8. Chen, L., Kudla, C.: Identity based authenticated key agreement protocols from pairings. In: Proceedings of 16th IEEE Computer Security Foundations Workshop, pp. 219–233 (2003)
9. Choo, K.-K.R., Chow, S.S.M.: Strongly-secure identity-based key agreement and anonymous extension. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) Information Security—ISC 2008, vol. 4779 of LNCS, pp. 203–220. Springer, Berlin (2007)
10. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **IT-22**(6), 644–654 (1976)
11. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **IT-31**(4), 469–472 (1985)
12. Fujioka, A., Suzuki, K., Ustaoğlu, B.: Utilizing postponed ephemeral and pseudo-static keys in tripartite and identity-based key agreement protocols. Cryptology ePrint Archive, Report 2009/423 (2009)
13. Günther, C.G.: An identity-based key-exchange protocol. In: Vandewaile, J., Quisquater, J.-J. (eds.) Advances in Cryptology —EUROCRYPT'89, vol. 434 of LNCS, pp. 29–37. Springer, Berlin (1989)
14. Huang, H., Cao, Z.: An ID-based authenticated key exchange protocol based on bilinear Diffie–Hellman problem. In: Safavi-Naini, R., Varadharajan, V. (eds.) ASIACCS '09: Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, pp. 333–342. ACM (2009)
15. Kelsey, J., Schneier, B., Wagner, D.: Protocol interactions and the chosen protocol attack. In: Christianson, B., Crispo, B., Lomas, M., Michael, R. (eds.) Security Protocols—5th International Workshop, vol. 1361 of LNCS, pp. 91–104. Springer, Berlin (1998)
16. Krawczyk, H.: HMQV: a high-performance secure Diffie–Hellman protocol. In: Cramer, R. (ed.) Advances in Cryptology—CRYPTO 2005, vol. 3621 of LNCS, pp. 546–566. Springer, Berlin (2005)
17. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) Provable Security: First International Conference, ProvSec 2007, vol. 4784 of LNCS, pp. 1–16. Springer, Berlin (2007)
18. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.A.: An efficient protocol for authenticated key agreement. Des. Codes Cryptogr. **28**(2), 119–134 (2003)
19. McCullagh, N., Barreto, P.S.L.M.: A new two-party identity-based authenticated key agreement. In: Menezes, A. (ed.) Topics in Cryptology—CT-RSA 2005, vol. 3376 of LNCS, pp. 262–274. Springer, Berlin (2005)
20. Okamoto, E., Tanaka, K.: Key distribution system based on identification information. IEEE J. Sel. Areas Commun. **7**(4), 481–485 (1989)
21. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
22. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology—CRYPTO 84, vol. 196 of LNCS, pp. 47–53. Springer, Berlin (1984)
23. Smart, N.P.: Identity-based authenticated key agreement protocol based on weil pairing. IET Electron. Lett. **38**(13), 630–632 (2002)
24. Ustaoğlu, B.: Comparing *SessionStateReveal* and *EphemeralKeyReveal* for Diffie–Hellman protocols. In: Pieprzyk, J., Zhang, F. (eds.) Provable Security: Third International Conference, ProvSec 2009, vol. 5848 of LNCS, pp. 183–197. Springer, Berlin (2009)
25. Wang, Y.: Efficient identity-based and authenticated key agreement protocol. Cryptology ePrint Archive, Report 2005/108 (2005)
26. Xie, G.: An ID-based key agreement scheme from pairing. Cryptology ePrint Archive, Report 2005/093 (2005)
27. Yuan, Q., Li, S.: A new efficient ID-based authenticated key agreement protocol. Cryptology ePrint Archive, Report 2005/309 (2005)