Ephemeral Key Leakage Resilient and Efficient ID-AKEs That Can Share Identities, Private and Master Keys

Atsushi Fujioka, Koutarou Suzuki, and Berkant Ustaoğlu

NTT Information Sharing Platform Laboratories
3-9-11 Midori-cho Musashino-shi Tokyo 180-8585, Japan
{fujioka.atsushi,suzuki.koutarou,ustaoglu.berkant}@lab.ntt.co.jp

Abstract. One advantage of identity-based (ID-based) primitives is the reduced overhead of maintaining multiple static key pairs and the corresponding certificates. However, should a party wish to participate in more than one protocol with the same identity (ID), say email address, the party has to share a state between distinct primitives which is contrary to the conventional *key separation* principle. Thus it is desirable to consider security of protocols when a public identity and a corresponding private key are utilized in different protocols.

We focus on authenticated key exchange (AKE) and propose a pair of two-party ID-based authenticate key exchange protocols (ID-AKE) that are secure even if parties use the same IDs, private keys and master keys to engage in either protocol. To our knowledge the only ID-AKE protocol formally resilient to ephemeral key leakage is due to Huang and Cao (the HC protocol), where a party's static key consists of two group elements. Our proposed protocols provide similar assurances and require a single group element both for static and ephemeral keys, and in that sense are optimal. From an efficiency perspective, they have the same number of pairing computations as the HC protocol. The security of all these protocols is established in the random oracle.

Keywords: ID-based AKE, shared keys, combined keys, pairings.

1 Introduction

In 1984 Shamir [23] proposed the idea of ID-based primitives, whereby a static public key consists of a party's undeniable identifier. Consequently, in ID-based schemes parties are not required to maintain public key certificates: the public keys are available as soon as the identities become known. This is advantageous since parties do not have to manage certificates.

In a typical ID-based protocol a key generation center creates a static private key corresponding to an identity. However, most ID-based protocols and their analyses do not account for the fact that a party would often use the same identifier in many different settings. For example, a party, that identifies itself via an email (or a web) address, is unlikely to maintain different addresses for different primitives such as signatures encryption. It may be the case that obtaining private keys for different identities is more prohibitive than obtaining certificates with different public keys bound to the same identity. Furthermore, users may be attached to their identity string and be reluctant to use other identifiers. Consequently, a party might use the same private key in multiple different protocols. Such reuse of private material goes against conventional cryptographic wisdom of separating keys for different primitives.

Key Separation. Key separation can be achieved by appending each identity string with information describing the primitive or assigning a single identity string with different private keys for different protocols; for each key there would be a different key generation center dealing with a particular protocol. Compared with certificates such solutions do not reduce the number of static private keys that a party has to manage and so reduce the attractiveness of ID-based solutions. Furthermore, for a given primitive such as signatures or key establishment, parties are usually given the choice of more than one protocol. For example, the NIST's SP800-56A standard [20] defines two key agreement protocols and allows parties to use the same certificate and static public key to engage in either of those protocol variants. There is little reason to use different static keys if two protocols achieve similar goals. But as shown by Chatterjee, Menezes and Ustaoğlu [9], sharing static information between authenticated key exchange protocols can have a negative effect on overall security even if the protocols are individually secure. It is therefore not clear a priori that two ID-AKE protocols can share identities without affecting each others security.

Compositions. Protocol composition can be achieved by describing conditions that if *violated* would break security; however users find surprising ways around such lists. Kelsey, Schneier and Wagner [15] outline *chosen protocol attacks* in which, given a target secure protocol, an attacker can create a different and stand-alone secure protocol which, when sharing state with the target protocol, results in security breaches. We focus on static information reuse that defines what is *allowed*, effectively adopting a conservative approach to shared states.

Pairings. Menezes, Okamoto and Vanstone [19] used pairing to solve the discrete logarithm problem on some elliptic curves. Sakai, Ohgishi and Kasahara [22] used them to devise an ID-based variant of the well-known Diffie-Hellman key agreement protocol [11]. With the work of Boneh and Franklin [3], ID-based primitives utilizing pairings gained widespread attention. They are used in ID-based encryption, signatures, signcryption and ring-signcryption protocols. However, to our knowledge the only result that is concerned with reusing the same public ID and the corresponding private key for two distinct primitives is due to González Vasco, Hess and Steinwandt [12]. We are not aware of any previous work dealing with shared IDs in ID-AKE.

Key Establishment. Key establishment is a fundamental cryptographic primitive and it is important to devise efficient protocols that satisfy strong security requirements. Security definitions for key agreement protocols were initially developed for two-party protocols by Bellare and Rogaway [1] and Blake-Wilson, Johnson and Menezes [2] in the shared-secret and public-key setting, respectively. Recent developments in two-party authenticated key exchange have improved the security models and definitions. These models better represent environments where protocols are deployed. Security definitions such as [16,17] allow leakage of information related to the test session; in addition, [18] accounts for relative timing of information leakage; lastly, [9] models the fact that government standards allow users to share static keys between different protocols. Security considerations for AKE protocols are similar to ID-AKE protocols. Therefore it is natural to adapt the strongest model to the ID-based setting and design protocols secure within these definitions.

While such an adoption is natural, much work remains to be done. There are alternatives to Sakai, Ohgishi and Kasahara [22], see for example [10,24]. Boyd and Choo [4] observe that many existing ID-based protocols are not as secure as we expect them to be. Also, to the best of our knowledge, the only ID-AKE work that formally considers ephemeral key leakage is due to Huang and Cao [14]; their protocols is henceforth referred to as the HC protocol. Unfortunately, the public keys of the HC protocol consists of two group elements so it is worthwhile to develop efficient and secure ID-AKE protocols resilient to ephemeral key leakage using shorter public keys. The protocols proposed by Boyd, Cliff, González-Nieto and Paterson [5] do not necessarily require pairing-based ID primitives, but in any case are not ephemeral key leakage resilient.

Universally composable security notion of key exchange is studied in [7], that considers security of key exchange composing with any protocol, while our security model considers only security of combination of two authenticated key exchange protocols. On the other hand, our security model captures leakage of static and ephemeral keys of test session, that is not captured in [7].

Our Contribution. In this work we extend the shared model of Chatterjee, Menezes and Ustaoğlu [9] to the ID-based setting. We propose two novel protocols that satisfy the new security definition and show that parties can safely reuse private keys in these two protocols. As the HC protocol we require rather strong assumption for formal security. Finally, we provide the efficiency comparison to the HC protocol.

Organization. In Section 2, we recall the bilinear group and the gap BDH assumption. In Section 3, we briefly outline our combined model. In Section 4, we propose our new protocols, give comments related to their security arguments and describe their design principles. In Section 5, we compare our protocols with existing relevant protocols. We conclude the paper in Section 6.

2 Preliminaries

Let κ be the security parameter and q be a 2κ-bit prime. Let $G = \langle g \rangle$ and G_T be cyclic groups of prime order q with generators g and g_T, respectively. Let

$e : G \times G \to G_T$ be a polynomial-time computable bilinear non-degenerate map called a pairing. We say that (G, G_T) is are bilinear groups with pairing e.

The gap BDH (Bilinear Diffie-Hellman) problem is as follows. The computational BDH function BDH : $G^3 \to G_T$ is $\mathrm{BDH}(U, V, W) = e(g, g)^{\log U \log V \log W}$ and the decisional BDH (Bilinear Diffie-Hellman) predicate BDDH : $G^4 \to \{0, 1\}$ 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 $U, V, W \in_U G$ selected uniformly random and oracle access to $\mathrm{BDDH}(\cdot,\cdot,\cdot,\cdot)$ oracle, and tries to compute $\mathrm{BDH}(U, V, W)$. For adversary $\mathcal{A}$, we define advantage

$$Adv^{\mathrm{gapBDH}}(\mathcal{A}) = \Pr[U, V, W \in_R G, \mathcal{A}^{\mathrm{BDDH}(\cdot,\cdot,\cdot,\cdot)}(U, V, W) = \mathrm{BDH}(U, V, W)],$$

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 satisfy the gap BDH assumption if, for all polynomial-time adversaries $\mathcal{A}$, advantage $Adv^{\mathrm{gapBDH}}(\mathcal{A})$ is negligible in security parameter κ.*

3 Shared Security Model for ID-Based AKE

Our model extends the shared static key model of Chatterjee, Menezes and Ustaoğlu [9] in a similar manny in which Huang and Cao [14] extend the LaMacchia, Lauter and Mityagin [17] model. Unlike Huang and Cao [14] we also allow our adversary to obtain private or public ephemeral session information before a session is initiated. Thus our model encompasses relative timing of ephemeral leakage.

We denote a party by U_i and the identifier of U_i by ID_i. We outline our model for two different two-pass Diffie-Hellman protocols, 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. We note that the order in which messages are exchanged is not important in practice as long as the session peers have consistent views about the information exchanged. However to simplify the exposition we assume a fixed order of message delivery. The model can be adapted to different protocols and number of rounds.

In the model, each party is a probabilistic polynomial-time Turing machine in security parameter κ and obtains a static private key corresponding to its identity string from a key generation center (KGC) via a secure and authentic channel. The center KGC uses a master secret key to generate individual private keys. We assume that the KGC never reveals the static private key for an identity string ID_i to two different parties. In other words, a malicious entity cannot obtain a static key corresponding to ID_i, unless the malicious entity is bound to ID_i.

Session. An invocation of a protocol is called a *session.* A session is activated via an incoming message of the forms $(\Pi_c, \mathcal{I}, ID_A, ID_B)$ or $(\Pi_c, \mathcal{R}, ID_A, ID_B, X_B)$, where Π_c is a protocol identifier. If U_A was activated with $(\Pi_c, \mathcal{I}, ID_A, ID_B)$, then U_A is the session *initiator*, otherwise the session *responder.* In the activation, Π_c identifies which protocol the party should execute. 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. A party U_A that has been successfully activated via $(\Pi_c, \mathcal{I}, ID_A, ID_B)$, can be further activated via $(\Pi_c, \mathcal{R}, ID_A, ID_B, X_A, X_B)$ to compute a session key. 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.

A session initiator U_A identifies the session via $(\Pi_c, \mathcal{I}, ID_A, ID_B, X_A, \times)$ or $(\Pi_c, \mathcal{I}, ID_A, ID_B, X_A, X_B)$. If U_A is the responder, the session is identified via $(\Pi_c, \mathcal{R}, ID_A, ID_B, X_B, X_A)$. For session $(\Pi_c, \mathcal{I}, ID_A, ID_B, X_A, X_B)$ the *matching session* has identifier $(\Pi_c, \mathcal{R}, ID_B, ID_A, X_A, X_B)$ 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. For further details on session activation, abortion and matching sessions with incomplete identifiers we refer to [9].

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_c, ID_A, ID_B), (Π_c, ID_A, ID_B, X_A), or $(\Pi_c, ID_A, ID_B, X_A, X_B)$; Π_c is a protocol identifier. Each party submits its responses to the adversary, who decides the global delivery order. Note that the adversary does not control the communication between parties and the key generation center. It is possible to incorporate into the model the ability of the adversary to time when a party obtains a static private key. For simplicity, we assume that identities and corresponding static keys are part of $\mathcal{A}$'s input.

A party's private information is not accessible to the adversary; however, leakage of private information is captured via the following adversary queries. For details relating to incomplete session identifiers and discussion related to EphemeralPublicKeyReveal we refer the reader to [18]:

- SessionKeyReveal(`sid`). The adversary obtains the session key for the session `sid`, provided that the session holds a session key.
- EphemeralPublicKeyReveal(ID_i). The adversary obtains the ephemeral public key that ID_i will use when a session is next activated at ID_i.
- 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 system.
- EstablishParty(ID_i). This query allows the adversary to register a static public key on behalf of a party U_i; the adversary totally controls that party. If

a party pid is established by an $\mathsf{EstablishParty}(ID_i)$ query issued by the adversary, then we call the party *dishonest*. If not, we call the party *honest*. This query models malicious insiders.

Our security definition requires the notion of "freshness". The protocols we consider have the same security attributes and a single definition suffices.

Definition 2 (Freshness). *Let* $\texttt{sid}^*$ *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* $\overline{\texttt{sid}^*}$ *be the session identifier of the matching session of* $\texttt{sid}^*$*. Define* $\texttt{sid}^*$ *to be fresh if none of the following conditions hold:*

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

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

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 $\mathsf{Test}(\texttt{sid}^*)$ and is given with equal probability either the session key held by $\texttt{sid}^*$ or a random key; the query does not terminate the experiment. 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 $\texttt{sid}^*$ is fresh at the end of $\mathcal{A}$'s execution and if $\mathcal{A}$'s guess was correct. Formally,

Definition 3 (security). *The advantage of the adversary* $\mathcal{A}$ *in the experiment with AKE protocols* Π_1 *and* Π_2 *is defined as*

$$\mathrm{Adv}^{\mathrm{AKE}}_{\Pi_1\Pi_2}(\mathcal{A}) = \Pr[\mathcal{A}\ wins] - \frac{1}{2}.$$

We say that Π_1 *and* Π_2 *are secure AKE protocols in the shared identity-based model if the following conditions hold:*

1. *If two honest parties complete matching* Π_c*-sessions, then, except with negligible probability in security parameter* κ*, they both compute the same session key.*
2. *For any probabilistic polynomial-time bounded adversary* $\mathcal{A}$*,* $\mathrm{Adv}^{\mathrm{AKE}}_{\Pi_1\Pi_2}(\mathcal{A})$ *is negligible in security parameter* κ*.*

Remark. If the adversary initiates sessions of only one protocol, issues neither $\mathsf{EphemeralPublicKeyReveal}(ID_i)$ queries and nor $\mathsf{EphemeralKeyReveal}(\mathtt{sid})$ queries except during the time between session initiation and completion, then the model is equivalent to the Huang-Cao [14] model.

4 Proposed ID-Based AKE Protocols

This section describes our ID-based protocols. We let $H : \{0,1\}^* \to \{0,1\}^k$, $H_1 : \{0,1\}^* \to G$, and $H_2 : \{0,1\}^* \to \mathbb{Z}_q$ be cryptographic hash function modeled as random oracles.

Key Generation Center. The KGC randomly selects master secret key $z \in_R \mathbb{Z}_q$ and publishes master public key $Z = g^z \in G$.

Private Key Generation. Given ID string $ID_i \in \{0,1\}^*$ of user U_i, the KGC computes $Q_i = H_1(ID_i)$ and returns static secret key $D_i = Q_i^z$.

4.1 Proposed ID-Based AKE Protocol 1 – Π_1

In this section, we describe the actions required to execute a Π_1 session.

Key Exchange. User U_A is the session initiator and user U_B is the session responder.

1. U_A chooses an ephemeral private key $x_A \in_R \mathbb{Z}_q$, computes the ephemeral public key $X_A = g^{x_A}$ and sends (Π_1, ID_A, ID_B, X_A) to U_B.
2. Upon receiving (Π_1, ID_A, ID_B, X_A), U_B chooses an ephemeral private key $x_B \in_R \mathbb{Z}_q$, computes the ephemeral public key $X_B = g^{x_B}$ and responds to U_A with $(\Pi_1, ID_A, ID_B, X_A, X_B)$.
   U_B also computes $e_A = H_2(X_A), e_B = H_2(X_B)$, the shared secrets
   $$\sigma_1 = e(Q_A^{e_A} X_A, D_B Z^{x_B}),\ \sigma_2 = e(Q_A X_A, D_B^{e_B} Z^{x_B}),\ \sigma_3 = X_A^{x_B},$$
   the session key $K = H(\sigma_1, \sigma_2, \sigma_3, \Pi_1, ID_A, ID_B, X_A, X_B)$. U_B completes the session with session key K.
3. Upon receiving $(\Pi_1, ID_A, ID_B, X_A, X_B)$, U_A computes $e_A = H_2(X_A), e_B = H_2(X_B)$, the shared secrets
   $$\sigma_1 = e(D_A^{e_A} Z^{x_A}, Q_B X_B),\ \sigma_2 = e(D_A Z^{x_A}, Q_B^{e_B} X_B),\ \sigma_3 = X_B^{x_A},$$
   the session key $K = H(\sigma_1, \sigma_2, \sigma_3, \Pi_1, ID_A, ID_B, X_A, X_B)$. U_A completes the session with session key K.

Both parties compute the shared secrets

$$\sigma_1 = g_T^{z(e_A \log(Q_A)+x_A)(\log(Q_B)+x_B)},$$
$$\sigma_2 = g_T^{z(\log(Q_A)+x_A)(e_B \log(Q_B)+x_B)},$$
$$\sigma_3 = g^{x_A x_B}$$

and therefore compute the same session key K.

It is worth noting that we could also construct secure ID-based AKE by modifying σ_1 and σ_2 to the following values

$$\sigma_1 = g_T^{z(e_A \log(Q_A)+x_A)(e_B \log(Q_B)+x_B)},\ \sigma_2 = g_T^{z(\log(Q_A)+x_A)(\log(Q_B)+x_B)};$$

however, we opt to define our second protocol via slightly more efficient algorithms.

4.2 Proposed ID-Based AKE Protocol 2 – Π_2

In this section, we describe the actions required to execute a Π_2 session.

Key Exchange. User U_A is the session initiator and user U_B is the session responder.

1. U_A chooses an ephemeral private key $x_A \in_R \mathbb{Z}_q$, computes the ephemeral public key $X_A = g^{x_A}$ and sends (Π_2, ID_A, ID_B, X_A) to U_B.
2. Upon receiving (Π_2, ID_A, ID_B, X_A), U_B chooses an ephemeral private key $x_B \in_R \mathbb{Z}_q$, computes the ephemeral public key $X_B = g^{x_B}$ and responds to U_A with $(\Pi_2, ID_A, ID_B, X_A, X_B)$.
   U_B also computes the shared secrets
   $$\sigma_1 = e(Q_A X_A, D_B Z^{x_B}),\ \sigma_2 = e(Q_A, D_B),\ \sigma_3 = X_A^{x_B}$$
   the session key $K = H(\sigma_1, \sigma_2, \sigma_3, \Pi_2, ID_A, ID_B, X_A, X_B)$. U_B completes the session with session key K.
3. Upon receiving $(\Pi_2, ID_A, ID_B, X_A, X_B)$, U_A computes the shared secrets
   $$\sigma_1 = e(D_A Z^{x_A}, Q_B X_B),\ \sigma_2 = e(D_A, Q_B),\ \sigma_3 = X_B^{x_A}$$
   the session key $K = H(\sigma_1, \sigma_2, \sigma_3, \Pi_2, ID_A, ID_B, X_A, X_B)$. U_A completes the session with session key K.

Both parties compute the shared secrets

$$\sigma_1 = g_T^{z(\log(Q_A)+x_A)(\log(Q_B)+x_B)},\ \sigma_2 = g_T^{z \log(Q_A) \log(Q_B)},\ \sigma_3 = g^{x_A x_B},$$

and therefore compute the same session key K.

4.3 Security

The security of the proposed ID-AKE protocols is established by the following theorem.

Theorem 1. *If (G, G_T) are groups where gap Bilinear Diffie-Hellman assumption holds and H, H_1 and H_2 are random oracles, the ID-AKE Protocols Π_1 and Π_2 are secure in the shared static key model described in Section 3.*

The proof of Theorem 1 is provided in Appendix A.

4.4 Shared Secrets Design

Protocol Π_1. Protocol Π_1 is an adaptation of the Unified Protocol in [25] to the ID-based scenario. We refer the reader to [25] for the rationale behind the definitions of σ_1 and σ_2. Unlike the certificated-based scenario in the ID-based scenario without σ_3 the protocol has no forward secrecy with respect to the master secret key. To provide assurance that the KGC cannot compute the session key we include σ_3. There are other viable alternatives; however, we opt for the basic Diffie-Hellman value due to its relative simplicity.

Protocol Π_2. Protocol Π_2 is an adaptation of the Unified Model protocol [2,20]. More precisely, σ_2 and σ_3 are the static and ephemeral shared secrets, respectively, that Alice and Bob can compute in the ID-based scenario. The certificate-based variant is vulnerable to some attacks such as key compromise impersonation, which are carried over to the ID-based setting. We include the value σ_1 to remove these Unified Model protocol drawbacks.

Computational cost. The naive count of operations for Π_1 and Π_2 show that the former requires $2P+4E$ and the latter $2P+2E$, where P stands for pairing computation and E stands for group exponentiation. We do not take into account the exponentiation required to prepare the outgoing ephemeral public key since in our analysis these can be pre-computed; it is also not included in the protocol comparison in Section 5. Since pairing computations are more costly than exponentiations [8,13] the computational cost of the two protocols is effectively the same. We include both measures because novel techniques may reduce the cost of pairing computations significantly.

Efficiency. The certificate-based motivating protocol for Π_1 is less efficient but provides stronger security assurances than the certificate-based motivating protocol for Π_2. Surprisingly, in the ID-based scenario Π_2 is marginally more efficient than Π_1 and has the same security attributes. This suggest that while certificate-based protocols can help in the design of ID-based protocols, it is not necessary that more efficient certificate based protocols result in more efficient ID-based protocols. A single pairing to produce a session key is a viable option. However, as in the certificate-based setting better efficiency may require larger group sizes for comparable security levels. In particular, an ID-based adaptation of the HMQV protocol [16] would suffer from a less tight security argument because of forking arguments. We therefore opt for protocols with security arguments that do not require forking lemma type arguments.

Protocol choice. One may naturally ask why use Π_1 when Π_2 already provides the same security attributes and has some efficiency advantages. SP800-56A [20] provides two techniques for key establishment and the "unified model" protocol is both less efficient and has fewer security attributes. But, it is advantageous to offer fall-back alternatives to accommodate any unexpected environmental reasons that prevent the use of a particular protocol. Our arguments show that sharing identities for Π_1 and Π_2 is cryptographically sound.

4.5 Further Observations and Comments

Theorem 1 implies that the protocols are secure even if they are used separately. The security argument for protocol Π_1 say is a specialization of the execution model where the adversary does not activate a session of protocol Π_2. Thus, the theorem provides stronger assurance than separate security arguments.

The essential set of message that Alice (U_A) sends to Bob (U_B) consists of four messages $(\Pi_c, role, ID_A, X_A)$ – the protocol identifier, the role Alice views for herself, the identifier she uses and her ephemeral public key. The response from Bob consists of a similar tuple $(\Pi_c', role', ID_B, X_B)$. The first two entries in Bob's case simply affirm that he agrees to Alice's choice of protocol and role. Our analysis is carried out only in the case where Alice sends her tuple at once and *then* receives all of Bob's response. It is possible that all these messages are interleaved: for example in the first flow Alice can send only $(\Pi_c, role, U_A)$ to Bob and reveal her identifier ID_A only after receiving Bob's response. For simplicity we have omitted those technical details required in the model description. However it appears that the only requirement is that parties associate correctly incoming messages with sessions.

As in the HC protocol, Alice does not need to know her static private key to complete the message exchange in our protocols. This is unlike the Okamoto-Tanaka protocol [21], where a party needs its static private key before being able to compute outgoing messages. The feature is useful in scenarios where users select identities "on the fly" depending on some ephemeral system setting such as the day or time.

5 Comparison

Next we compare our protocols with related ID-AKE protocols in terms of underlying assumption, computational efficiency, and security model. In Table 1, number of pairing computations, the number of exponentiations in G, number of static public keys in terms of group elements, and number of ephemeral public key in terms of group elements are denoted by P, E, #sPK, and #ePK, respectively. Furthermore, id-CK and id-eCK denotes ID-based versions of the well-know Canetti-Krawzcyk [6] (CK) and LaMacchia, Lauter and Mityagin [17] (eCK) security models, respectively. Our model is denoted by id-eCK*; KCI denotes key compromise impersonation resistance and mSk-fs denotes master secret key forward security.

For further comparison we refer the reader to [14, Table 1]. It is plausible that the HC protocol is also secure if ephemeral public keys are revealed to the adversary before used in a session. Our protocols are as efficient as the HC protocol (the cost of a single exponentiation can be safely ignored here) and therefore the trade-off lies in the size of the private keys and the underlying assumptions. Such a comparison is very subjective but we believe that for practical purposes the use of shorter keys justifies the invocation of the gap assumption.

Table 1. Comparison with the existing protocols

Protocol	Computation	#sPk	#ePk	Security Model	Assumption
SCK [10]	2P+3E	1	1	id-CK,KCI,mSk-fs	BDH
SYL [10]	1P+3E	1	1	id-CK,KCI,mSk-fs	BDH
HC [14]	2P+3E	2	1	id-eCK	BDH
Π_1	2P+4E	1	1	id-eCK*	gap BDH
Π_2	2P+2E	1	1	id-eCK*	gap BDH

6 Conclusion

In this paper, we proposed the first secure ID-based AKE protocols that used a single group element as a static secret key, resist ephemeral key leakage, and can safely share static private information. Moreover, our protocols are efficient in terms of communication and computations and thus are suitable for practical applications. It is an interesting problem to consider developing a protocol with the same setup and communication message, but which requires a single pairing operation, such as an adaptation of the HMQV [16] protocol to the ID-based setting. Additionally, it is worth developing a new protocol that can be used in conjunction with protocol Π_2, and uses a single random oracle call during the session key computation stage.

Acknowledgments

We thank Alfred Menezes for helpful comments.

References

1. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994); Full version available at `http://www.cs.ucdavis.edu/~rogaway/papers/eakd-abstract.html`
2. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Darnell, M. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 30–45. Springer, Heidelberg (1997)
3. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
4. Boyd, C., Choo, K.-K.R.: Security of two-party identity-based key agreement. In: Dawson, E., Vaudenay, S. (eds.) MYCRYPT 2005. LNCS, vol. 3715, pp. 229–243. Springer, Heidelberg (2005)
5. Boyd, C., Cliff, Y., González Nieto, J.M., Paterson, K.: Efficient one-round key exchange in the standard model. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 69–83. Springer, Heidelberg (2008); Full version available at `http://eprint.iacr.org/2008/007/`

6. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001); Full version available at `http://eprint.iacr.org/2001/040/`
7. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)
8. Chatterjee, S., Hankerson, D., Knapp, E., Menezes, A.: Comparing two pairing-based aggregate signature schemes. Designs, Codes and Cryptography 55(2), 141–167 (2010)
9. Chatterjee, S., Menezes, A., Ustaoğlu, B.: Reusing static keys in key agreement protocols. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 39–56. Springer, Heidelberg (2009)
10. Chen, L., Cheng, Z., Smart, N.P.: Identity-based key agreement protocols from pairings. International Journal of Information Security 6(4), 213–241 (2007)
11. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory IT-22(6), 644–654 (1976)
12. González Vasco, M.I., Hess, F., Steinwandt, R.: Combined (identity-based) public key schemes. Cryptology ePrint Archive, Report 2008/466 (2008), `http://eprint.iacr.org/2008/466`
13. Hankerson, D., Menezes, A., Scott, M.: Software implementation of pairings. In: Joye, M., Neven, G. (eds.) Identity-Based Cryptography. Cryptology and Information Security, vol. 2, ch. XII, pp. 188–206. IOS Press, Amsterdam (2008)
14. Huang, H., Cao, Z.: An id-based authenticated key exchange protocol based on bilinear diffie-hellman problem. In: Safavi-Naini, R., Varadharajan, V. (eds.) ASIACCS 2009: Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, New York, NY, USA, pp. 333–342 (2009)
15. Kelsey, J., Schneier, B., Wagner, D.: Protocol interactions and the chosen protocol attack. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) SP 1997. LNCS, vol. 1361, pp. 91–104. Springer, Heidelberg (1998)
16. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Cramer, R. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
17. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
18. Menezes, A., Ustaoğlu, B.: Comparing the pre- and post-specified peer models for key agreement. International Journal of Applied Cryptography (IJACT) 1(3), 236–250 (2009)
19. Menezes, A.J., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory 39(5), 1639–1646 (1993)
20. NIST National Institute of Standards and Technology. Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (March 2007), `http://csrc.nist.gov/publications/PubsSPs.html`
21. Okamoto, E., Tanaka, K.: Key distribution system based on identification information. IEEE Journal on Selected Arean in Communications 7(4), 481–485 (1989)
22. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairings. In: The 2000 Symposium on Cryptography and Information Security (2000)

23. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1984)
24. Smart, N.P.: Identity-based authenticated key agreement protocol based on weil pairing. Electronic Letters 38(13), 630–632 (2002)
25. Ustaoğlu, B.: Comparing *SessionStateReveal* and *EphemeralKeyReveal* for Diffie-Hellman protocols. In: Pieprzyk, J., Zhang, F. (eds.) ProvSec 2009. LNCS, vol. 5848, pp. 183–197. Springer, Heidelberg (2009)

A Proof of Theorem 1

We call the variant of the gap BDH assumption where one tries to compute $\mathrm{BDH}(U, U, W)$ instead of $\mathrm{BDH}(U, V, W)$ as the square gap BDH assumption. The variant is equivalent to the gap BDH assumption as follows. Given a challenge U, W of the square gap BDH assumption, one sets $V = U^s$ for random integers $s \in_R [1, p-1]$, and then can compute $\mathrm{BDH}(U, V, W)^{1/s} = \mathrm{BDH}(U, U, W)$. Given a challenge (U, V, W) of the gap BDH assumption, one can set $U_1 = UVW$, $U_2 = UVW^{-1}$, $U_3 = UV^{-1}W$, $U_4 = UV^{-1}W^{-1}$, and then compute $\mathrm{BDH}(U, V, W)$ from $\mathrm{BDH}(U_i, U_i, U_i^s)^{1/s}$, $i = 1, \ldots, 4$.

We will show that no polynomially bounded adversary can distinguish the session key of a fresh session from a randomly chosen session key.

Let κ denote the security parameter, and let $\mathcal{A}$ be a polynomially (in κ) bounded adversary. We use $\mathcal{A}$ to construct a gap BDH solver $\mathcal{S}$ that succeeds with non-negligible probability. The adversary $\mathcal{A}$ is said to be successful with non-negligible probability if $\mathcal{A}$ wins the distinguishing game with probability $\frac{1}{2} + f(\kappa)$, where $f(\kappa)$ is non-negligible, and the event M denotes a successful $\mathcal{A}$.

Assume that $\mathcal{A}$ succeeds in an environment with n users, activates at most s sessions within a user, makes at most q_H, q_{H_1}, q_{H_2} queries to oracles H, H_1, H_2, respectively.

We denote the master secret and public keys by $z, Z = g^z$. For user U_i, we denote the identity by ID_i, the static secret and public keys by $D_i = Q_i^z, Q_i = H_1(ID_i)$, and the ephemeral secret and public keys by $x_i, X_i = g^{x_i}$. We also denote the session key by K.

Let the test session be $\mathtt{sid}^* = (\Pi_c, ID_A, ID_B, X_A, X_B)$, where users U_A, U_B are initiator and responder of the test session $\mathtt{sid}^*$. Let H^* be the event that $\mathcal{A}$ queries $(\sigma_1, \sigma_2, \sigma_3, \mathtt{sid}^*)$ 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 $sid \neq \mathtt{sid}^*$ 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, $\mathcal{A}$ 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 $\Pr(M \wedge H^*) \geq p(\kappa)$. Henceforth the event $M \wedge H^*$ is denoted by M^*.

We will consider the not exclusive classification of all possible events in the following Table 2 and 3. Table 2 classifies events when Q_A, Q_B are distinct, and Table 3 classifies events when $Q_A = Q_B$. We denote by z master secret key, and by $(D_i, x_i)_{i=A,B}$ static and ephemeral secret keys of users U_A, U_B who are initiator and responder of the test session sid*. In these tables, "ok" means the static key is not revealed, or the matching session exists and the ephemeral key is not revealed. "r" means the static or ephemeral key may be revealed. "r/n" means the ephemeral key may be revealed if the matching session exists or no matching session exists. "instance embedding" row shows how simulator embeds a instance of gap BDH problem. "succ. prob." row shows the probability of success of solver $\mathcal{S}$, where $p_{xy} = Pr(E_{xy} \wedge M^*)$ and n and s is the number of parties and sessions.

Since the classification covers all possible events, at least one event $E_{xy} \wedge M^*$ 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, and that means we shows that the proposed protocol is secure under the gap BDH assumption. We will investigate each of these events in the following subsections.

Table 2. Classification of attacks, when Q_A, Q_B are distinct. "ok" means the static key is not revealed, or the matching session exists and the ephemeral key is not revealed. "r" means the static or ephemeral key may be revealed. "r/n" means the ephemeral key may be revealed if the matching session exists or no matching session exists. "instance embedding" row shows how simulator embeds a instance of gap BDH problem. "succ. prob." row shows the probability of success of solver $\mathcal{S}$, where $p_{xy} = Pr(E_{xy} \wedge M^*)$ and n and s is the number of parties and sessions.

	z	D_A	x_A	D_B	x_B	instance embedding	succ. prob.
E_{1a}	ok	r	ok	ok	r/n	$Z = U, X_A = V, Q_B = W$	$p_{1a}/n^2 s$
E_{1b}	ok	ok	r	ok	r/n	$Z = U, Q_A = V, Q_B = W$	p_{1b}/n^2
E_{2a}	r	r	ok	r	ok	$X_A = V, X_B = W$	$p_{2a}/n^2 s^2$
E_{2b}	ok	ok	r	r	ok	$Z = U, Q_A = V, X_B = W$	$p_{2b}/n^2 s$

Table 3. Classification of attacks, when $Q_A = Q_B$

	z	D_A	x_A	D_A	x_B	instance embedding	succ.prob.
E'_{1b}	ok	ok	r	ok	r/n	$Z = U, Q_A = V$	p_{1b}/n
E'_{2a}	r	r	ok	r	ok	$X_A = V, X_B = W$	$p_{2a}/n^2 s^2$

A.1 E_{1a}

Setup. The gap BDH solver $\mathcal{S}$ begins by establishing n honest users that are assigned random static key pairs. For each honest user U_i, $\mathcal{S}$ maintains list L_{EK} of at most s ephemeral key pairs, and two markers – a user marker and an adversary marker. The list L_{EK} is initially empty, and the markers initially point to the first entry of the list L_{EK}. Whenever U_i is activated to create a new session, $\mathcal{S}$ checks if the user marker points to an empty entry. If so, $\mathcal{S}$ selects a new ephemeral key pair on behalf of U_i as described in the Π_c $(c = 1, 2)$ protocol.

If the list entry is not empty, then $\mathcal{S}$ uses the ephemeral key pair in that list entry for the newly created session. In either case the user marker is updated to point to the next list entry, and the adversary marker is also advanced if it points to an earlier entry. If $\mathcal{A}$ issues an EphemeralPublicKeyReveal query, then $\mathcal{S}$ selects a new ephemeral key pair on behalf of U_i as described in the Π_c protocol. $\mathcal{S}$ stores the key pair in the entry pointed to by the adversary marker, returns the public key as the query response, and advances the adversary marker.

In addition to the above steps, $\mathcal{S}$ embeds instance (U, V, W) of gap BDH problem as follows. $\mathcal{S}$ randomly selects two users U_A, U_B and integer $t \in_R [1, s]$. Public master key Z is chosen to be U, and D_A is chosen to be Z^{q_A} for randomly selected $q_A = \log(Q_A)$. $\mathcal{S}$ simulates oracle H_1 by selecting a random integer c in the interval $[1, q]$ and setting $H_1(ID_C) = g^c$; the static private key corresponding to ID_C is U^c. $\mathcal{S}$ selects static and ephemeral key pairs on behalf of honest users as described above with the following exceptions. The i-th ephemeral public key X selected on behalf of U_A is chosen to be V, and the static public key Q_B selected on behalf of U_B is chosen to be W. $\mathcal{S}$ does not possess the corresponding static and ephemeral private keys.

The algorithm $\mathcal{S}$ activates $\mathcal{A}$ 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.

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

1. Send$(\Pi_c, \mathcal{I}, ID_i, ID_j)$: $\mathcal{S}$ picks ephemeral public key X_i from the list L_{EK}, and records (Π_c, ID_i, ID_j, X_i) and returns it.
2. Send$(\Pi_c, \mathcal{R}, ID_j, ID_i, X_i)$: $\mathcal{S}$ picks ephemeral public key X_j from the list L_{EK}, and records $(\Pi_c, ID_i, ID_j, X_i, X_j)$ and returns it.
3. Send$(\Pi_c, \mathcal{I}, ID_i, ID_j, X_i, X_j)$: If (Π_c, ID_i, ID_j, X_i) is not recorded, then $\mathcal{S}$ records the session $(\Pi_1, \mathcal{I}, ID_i, ID_j, X_i, X_j)$ as not completed. Otherwise, $\mathcal{S}$ records the session as completed.
4. $H(\sigma_1, \sigma_2, \sigma_3, \Pi_1, ID_i, ID_j, X_i, X_j)$:
   (a) If $(\sigma_1, \sigma_2, \sigma_3, \Pi_1, ID_i, ID_j, X_i, X_j)$ is recorded in list L_H, then $\mathcal{S}$ returns recorded value K.
   (b) Else if the session $(\Pi_1, \mathcal{I}, ID_i, ID_j, X_i, X_j)$ or $(\Pi_1, \mathcal{R}, ID_j, ID_i, X_i, X_j)$ is recorded in list L_H, then $\mathcal{S}$ checks that $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, i.e.,

$$\mathrm{BDDH}(Z, Q_i^{e_i} X_i, Q_j X_j, \sigma_1) = 1,$$
$$\mathrm{BDDH}(Z, Q_i X_i, Q_j^{e_j} X_j, \sigma_2) = 1$$

   and $e(X_i, X_j) = \sigma_3$. If $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, then $\mathcal{S}$ returns recorded value K and records it in list L_H.

(c) Else if $i = A, j = B, X_A = V, Q_B = W$, then $\mathcal{S}$ checks that $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, i.e.,

$$\mathrm{BDDH}(Z, Q_i^{e_i} X_i, Q_j X_j, \sigma_1) = 1,$$
$$\mathrm{BDDH}(Z, Q_i X_i, Q_j^{e_j} X_j, \sigma_2) = 1,$$

and $e(X_i, X_j) = \sigma_3$. If $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, then since $\mathcal{S}$ knows $\log(Q_A)$, $\mathcal{S}$ can compute the answer to the gap BDH instance

$$((\sigma_1')(\sigma_2')^{-1})^{1/(1-e_B)} = g_T^{zx_A \log(Q_B)} = \mathrm{BDH}(Z, X_A, Q_B),$$

where

$$\sigma_1' = \sigma_1 e(Z, Q_B X_B)^{-e_A \log(Q_A)} = g_T^{zx_A(\log(Q_B)+x_B)},$$
$$\sigma_2' = \sigma_2 e(Z, Q_B^{e_B} X_B)^{-\log(Q_A)} = g_T^{zx_A(e_B \log(Q_B)+x_B)},$$

and stops successfully by outputting the answer.

(d) Otherwise, $\mathcal{S}$ returns random value K and records it in list L_H.

5. $H(\sigma_1, \sigma_2, \sigma_3, \Pi_2, ID_i, ID_j, X_i, X_j)$:

(a) If $(\sigma_1, \sigma_2, \sigma_3, \Pi_2, ID_i, ID_j, X_i, X_j)$ is recorded in list L_H, then $\mathcal{S}$ returns recorded value K.

(b) Else if the session $(\Pi_2, \mathcal{I}, ID_i, ID_j, X_i, X_j)$ or $(\Pi_2, \mathcal{R}, ID_j, ID_i, X_i, X_j)$ is recorded in list L_H, then $\mathcal{S}$ checks that $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, i.e.,

$$\mathrm{BDDH}(Z, Q_i X_i, Q_j X_j, \sigma_1) = 1,$$
$$\mathrm{BDDH}(Z, Q_i, Q_j, \sigma_2) = 1,$$

and $e(X_i, X_j) = \sigma_3$. If $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, then $\mathcal{S}$ returns recorded value K and records it in list L_H.

(c) Else if $i = A, j = B, X_A = V, Q_B = W$, then $\mathcal{S}$ checks that $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, i.e.,

$$\mathrm{BDDH}(Z, Q_i X_i, Q_j X_j, \sigma_1) = 1,$$
$$\mathrm{BDDH}(Z, Q_i, Q_j, \sigma_2) = 1,$$

and $e(X_i, X_j) = \sigma_3$. If $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, then since $\mathcal{S}$ knows $\log(Q_A)$, $\mathcal{S}$ can compute the answer of the gap BDH instance

$$\sigma_1 \tau_1^{-1} \tau_2^{-1} \tau_3^{-1} = g_T^{zx_A \log(Q_B)} = \mathrm{BDH}(Z, X_A, Q_B),$$

where

$$\tau_1 = \sigma_2 = g_T^{z \log(Q_A) \log(Q_B)},$$
$$\tau_2 = e(Z, \sigma_3) = g_T^{zx_A x_B},$$
$$\tau_3 = e(Z, X_B)^{\log(Q_A)} = g_T^{z \log(Q_A) x_B},$$

and stops successfully by outputting the answer.

(d) Otherwise, $\mathcal{S}$ returns random value K and records it in list L_H.

6. SessionKeyReveal($(\Pi_1, \mathcal{I}, ID_i, ID_j, X_i, X_j)$ or $(\Pi_1, \mathcal{R}, ID_j, ID_i, X_i, X_j)$):
   (a) If the session `sid` is not completed, $\mathcal{S}$ returns error.
   (b) Else if the session `sid` is recorded in list L_S, then $\mathcal{S}$ returns recorded value K.
   (c) Else if $(\sigma_1, \sigma_2, \sigma_3, \Pi_1, ID_i, ID_j, X_i, X_j)$ is recorded in list L_H, then $\mathcal{S}$ checks that $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, i.e.,

   $$\begin{aligned} \mathrm{BDDH}(Z, Q_i^{e_i} X_i, Q_j X_j, \sigma_1) &= 1, \\ \mathrm{BDDH}(Z, Q_i X_i, Q_j^{e_j} X_j, \sigma_2) &= 1, \end{aligned}$$

   and $e(X_i, X_j) = \sigma_3$. If $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, then $\mathcal{S}$ returns recorded value K and records it in list L_S.
   (d) Otherwise, $\mathcal{S}$ returns random value K and records it in list L_S.
7. SessionKeyReveal($(\Pi_2, \mathcal{I}, ID_i, ID_j, X_i, X_j)$ or $(\Pi_2, \mathcal{R}, ID_j, ID_i, X_i, X_j)$):
   (a) If the session `sid` is not completed, $\mathcal{S}$ returns error.
   (b) Else if the session `sid` is recorded in list L_S, then $\mathcal{S}$ returns recorded value K.
   (c) Else if $(\sigma_1, \sigma_2, \sigma_3, \Pi_2, ID_i, ID_j, X_i, X_j)$ is recorded in list L_H, then $\mathcal{S}$ checks that $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, i.e.,

   $$\begin{aligned} \mathrm{BDDH}(Z, Q_i X_i, Q_j X_j, \sigma_1) &= 1, \\ \mathrm{BDDH}(Z, Q_i, Q_j, \sigma_2) &= 1, \end{aligned}$$

   and $e(X_i, X_j) = \sigma_3$. If $\sigma_1, \sigma_2, \sigma_3$ are correctly formed, then $\mathcal{S}$ returns recorded value K and records it in list L_S.
   (d) Otherwise, $\mathcal{S}$ returns random value K and records it in list L_S.
8. $H_1(ID_C)$: If $C = B$, $\mathcal{S}$ returns $Q_B = W$, otherwise selects a random integer c in the interval $[1, q]$ and return g^c.
9. $H_2(X_i)$: $\mathcal{S}$ simulates random oracle in the usual way.
10. EphemeralPublicKeyReveal(`sid`): $\mathcal{S}$ picks ephemeral public key X from the list L_{EK}, and returns X.
11. EphemeralKeyReveal(`sid`): $\mathcal{S}$ picks ephemeral secret key x from the list L_{EK}. If the corresponding ephemeral public key is V, then $\mathcal{S}$ aborts with failure, otherwise returns x.
12. StaticKeyReveal(ID_i): If static public key Q_i of user U_i is W, then $\mathcal{S}$ aborts with failure, otherwise responds to the query faithfully.
13. MasterKeyReveal(): $\mathcal{S}$ aborts with failure.
14. EstablishParty(ID_i): $\mathcal{S}$ responds to the query faithfully.
15. Test(`sid`): If ephemeral public key of a user is not V and static public key of the other user not W in session `sid`, then $\mathcal{S}$ aborts with failure, otherwise responds to the query faithfully.
16. If $\mathcal{A}$ outputs a guess γ, $\mathcal{S}$ aborts with failure.

Analysis. The simulation of $\mathcal{A}$ environment is perfect except with negligible probability. The probability that $\mathcal{A}$ selects the session, where ephemeral public key of a user is V and static public keys of the other users are W, as the test session $\texttt{sid}^*$ is at least $\frac{1}{n^2 s}$. Suppose this is indeed the case, $\mathcal{S}$ does not abort in Step 15, and suppose event $E_{1a} \wedge M^*$ occurs, $\mathcal{S}$ does not abort in Step 11, Step 12, and Step 13.

Under event M^* except with negligible probability, $\mathcal{A}$ queries H with correctly formed $\sigma_1, \sigma_2, \sigma_3$. Therefore $\mathcal{S}$ is successful as described in Step 4c or Step 5c and does not abort as in Step 16.

Hence, $\mathcal{S}$ is successful with probability $Pr(S) \geq \frac{p_{1a}}{n^2 s}$, where p_{1a} is probability that $E_{1a} \wedge M^*$ occurs.

A.2 E_{1b}

Same as the event $E_{1a} \wedge M^*$ in Subsection A.1, except the following points.

In Setup, $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $Z = U, Q_A = V, Q_B = W$, and selects randomly x_A and simulates as $X_A = g^{x_A}$.

In Simulation of H, $\mathcal{S}$ extracts BDH(U, V, W) as follows. In Step 4c, since $\mathcal{S}$ knows x_A, $\mathcal{S}$ extracts

$$((\sigma_1')^{1/e_A}(\sigma_2')^{-1})^{1/(1-e_B)} = g_T^{z \log(Q_A) \log(Q_B)} = \mathrm{BDH}(Z, Q_A, Q_B),$$

where

$$\sigma_1' = \sigma_1 e(Z, Q_B X_B)^{-x_A} = g_T^{z e_A \log(Q_A)(\log(Q_B)+x_B)},$$
$$\sigma_2' = \sigma_2 e(Z, Q_B^{e_B} X_B)^{-x_A} = g_T^{z \log(Q_A)(e_B \log(Q_B)+x_B)}.$$

In Step 5c, $\mathcal{S}$ extracts

$$\sigma_2 = g_T^{z \log(Q_A) \log(Q_B)} = \mathrm{BDH}(Z, Q_A, Q_B).$$

A.3 E_{2a}

Same as the event $E_{1a} \wedge M^*$ in Subsection A.1, except the following points.

In Setup, $\mathcal{S}$ embeds gap CDH instance (V, W) as $X_A = V, X_B = W$, selects uniformly at random master private key z and computes the corresponding master public key $Z = g^z$. The oracle H_1 is simulated honestly private keys of all parties are computed using z.

In Simulation of H, $\mathcal{S}$ extracts BDH(U, V, W) as follows. In Step 4c, since $\mathcal{S}$ knows $\log(Q_A)$, $\mathcal{S}$ extracts

$$((\sigma_1')^{e_B}(\sigma_2')^{-1})^{1/(e_B-1)} = g_T^{z x_A x_B} = \mathrm{BDH}(Z, X_A, X_B),$$

where

$$\sigma_1' = \sigma_1 e(Z, Q_B X_B)^{-e_A \log(Q_A)} = g_T^{z x_A(\log(Q_B)+x_B)},$$
$$\sigma_2' = \sigma_2 e(Z, Q_B^{e_B} X_B)^{-\log(Q_A)} = g_T^{z x_A(e_B \log(Q_B)+x_B)}.$$

In Step 5c, $\mathcal{S}$ extracts

$$e(Z, \sigma_3) = g_T^{z x_A x_B} = \mathrm{BDH}(Z, X_A, X_B).$$

A.4 E_{2b}

Same as the event $E_{1a} \wedge M^*$ in Subsection A.1, except the following points.

In Setup, $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $Z = U, Q_A = V, X_B = W$, and selects randomly $x_A, q_B = \log(Q_B)$ and simulates as $X_A = g^{x_A}, D_B = Z^{q_B}$.

In Simulation of H, $\mathcal{S}$ extracts BDH(U, V, W) as follows. In Step 4c, since $\mathcal{S}$ knows x_A, $\mathcal{S}$ extracts

$$((\sigma_1')^{e_B/e_A}(\sigma_2')^{-1})^{1/(e_B-1)} = g_T^{z\log(Q_A)x_B} = \mathrm{BDH}(Z, Q_A, X_B),$$

where

$$\sigma_1' = \sigma_1 e(Z, Q_B X_B)^{-x_A} = g_T^{ze_A\log(Q_A)(\log(Q_B)+x_B)},$$
$$\sigma_2' = \sigma_2 e(Z, Q_B^{e_B} X_B)^{-x_A} = g_T^{z\log(Q_A)(e_B\log(Q_B)+x_B)}.$$

In Step 5c, since $\mathcal{S}$ knows x_A, $\mathcal{S}$ extracts

$$\sigma_1\tau_1^{-1}\tau_2^{-1}\tau_3^{-1} = g_T^{z\log(Q_A)x_B} = \mathrm{BDH}(Z, Q_A, X_B),$$

where

$$\tau_1 = \sigma_2 = g_T^{z\log(Q_A)\log(Q_B)},$$
$$\tau_2 = e(Z, \sigma_3) = g_T^{zx_Ax_B},$$
$$\tau_3 = e(Z, Q_B)^{x_A} = g_T^{zx_A\log(Q_B)}.$$

A.5 Other Cases

Event E_{1b}' in Table 3 can be handled same as event E_{1b} in Table 2, with condition $Q_A = Q_B$ under the square gap BDH assumption that is equivalent to the gap BDH assumption.

Events E_{2a}', E_3' in Table 3 can be handled same as events E_{2a}, E_3 in Table 2, with condition $Q_A = Q_B$ under the gap BDH assumption.