PAPER Special Section on Discrete Mathematics and Its Applications

id-eCK Secure ID-Based Authenticated Key Exchange on Symmetric and Asymmetric Pairing**

Atsushi FUJIOKA[†*a)], ***Member***, **Fumitaka HOSHINO**[†b)], ***Nonmember***, **Tetsutaro KOBAYASHI**[†c)], **Koutarou SUZUKI**[†d)], ***Members***, **Berkant USTAOĞLU**[††e)], ***Nonmember***, ***and*** **Kazuki YONEYAMA**[†f)], ***Member***

SUMMARY In this paper, we propose an identity-based authenticated key exchange (ID-AKE) protocol that is secure in the identity-based extended Canetti-Krawczyk (id-eCK) model in the random oracle model under the gap Bilinear Diffie-Hellman assumption. The proposed ID-AKE protocol is the most efficient among the existing ID-AKE protocols that is id-eCK secure, and it can be extended to use in asymmetric pairing.

key words: *identity-based authenticated key exchange, gap Bilinear Diffie-Hellman assumption, identity-based extended Canetti-Krawczyk model, random oracle model*

1. Introduction

Key exchange is one of the important cryptographic protocols since it can be used to establish secure channels. Recent progress in its research has reached a formulation, *authenticated key exchange* (AKE), where AKE enables two parties to share a key via a public communication channel, and both parties are assured that only their intended peers can derive the session key.

In ordinary AKE, public-key infrastructure (PKI) is necessary since each static public-key needs to be linked with the identity of each user to provide authenticity, and such AKE is called *PKI-based* AKE. It is natural to introduce an identity-based version of AKE, called *identity-based authenticated key exchange* (ID-AKE). In order to avoid such requirement like a PKI system, a *key generation center* (KGC) in an ID-AKE protocol generates a pair of master public and secret keys, and the KGC extracts each user's secret key corresponding to the user's identity. Every user has own secret key and can use the identity as public information. Then, a party, called *initiator*, who wants to share a key with another party, called *responder*, sends ephemeral public information to the responder, the responder sends back another ephemeral public information to the initiator. This type of ID-AKE is called *two-pass* ID-AKE. Each party generates the session key from the master public key, own secret key given by the KGC, own secret values of ephemeral information, the peer's identity, and the received ephemeral information.

The several security models of ID-AKE have been investigated, and they are influenced by the security models of PKI-based AKE. For PKI-based AKE, the Canetti-Krawczyk (CK) model [11] and the extended Canetti-Krawczyk (eCK) model [26] followed the Bellare-Rogaway (BR) model [6], which is the first formal security model for AKE. The security notions defined in the above models are given as an indistinguishability game, where an adversary is required to differentiate between a random key and a session key of a session. The session is called *target session*, and is chosen by the adversary. Based on these models, the id-BR*** [8], [13], id-CK [9], and id-eCK [25] models were defined, respectively. Note that in ID-AKE, although the KGC has much power than users, any session key between users should not be revealed even to the KGC. This property is called *forward secrecy against KGC* (KGC-FS). The security and the eCK security are stronger than the BR security [11], [15], [26], however, the CK security and the eCK security are incompatible [16], [17]. These relations hold in the security definitions of ID-AKE, also.

We adopt the id-eCK model since it captures session key indistinguishability under the leakage of all non-trivial combination of master, static, and ephemeral secret keys.

Most ID-AKE protocols [21], [22], [25] are realized by using *pairing* as an important element. Pairing is a function from two groups, G_1 and G_2, to another group, G_T, and pairing functions are classified into two types: *symmetric* pairing and *asymmetric* pairing. The domain groups, G_1 and G_2, in symmetric pairings are the same (i.e., $G_1 = G_2$), however, those in asymmetric pairing are different (i.e., $G_1 \neq G_2$). The ID-AKE protocols in [21], [22], [25] are using symmetric pairing. Tate pairing on super-singular elliptic curves and η_T pairing are examples of symmetric pairing, and Tate pairing on MNT curves [31], Tate pairing on BN curves [32] and Ate pairing [24] are those of asymmetric pairing. Recent progress in attacks against super-singular elliptic curves in addition to researches of R-ate pairing [30] on BN curves

Manuscript received September 24, 2012.
Manuscript revised January 7, 2013.

[†]The authors are with NTT Secure Platform Laboratories, NTT Corporation, Musashino-shi, 180-8585 Japan.
[††]The author is with the IYTE, Turkey.
[*]Presently, with Kanagawa University, Japan.

[**]A part of this work is appeared in [22], and was done while the first and the fifth authors were with NTT Laboratories.

a) E-mail: fujioka@kanagawa-u.ac.jp
b) E-mail: hoshino.fumitaka@lab.ntt.co.jp
c) E-mail: kobayashi.tetsutaro@lab.ntt.co.jp
d) E-mail: suzuki.koutarou@lab.ntt.co.jp
e) E-mail: bustaoglu@uwaterloo.ca
f) E-mail: yoneyama.kazuki@lab.ntt.co.jp

DOI: 10.1587/transfun.E96.A.1139

[***]The BR model is defined in symmetric key setting, and the model in public key setting was proposed by Blake-Wilson, Johnson, and Menezes [7].

and Optimal-ate pairing [34] will increase significance of asymmetric pairing in near future.

In this paper, we propose an identity-based authenticated key exchange (ID-AKE) protocol Π^{sym} that is secure in the identity-based extended Canetti-Krawczyk (id-eCK) model in the random oracle model under the gap Bilinear Diffie-Hellman (gap BDH) assumption using symmetric pairing. The proposed ID-AKE protocol Π^{sym} using symmetric pairing is the most efficient among the existing ID-AKE protocol that is id-eCK secure (see Sect. 6.2 for details).

Based on the protocol Π^{sym}, we propose an ID-AKE protocol Π^{asym} that is secure in id-eCK model in the random oracle model under the gap BDH assumption using asymmetric pairing. The proposed ID-AKE protocol Π^{asym} using asymmetric pairing is the first id-eCK secure ID-AKE protocol using asymmetric pairing (see Sect. 6.2 for details).

1.1 Related Works

In a literature, many ID-AKE protocols secure in the id-BR model have been investigated. (See survey papers for them [8], [13].)

However, in PKI-based AKE, the BR model does not consider exposure of secret information, and then, the CK model was formulated to treat such situations. The CK model considers leakage of information in sessions, and the adversary in the CK model is allowed to access the state information, named *session state*. The session state contains not only the ephemeral secret key but also internal values computed with the static secret key, however, the adversary is not allowed to access the static secret key itself. The same discussion can be applied to the id-BR and id-CK models, and several id-CK secure protocols [9], [19] have followed the first attempt by Chow and Choo [14]. It is worthy to note that it is complicated to precisely define state information in the (id-)CK model.

For (ID-)AKE, several security notions have been proposed in addition to the (id-)CK security: The *weak Perfect Forward Secrecy* (wPFS) implies that an adversary cannot recover an already established session key before the compromise of static secret keys. The resilience to *Key Compromised Impersonation* (KCI) means that given a static secret key an adversary tries to impersonate some honest party against the owner of the leaked secret key. The resilience to *Reflection Attack* (REF) implies that the indistinguishability test must be passed even when the adversary is allowed to make a session between the same party. The resilience to *Maximal EXposure attack* (MEX) means that an adversary tries to distinguish the session key from a random value under the disclosure of any pair of secret static keys and ephemeral secret keys of the initiator and the responder in the session except both static and ephemeral secret key of the initiator or the responder.

Regarding to the id-eCK security, Huang and Cao defined the model, and proposed an id-eCK secure ID-AKE protocol using symmetric pairing [25]. The extended Canetti-Krawczyk (eCK) model for AKE was defined by LaMacchia, Lauter, and Mityagin [26], and roughly speaking, the eCK security assures that any adversary cannot distinguish the session key of a target session from a random value even when the adversary is allowed to access either the static secret key or the ephemeral secret key of each party establishing the target session, and the similar assurance is achieved in the id-eCK model.

Fujioka, Suzuki, and Ustaoğlu [22] proposed a shared security model for ID-AKE, where two different protocols using symmetric pairing can be securely carried out even when the identity and the secret key are shared in these protocols. They adopted and modified the id-eCK model, and proved that their proposed protocols are secure in both id-eCK and shared security models.

For ID-AKE, it is required that the KGC-FS security should be ensured, also, and the id-eCK security guarantees the all previous security notions: wPFS, KGC-FS, KCI, REF, and MEX.

Fujioka and Suzuki [21] provide a sufficient condition for ID-AKE security in the id-eCK model introducing the notion of admissible polynomials proposed in [20]. The resulting ID-AKE protocols use symmetric pairing, also.

1.2 Organization

In Sect. 2, we provide preliminaries. In Sect. 3, we describe the id-eCK security model for ID-AKE. In Sect. 4, we propose an id-eCK secure ID-AKE protocol using symmetric pairing. In Sect. 5, we extend it to the case using asymmetric pairing. In Sect. 6, we provide comparison with existing protocols and discuss about twin DH technique. In Sect. 7, we conclude the paper. In Appendix, we provide the security proof.

2. Preliminaries

For the security of the proposed protocol Π^{sym} with symmetric pairing, we need the gap Bilinear Diffie-Hellman (gap BDH) assumption described below, which is introduced by [27] and used [2], [28], [29]. Let κ be the security parameter. Let G and G_T be cyclic groups with generator g and $g_T = e(g, g)$ and of order κ-bit prime q, and $e : G \times G \to G_T$ be symmetric pairing, i.e., computable bilinear non-degenerate map. The computational BDH function $\mathrm{BCDH} : G^3 \to G_T$ is $\mathrm{BCDH}(g^u, g^v, g^w) = e(g, g)^{uvw}$, and the decisional BDH predicate $\mathrm{BDDH} : G^3 \times G_T \to \{0, 1\}$ is a function which takes an input $(g^u, g^v, g^w, e(g, g)^x)$ and returns the bit 1 if $uvw = x \bmod q$ and the bit 0 otherwise. An adversary $\mathcal{A}$ is given input $g^u, g^v, g^w \in G$, where $u, v, w \in_U \mathbb{Z}_q$ is selected uniformly random, and oracle access to $\mathrm{BDDH}(\cdot, \cdot, \cdot, \cdot)$ oracle, and tries to compute $\mathrm{BCDH}(g^u, g^v, g^w)$. For adversary $\mathcal{A}$, we define advantage

$$Adv^{\mathrm{gapBDH}}(\mathcal{A}) = \Pr[u, v, w \in_U \mathbb{Z}_q, \mathcal{A}^{\mathrm{BDDH}(\cdot,\cdot,\cdot,\cdot)}(g^u, g^v, g^w) = \mathrm{BCDH}(g^u, g^v, g^w)],$$

where the probability is taken over the choices of u, v, w and

$\mathcal{A}$'s random tape.

Definition 1 (gap BDH assumption): We say that G, G_T satisfy the gap BDH assumption if, for all polynomial-time adversaries $\mathcal{A}$, advantage $Adv^{\text{gapBDH}}(\mathcal{A})$ is negligible in security parameter κ.

For the security of the proposed protocol Π^{asym} with asymmetric pairing, we also need the asymmetric gap Bilinear Diffie-Hellman (asymmetric gap BDH) assumption described below. Let κ be the security parameter. Let G_1, G_2, and G_T be cyclic groups with generator g_1, g_2, and $g_T = \hat{e}(g_1, g_2)$ and of order κ-bit prime q, and $\hat{e} : G_1 \times G_2 \to G_T$ be asymmetric pairing, i.e., computable bilinear non-degenerate map. The computational BDH function BCDH : $G_1^3 \times G_2^3 \to G_T$ is $\text{BCDH}(g_1^u, g_1^v, g_1^w, g_2^u, g_2^v, g_2^w) = e(g_1, g_2)^{uvw}$, and the decisional BDH predicate $\text{BDDH}^{a,b,c} : G_a \times G_b \times G_c \times G_T \to \{0, 1\}$ is a function which takes an input $(g_a^u, g_b^v, g_c^w, e(g_1, g_2)^x)$ $(a, b, c = 1, 2)$ and returns the bit 1 if $uvw = x \bmod q$ and the bit 0 otherwise. Let $\phi \neq I_1, I_2, I_3 \subset \{1, 2\}$ be the sets of indices of input for adversary $\mathcal{A}$ and $\phi \neq O \subset \{1, 2\}^3$ be the sets of indices of BDDH oracle for adversary $\mathcal{A}$. The adversary $\mathcal{A}$ is given input $\{g_a^u\}_{a \in I_1}$, $\{g_b^v\}_{b \in I_2}$, and $\{g_c^w\}_{c \in I_3}$ where $u, v, w \in_U \mathbb{Z}_q$ is selected uniformly random, and accesses oracles $\{\text{BDDH}^{a,b,c}(\cdot, \cdot, \cdot, \cdot)\}_{(a,b,c) \in O}$, and tries to compute $\text{BCDH}(g_1^u, g_1^v, g_1^w, g_2^u, g_2^v, g_2^w)$. For adversary $\mathcal{A}$, we define advantage

$$Adv^{\text{asymgapBDH}}(\mathcal{A}) = \Pr[u, v, w \in_U \mathbb{Z}_q, \mathcal{A}^{\{\text{BDDH}^{a,b,c}(\cdot,\cdot,\cdot,\cdot)\}_{(a,b,c) \in O}}(\{g_a^u\}_{a \in I_1}, \{g_b^v\}_{b \in I_2}, \{g_c^w\}_{c \in I_3}) = \text{BCDH}(g_1^u, g_1^v, g_1^w, g_2^u, g_2^v, g_2^w)],$$

where the probability is taken over the choices of u, v, w and $\mathcal{A}$'s random tape.

Definition 2 (asymmetric gap BDH assumption): We say that G_1, G_2, G_T satisfy the asymmetric gap BDH assumption with I_1, I_2, I_3, O if, for all polynomial-time adversaries $\mathcal{A}$, advantage $Adv^{\text{asymgapBDH}}(\mathcal{A})$ is negligible in security parameter κ.

Notice that the asymmetric gap BDH assumption is identical to the (symmetric) gap BDH assumption if $G_1 = G_2$.

For the security of the proposed protocol Π^{asym} with asymmetric pairing, the asymmetric gap BDH assumption with $I_1 = \{1, 2\}$, $I_2 = \{1, 2\}$, $I_3 = \{1, 2\}$, $O = \{(1, 1, 2)\}$ is required.

In ID-based AKE protocols Smart-1, Smart-2, and CK, the asymmetric gap BDH assumption is used for their security proofs [13].

Finally, we describe how to construct hash functions to groups G_1 and G_2 (see Sect. 2 of [13] for details). In this paper, we use Type 3 pairing described in Sect. 2 of [13], where we can compute hash functions to groups G_1 and G_2. In the setting of Type 3 pairing, group $E[q]$ of points of order q is decomposed as product of $G_1 \subset E(\mathbb{F}_p)$ and $G_2 = Ker(Tr) \subset E(\mathbb{F}_{p^k})$, i.e., G_2 is kernel of trace map Tr. We denote based point of G_1 and G_2 as P_1 and P_2. Notice that, in the setting of Type 3 pairing, we can compute projection of $Q = aP_1 + bP_2$ by $aP_1 = 1/k \cdot Tr(Q) \in G_1$ and $bP_2 = Q - aP_1 \in G_2$.

We can compute hash function to G_1 or G_2 as follows: 1. we generate random $x \in \mathbb{F}_{p^k}$ by applying hash function to the input and a counter and obtain $(x, y) \in E(\mathbb{F}_{p^k})$ by solving the defining equation of E (if there is no solution, increment the counter and retry), 2. by multiplying cofactor c, i.e., $\#E(\mathbb{F}_{p^k}) = q^2 c$, to the point $(x, y) \in E(\mathbb{F}_{p^k})$, we obtain $Q \in E[q] = G_1 \times G_2$, 3. by applying the above projection map to group G_1 or G_2, we have the hash value in group G_1 or G_2.

For general curve parameters, step 1 requires the square root operation whose const is comparable to scalar multiplication, step 2 requires the scalar multiplication in extension fields, and step 3 requires the computation of Frobenius map whose const is almost free comapring with scalar multiplication. For special tuned parameter, the computational consts of hashing is almost same as of scalar multiplication (see Table 5 in [13]).

3. Security Model for ID-Based AKE

We recall the id-eCK security model for ID-AKE by Huang and Cao [25] that is the ID-based version of the eCK security model by LaMacchia, Lauter and Mityagin [26].

We denote a party by U_i and the identifier of U_i by ID_i. We outline our model for two-pass ID-AKE protocol, where parties U_A and U_B exchange ephemeral public keys X_A and X_B, i.e., U_A sends X_A to U_B and U_B sends X_B to U_A, and thereafter compute a session key. The session key depends on the exchanged ephemeral keys, identities of the parties, the static keys corresponding to these identities and the protocol instance that is used.

In the model, each party is a probabilistic polynomial-time Turing machine in security parameter κ and obtains a static secret key corresponding to its identity string from a key generation center (KGC) via a secure and authenticated channel. The KGC uses a master secret key to generate individual secret keys.

Session.

An invocation of a protocol is called a *session*. A session is activated via an incoming message of the forms $(\Pi, \mathcal{I}, ID_A, ID_B)$ or $(\Pi, \mathcal{R}, ID_A, ID_B, X_B)$, where Π is a protocol identifier. If U_A was activated with $(\Pi, \mathcal{I}, ID_A, ID_B)$, then U_A is the session *initiator*, otherwise the session *responder*. After activation, U_A appends an ephemeral public key X_A to the incoming message and sends it as an outgoing response. If U_A is the responder, U_A computes a session key. If U_A is the initiator, U_A that has been successfully activated via $(\Pi, \mathcal{I}, ID_A, ID_B)$ can be further activated via $(\Pi, \mathcal{R}, ID_A, ID_B, X_A, X_B)$ to compute a session key.

If U_A is the initiator, the session is identified via $(\Pi, \mathcal{I}, ID_A, ID_B, X_A, \times)$ or $(\Pi, \mathcal{I}, ID_A, ID_B, X_A, X_B)$. If U_A is the responder, the session is identified via $(\Pi, \mathcal{R}, ID_A, ID_B, X_B, X_A)$.

We say that U_A is *owner* of session `sid` if the third coordinate of session `sid` is ID_A. We say that U_A is *peer* of

session sid if the fourth coordinate of session sid is ID_A. We say that a session is *completed* if its owner computes a session key. For session $(\Pi, \mathcal{I}, ID_A, ID_B, X_A, X_B)$ the *matching session* has identifier $(\Pi, \mathcal{R}, ID_B, ID_A, X_A, X_B)$ and vice versa.

From now on we omit $\mathcal{I}$ and $\mathcal{R}$ since these "role markers" are implicitly defined by the order of X_A and X_B.

Adversary.

The adversary $\mathcal{A}$ is modeled as a probabilistic Turing machine that controls all communications between parties including session activation, performed via a Send(message) query. The message has one of the following forms: (Π, ID_A, ID_B), (Π, ID_A, ID_B, X_A), or $(\Pi, ID_A, ID_B, X_A, X_B)$. Each party submits its responses to the adversary, who decides the global delivery order. Note that the adversary does not control the communication between each party and the key generation center.

A party's secret information is not accessible to the adversary, however, leakage of secret information is captured via the following adversary queries.

- SessionKeyReveal(sid) The adversary obtains the session key for the session sid, provided that the session holds a session key.
- EphemeralKeyReveal(sid) The adversary obtains the ephemeral secret key associated with the session sid.
- StaticKeyReveal(ID_i) The adversary learns the static secret key of party U_i.
- MasterKeyReveal() The adversary learns the master secret key of the KGC.
- EstablishParty(ID_i) This query allows the adversary to register ID_i on behalf of a party U_i, the adversary totally controls that party. If a party is established by an EstablishParty(ID_i) query issued by the adversary, then we call the party *dishonest*, and if not, we call the party *honest*. This query models malicious insiders.

Freshness. Our security definition requires the notion of "freshness".

Definition 3 (Freshness): Let $\mathtt{sid}^*$ be the session identifier of a completed session, owned by an honest party U_A with peer U_B, who is also honest. If the matching session exists, then let $\overline{\mathtt{sid}^*}$ be the session identifier of the matching session of $\mathtt{sid}^*$. Define $\mathtt{sid}^*$ to be fresh if none of the following conditions hold:

1. $\mathcal{A}$ issues SessionKeyReveal($\mathtt{sid}^*$) or SessionKeyReveal($\overline{\mathtt{sid}^*}$) (if $\overline{\mathtt{sid}^*}$ exists).
2. $\overline{\mathtt{sid}^*}$ exists and $\mathcal{A}$ makes either of the following queries
   - both StaticKeyReveal(ID_A) and EphemeralKeyReveal($\mathtt{sid}^*$), or
   - both StaticKeyReveal(ID_B) and EphemeralKeyReveal($\overline{\mathtt{sid}^*}$).
3. $\overline{\mathtt{sid}^*}$ does not exist and $\mathcal{A}$ makes either of the following queries
   - both StaticKeyReveal(ID_A) and EphemeralKeyReveal($\mathtt{sid}^*$), or
   - StaticKeyReveal(ID_B).

Note that if $\mathcal{A}$ issues MasterKeyReveal(), we regard $\mathcal{A}$ as having issued both StaticKeyReveal(ID_A) and StaticKeyReveal(ID_B).

Security Experiment. The adversary $\mathcal{A}$ starts with a set of honest parties, for whom $\mathcal{A}$ adaptively selects identifiers. The adversary makes an arbitrary sequence of the queries described above. During the experiment, $\mathcal{A}$ makes a special query Test($\mathtt{sid}^*$) and is given with equal probability either the session key held by $\mathtt{sid}^*$ or a random key. The experiment continues until $\mathcal{A}$ makes a guess whether the key is random or not. The adversary *wins* the game if the test session $\mathtt{sid}^*$ is fresh at the end of $\mathcal{A}$'s execution and if $\mathcal{A}$'s guess was correct.

Definition 4 (security): The advantage of the adversary $\mathcal{A}$ in the experiment with ID-AKE protocol Π is defined as

$$\mathrm{Adv}_{\Pi}^{\text{ID-AKE}}(\mathcal{A}) = \Pr[\mathcal{A}\ wins] - \frac{1}{2}.$$

We say that Π is secure ID-AKE protocols in the id-eCK model if the following conditions hold.

1. If two honest parties complete matching session, then, except with negligible probability in security parameter κ, they both compute the same session key.
2. For any probabilistic polynomial-time bounded adversary $\mathcal{A}$, $\mathrm{Adv}_{\Pi}^{\text{ID-AKE}}(\mathcal{A})$ is negligible in security parameter κ.

4. Proposed ID-Based AKE Protocol Using Symmetric Pairing

4.1 Proposed ID-Based AKE Protocol

We propose an ID-AKE protocol, Π^{sym}, and prove in Theorem 5 that Π^{sym} is id-eCK secure in the random oracle model under the gap BDH assumption.

The proposed ID-AKE protocol, Π^{sym}, is described as follows: Let κ be the security parameter. Let G and G_T be cyclic groups with generators g and $g_T = e(g, g)$ both of order κ-bit prime q, respectively, and $e : G \times G \to G_T$ be symmetric pairing. Let $H : \{0,1\}^* \to \{0,1\}^\kappa$ and $H_1 : \{0,1\}^* \to G$ be cryptographic hash functions modeled as random oracles. Let Π be the protocol identifier of protocol Π^{sym}.

KGC randomly selects master secret key $z \in \mathbb{Z}_q$, and publishes master public key $Z = g^z \in G$. Z is provided as a part of the system parameters.

User U_i with identity ID_i is assigned static secret key $D_i = Q_i^z \in G$, where $Q_i = H_1(ID_i) = g^{q_i} \in G$.

Thus, U_A's identity and static secret key are ID_A and $D_A = Q_A^z = H_1(ID_A)^z = g^{zq_A} \in G$, and U_B's identity and static secret key are ID_B and $D_B = Q_B^z = H_1(ID_B)^z = g^{zq_B} \in G$.

In the description, user U_A with static secret key D_A is the session initiator and user U_B with static secret key D_B is the session responder.

1. U_A selects a random ephemeral secret key $x_A \in_U \mathbb{Z}_q$, computes the ephemeral public key $X_A = g^{x_A}$, and sends (Π, ID_B, ID_A, X_A) to U_B.
2. Upon receiving (Π, ID_B, ID_A, X_A), U_B selects a random ephemeral secret key $x_B \in_U \mathbb{Z}_q$, computes the ephemeral public key $X_B = g^{x_B}$, and sends $(\Pi, ID_A, ID_B, X_A, X_B)$ to U_A.
   U_B computes shared values
   $$\sigma_1 = e(Q_A, D_B),\ \sigma_2 = e(Q_A X_A, D_B Z^{x_B}),$$
   $$\sigma_3 = X_A^{x_B}$$
   computes the session key $K = H(\sigma_1, \sigma_2, \sigma_3, \Pi, ID_A, ID_B, X_A, X_B)$, and completes the session.
3. Upon receiving $(\Pi, ID_A, ID_B, X_A, X_B)$, U_A checks whether U_A has sent (Π, ID_B, ID_A, X_A) to U_B or not, and aborts the session if not.
   U_A computes shared values
   $$\sigma_1 = e(D_A, Q_B),\ \sigma_2 = e(D_A Z^{x_A}, Q_B X_B),$$
   $$\sigma_3 = X_B^{x_A}$$
   computes the session key $K = H(\sigma_1, \sigma_2, \sigma_3, \Pi, ID_A, ID_B, X_A, X_B)$, and completes the session.

   Both parties compute the same shared values

$$\sigma_1 = g_T^{z q_A q_B},\ \sigma_2 = g_T^{z(q_A + x_A)(q_B + x_B)},$$
$$\sigma_3 = g^{x_A x_B}$$

and compute the same session key K.

Protocol Π^{sym} requires 2 pairing operations, 3 exponentiations (including the exponentiation for the ephemeral public key), and 3 shared values. The protocol has been proposed as protocol Π_2 in [22], and, to the best of our knowledge, is the most efficient id-eCK secure ID-AKE protocol based on pairing[†]. Only known id-eCK secure ID-AKE protocol based on pairing, except ones in [22], is the protocol in [25], and it requires 2 static keys while protocol Π^{sym} requires 1 static key. Other computational properties regarding numbers of the pairing operation, the exponentiation, and the shared secrets are equivalent.

4.2 Security

The proposed ID-AKE protocol is secure in the id-eCK model [25] in the random oracle model under the gap BDH assumption.

Theorem 5: If G, G_T are groups where the gap BDH assumption holds, and H, H_1 are random oracles, the proposed ID-AKE protocol, Π^{sym}, is secure in the id-eCK model.

The proof of Theorem 5 is provided in Appendix A. We provide a intuitive discussion here.

Proof: (Sketch) The gap BDH solver $\mathcal{S}$ extracts the answer g_T^{uvw} of an instance $(U = g^u, V = g^v, W = g^w)$ of the gap BDH problem using adversary $\mathcal{A}$. For instance, we assume the case that test session $\mathtt{sid}^*$ has no matching session $\overline{\mathtt{sid}^*}$, adversary $\mathcal{A}$ is given D_A, and adversary $\mathcal{A}$ does not obtain x_A and D_B from the condition of freshness, i.e., event E_1 in Table A· 1. In this case, solver $\mathcal{S}$ selects random q_A and sets $Q_A = H_1(ID_A) = g^{q_A}$ and $D_A = Z^{q_A}$, embeds the instance as $Z = U\ (= g^u)$, $X_A = V\ (= g^v)$ and $Q_B = W\ (= g^w)$, and extracts $g_T^{z x_A q_B} = g_T^{uvw}$ from the shared values $\sigma_1, \sigma_2, \sigma_3$.

Solver $\mathcal{S}$ can perfectly simulate StaticKeyReveal query by selecting random q_i and setting $Q_i = H_1(ID_i) = g^{q_i}$ and $D_i = Z^{q_i}$. Solver $\mathcal{S}$ can perfectly simulate EphemeralKeyReveal query by selecting random x_i and setting $X_i = g^{x_i}$.

The solver $\mathcal{S}$ can extract the answer of the gap BDH instance as follows. By eliminating the terms including q_A using the knowledge of q_A, solver $\mathcal{S}$ can obtain

$$\sigma_2' = \sigma_2 / e(Z, Q_B X_B)^{q_A} = g_T^{z x_A (q_B + x_B)},$$
$$\sigma_3' = e(Z, \sigma_3) = g_T^{z x_A x_B},$$

By using this term, solver $\mathcal{S}$ can extract the answer $g_T^{z x_A q_B} = g_T^{uvw}$ as

$$\sigma_2' / \sigma_3' = g_T^{z x_A q_B}.$$

Notice that, in other cases in Table A· 1, the solver $\mathcal{S}$ can eliminate the terms including the specific secret key using the knowledge of the secret key, can obtain 2 linearly independent terms, and can extract the answer g_T^{uvw} by solving the linear equation.

The solver $\mathcal{S}$ can check whether the shared secrets are correctly formed w.r.t. static and ephemeral public keys, and can simulate H and SessionKeyReveal queries consistently. More precisely, in the simulation of the $H(\sigma_1, \sigma_2, \sigma_3, \Pi, ID_A, ID_B, X_A, X_B)$ query, solver $\mathcal{S}$ needs to check that the shared secrets $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, and if so return session key K being consistent with the previously answered SessionKeyReveal(Π, $\mathcal{I}$, ID_A, ID_B, X_A, X_B) and SessionKeyReveal(Π, $\mathcal{R}$, ID_B, ID_A, X_A, X_B) queries. The solver $\mathcal{S}$ can check whether shared secrets $\sigma_1, \sigma_2, \sigma_3$ are correctly formed w.r.t. the static and ephemeral public keys by asking BDDH oracle

$$\mathrm{BDDH}(Z, Q_A, Q_B, \sigma_1) = 1,$$
$$\mathrm{BDDH}(Z, Q_A X_A, Q_B X_B, \sigma_2) = 1,$$
$$e(X_A, X_B) = e(g, \sigma_3),$$

and this implies $\sigma_1, \sigma_2, \sigma_3$ are correctly formed.

Notice that, in other cases in Table A· 1, the solver $\mathcal{S}$ can check whether shared secrets $\sigma_1, \sigma_2, \sigma_3$ are correctly formed or not by the same procedure.

Finally, we assume the case that adversary $\mathcal{A}$ issues MasterKeyReveal query. In this case, solver $\mathcal{S}$ embeds the instance as $X_A = V\ (= g^v)$, $X_B = W\ (= g^w)$, and extracts answer $e(U, \sigma_3) = g_T^{u x_A x_B} = g_T^{uvw}$ of the gap BDH problem

[†]Note that protocol Π_1 in [22] is less efficient than protocol Π_2 (Π^{sym}) as Π_1 requires 5 exponentiations however Π_2 requires 3 exponentiations.

from the shared values $\sigma_3 = g^{x_A x_B}$.

□

5. Extension to Asymmetric Pairing Case

5.1 Proposed ID-Based AKE Protocol Using Asymmetric Pairing

We propose another ID-AKE protocol, Π^{asym}, using asymmetric pairing and give a theorem that Π^{asym} is id-eCK secure in the random oracle model under the asymmetric gap BDH assumption.

The proposed ID-AKE protocol, Π^{asym}, is described as follows: Let κ be the security parameter. Let G_1, G_2, and G_T be cyclic groups with generators g_1, g_2, and $g_T = \hat{e}(g_1, g_2)$ all of order κ-bit prime q, respectively, and $\hat{e} : G_1 \times G_2 \to G_T$ be asymmetric pairing. Let $H : \{0,1\}^* \to \{0,1\}^\kappa$, $H_1 : \{0,1\}^* \to G_1$, and $H_2 : \{0,1\}^* \to G_2$ be cryptographic hash functions modeled as random oracles. Let Π be the protocol identifier of protocol Π^{asym}.

KGC randomly selects master secret key $z \in \mathbb{Z}_q$, and publishes master public keys $Z_t = g_t^z \in G_t$ $(t = 1, 2)$. Z_1 and Z_2 are provided as a part of the system parameters.

User U_i with identity ID_i is assigned static secret key $D_{i,t} = Q_{i,t}^z \in G_t$ $(t = 1, 2)$, where $Q_{i,t} = H_t(ID_i) = g_t^{q_{i,t}} \in G_t$ $(t = 1, 2)$.

Thus, U_A's identity and static secret key are ID_A and $D_{A,t} = Q_{A,t}^z = H_t(ID_A)^z = g_t^{zq_{A,t}} \in G_t$ $(t = 1, 2)$, and U_B's identity and static secret key are ID_B and $D_{B,t} = Q_{B,t}^z = H_t(ID_B)^z = g_t^{zq_{B,t}} \in G_t$ $(t = 1, 2)$.

In the description, user U_A with static secret key $D_{A,1} \in G_1$ is the session initiator and user U_B with static secret key $D_{B,2} \in G_2$ is the session responder.

1. U_A selects a random ephemeral secret key $x_A \in_U \mathbb{Z}_q$, computes the ephemeral public key $X_{A,t} = g_t^{x_A}$ $(t = 1, 2)$, and sends $(\Pi, ID_B, ID_A, X_{A,1}, X_{A,2})$ to U_B.
2. Upon receiving $(\Pi, ID_B, ID_A, X_{A,1}, X_{A,2})$, U_B verifies $\hat{e}(X_{A,1}, g_2) = \hat{e}(g_1, X_{A,2})$. U_B selects a random ephemeral secret key $x_B \in_U \mathbb{Z}_q$, computes the ephemeral public key $X_{B,t} = g_t^{x_B}$ $(t = 1, 2)$, and sends $(\Pi, ID_A, ID_B, X_{A,1}, X_{A,2}, X_{B,1}, X_{B,2})$ to U_A.
   U_B computes shared values
   $$\sigma_1 = \hat{e}(Q_{A,1}, D_{B,2}),\ \sigma_2 = \hat{e}(Q_{A,1}X_{A,1}, D_{B,2}Z_2^{x_B}),$$
   $$\sigma_3 = X_{A,1}^{x_B},\ \sigma_4 = X_{A,2}^{x_B}$$
   computes the session key $K = H(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \Pi, ID_A, ID_B, X_{A,1}, X_{A,2}, X_{B,1}, X_{B,2})$, and completes the session.
3. Upon receiving $(\Pi, ID_A, ID_B, X_{A,1}, X_{A,2}, X_{B,1}, X_{B,2})$, U_A checks whether U_A has sent $(\Pi, ID_B, ID_A, X_{A,1}, X_{A,2})$ to U_B or not and verifies $\hat{e}(X_{B,1}, g_2) = \hat{e}(g_1, X_{B,2})$.
   U_A computes shared values
   $$\sigma_1 = \hat{e}(D_{A,1}, Q_{B,2}),\ \sigma_2 = \hat{e}(D_{A,1}Z_1^{x_A}, Q_{B,2}X_{B,2}),$$
   $$\sigma_3 = X_{B,1}^{x_A},\ \sigma_4 = X_{B,2}^{x_A}$$
   computes the session key $K = H(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \Pi, ID_A, ID_B, X_{A,1}, X_{A,2}, X_{B,1}, X_{B,2})$, and completes the session.

   Both parties compute the same shared values
   $$\sigma_1 = g_T^{zq_{A,1}q_{B,2}},\ \sigma_2 = g_T^{z(q_{A,1}+x_A)(q_{B,2}+x_B)},$$
   $$\sigma_3 = g_1^{x_A x_B},\ \sigma_4 = g_2^{x_A x_B}$$
   and compute the same session key K.

Protocol Π^{asym} requires 4 pairing operations, 2.5 exponentiations in G_1 and 2.5 exponentiations in G_2 (including the exponentiation for the ephemeral public key), and 4 shared values. Note that 2.5 means average, i.e., 3 exponentiations in G_1 and 2 exponentiations in G_2 are requited for U_A and 2 exponentiations in G_1 and 3 exponentiations in G_2 are requited for U_B. In asymmetric pairing, cost of exponentiation in G_1 becomes negligible asymptotically compared with that in G_2, so it could be concluded that protocol Π^{asym} requires 4 pairing operations and 2.5 exponentiations in G_2. This means that Π^{sym} is more efficient than Π^{asym} w.r.t. pairing operations and Π^{asym} is more efficient than Π^{sym} w.r.t. exponentiations.

5.2 Security

The proposed ID-AKE protocol is secure in the id-eCK model [25] in the random oracle model under the asymmetric gap BDH assumption.

Theorem 6: If G_1, G_2, G_T are groups where the asymmetric gap BDH assumption with $I_1 = \{1, 2\}$, $I_2 = \{1, 2\}$, $I_3 = \{1, 2\}$, $O = \{(1, 1, 2)\}$ holds, and H, H_1 are random oracles, the proposed ID-AKE protocol, Π^{asym}, is secure in the id-eCK model.

The proof of Theorem 6 is provided in Appendix B. We provide a intuitive discussion here.

Proof: (Sketch) The gap BDH solver $\mathcal{S}$ extracts the answer g_T^{uvw} of an instance $(U_1 = g_1^u, U_2 = g_2^u, V_1 = g_1^v, V_2 = g_2^v, W_1 = g_1^w, W_2 = g_2^w)$ of the asymmetric gap BDH problem using adversary $\mathcal{A}$. For instance, we assume the case that test session $\mathtt{sid}^*$ has no matching session $\overline{\mathtt{sid}^*}$, adversary $\mathcal{A}$ is given $D_{A,1} \in G_1$, and adversary $\mathcal{A}$ does not obtain x_A and $D_{B,2} \in G_2$ from the condition of freshness, i.e., event E_1 in Table A·1. In this case, solver $\mathcal{S}$ selects random $q_{A,1}$ and sets $Q_{A,1} = H_1(ID_A) = g_1^{q_{A,1}}$ and $D_{A,1} = Z^{q_{A,1}}$, embeds the instance as $Z_1 = U_1$ $(= g_1^u)$, $Z_2 = U_2$ $(= g_2^u)$, $X_{A,1} = V_1$ $(= g_1^v)$, $X_{A,2} = V_2$ $(= g_2^v)$, and $Q_{B,2} = W_2$ $(= g_2^w)$, and extracts $g_T^{zx_A q_{B,2}} = g_T^{uvw}$ from the shared values $\sigma_1, \sigma_2, \sigma_3, \sigma_4$.

Solver $\mathcal{S}$ can perfectly simulate StaticKeyReveal query by selecting random $q_{i,t}$ and setting $Q_{i,t} = H_t(ID_i) = g_t^{q_{i,t}}$ and $D_{i,t} = Z_t^{q_{i,t}}$ $(t = 1, 2)$. Solver $\mathcal{S}$ can perfectly simulate EphemeralKeyReveal query by selecting random x_i and setting $X_{i,t} = g_t^{x_i}$ $(t = 1, 2)$. In the simulation of Send query, solver $\mathcal{S}$ checks $\hat{e}(X_{i,1}, g_2) = \hat{e}(g_1, X_{i,2})$ and if not aborts the session and so the session is never completed.

The solver $\mathcal{S}$ can extract the answer of the gap BDH instance as follows. By eliminating the terms including $q_{A,1}$ using the knowledge of $q_{A,1}$, solver $\mathcal{S}$ can obtain

$$\sigma_2' = \sigma_2/\hat{e}(Z_1, Q_{B,2}X_{B,2})^{q_{A,1}} = g_T^{z x_A (q_{B,2}+x_B)},$$
$$\sigma_4' = \hat{e}(Z_1, \sigma_4) = g_T^{z x_A x_B},$$

By using this term, solver $\mathcal{S}$ can extract the answer $g_T^{z x_A q_{B,2}} = g_T^{uvw}$ as

$$\sigma_2'/\sigma_4' = g_T^{z x_A q_{B,2}}.$$

Notice that, in other cases in Table A· 1, the solver $\mathcal{S}$ can eliminate the terms including the specific secret key using the knowledge of the secret key, can obtain 2 linearly independent terms, and can extract the answer g_T^{uvw} by solving the linear equation.

The solver $\mathcal{S}$ can check whether the shared secrets are correctly formed w.r.t. static and ephemeral public keys, and can simulate H and SessionKeyReveal queries consistently. More precisely, in the simulation of the $H(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \Pi, ID_A, ID_B, X_A, X_B)$ query, solver $\mathcal{S}$ needs to check that the shared secrets $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed, and if so return session key K being consistent with the previously answered SessionKeyReveal(Π, $\mathcal{I}$, ID_A, ID_B, X_A, X_B) and SessionKeyReveal(Π, $\mathcal{R}$, ID_B, ID_A, X_A, X_B) queries. The solver $\mathcal{S}$ can check whether shared secrets $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed w.r.t. the static and ephemeral public keys by asking BDDH^{1,1,2} oracle

$$\text{BDDH}^{1,1,2}(Z_1, Q_{A,1}, Q_{B,2}, \sigma_1) = 1,$$
$$\text{BDDH}^{1,1,2}(Z_1, Q_{A,1}X_{A,1}, Q_{B,2}X_{B,2}, \sigma_2) = 1,$$
$$\hat{e}(X_{A,1}, g_2) = \hat{e}(g_1, X_{A,2}),\ \hat{e}(X_{B,1}, g_2) = \hat{e}(g_1, X_{B,2}),$$
$$\hat{e}(X_{A,1}, X_{B,2}) = \hat{e}(\sigma_3, g),\ \hat{e}(X_{A,1}, X_{B,2}) = \hat{e}(g, \sigma_4),$$

and this implies $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed.

Notice that, in other cases in Table A· 1, the solver $\mathcal{S}$ can check whether shared secrets $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed or not by the same procedure.

Finally, we assume the case that adversary $\mathcal{A}$ issues MasterKeyReveal query. In this case, solver $\mathcal{S}$ embeds the instance as $X_{A,1} = V_1$ $(= g_1^v)$, $X_{A,2} = V_2$ $(= g_2^v)$, $X_{B,1} = W_1$ $(= g_1^w)$, $X_{B,2} = W_2$ $(= g_2^w)$, and extracts answer $\hat{e}(U, \sigma_4) = g_T^{u x_A x_B} = g_T^{uvw}$ of the gap BDH problem from the shared values $\sigma_4 = g_2^{x_A x_B}$.

□

6. Discussions

6.1 Twin DH Technique

The proposed protocols Π^{sym} and Π^{asym} need the gap BDH assumption in order to prove security. We can extend the protocols to be based on just the BDH assumption (i.e., the simulator does not need the BDDH oracle) with the *twin DH* technique [12]. The twin DH technique enables the simulator to be able to check the correspondence between static and ephemeral public keys and shared values with the trapdoor test. We show the case of the protocol Π^{asym} using asymmetric pairings in the following.

We recall the asymmetric version of the trapdoor test in [25].

Lemma 7 (asymmetric version of Theorem 1 in [25]): Suppose $W \in G_1$ and $r, s \in \mathbb{Z}_q$ are mutually independent random variables and $r, s \in \mathbb{Z}_q$ are uniformly distributed over $\mathbb{Z}_q$. Define the random variable $W' = g_1^s/W^r \in G_1$. Furthermore, suppose that $X \in G_1$ and $Y \in G_2$ are random variables, and $\sigma_1, \sigma_2 \in G_T$ are random variables, each of which is defined as some function of W, W'. Then we have

1. W' is uniformly distributed over G_1,
2. W and W' are independent,
3. if $W = g_1^w$ and $W' = g_1^{w'}$, then the truth value of $\sigma_1^r \cdot \sigma_2 \stackrel{?}{=} \hat{e}(X, Y)^s$ agrees with the truth value of $\sigma_1 \stackrel{?}{=} \hat{e}(X, Y)^w \wedge \sigma_2 \stackrel{?}{=} \hat{e}(X, Y)^{w'}$ except probability $1/q$.

Thus, we can verify $\sigma_1 \stackrel{?}{=} \hat{e}(X, Y)^w$ without knowing the value w with σ_2, r, s, and we can implement the BDDH^{1,1,2} oracle, which on input $(X = g_1^x, W = g_1^w, Y = g_2^y, \sigma_1 = \hat{e}(g_1, g_2)^r)$ returns bit 1 if $xwy = r$ and bit 0 otherwise.

The extended protocol based on the twin DH technique is different from the proposed protocols Π^{sym} and Π^{asym} in the generation of static keys and shared values. We only describe the difference from the case of Π^{asym} in the following.

Let $H_1, H_1' : \{0,1\}^* \to G_1$, and $H_2, H_2' : \{0,1\}^* \to G_2$ be cryptographic hash functions modeled as random oracles.

User U_i with identity ID_i is assigned static secret key $D_{i,t} = (Q_{i,t})^z \in G_t$ $(t = 1, 2)$ and $D'_{i,t} = (Q'_{i,t})^z \in G_t$ $(t = 1, 2)$, where $Q_{i,t} = H_t(ID_i) = g_t^{q_{i,t}} \in G_t$ $(t = 1, 2)$ and $Q'_{i,t} = H'_t(ID_i) = g_t^{q'_{i,t}} \in G_t$ $(t = 1, 2)$.

User U_A with static secret key $D_{A,1}, D'_{A,1} \in G_1$ is the session initiator and user U_B with static secret key $D_{B,2}, D'_{B,2} \in G_2$ is the session responder, in a session of key exchange.

In the session, U_B computes shared values

$$\sigma_{1,1} = \hat{e}(Q_{A,1}, D_{B,2}),$$
$$\sigma_{1,2} = \hat{e}(Q'_{A,1}, D_{B,2}),$$
$$\sigma_{1,3} = \hat{e}(Q_{A,1}, D'_{B,2}),$$
$$\sigma_{1,4} = \hat{e}(Q'_{A,1}, D'_{B,2}),$$
$$\sigma_{2,1} = \hat{e}(Q_{A,1}X_{A,1}, D_{B,2}Z_2^{x_B}),$$
$$\sigma_{2,2} = \hat{e}(Q'_{A,1}X_{A,1}, D_{B,2}Z_2^{x_B}),$$
$$\sigma_{2,3} = \hat{e}(Q_{A,1}X_{A,1}, D'_{B,2}Z_2^{x_B}),$$
$$\sigma_{2,4} = \hat{e}(Q'_{A,1}X_{A,1}, D'_{B,2}Z_2^{x_B}),$$
$$\sigma_3 = X_{A,1}^{x_B},\ \sigma_4 = X_{A,2}^{x_B}$$

computes the session key $K = H(\sigma_{1,1}, ..., \sigma_{1,4}, \sigma_{2,1}, ..., \sigma_{2,4}, \sigma_3, \sigma_4, \Pi, ID_A, ID_B, X_{A,1}, X_{A,2}, X_{B,1}, X_{B,2})$.

In the session, U_A computes shared values

$$\sigma_{1,1} = \hat{e}(D_{A,1}, Q_{B,2}),$$
$$\sigma_{1,2} = \hat{e}(D'_{A,1}, Q_{B,2}),$$
$$\sigma_{1,3} = \hat{e}(D_{A,1}, Q'_{B,2}),$$
$$\sigma_{1,4} = \hat{e}(D'_{A,1}, Q'_{B,2}),$$
$$\sigma_{2,1} = \hat{e}(D_{A,1}Z_1^{x_A}, Q_{B,2}X_{B,2}),$$
$$\sigma_{2,2} = \hat{e}(D'_{A,1}Z_1^{x_A}, Q_{B,2}X_{B,2}),$$

Table 1 Comparison with the existing protocols.

Protocol	Computation	Storage	Commu.	Achieved Security Level	Assumption	Pairing Type	Bit Length
SCK [13]	2P+3E+1H	1	1	id-BR,KCI,mFS	BDH	Asym/Sym	256
SYL [13]	1P+3E+1H	1	1	id-BR,KCI,mFS	BDH	Sym	512
FG [19]	0P+4E	2	2	id-CK,KCI,mFS	strong DH	not required	256
PKI+Schnorr	0P+5E	1	4	–	–	not required	256
PKI+BLS	2P+3E+1H	1	3	–	–	Asym/Sym	256
HC [25]	2P+3E+2H	2	1	id-eCK	BDH	Sym	512
Π^{sym} (FSU [22])	2P+3E+1H	1	1	id-eCK	gap BDH	Sym	512
Π^{asym}	4P+5E+1H	2	2	id-eCK	gap BDH	Asym/Sym	256
Π^{sym}+TwinDH	4P+4E+2H	2	1	id-eCK	BDH	Sym	512
Π^{asym}+TwinDH	10P+8E+2H	4	2	id-eCK	BDH	Asym/Sym	256

The number of pairing computations, the number of exponentiations in G, and the number of hashing to G are denited by "P", "E", and "H". (The computational costs include the computation of the ephemeral public key.) The length of stored static secret keys in terms of group elements and the length of communicated message in terms of group elements are denoted by "Storage" and "Commu.".
In the column of Achieved Security Level, "id-BR", "id-CK", and "id-eCK" denotes ID-based versions of the Bellare-Rogaway [6] (BR), Canetti-Krawczyk [11] (CK), and LaMacchia, Lauter and Mityagin [26] (eCK) security models, respectively. "KCI" denotes key compromise impersonation resistance and "mFS" denotes master secret key forward security.
In the column of Pairing Type, "Asym/Sym" means that the protocol can be realized using both asymmetric and symmetric pairing, and "Sym" means that the protocol can be realized using only symmetric pairing. "not required" means pairing is not required for the protocol.
In the column of Bit Length, the bit length of one group element of elliptic curve without pairing, elliptic curve with asymmetric pairing, and elliptic curve with symmetric pairing for 2^{128} security.

$$\sigma_{2,3} = \hat{e}(D_{A,1}Z_1^{x_A}, Q'_{B,2}X_{B,2}),$$
$$\sigma_{2,4} = \hat{e}(D'_{A,1}Z_1^{x_A}, Q'_{B,2}X_{B,2}),$$
$$\sigma_3 = X_{B,1}^{x_A},\ \sigma_4 = X_{B,2}^{x_A}$$

computes the session key $K = H(\sigma_{1,1}, ..., \sigma_{1,4}, \sigma_{2,1}, ..., \sigma_{2,4}, \sigma_3, \sigma_4, \Pi, ID_A, ID_B, X_{A,1}, X_{A,2}, X_{B,1}, X_{B,2})$.

In the security proof, the solver $\mathcal{S}$ with an instance $(U_1 = g_1^u, U_2 = g_2^u, V_1 = g_1^v, V_2 = g_2^v, W_1 = g_1^w, W_2 = g_2^w)$ of the asymmetric BDH problem checks BDH relation using twin DH technique as follows. For instance, we assume the case that test session $\mathtt{sid}^*$ has no matching session $\overline{\mathtt{sid}^*}$, adversary $\mathcal{A}$ is given $x_A \in \mathbb{Z}_q$, and adversary $\mathcal{A}$ does not obtain $D_{A,1} \in G_1$ and $D_{B,2} \in G_2$ from the condition of freshness.

In this case, solver $\mathcal{S}$ selects random $x_A \in \mathbb{Z}_q$ and sets $X_{A,1} = g_1^{x_A}$ and $X_{A,2} = g_2^{x_A}$. Solver $\mathcal{S}$ selects random $r_{A,t}, s_{A,t}, r_{B,t}, s_{B,t} \in \mathbb{Z}_q$ $(t = 1, 2)$ and embeds the instance as $Z_1 = U_1$ $(= g_1^u)$, $Z_2 = U_2$ $(= g_2^u)$, and $Q_{A,t} = H_t(ID_A) = V_t \in G_t$ $(t = 1, 2)$, $Q'_{A,t} = H'_t(ID_A) = g_t^{s_{A,t}}/V_t^{r_{A,t}} \in G_t$ $(t = 1, 2)$, and $Q_{B,t} = H_t(ID_B) = W_t \in G_t$ $(t = 1, 2)$ $Q'_{B,t} = H'_t(ID_B) = g_t^{s_{B,t}}/W_t^{r_{B,t}} \in G_t$ $(t = 1, 2)$.

To check $\sigma_{1,1}, \sigma_{1,2}, \sigma_{1,3}, \sigma_{1,4}$ are correctly formed, solver $\mathcal{S}$ verifies

$$\sigma_{1,1}^{r_{A,1}} \cdot \sigma_{1,2} \stackrel{?}{=} \hat{e}(Z_1, Q_{B,2})^{s_{A,1}},$$
$$\sigma_{1,3}^{r_{A,1}} \cdot \sigma_{1,4} \stackrel{?}{=} \hat{e}(Z_1, Q'_{B,2})^{s_{A,1}}.$$

To check $\sigma_{2,1}, \sigma_{2,2}, \sigma_{2,3}, \sigma_{2,4}$ are correctly formed, solver $\mathcal{S}$ computes

$$\bar{\sigma}_{2,1} = \sigma_{2,1}/(\sigma_{1,1}\hat{e}(Z_1, Q_{B,2}X_{B,2})^{x_A}),$$
$$\bar{\sigma}_{2,2} = \sigma_{2,2}/(\sigma_{1,2}\hat{e}(Z_1, Q_{B,2}X_{B,2})^{x_A}),$$
$$\bar{\sigma}_{2,3} = \sigma_{2,3}/(\sigma_{1,3}\hat{e}(Z_1, Q'_{B,2}X_{B,2})^{x_A}),$$
$$\bar{\sigma}_{2,4} = \sigma_{2,4}/(\sigma_{1,4}\hat{e}(Z_1, Q'_{B,2}X_{B,2})^{x_A}),$$

and verifies

$$\bar{\sigma}_{2,1}^{r_{A,1}} \cdot \bar{\sigma}_{2,2} \stackrel{?}{=} \hat{e}(Z_1, X_{B,2})^{s_{A,1}},$$
$$\bar{\sigma}_{2,3}^{r_{A,1}} \cdot \bar{\sigma}_{2,4} \stackrel{?}{=} \hat{e}(Z_1, X_{B,2})^{s_{A,1}}.$$

6.2 Comparison with Existing Protocols

(1) Explanation of Table 1.

We compare the proposed Π^{sym} (FSU [22]) protocol, Π^{asym} protocol, and their twin-DH versions with known ID-AKE protocols in terms of computational efficiency, security model, and underlying assumption in Table 1.

In Table 1, the number of pairing computations, the number of exponentiations in group G, and the number of hashing to group G are denoted by "P", "E", and "H". The computational costs of hashing to group G is almost same as of scalar multiplication (see Table 5 in [13]). In the column of Bit Length of Table 1, the bit length of one group element of elliptic curve without pairing, elliptic curve with asymmetric pairing, and elliptic curve with symmetric pairing for 2^{128} security. For elliptic curve without pairing, we can use 256-bit curves for 128-bit security. For elliptic curve with asymmetric pairing, we can use 256-bit curves with embedding degree 12 for 128-bit security. For elliptic curve with symmetric pairing, we can use 512-bit curves with embedding degree 6 for 128-bit security.

SCK protocol [13] requires 2 pairing operations, 3 exponentiations (including the exponentiation for the ephemeral public key), and uses 1 static keys and 1 ephemeral key.

SYL protocol [13] requires 1 pairing operations, 3 exponentiations (including the exponentiation for the ephemeral public key), and uses 1 static keys and 1 ephemeral key.

FG protocol [19], which is constructed without pairings, requires 2 group elements as static secret key and 2

group elements as ephemeral public key, where the static key is a kind of a certificate and 1 group element in the ephemeral public key is a part of the certificate. Comparing FG protocol [19], the proposed protocol Π^{sym} has an advantage in the number of static and ephemeral keys, though it requires pairing operations.

We may consider PKI-based construction of ID-AKE, where certificate authority (CA) generates certificate that bind ID and public key of a user and the user sends the certificate and the public key with ephemeral public key. Since, as far as we know, no PKI-based construction of ID-AKE have been proposed, we show the estimation of PKI-based construction of ID-AKE using Schnorr signature and HMQV authenticated key exchange protocol (PKI+Schnorr) and using BLS[1] signature and HMQV (PKI+BLS) as an example. In PKI+Schnorr protocol using Schnorr signature and HMQV, 4 group elements (2 for Schnorr signature for certificate, 1 for static public key, and 1 for ephemeral public key) need to be sent, and 5 exponentiations (2 for verification of Schnorr signature, 1 for computation of ephemeral public key, and 2 for computation of HMQV shared key) need to be computed. In PKI+BLS protocol using BLS signature and HMQV, 3 group elements (1 for BLS signature for certificate, 1 for static public key, and 1 for ephemeral public key) need to be sent, and 2 pairings 3 exponentiations (2 pairings for verification of BLS signature, 1 for computation of ephemeral public key, and 2 for computation of HMQV shared key) need to be computed. Since no PKI-based construction of ID-AKE have been proposed, achieved security level and required assumption are not known.

HC protocol [25], which is the only known existing id-eCK secure ID-AKE protocol, requires 2 pairing operations, 3 exponentiations (including the exponentiation for the ephemeral public key), and uses 2 static keys and 1 ephemeral key.

The protocol Π^{sym}, that is proposed in [22], requires 2 pairing operations, 3 exponentiations (1 exponentiation for computation of ephemeral public key, and 2 pairing operations and 2 exponentiations for computation of shared values), and uses 1 static keys and 1 ephemeral key.

The protocol Π^{asym}, that is asymmetric pairing version of Π^{sym}, requires 4 pairing operations, 5 exponentiations (2 pairing operations for verification of ephemeral public keys, 2 exponentiations for computation of ephemeral public keys, and 2 pairing operations and 3 exponentiations for computation of shared values), and uses 2 static keys and 2 ephemeral key.

The protocol Π^{sym}+TwinDH, that is twin-DH version of Π^{sym}, requires 4 pairing operations, 4 exponentiations (1 exponentiations for computation of ephemeral public keys, and 4 pairing operations and 3 exponentiations for computation of shared values), and uses 2 static keys and 1 ephemeral key.

The protocol Π^{sym}+TwinDH, that is twin-DH version of Π^{asym}, requires 10 pairing operations, 8 exponentiations (2 pairing operations for verification of ephemeral public keys, 2 exponentiations for computation of ephemeral public keys, and 8 pairing operations and 6 exponentiations for computation of shared values), and uses 4 static keys and 2 ephemeral key.

(2) Comparison with the existing protocols.

From Table 1, we have the following comparison results of the proposed protocols with the existing protocols.

The proposed protocol Π^{sym} is the most efficient among the protocols satisfying id-eCK security. Only known existing id-eCK secure ID-AKE protocol is HC protocol [25], that requires 2 static keys while protocol Π^{sym} requires 1 static key, and the numbers of ephemeral keys, pairing operations, and exponentiations are equivalent to Π^{sym}. Thus, the protocol Π^{sym}, to the best of our knowledge, is the most efficient id-eCK secure ID-based AKE protocol based on symmetric pairings.

Comparing with the SCK and SYL protocols based on elliptic curve with pairing, the proposed protocol Π^{sym} is proven to be id-eCK secure while the SCK and SYL protocol are not, although they are comparable in communication complexity and they require more computation and storage.

Comparing with the FG protocol based on elliptic curve without pairing, the proposed protocol Π^{sym} is proven to be id-eCK secure while the FG protocol is not, although it is comparable in communication complexity and it requires more computation and storage. From the view point of the number of the ephemeral public keys, the proposed protocol Π^{sym} has advantages comparing with FG protocol.

Comparing with the PKI+Schnorr and PKI+BLS protocols based on PKI-based AKE, the proposed protocol Π^{sym} is proven to be id-eCK secure while PKI+Schnorr and PKI+BLS are not proven secure, although they are comparable in communication complexity and they require more computation and storage. From the view point of the number of the ephemeral public keys, the proposed protocol Π^{sym} has advantages comparing with PKI+Schnorr and PKI+BLS protocol.

The proposed protocol Π^{asym} is the first id-eCK secure ID-based AKE protocol based on asymmetric pairings, since only known existing id-eCK secure ID-AKE protocol HC protocol [25] uses symmetric pairings.

(3) Comparison with the HC protocol.

Here, we compare more precisely the proposed protocols with the HC protocol [25], since it is the only known existing id-eCK secure ID-AKE protocol.

First, we compare Π^{sym}+TwinDH protocol, that is twin-DH version of Π^{sym}, with HC protocol, since both protocols achieve the same security level, i.e. id-eCK security, with the same assumption, i.e. BDH assumption. Since Π^{sym} protocol is not designed to combine with twin-DH technique, HC protocol is more efficient than Π^{sym}+TwinDH protocol. So comparing with HC protocol, Π^{sym} protocol achieves better efficiency with stronger assumption, that is trade-off between efficiency and assumption.

Next, we compare Π^{sym} protocol with 1 static key ver-

sion of HC protocol, since both protocols achieve the same security level, i.e. id-eCK security, with the same assumption, i.e. gap BDH assumption. Unfortunately, straightforward modification of HC protocol cannot yield id-eCK secure protocol as follows. If we apply straightforward modification, i.e. to make two static keys identical, to HC protocol, we have 1 static key version of HC protocol, where each party computes shared values $g_T^{z(q_A+x_A)(q_B+x_B)}$ and $g^{x_A x_B}$. However, these two shared values are not enough to extract the answer of of gap BDH problem. For instance, in the case that simulator embeds instance of gap BDH problem into (g^z, g^{q_A}, g^{q_B}) and x_A is generated by simulator and x_B is generated by adversary in the security proof, simulator cannot extract answer $g_T^{zq_Aq_B}$ of gap BDH problem. So, to overcome the difficulty, we add shared value $g_T^{zq_Aq_B}$ in Π^{sym} protocol s.t. simulator can extract answer in all cases in the security proof.

Finally, we compare Π^{asym} protocol with asymmetric pairing version of HC protocol, since both protocols achieve the same security level, i.e. id-eCK security, with asymmetric pairings. Unfortunately, straightforward modification of HC protocol cannot yield id-eCK secure protocol with asymmetric pairings as follows. If we apply straightforward modification, i.e. party A uses group G_1 and party B uses group G_2, to HC protocol, we have asymmetric pairing version of HC protocol, where $Q_A, X_A \in G_1$ and $Q_B, X_B \in G_2$. However, we have difficulty comes from shared value $g^{x_A x_B}$, that is needed to guarantee the security when the master secret key is leaked, in original HC protocol. In asymmetric pairing setting, $X_A^{x_B} = g_1^{x_A x_B} \in G_1$ and $X_B^{x_A} = g_2^{x_A x_B} \in G_2$ are not equal, so party A and B cannot compute the same shared value. So, to overcome the difficulty, we introduce two ephemeral public keys $X_{A,1} = g_1^{x_A}, X_{A,2} = g_2^{x_A}$ and $X_{B,1} = g_1^{x_B}, X_{B,2} = g_2^{x_B}$ in Π^{asym} protocol, where each party checks correctness of these ephemeral public keys using pairing and can compute same shared values $g_1^{x_A x_B}$ and $g_2^{x_A x_B}$.

7. Conclusion

We presented an id-eCK secure two-pass AKE protocols based on symmetric pairing with a single static (secret) key and a single ephemeral key using a single hash function. We proposed the other id-eCK secure two-pass AKE protocols based on asymmetric pairing, also. Their security proofs do not depend on the Forking Lemma. As a result, our protocols provide strong security assurances without compromising too much on efficiency.

References

[1] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," J. Cryptology, vol.17, no.4, pp.297–319, 2004.
[2] J. Baek, R. Safavi-Naini, and W. Susilo, "Efficient multi-receiver identity-based encryption and its application to broadcast encryption," PKC 2005, pp.380–397, 2005.
[3] J. Baek, R. Safavi-Naini, and W. Susilo, "Universal designated verifier signature proof (or how to efficiently prove knowledge of a signature)," Asiacrypt 2005, pp.644–661, 2005.
[4] F. Bao, R.H. Deng, and H. Zhu, "Variations of Diffie-Hellman problem," ICICS 2003, pp.301–312, 2003.
[5] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," ACM CCS'93, pp.62–73, 1993.
[6] M. Bellare and P. Rogaway, "Entity authentication and key distribution," Crypto 1993, pp.110–125, 1993.
[7] S. Blake-Wilson, D. Johnson, and A. Menezes, "Key agreement protocols and their security analysis," IMA CC'97, pp.30–45, 1997.
[8] C. Boyd and K.-K.R. Choo, "Security of two-party identity-based key agreement," Mycrypt 2005, pp.229–243, 2005.
[9] C. Boyd, Y. Cliff, J.M. González Nieto, and K. Paterson, "Efficient one-round key exchange in the standard model," ACISP 2008, pp.69–83, 2008.
[10] E. Bresson, Y. Lakhnech, L. Mazaré, and B. Warinschi, "A Generalization of DDH with applications to protocol analysis and computational soundness," Crypto 2007, pp.482–499, 2007.
[11] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," Eurocrypt 2001, pp.453–474, 2001.
[12] D. Cash, E. Kiltz, and V. Shoup, "The twin Diffie-Hellman problem and applications," Eurocrypt 2008, pp.127–145, 2008.
[13] L. Chen, Z. Cheng, and N.P. Smart, "Identity-based key agreement protocols from pairings," Int. J. Information Security, vol.6, no.4, pp.213–241, 2007.
[14] S.S.M. Chow and K.-K.R. Choo, "Strongly-secure identity-based key agreement and anonymous extension," ISC 2007, pp.203–220, 2007.
[15] K.-K.R. Choo, C. Boyd, and Y. Hitchcock, "Examining indistinguishability-based proof models for key establishment protocols," Asiacrypt 2005, pp.585–604, 2005.
[16] C.J.F. Cremers, "Session-stateReveal is stronger than EphemeralKeyReveal: Attacking the NAXOS authenticated key exchange protocol," ACNS 2009, pp.20–33, 2009.
[17] C.J.F. Cremers, "Examining indistinguishability-based security models for key exchange protocols: The case of CK, CK-HMQV, and eCK," 6th ACM Symposium on Information, Computer and Communications Security, pp.80–91, 2011.
[18] W. Diffie and H. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theory, vol.IT-22, no.6, pp.644–654, 1976.
[19] D. Fiore and R. Gennaro, "Making the Diffie-Hellman protocol identity-based," CT-RSA 2010, pp.165–178, 2010.
[20] A. Fujioka and K. Suzuki, "Designing efficient authenticated key exchange resilient to leakage of ephemeral secret keys," CT-RSA 2011, pp.121–141, 2011.
[21] A. Fujioka and K. Suzuki, "Sufficient condition for identity-based authenticated key exchange resilient to leakage of secret keys," ICISC 2011, pp.490–509, 2011.
[22] A. Fujioka, K. Suzuki, and B. Ustaoğlu, "Ephemeral key leakage resilient and efficient ID-AKEs that can share identities, private and master keys," Pairing 2010, pp.187–205, 2010.
[23] T. Hayashi, N. Shinohara, L. Wang, S. Matsuo, M. Shirase, and T. Takagi, "Solving a 676-bit discrete logarithm problem in GF(3^{6n})," PKC 2010, pp.351–367, 2010.
[24] F. Hess, N.P. Smart, and F. Vercauteren, "The Eta pairing revisited," IEEE Trans. Inf. Theory, vol.52, no.10, pp.4595–4602, 2006.
[25] H. Huang and Z. Cao, "An ID-based authenticated key exchange protocol based on bilinear Diffie-Hellman problem," ASIACCS'09, pp.333–342, 2009.
[26] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger security of authenticated key exchange," ProvSec 2007, pp.1–16, 2007.
[27] B. Libert and J.J. Quisquater, "Identity based undeniable signatures," CT-RSA 2004, pp.112–125, 2004.
[28] F. Laguillaumie and D. Vergnaud, "Designated verifier signatures: Anonymity and efficient construction from any bilinear map," SCN 2004, pp.105–119, 2004.

[29] F. Laguillaumie and D. Vergnaud, "Multi-designated verifiers signatures," ICICS 2004, pp.495–507, 2004.
[30] E. Lee, H. Lee, and C. Park, "Efficient and generalized pairing computation on abelien varieties," IEEE Trans. Inf. Theory, vol.55, no.4, pp.1793–1803, 2009.
[31] A. Miyaji, M. Nakabayashi, and S. Takano, "New explicit conditions of elliptic curve traces for FR-reduction," IEICE Trans. Fundamentals, vol.E84-A, no.5, pp.1234–1243, May 2001.
[32] Paulo S.L.M. Barreto, and Michael Naehrig, "Pairing-friendly elliptic curves of prime order," SAC 2005, pp.319–331, 2006.
[33] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," J. Cryptology, vol.13, no.3, pp.361–396, 2000.
[34] F. Vercauteren, "Optimal pairings," IEEE Trans. Inf. Theory, vol.56, no.1, pp.455–461, 2010.

Appendix A: Proof of Theorem 5

We need the gap Bilinear Diffie-Hellman (gap BDH) assumption in pairing groups G, G_T of order q with generator g, g_T, where one tries to compute BCDH(U, V, W) accessing the BDDH oracle. Here, we denote BCDH(g^u, g^v, g^w) = g_T^{uvw}, and the BDDH oracle on input (g^u, g^v, g^w, g_T^x) returns bit 1 if $uvw = x$, or bit 0 otherwise. We also need a variant of the gap BDH assumption, where one tries to compute BCDH(U, V, V) instead of BCDH(U, V, W). We call the variant as the *square gap BDH assumption*, which is equivalent to the gap BDH assumption if groups G, G_T have prime order q [4] as follows. Given a challenge U, V of the square gap BDH assumption, one sets $W = V^s$ for random integers $s \in_R [1, q-1]$ and can compute BCDH(U, V, V) = BCDH(U, V, W)$^{1/s}$. Given a challenge U, V, W of the gap BDH assumption, one sets $V_1 = VW, V_2 = VW^{-1}$ and can compute BCDH(U, V, W) = (BCDH(U, V_1, V_1)/BCDH(U, V_2, V_2))$^{1/4}$.

We show that if polynomially bounded adversary $\mathcal{A}$ can distinguish the session key of a fresh session from a randomly chosen session key, we can solve the gap BDH problem. Let κ denote the security parameter, and let $\mathcal{A}$ be a polynomial-time bounded adversary w.r.t. security parameter κ. We use adversary $\mathcal{A}$ to construct the gap BDH solver $\mathcal{S}$ that succeeds with non-negligible probability. Adversary $\mathcal{A}$ is said to be successful with non-negligible probability if adversary $\mathcal{A}$ wins the distinguishing game with probability $\frac{1}{2} + f(\kappa)$, where $f(\kappa)$ is non-negligible, and the event M denotes that an adversary $\mathcal{A}$ is successful.

Let the test session be $\mathtt{sid}^* = (\Pi, \mathcal{I}, ID_A, ID_B, X_A, X_B)$ or $(\Pi, \mathcal{R}, ID_B, ID_A, X_A, X_B)$, which is a completed session between honest users U_A and U_B, where user U_A is the initiator and user U_B is the responder of the test session $\mathtt{sid}^*$. Let H^* be the event that adversary $\mathcal{A}$ queries $(\sigma_1, \sigma_2, \sigma_3, \Pi, ID_A, ID_B, X_A, X_B)$ to H. Let $\overline{H^*}$ be the complement of event H^*. Let $\mathtt{sid}$ be any completed session owned by an honest user such that $\mathtt{sid} \neq \mathtt{sid}^*$ and $\mathtt{sid}$ is non-matching to $\mathtt{sid}^*$. Since $\mathtt{sid}$ and $\mathtt{sid}^*$ are distinct and non-matching, the inputs to the key derivation function H are different for $\mathtt{sid}$ and $\mathtt{sid}^*$. Since H is a random oracle, adversary $\mathcal{A}$ cannot obtain any information about the test session key from the session keys of non-matching sessions. Hence, $\Pr(M \wedge \overline{H^*}) \leq \frac{1}{2}$ and $\Pr(M) = \Pr(M \wedge H^*) + \Pr(M \wedge \overline{H^*}) \leq \Pr(M \wedge H^*) + \frac{1}{2}$, whence $f(\kappa) \leq \Pr(M \wedge H^*)$. Henceforth, the event $M \wedge H^*$ is denoted by M^*.

Table A·1 Classification of attacks when IDs ID_A, ID_B are distinct. "ok" means the secret key is not revealed. "r" means the secret key may be revealed. "n" means no matching session exists. The "instance embedding" row shows how the simulator embeds an instance of the gap BDH problem.

	z	D_A	x_A	D_B	x_B	instance embedding
E_1	ok	r	ok	ok	n	$Z = U, X_A = V, Q_B = W$
E_2	ok	ok	r	ok	n	$Z = U, Q_A = V, Q_B = W$
E_3	ok	r	ok	ok	r	$Z = U, X_A = V, Q_B = W$
E_4	ok	ok	r	ok	r	$Z = U, Q_A = V, Q_B = W$
E_5	r	r	ok	r	ok	$X_A = V, X_B = W$
E_6	ok	ok	r	r	ok	$Z = U, Q_A = V, X_B = W$

Table A·2 Classification of attacks when IDs $ID_A = ID_B$ are the same.

	z	D_A	x_A	D_B	x_B	instance embedding
E_2'	ok	ok	r	ok	n	$Z = U, Q_A = V, Q_B = V$
E_4'	ok	ok	r	ok	r	$Z = U, Q_A = V, Q_B = V$
E_5'	r	r	ok	r	ok	$X_A = V, X_B = V$

We denote a user as U_i, and user U_i and other parties are modeled as probabilistic polynomial-time Turing machines w.r.t. security parameter κ. We denote master secret (public) key as z (Z). For user U_i, we denote static secret keys as D_i and ephemeral secret (public) keys as x_i (X_i, respectively). We also denote the session key as K. Assume that adversary $\mathcal{A}$ succeeds in an environment with n users and activates at most s sessions within a user.

We consider the non-exclusive classification of all possible events in Tables A·1 and A·2. Here, users U_A and U_B are initiator and responder of the test session $\mathtt{sid}^*$, respectively. Table A·1 classifies events, named $E_1, E_2, E_3, E_4, E_5, E_6$, when identities ID_A, ID_B are distinct, and Table A·2 classifies events, named E_2', E_4', E_5', when identities $ID_A = ID_B$ are the same, i.e., reflection attacks. In these tables, "ok" means the secret key is not revealed, or the matching session exists and the ephemeral key is not revealed. "r" means the secret key may be revealed. "n" means no matching session exists. The "instance embedding" row shows how the simulator embeds an instance of the gap BDH problem.

Since the classification covers all possible events, at least one event $E_i \wedge M^*$ in the tables occurs with non-negligible probability if event M^* occurs with non-negligible probability. Thus, the gap BDH problem can be solved with non-negligible probability, which means that the proposed protocol is secure under the gap BDH assumption. We investigate each of these events in the following subsections.

A.1 Event $E_1 \wedge M^*$

In event E_1, test session $\mathtt{sid}^*$ has no matching session $\overline{\mathtt{sid}^*}$,

adversary $\mathcal{A}$ obtains D_A, and adversary $\mathcal{A}$ does not obtain x_A and D_B from the condition of freshness. In this case, solver $\mathcal{S}$ embeds the instance as $Z = U\ (= g^u)$, $X_A = V\ (= g^v)$ and $Q_B = W\ (= g^w)$, and extracts g_T^{uvw} from the shared values $\sigma_1 = g_T^{zq_Aq_B}$, $\sigma_2 = g_T^{z(q_A+x_A)(q_B+x_B)}$, $\sigma_3 = g^{x_Ax_B}$ using the knowledge of q_A. In event $E_1 \wedge M^*$, solver $\mathcal{S}$ performs the following steps.

A.1.1 Setup

The gap BDH solver $\mathcal{S}$ embeds instance $(U = g^u, V = g^v, W = g^w)$ of the gap BDH problem as follows. $\mathcal{S}$ set the master public key $Z = U$, establishes n honest users $U_1, ..., U_n$. $\mathcal{S}$ randomly selects two users U_A and U_B and integer $t \in_R [1, s]$, which is a guess of the test session with probability $1/n^2s$, i.e., the test session is the t-th session initialized by Send query as initiator or responder, which is owned by U_A and with peer U_B. $\mathcal{S}$ sets the ephemeral public key of the t-th session of user U_A as $X_A = V$, sets hash value $Q_B = W$ of ID_B of of user U_B, and selects random q_A and sets $Q_A = H_1(ID_A) = g^{q_A}$ and $D_A = Z^{q_A}$.

$\mathcal{S}$ activates adversary $\mathcal{A}$ on this set of users and awaits the actions of $\mathcal{A}$. We next describe the actions of $\mathcal{S}$ in response to user activations and oracle queries.

A.1.2 Simulation

$\mathcal{S}$ maintains a list L_H that contains queries and answers of H oracle, and a list L_S that contains queries and answers of SessionKeyReveal. $\mathcal{S}$ simulates oracle queries as follows.

1. Send$(\Pi, \mathcal{I}, ID_i, ID_j)$: If $i = A$, $j = B$ and the session is the test session, i.e., the t-th initialized session, $\mathcal{S}$ sets $X_A = V$, records (Π, ID_i, ID_j, X_i) in list L_{send}, and returns X_A. Otherwise, $\mathcal{S}$ selects ephemeral secret key $x_i \in_U \mathbb{Z}_q$, computes ephemeral public key X_i honestly, records (Π, ID_i, ID_j, X_i) in list L_{send}, and returns X_i.
2. Send$(\Pi, \mathcal{R}, ID_j, ID_i, X_i)$: $\mathcal{S}$ selects ephemeral secret key $x_j \in_U \mathbb{Z}_q$, computes ephemeral public key X_j honestly, records $(\Pi, ID_i, ID_j, X_i, X_j)$ in list L_{send}, and returns X_j.
3. Send$(\Pi, \mathcal{I}, ID_i, ID_j, X_i, X_j)$: If (Π, ID_i, ID_j, X_i) is not recorded in list L_{send}, $\mathcal{S}$ records the session $(\Pi, \mathcal{I}, ID_i, ID_j, X_i, X_j)$ as not completed in list L_{send}. Otherwise, $\mathcal{S}$ records the session as completed in list L_{send}.
4. $H(\sigma_1, \sigma_2, \sigma_3, \Pi, ID_i, ID_j, X_i, X_j)$:
   a. If $(\sigma_1, \sigma_2, \sigma_3, \Pi, ID_i, ID_j, X_i, X_j)$ is recorded in list L_H, then return value K recorded in list L_H.
   b. Else if the session $(\Pi, \mathcal{I}, ID_i, ID_j, X_i, X_j)$ or $(\Pi, \mathcal{R}, ID_j, ID_i, X_i, X_j)$ is recorded in list L_S, then $\mathcal{S}$ checks that the shared values $\sigma_1, \sigma_2, \sigma_3$ are correctly formed w.r.t. IDs and ephemeral public keys ID_i, ID_j, X_i, X_j by the procedure Check described below.
   If the shared values are correctly formed, then return value K recorded in list L_S and record it in list L_H. Otherwise $\mathcal{S}$ returns random value K and records it in list L_H.
   c. Else if $i = A$, $j = B$, and the session is the test session, i.e., $X_i = V$, then $\mathcal{S}$ checks that the shared values $\sigma_1, \sigma_2, \sigma_3$ are correctly formed w.r.t. IDs and ephemeral public keys ID_A, ID_B, X_A, Y_B by the procedure Check described below.
   If the shared values are correctly formed, then $\mathcal{S}$ computes the answer of the gap BDH instance from the shared values and public keys using knowledge of secret key q_A by the procedure Extract described below, and is successful by outputting the answer. Otherwise $\mathcal{S}$ returns random value K and records it in list L_H.
   d. Otherwise, $\mathcal{S}$ returns random value K and records it in list L_H.
5. SessionKeyReveal$((\Pi, \mathcal{I}, ID_i, ID_j, X_i, X_j)$ or $(\Pi, \mathcal{R}, ID_j, ID_i, X_i, X_j))$:
   a. If the session $(\Pi, \mathcal{I}, ID_i, ID_j, X_i, X_j)$ or $(\Pi, \mathcal{R}, ID_j, ID_i, X_i, X_j)$ (= `sid`) is not completed, return error.
   b. Else if `sid` is recorded in list L_S, then return value K recorded in list L_S.
   c. Else if $(\sigma_1, \sigma_2, \sigma_3, \Pi, ID_i, ID_j, X_i, X_j)$ is recorded in list L_H, then $\mathcal{S}$ checks that the shared values $\sigma_1, \sigma_2, \sigma_3$ are correctly formed w.r.t. IDs and ephemeral public keys ID_i, ID_j, X_i, X_j by the procedure Check described below.
   If the shared values are correctly formed, then return value K recorded in list L_H and record it in list L_S. Otherwise $\mathcal{S}$ returns random value K and records it in list L_S.
   d. Otherwise, $\mathcal{S}$ returns random value K and records it in list L_S.
6. EphemeralKeyReveal(`sid`): If the session `sid` is the test session, i.e., $i = A$, $j = B$, $X_i = V$, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ selects random x_i, sets $X_i = g^{x_i}$, and returns x_i.
7. StaticKeyReveal(U_i): If $i = B$, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ selects random q_i, sets $Q_i = H_1(ID_i) = g^{q_i}$ and records (ID_i, q_i, Q_i) in list L_{H_1}, sets $D_i = Z^{q_i}$, and returns D_i.
8. MasterKeyReveal(): $\mathcal{S}$ aborts with failure.
9. Test(`sid`): If `sid` is not t-th session of U_A, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ responds to the query faithfully.
10. If adversary $\mathcal{A}$ outputs a guess γ, $\mathcal{S}$ aborts with failure.
11. $H_1(ID_i)$: If (ID_i, q_i, Q_i) is recorded in list L_{H_1}, then $\mathcal{S}$ returns recorded Q_i. Otherwise, $\mathcal{S}$ selects random q_i, sets $Q_i = H_1(ID_i) = g^{q_i}$ and records (ID_i, q_i, Q_i) in list L_{H_1}, and returns Q_i.
12. EstablishParty(ID_i): If ID_i is already registered, $\mathcal{S}$ returns reject message. Otherwise, $\mathcal{S}$ register ID_i as identity of dishonest user U_i and returns accept message.

(1) Extract.

The procedure Extract computes the answer of the gap BDH instance as follows. By eliminating the terms including q_A using the knowledge of q_A, solver $\mathcal{S}$ can obtain

$$\sigma_2' = \sigma_2 / e(Z, Q_B X_B)^{q_A} = g_T^{z x_A (q_B + x_B)},$$
$$\sigma_3' = e(Z, \sigma_3) = g_T^{z x_A x_B},$$

By using this term, solver $\mathcal{S}$ can extract the answer $g_T^{z x_A q_B} = g_T^{uvw}$ as

$$\sigma_2' / \sigma_3' = g_T^{z x_A q_B}.$$

Notice that, in other cases in Table A· 1, the solver $\mathcal{S}$ can eliminate the terms including the specific secret key using the knowledge of the secret key, can obtain 2 linearly independent terms, and can extract the answer g_T^{uvw} by solving the linear equation.

(2) Check.

The procedure Check checks whether the shared secrets are correctly formed w.r.t. static and ephemeral public keys, and can simulate H and SessionKeyReveal queries consistently. More precisely, in the simulation of the $H(\sigma_1, \sigma_2, \sigma_3, \Pi, ID_A, ID_B, X_A, X_B)$ query, solver $\mathcal{S}$ needs to check that the shared secrets $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, and if so return session key K being consistent with the previously answered SessionKeyReveal(Π, $\mathcal{I}$, ID_A, ID_B, X_A, X_B) and SessionKeyReveal(Π, $\mathcal{R}$, ID_B, ID_A, X_A, X_B) queries. The solver $\mathcal{S}$ can check if shared secrets $\sigma_1, \sigma_2, \sigma_3$ are correctly formed w.r.t. the static and ephemeral public keys by asking BDDH oracle

$$\mathrm{BDDH}(Z, Q_A, Q_B, \sigma_1) = 1,$$
$$\mathrm{BDDH}(Z, Q_A X_A, Q_B X_B, \sigma_2) = 1,$$
$$e(X_A, X_B) = e(g, \sigma_3),$$

and this implies $\sigma_1, \sigma_2, \sigma_3$ are correctly formed.

Notice that, in other cases in Table A· 1, the solver $\mathcal{S}$ can check if shared secrets $\sigma_1, \sigma_2, \sigma_3$ are correctly formed or not by the same procedure.

A.1.3 Analysis

The simulation of the environment for adversary $\mathcal{A}$ is perfect except with negligible probability. The probability that adversary $\mathcal{A}$ selects the session, where U_A is initiator, U_B is responder, and ephemeral public key X_A is V, as the test session $\mathtt{sid}^*$ is at least $\frac{1}{n^2 s}$. Suppose this is indeed the case, solver $\mathcal{S}$ does not abort in Step 9.

Suppose event E_1 occurs, solver $\mathcal{S}$ does not abort in Steps 6, 7 and 8.

Suppose event M^* occurs, adversary $\mathcal{A}$ queries correctly formed $\sigma_1, \sigma_2, \sigma_3$ to H. Therefore, solver $\mathcal{S}$ is successful as described in Step 4c since $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, and does not abort as in Step 10.

Hence, solver $\mathcal{S}$ is successful with probability $Pr(S) \geq \frac{p_1}{n^2 s}$, where p_1 is probability that $E_1 \wedge M^*$ occurs.

A.2 Event $E_2 \wedge M^*$

In event E_2, test session $\mathtt{sid}^*$ has no matching session $\overline{\mathtt{sid}^*}$, $\mathcal{A}$ obtains x_A, and $\mathcal{A}$ does not obtain either D_A or D_B. The reduction to the gap BDH assumption is similar to event $E_1 \wedge M^*$ in Sect. A.1, except the following points.

In Setup and Simulation, $\mathcal{S}$ embeds gap BDH instance U, V, W as $Z = U$, $Q_A = V$, and $Q_B = W$.

In Simulation, using knowledge of x_A, $\mathcal{S}$ extracts answer $g_T^{z q_A q_B}$ of the gap BDH problem.

A.3 Event $E_3 \wedge M^*$

In event E_3, test session $\mathtt{sid}^*$ has matching session $\overline{\mathtt{sid}^*}$, $\mathcal{A}$ obtains D_A and x_B, and $\mathcal{A}$ does not obtain either x_A or D_B. The reduction to the gap BDH assumption is similar to event $E_1 \wedge M^*$ in Sect. A.1, except the following points.

In Setup and Simulation, $\mathcal{S}$ embeds gap BDH instance U, V, W as $Z = U$, $X_A = V$, and $Q_B = W$.

In Simulation, using knowledge of q_A or x_B, $\mathcal{S}$ extracts answer $g_T^{z x_A q_B}$ of the gap BDH problem.

A.4 Event $E_4 \wedge M^*$

In event E_4, test session $\mathtt{sid}^*$ has matching session $\overline{\mathtt{sid}^*}$, $\mathcal{A}$ obtains x_A and x_B, and $\mathcal{A}$ does not obtain either D_A or D_B. The reduction to the gap BDH assumption is similar to event $E_1 \wedge M^*$ in Sect. A.1, except the following points.

In Setup and Simulation, $\mathcal{S}$ embeds gap BDH instance U, V, W as $Z = U$, $Q_A = V$, and $Q_B = W$.

In Simulation, using knowledge of x_A or x_B, $\mathcal{S}$ extracts answer $g_T^{z q_A q_B}$ of the gap BDH problem.

A.5 Event $E_5 \wedge M^*$

In event E_5, test session $\mathtt{sid}^*$ has matching session $\overline{\mathtt{sid}^*}$, $\mathcal{A}$ obtains z, D_A, and D_B, and $\mathcal{A}$ does not obtain either x_A or x_B. The reduction to the gap BDH assumption is similar to event $E_1 \wedge M^*$ in Sect. A.1, except the following points.

In Setup and Simulation, $\mathcal{S}$ embeds gap BDH instance U, V, W as $X_A = V$ and $X_B = W$.

In Simulation, using knowledge of q_A or q_B, $\mathcal{S}$ extracts $g^{x_A x_B}$ from shared value σ_3 and can compute answer $e(U, g^{x_A x_B})$ of the gap BDH problem.

A.6 Event $E_6 \wedge M^*$

In event E_6, test session $\mathtt{sid}^*$ has matching session $\overline{\mathtt{sid}^*}$, $\mathcal{A}$ obtains x_A and D_B, and $\mathcal{A}$ does not obtain either D_A or x_B. The reduction to the gap BDH assumption is similar to event $E_1 \wedge M^*$ in Sect. A.1, except the following points.

In Setup and Simulation, $\mathcal{S}$ embeds gap BDH instance U, V, W as $Z = U$, $Q_A = V$, and $X_B = W$.

In Simulation, using knowledge of x_A or q_B, $\mathcal{S}$ extracts answer $g_T^{z q_A x_B}$ of the gap BDH problem.

A.7 $ID_A = ID_B$ Cases

In the case of $ID_A = ID_B$, i.e., $Q_A = Q_B$, we make the reduction to the square gap BDH assumption.

In event E_2', the reduction to the square gap BDH assumption is similar to event E_2. In Setup and Simulation, $\mathcal{S}$ embeds square gap BDH instance U, V as $Z = U$, $Q_A = V$, and $Q_B = V$. In Simulation, using knowledge of x_A, $\mathcal{S}$ extracts answer $g_T^{zq_Aq_B}$ of the square gap BDH problem.

In event E_4', the reduction to the square gap BDH assumption is similar to event E_4. In Setup and Simulation, $\mathcal{S}$ embeds square gap BDH instance U, V as $Z = U$, $Q_A = V$, and $Q_B = V$. In Simulation, using knowledge of x_A or x_B, $\mathcal{S}$ extracts answer $g_T^{zq_Aq_B}$ of the square gap BDH problem.

In event E_5', the reduction to the square gap BDH assumption is similar to event E_5. In Setup and Simulation, $\mathcal{S}$ embeds square gap BDH instance U, V as $X_A = V$ and $X_B = V^r$, where r is random. In Simulation, using knowledge of q_A or q_B and r, $\mathcal{S}$ extracts answer $g_T^{ux_Ax_B/r} = e(U, \sigma_3)^{1/r}$ of the square gap BDH problem.

□

Appendix B: Proof of Theorem 6

The outline of the proof is almost same as the proof of Theorem 5. So, we only describe the part of the proof that is differ from the proof of Theorem 5 in Appendix A in the following.

B.1 Event $E_1 \wedge M^*$

In event E_1, test session $\mathtt{sid}^*$ has no matching session $\overline{\mathtt{sid}^*}$, adversary $\mathcal{A}$ obtains D_A, and adversary $\mathcal{A}$ does not obtain x_A and D_B from the condition of freshness. In this case, solver $\mathcal{S}$ embeds the instance as $Z_1 = U_1$ $(= g_1^u)$, $Z_2 = U_2$ $(= g_2^u)$, $X_{A,1} = V_1$ $(= g_1^v)$, $X_{A,2} = V_2$ $(= g_2^v)$, and $Q_{B,2} = W_2$ $(= g_2^w)$, and extracts g_T^{uvw} from the shared values $\sigma_1 = g_T^{zq_{A,1}q_{B,2}}$, $\sigma_2 = g_T^{z(q_{A,1}+x_A)(q_{B,2}+x_B)}$, $\sigma_3 = g_1^{x_Ax_B}$, $\sigma_4 = g_2^{x_Ax_B}$ using the knowledge of $q_{A,1}$. In event $E_1 \wedge M^*$, solver $\mathcal{S}$ performs the following steps.

B.1.1 Setup

The gap BDH solver $\mathcal{S}$ embeds instance $(U_1 = g_1^u, U_2 = g_2^u, V_1 = g_1^v, V_2 = g_2^v, W_1 = g_1^w, W_2 = g_2^w)$ of the asymmetric gap BDH problem as follows. $\mathcal{S}$ set the master public key $Z_1 = U_1$ $(= g_1^u)$, $Z_2 = U_2$ $(= g_2^u)$, establishes n honest users $U_1, ..., U_n$. $\mathcal{S}$ randomly selects two users U_A and U_B and integer $t \in_R [1, s]$, which is a guess of the test session with probability $1/n^2s$, i.e., the test session is the t-th session initialized by Send query as initiator or responder, which is owned by U_A and with peer U_B. $\mathcal{S}$ sets the ephemeral public key of t-th session of user U_A as $X_{A,1} = V_1$ $(= g_1^v)$, $X_{A,2} = V_2$ $(= g_2^v)$, sets hash value $Q_{B,2} = W_2$ $(= g_2^w)$ of ID_B of of user U_B, and selects random $q_{A,1}$ and sets $Q_{A,1} = H_1(ID_A) = g_1^{q_{A,1}}$ and $D_{A,1} = Z_1^{q_{A,1}}$.

$\mathcal{S}$ activates adversary $\mathcal{A}$ on this set of users and awaits the actions of $\mathcal{A}$. We next describe the actions of $\mathcal{S}$ in response to user activations and oracle queries.

B.1.2 Simulation

$\mathcal{S}$ maintains a list L_H that contains queries and answers of H oracle, and a list L_S that contains queries and answers of SessionKeyReveal. $\mathcal{S}$ simulates oracle queries as follows.

1. Send$(\Pi, \mathcal{I}, ID_i, ID_j)$: If $i = A$, $j = B$ and the session is the test session, i.e., the t-th initialized session, $\mathcal{S}$ sets $X_{A,1} = V_1, X_{A,2} = V_2$, records $(\Pi, ID_i, ID_j, X_{i,1}, X_{i,2})$ in list L_{send}, and returns $X_{A,1}, X_{A,2}$. Otherwise, $\mathcal{S}$ selects ephemeral secret key $x_i \in_U \mathbb{Z}_q$, computes ephemeral public key $X_{i,1}, X_{i,2}$ honestly, records $(\Pi, ID_i, ID_j, X_{i,1}, X_{i,2})$ in list L_{send}, and returns $X_{i,1}, X_{i,2}$.
2. Send$(\Pi, \mathcal{R}, ID_j, ID_i, X_{i,1}, X_{i,2})$: $\mathcal{S}$ selects ephemeral secret key $x_j \in_U \mathbb{Z}_q$, computes ephemeral public key $X_{j,1}, X_{j,2}$ honestly, records $(\Pi, ID_i, ID_j, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2})$ in list L_{send}, and returns $X_{j,1}, X_{j,2}$.
3. Send$(\Pi, \mathcal{I}, ID_i, ID_j, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2})$: If $(\Pi, ID_i, ID_j, X_{i,1}, X_{i,2})$ is not recorded in list L_{send}, $\mathcal{S}$ records the session $(\Pi, \mathcal{I}, ID_i, ID_j, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2})$ as not completed in list L_{send}. If $\hat{e}(X_{i,1}, g_2) = \hat{e}(g_1, X_{i,2})$ is not held, $\mathcal{S}$ records the session $(\Pi, \mathcal{I}, ID_i, ID_j, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2})$ as not completed in list L_{send}. Otherwise, $\mathcal{S}$ records the session as completed in list L_{send}.
4. $H(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \Pi, ID_i, ID_j, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2})$:
   a. If $(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \Pi, ID_i, ID_j, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2})$ is recorded in list L_H, then return value K recorded in list L_H.
   b. Else if the session $(\Pi, \mathcal{I}, ID_i, ID_j, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2})$ or $(\Pi, \mathcal{R}, ID_j, ID_i, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2})$ is recorded in list L_S, then $\mathcal{S}$ checks that the shared values $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed w.r.t. IDs and ephemeral public keys $ID_i, ID_j, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2}$ by the procedure Check described below. If the shared values are correctly formed, then return value K recorded in list L_S and record it in list L_H. Otherwise $\mathcal{S}$ returns random value K and records it in list L_H.
   c. Else if $i = A$, $j = B$, and the session is the test session, i.e., $X_i = V$, then $\mathcal{S}$ checks that the shared values $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed w.r.t. IDs and ephemeral public keys ID_A, ID_B, X_A, Y_B by the procedure Check described below.
   If the shared values are correctly formed, then $\mathcal{S}$ computes the answer of the gap BDH instance from the shared values and public keys using knowledge of secret key $q_{A,1}$ by the procedure Extract described below, and is successful by outputting the answer. Otherwise $\mathcal{S}$ returns random value K and records it in list L_H.
   d. Otherwise, $\mathcal{S}$ returns random value K and records it in list L_H.
5. SessionKeyReveal$((\Pi, \mathcal{I}, ID_i, ID_j, X_{i,1}, X_{i,2}, X_{j,1},$

$X_{j,2}$) or $(\Pi, \mathcal{R}, ID_j, ID_i, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2})$):

a. If the session $(\Pi, \mathcal{I}, ID_i, ID_j, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2})$ or $(\Pi, \mathcal{R}, ID_j, ID_i, X_{i,1}, X_{i,2}, X_{j,1}, X_{j,2})$ (= sid) is not completed, return error.
b. Else if sid is recorded in list L_S, then return value K recorded in list L_S.
c. Else if (σ_1, σ_2, σ_3, σ_4, Π, ID_i, ID_j, $X_{i,1}$, $X_{i,2}$, $X_{j,1}$, $X_{j,2}$) is recorded in list L_H, then $\mathcal{S}$ checks that the shared values $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed w.r.t. IDs and ephemeral public keys ID_i, ID_j, $X_{i,1}$, $X_{i,2}$, $X_{j,1}$, $X_{j,2}$ by the procedure Check described below.
If the shared values are correctly formed, then return value K recorded in list L_H and record it in list L_S. Otherwise $\mathcal{S}$ returns random value K and records it in list L_S.
d. Otherwise, $\mathcal{S}$ returns random value K and records it in list L_S.

6. EphemeralKeyReveal(sid): If the session sid is the test session, i.e., $i = A$, $j = B$, $X_{i,1} = V_1, X_{i,2} = V_2$, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ selects random x_i, sets $X_{i,1} = g_1^{x_i}$, $X_{i,2} = g_2^{x_i}$, and returns x_i.
7. StaticKeyReveal(U_i): If $i = B$, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ selects randoms $q_{i,1}$, $q_{i,2}$, sets $Q_{i,1} = H_1(ID_i) = g_1^{q_{i,1}}$, $Q_{i,2} = H_2(ID_i) = g_2^{q_{i,2}}$ and records $(ID_i, q_{i,1}, Q_{i,1})$ in list L_{H_1} and $(ID_i, q_{i,2}, Q_{i,2})$ in list L_{H_2}, sets $D_{i,1} = Z_1^{q_{i,1}}$, $D_{i,2} = Z_2^{q_{i,2}}$, and returns $D_{i,1}, D_{i,2}$.
8. MasterKeyReveal(): $\mathcal{S}$ aborts with failure.
9. Test(sid): If sid is not t-th session of U_A, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ responds to the query faithfully.
10. If adversary $\mathcal{A}$ outputs a guess γ, $\mathcal{S}$ aborts with failure.
11. $H_t(ID_i)$ $(t = 1, 2)$: If $(ID_i, q_{i,t}, Q_{i,t})$ is recorded in list L_{H_t}, then $\mathcal{S}$ returns recorded $Q_{i,t}$. Otherwise, $\mathcal{S}$ selects random $q_{i,t}$, sets $Q_{i,t} = H_t(ID_i) = g^{q_{i,t}}$ and records $(ID_i, q_{i,t}, Q_{i,t})$ in list L_{H_t}, and returns $Q_{i,t}$.
12. EstablishParty(ID_i): If ID_i is already registered, $\mathcal{S}$ returns reject message. Otherwise, $\mathcal{S}$ register ID_i as identity of dishonest user U_i and returns accept message.

(1) Extract.

The procedure Extract computes the answer of the gap BDH instance as follows. By eliminating the terms including $q_{A,1}$ using the knowledge of $q_{A,1}$, solver $\mathcal{S}$ can obtain

$$\sigma_2' = \sigma_2 / \hat{e}(Z_1, Q_{B,2} X_{B,2})^{q_{A,1}} = g_T^{z x_A (q_{B,2} + x_B)},$$
$$\sigma_4' = \hat{e}(Z_1, \sigma_4) = g_T^{z x_A x_B},$$

By using this term, solver $\mathcal{S}$ can extract the answer $g_T^{z x_A q_{B,2}} = g_T^{uvw}$ as

$$\sigma_2' / \sigma_4' = g_T^{z x_A q_{B,2}}.$$

Notice that, in other cases in Table A· 1, the solver $\mathcal{S}$ can eliminate the terms including the specific secret key using the knowledge of the secret key, can obtain 2 linearly independent terms, and can extract the answer g_T^{uvw} by solving the linear equation.

(2) Check.

The procedure Check checks whether the shared secrets are correctly formed w.r.t. static and ephemeral public keys, and can simulate H and SessionKeyReveal queries consistently. More precisely, in the simulation of the $H(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \Pi, ID_A, ID_B, X_A, X_B)$ query, solver $\mathcal{S}$ needs to check that the shared secrets $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed, and if so return session key K being consistent with the previously answered SessionKeyReveal(Π, $\mathcal{I}$, ID_A, ID_B, X_A, X_B) and SessionKeyReveal(Π, $\mathcal{R}$, ID_B, ID_A, X_A, X_B) queries. The solver $\mathcal{S}$ can check if shared secrets $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed w.r.t. the static and ephemeral public keys by asking $\mathrm{BDDH}^{1,1,2}$ oracle

$$\mathrm{BDDH}^{1,1,2}(Z_1, Q_{A,1}, Q_{B,2}, \sigma_1) = 1,$$
$$\mathrm{BDDH}^{1,1,2}(Z_1, Q_{A,1} X_{A,1}, Q_{B,2} X_{B,2}, \sigma_2) = 1,$$
$$\hat{e}(X_{A,1}, g_2) = \hat{e}(g_1, X_{A,2}),\ \hat{e}(X_{B,1}, g_2) = \hat{e}(g_1, X_{B,2}),$$
$$\hat{e}(X_{A,1}, X_{B,2}) = \hat{e}(\sigma_3, g),\ \hat{e}(X_{A,1}, X_{B,2}) = \hat{e}(g, \sigma_4),$$

and this implies $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed.

Notice that, in other cases in Table A· 1, the solver $\mathcal{S}$ can check if shared secrets $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed or not by the same procedure.

B.1.3 Analysis

The simulation of the environment for adversary $\mathcal{A}$ is perfect except with negligible probability. The probability that adversary $\mathcal{A}$ selects the session, where U_A is initiator, U_B is responder, and ephemeral public key X_A is V, as the test session $\mathtt{sid}^*$ is at least $\frac{1}{n^2 s}$. Suppose this is indeed the case, solver $\mathcal{S}$ does not abort in Step 9.

Suppose event E_1 occurs, solver $\mathcal{S}$ does not abort in Steps 6, 7 and 8.

Suppose event M^* occurs, adversary $\mathcal{A}$ queries correctly formed $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ to H. Therefore, solver $\mathcal{S}$ is successful as described in Step 4c since $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed, and does not abort as in Step 10.

Hence, solver $\mathcal{S}$ is successful with probability $Pr(S) \geq \frac{p_1}{n^2 s}$, where p_1 is probability that $E_1 \wedge M^*$ occurs.

B.2 Event $E_2 \wedge M^*$

In event E_2, test session $\mathtt{sid}^*$ has no matching session $\overline{\mathtt{sid}^*}$, $\mathcal{A}$ obtains x_A, and $\mathcal{A}$ does not obtain either D_A or D_B. The reduction to the gap BDH assumption is similar to event $E_1 \wedge M^*$ in Sect. B.1, except the following points.

In Setup and Simulation, $\mathcal{S}$ embeds gap BDH instance $(U_1, U_2, V_1, V_2, W_1, W_2)$ as $Z_1 = U_1$, $Z_2 = U_2$, $Q_{A,1} = V_1$, and $Q_{B,2} = W_2$.

In Simulation, using knowledge of x_A, $\mathcal{S}$ extracts answer $g_T^{z q_{A,1} q_{B,2}}$ of the gap BDH problem.

B.3 Event $E_3 \wedge M^*$

In event E_3, test session $\mathtt{sid}^*$ has matching session $\overline{\mathtt{sid}^*}$, $\mathcal{A}$ obtains D_A and x_B, and $\mathcal{A}$ does not obtain either x_A or D_B. The reduction to the gap BDH assumption is similar to event $E_1 \wedge M^*$ in Sect. B.1, except the following points.

In Setup and Simulation, $\mathcal{S}$ embeds gap BDH instance $(U_1, U_2, V_1, V_2, W_1, W_2)$ as $Z_1 = U_1$, $Z_2 = U_2$, $X_{A,1} = V_1$, $X_{A,2} = V_2$, and $Q_{B,2} = W_2$.

In Simulation, using knowledge of $q_{A,1}$ or x_B, $\mathcal{S}$ extracts answer $g_T^{zx_Aq_{B,2}}$ of the gap BDH problem.

B.4 Event $E_4 \wedge M^*$

In event E_4, test session $\mathtt{sid}^*$ has matching session $\overline{\mathtt{sid}^*}$, $\mathcal{A}$ obtains x_A and x_B, and $\mathcal{A}$ does not obtain either D_A or D_B. The reduction to the gap BDH assumption is similar to event $E_1 \wedge M^*$ in Sect. B.1, except the following points.

In Setup and Simulation, $\mathcal{S}$ embeds gap BDH instance $(U_1, U_2, V_1, V_2, W_1, W_2)$ as $Z_1 = U_1$, $Z_2 = U_2$, $Q_{A,1} = V_1$, and $Q_{B,2} = W_2$.

In Simulation, using knowledge of x_A or x_B, $\mathcal{S}$ extracts answer $g_T^{zq_{A,1}q_{B,2}}$ of the gap BDH problem.

B.5 Event $E_5 \wedge M^*$

In event E_5, test session $\mathtt{sid}^*$ has matching session $\overline{\mathtt{sid}^*}$, $\mathcal{A}$ obtains z, D_A, and D_B, and $\mathcal{A}$ does not obtain either x_A or x_B. The reduction to the gap BDH assumption is similar to event $E_1 \wedge M^*$ in Sect. B.1, except the following points.

In Setup and Simulation, $\mathcal{S}$ embeds gap BDH instance $(U_1, U_2, V_1, V_2, W_1, W_2)$ as $X_{A,1} = V_1$, $X_{A,2} = V_2$, $X_{B,1} = W_1$, and $X_{B,2} = W_2$.

In Simulation, using knowledge of $q_{A,1}$ or $q_{B,2}$, $\mathcal{S}$ extracts answer $g_T^{ux_Ax_B} = e(U_1, \sigma_4)$ of the gap BDH problem.

B.6 Event $E_6 \wedge M^*$

In event E_6, test session $\mathtt{sid}^*$ has matching session $\overline{\mathtt{sid}^*}$, $\mathcal{A}$ obtains x_A and D_B, and $\mathcal{A}$ does not obtain either D_A or x_B. The reduction to the gap BDH assumption is similar to event $E_1 \wedge M^*$ in Sect. B.1, except the following points.

In Setup and Simulation, $\mathcal{S}$ embeds gap BDH instance $(U_1, U_2, V_1, V_2, W_1, W_2)$ as $Z_1 = U_1$, $Z_2 = U_2$, $Q_{A,1} = V_1$, $X_{B,1} = W_1$, and $X_{B,2} = W_2$.

In Simulation, using knowledge of x_A or $q_{B,2}$, $\mathcal{S}$ extracts answer $g_T^{zq_{A,1}x_B}$ of the gap BDH problem.

B.7 $ID_A = ID_B$ Cases

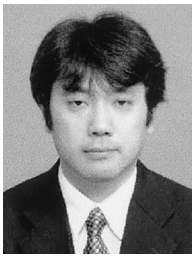
In the case of $ID_A = ID_B$, i.e., $Q_A = Q_B$, we make the reduction to the square gap BDH assumption.

In event E_2', the reduction to the square gap BDH assumption is similar to event E_2. In Setup and Simulation, $\mathcal{S}$ embeds square gap BDH instance (U_1, U_2, V_1, V_2) as $Z_1 = U_1$, $Z_2 = U_2$, $Q_{A,1} = V_1$, and $Q_{B,2} = V_2^r$, where r is random. In Simulation, using knowledge of x_A and r, $\mathcal{S}$ extracts answer $g_T^{zq_{A,1}q_{B,2}}$ of the square gap BDH problem.

In event E_4', the reduction to the square gap BDH assumption is similar to event E_4. In Setup and Simulation, $\mathcal{S}$ embeds square gap BDH instance (U_1, U_2, V_1, V_2) as $Z_1 = U_1$, $Z_2 = U_2$, $Q_{A,1} = V_1$, and $Q_{B,2} = V_2^r$, where r is random. In Simulation, using knowledge of x_A or x_B and r, $\mathcal{S}$ extracts answer $g_T^{zq_{A,1}q_{B,2}}$ of the square gap BDH problem.

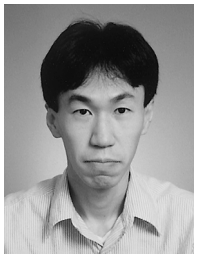
In event E_5', the reduction to the square gap BDH assumption is similar to event E_5. In Setup and Simulation, $\mathcal{S}$ embeds square gap BDH instance (U_1, U_2, V_1, V_2) as $X_{A,1} = V_1$, $X_{A,2} = V_2$, $X_{B,1} = V_1^r$, and $X_{B,2} = V_2^r$, where r is random. In Simulation, using knowledge of $q_{A,1}$ or $q_{B,2}$ and r, $\mathcal{S}$ extracts answer $g_T^{ux_Ax_B/r} = e(U_1, \sigma_4)^{1/r}$ of the square gap BDH problem. □

Atsushi Fujioka received his B.E., M.E. and D.E. degrees in electrical and electronic engineering from Tokyo Institute of Technology in 1985, 1987 and 1990, respectively. Dr. Fujioka joined NTT, Nippon Telegraph and Telephone Corporation, in 1990, and was an academic guest in Swiss Federal Institute of Technology, Zürich in 1993–1994. He received the 30th Best Achievement Award (and the 9th Kobayashi Memorial Award) from IEICE. Dr. Fujioka is presently engaged in research on cryptography and information security as a professor in Kanagawa University. He is a member of IPSJ and IACR.



Fumitaka Hoshino received his B.Eng. and M.Eng. degrees from Tokyo University, Tokyo, Japan, in 1996 and 1998, respectively. He is a Researcher in the NTT Secure Platform Laboratories.

Tetsutaro Kobayashi received his B.Eng. and M.Eng. degrees from Tokyo Institute of Technology, Tokyo, Japan, in 1993 and 1995, respectively, and received his Ph.D. degrees from the University of Tokyo, Tokyo, Japan, in 2005. He is a researcher in the NTT Secure Platform Laboratories. He is presently engaged in research on information security. He was awarded the SCIS'00 paper prize.



Koutarou Suzuki received the B.Sc., M.Sc., and Ph.D. degrees from the University of Tokyo in 1994, 1996, and 1999, respectively. He joined NTT, Nippon Telegraph and Telephone Corporation, in 1999. He is a senior research scientist in the NTT Secure Platform Laboratories. He is engaged in research on public key cryptography. He received the SCIS Paper Award in 2002. He is a member of the Information Processing Society of Japan (IPSJ).

Berkant Ustaoğlu received his B.S. degree in Mathematics in 2002 from Boğaziçi University, Turkey. In 2003 and 2008 he received his BMath and Ph.D. degrees from University of Waterloo, Waterloo, Canada. Dr. Ustaoğlu was a postdoctoral fellow at NTT, Japan and a visitor in Sabancı University, Turkey. At the present he is a faculty member at IYTE, Turkey.

Kazuki Yoneyama received the B.E., M.E. and Ph.D. degrees from the University of Electro-Communications, Tokyo, Japan, in 2004, 2006 and 2008, respectively. He is presently engaged in research on cryptography at NTT Secure Platform Laboratories, since 2009. He is a member of the International Association for Cryptologic Research, IPSJ and JSIAM.