Making Protocol FSU Revocable

Kazuma Wariki[1], Atsushi Fujioka[1], Akira Nagai[2], and Kan Yasuda[2]

[1] Kanagawa University {r202470194te,fujioka}@jindai.jp
[2] NTT Social Informatics Laboratories {akira.nagai,kan.yasuda}@ntt.com

Abstract. This paper examines whether a revocation function can be added to a protocol, protocol FSU, which is an asymmetric pairing variant of a protocol that has been adopted as an international standard, ISO/IEC11770-3. Protocol FSU is an identity-based authenticated-key exchange protocol based on a mathematical problem, an asymmetric gap bilinear Diffie–Hellman (GBDH) problem. To make protocol FSU revocable, a generic technique is applied, which converts an identity-based encryption scheme to a revocable identity-based encryption scheme by introducing a symmetric-key encryption scheme. In addition, to make the converted revocable identity-based authenticated-key exchange protocol efficient, we reduce ephemeral information exchanged in the protocol, and introduce an additional parameter to the master public-key where the secret information of the additional parameter is not needed to include in the master secret-key.
We discuss the security of the resultant protocol, and prove that it is rid-eCK secure under the asymmetric GBDH assumption.

Keywords: Identity-based authenticated key exchange · revocability · asymmetric gap bilinear Diffie–Hellman assumption · protocol FSU.

1 Introduction

Key exchange is one of the most important topics in cryptography. In a key exchange protocol, two parties exchange ephemeral information over a public channel, and then, both parties can establish a shared key only known by them. The shared key can be used to guarantee confidentiality and authenticity as a *session key*.

Authenticated-key exchange (AKE) is an evolution of key exchange. In an AKE protocol, each party has a static public-key, and the key is linked to the party's identity by a certificate issued by a certification authority (CA) in the public key infrastructure (PKI). To establish a session key, a party generates a pair of ephemeral public- and secret-keys. The party sends the ephemeral public-key to another party over a public channel, and receives another ephemeral public-key from the peer over the public channel. Each party can compute the same key using its own static secret-key, the own ephemeral secret-key, the peer's static public-key, and the peer's ephemeral public-key. AKE can guarantee that the session key the party computes is indeed shared with the intended party.

In a PKI-based AKE protocol, revocation is essential as its security is related to the linkage between a party and the static public-key.

Identity-based authenticated-key exchange (IB-AKE) is a variant of AKE designed for the identity-based setting. In an IB-AKE protocol, it is assumed that a trusted third party, called *private-key generator* (PKG), exists, and each party uses identity information (such as e-mail address or phone number) as a public key. The PKG generates a pair of master public- and secret-keys, and computes a secret-key of a party based on the identifier of the party.

In IB-AKE protocols, a *key revocation* mechanism is also necessary similar to PKI-based AKE protocols as a party may act maliciously or may compromise the static secret-key accidentally, thereby reducing the reliability of the system.

1.1 Revocable Identity-Based Authenticated-Key Exchange

The key revocation in IB-AKE was first addressed in [7], where the use of hierarchical identities, i.e., *revocable hierarchical identity-based authenticated-key exchange* (RHIB-AKE), was discussed since it is not practical for the PKG to manage all users. Okano et al. give a generic construction of a *revocable hierarchical identity-based authenticated-key exchange* (RHIB-AKE) protocol from a Revocable Hierarchical Identity-Based Encryption (RHIBE) scheme [9, 2]. They define the rhid-eCK security, and give two instantiations of the rhid-eCK secure RHIB-AKE protocols based on the pairing and the lattice, respectively.

Later, Nakagawa et al. simplified the rhid-eCK security to the rid-eCK security, i.e., a security notion in the identity-based setting, and formulated *revocable identity-based authenticated-key exchange* (RIB-AKE) protocol because of the protocol complexity [6]. They devised a rid-eCK secure RIB-AKE protocol that does not use pairing, to apply it to IoT devices.

In an RIB-AKE protocol, the PKG generates a pair of master public- and secret-keys. The PKG computes a secret-key of a party based on the identifier of the party. The PKG manages a revocation list and its time-period.

When a revocation occurs, the PKG updates the revocation list and its time-period. The PKG computes key update information and broadcasts over a public channel. Each party computes a current secret-key by using the static secret-key and the key update information.

To establish a session key, a party generates a pair of ephemeral public- and secret-keys. The party sends the ephemeral public-key to another party over a public channel, and receives another ephemeral public-key from the peer over the public channel. Each party can compute the same session key using its own current secret-key, the own ephemeral secret-key, the peer's identifier, and the peer's ephemeral public-key.

In other words, to generate a session key, an RIB-AKE protocol uses a current secret-key rather than a static secret-key is used in an IB-AKE protocol.

1.2 Contribution

This paper examines whether a revocation function can be added to a protocol, protocol FSU, which is an asymmetric pairing variant of a protocol that has been adopted as an international standard, ISO/IEC11770-3 [5][3]. Protocol FSU is an IB-AKE protocol based on a mathematical problem, an asymmetric gap bilinear Diffie–Hellman (GBDH) problem [3].

To make protocol FSU revocable, we apply a generic technique, invented by Seo and Emura [8], to protocol FSU. Here, the generic technique converts an identity-based encryption scheme to a revocable identity-based encryption scheme by introducing a symmetric-key encryption scheme. A secret-key of the symmetric-key encryption scheme is generated by the PKG, and it is included in a static secret-key of a party together with a usual static secret-key of the identity-based encryption scheme. When the PKG updates a current

[3] In [3], the protocol, proposed in [4] and adopted in [5], is named Π^{sym} and its asymmetric pairing variant is named Π^{asym}. In this paper, the latter is called protocol FSU.

secret-key of the party, the PKG encrypts the current secret-key using the symmetric-key encryption scheme. Each party can obtain the current secret-key by decrypting the ciphertext from the PKG.

In addition, to make the converted RIB-AKE protocol efficient, we reduce ephemeral information exchanged in the protocol. Each party sends two ephemeral public-keys in protocol FSU, that is, in the converted RIB-AKE protocol, whereas each party sends a single ephemeral public-key in the reduced protocol. Unfortunately, this resultant protocol becomes insecure. To overcome this, we introduce an additional parameter to the master public-key where the secret information of the additional parameter is not needed to include in the master secret-key.

We discuss the security of the final protocol, and prove that it is rid-eCK secure under the asymmetric GBDH assumption.

2 Preliminaries

2.1 Mathematical Assumption

We introduce the asymmetric gap Bilinear Diffie–Hellman (GBDH) assumption [3] described as follows: Let λ be the security parameter. Let G_1, G_2, and G_T be cyclic groups with generators g_1, g_2, and g_T $(= \hat{e}(g_1, g_2))$, respectively, and $\hat{e} : G_1 \times G_2 \to G_T$ be asymmetric pairing. Here, the order of all generators is λ-bit prime q. Choose $u, v, w \in_U \mathbb{Z}_q$ and let $U_1 := g_1^u, U_2 := g_2^u, V_1 := g_1^v, V_2 := g_2^v, W_1 := g_1^w$ and $W_2 := g_2^w$. Now, we consider the oracle $\mathrm{DBDH}^{1,1,2}(\cdot,\cdot,\cdot,\cdot)$ that on input $U_1, V_1, W_2, \hat{e}(g_1, g_2)^x$, return 1 if $uvw = x \bmod q$ and 0 otherwise. Let $\mathcal{S}$ be a solver who can access this oracle, $\mathrm{DBDH}^{1,1,2}(\cdot,\cdot,\cdot,\cdot)$, and who tries to compute $\hat{e}(g_1, g_2)^{uvw}$ given U_1, U_2, V_1, V_2, W_1, and W_2.

For solver $\mathcal{S}$, we define advantage

$$\mathrm{Adv}^{\mathrm{aGBDH}}(\mathcal{S}) = \Pr[\mathcal{S}^{\mathrm{DBDH}^{1,1,2}(\cdot,\cdot,\cdot,\cdot)}(U_1, U_2, V_1, V_2, W_1, W_2) = \hat{e}(g_1, g_2)^{uvw}].$$

Definition 1 (asymmetric GBDH assumption). *We say that the* asymmetric GBDH assumption *holds if, for any probabilistic polynomial-time solver, $\mathcal{S}$, $\mathrm{Adv}^{\mathrm{aGBDH}}(\mathcal{S})$ is negligible in security parameter λ.*

2.2 Syntax of RIB-AKE

The syntax of RIB-AKE follows that described in [6]. A RIB-AKE protocol, Π, consists of the following seven probabilistic polynomial-time (PPT) algorithms:

– **ParGen**$(1^\lambda, N) \to (MSK, MPK, RL, T)$: The parameter generation algorithm is executed only once by the PKG. With the security parameter, λ, and the maximum number of parties, N, as input, it outputs the master secret-key, MSK, the master public-key, MPK, the initial revocation list, RL, and the time counter, T.
The master public-key, MPK, is distributed to all parties via a public channel. The master secret-key, MSK, is the secret information of the PKG. The revocation list, RL, is not secret information but only used by the PKG, so it does not necessarily have to be distributed. Assume $RL = \emptyset$ and $T = 0$ as the initial state. (We assume to include MPK in the input of all algorithms below.)

– **SSKGen**$(MSK, ID) \rightarrow ssk_{ID}$: The static secret-key generation algorithm is performed by the PKG only once for each party. It takes the master secret-key, MSK, and the party's identifier, ID, as input and outputs the static secret-key, ssk_{ID}, corresponding to ID.
  Each static secret-key is distributed to each party via a secret channel.
– **Revoke**(RL, T, rl): The algorithm for updating the revocation list is executed by the PKG at certain intervals. It receives the list of newly revoked user's identifiers, rl. Then, it updates $RL \leftarrow RL \cup rl$. In addition, it increments the time counter $T \leftarrow T+1$.
– **KeyUp**$(MSK, T, RL) \rightarrow ku_T$: The algorithm for generating key update information is executed by the PKG after executing **Revoke**. It takes the master secret-key, MSK, the time counter, T, and the revocation list, RL, as input and outputs the key update information, ku_T.
  The key update information with the time counter, (ku_T, T), is distributed to all parties via a public channel.
– **CSKGen**$(ID, T, ssk_{ID}, ku_T) \rightarrow csk_{ID,T}$: The current secret-key generation algorithm is executed by each party after receiving (ku_T, T). It takes ID, the time counter, T, the static secret-key, ssk_{ID}, and the key update information, ku_T, as input and outputs the current secret-key, $csk_{ID,T}$, or $\perp$. The $\perp$ means that the user has been revoked.
– **EKGen**$(ID_A, ID_B, T, csk_{A,T}) \rightarrow (esk_A, epk_A)$: The ephemeral key generation algorithm is executed by each party for each session. It takes as input the identifier, ID_A, of executor P_A, the identifier, ID_B, of communication partner P_B, the time counter, T, and the current secret-key, $csk_{A,T}$, of executor P_A. It outputs the ephemeral secret/public-key pair, (esk_A, epk_A), of executor P_A for the session.
  The ephemeral public-key, epk_A, is distributed to communication partner P_B via a public channel.
– **SKGen**$(ID_A, ID_B, T, csk_{A,T}, esk_A, epk_B) \rightarrow SK$: The session key generation algorithm is executed by each party for each session. It takes as input the identifier, ID_A, of executor P_A, the identifier, ID_B, of communication partner P_B, the time counter, T, the current secret-key, $csk_{A,T}$, of executor P_A, the ephemeral secret-key, esk_A, of executor P_A, and the ephemeral public-key, epk_B, of communication partner P_B. It outputs the session key, SK.

We show the overall behavior of the RIB-AKE protocol in Fig. 1.

2.3 Session

The rid-eCK security model follows that described in [6]. An invocation of a protocol is called a *session*. A session is activated via an incoming message of the form, $(\Pi, \mathcal{I}, T, ID_A, ID_B)$ or $(\Pi, \mathcal{R}, T, ID_A, ID_B)$, where Π is the protocol identifier, $\mathcal{I}$ and $\mathcal{R}$ are role identifiers, T is the time counter, and ID_A and ID_B are user identifiers of user P_A and P_B, respectively. When P_A is activated with $(\Pi, \mathcal{I}, T, ID_A, ID_B)$, we call P_A an *initiator*. When P_A is activated with $(\Pi, \mathcal{R}, T, ID_A, ID_B)$, we call P_A a *responder*.

On activation, an initiator (resp. responder) P_A returns epk_A. Receiving an incoming message $(\Pi, \mathcal{I}, T, ID_A, ID_B, epk_B)$ (resp. $(\Pi, \mathcal{R}, T, ID_A, ID_B, epk_B)$) from the responder (resp. initiator), P_B, P_A computes the session key, SK.

If P_A is the initiator, the session identifier, sid, is $(\Pi, \mathcal{I}, T, ID_A, ID_B, epk_A, \cdot, s)$ or $(\Pi, \mathcal{I}, T, ID_A, ID_B, epk_A, epk_B, s)$ where s means that the session is the s-th initialized

Parameter Setting
PKG's Computation
$(MSK, MPK, RL, T) \leftarrow \mathbf{ParGen}(1^\lambda, N)$
PKG's secret-key: MSK, PKG's public-key: MPK, Revocation list: RL, Time counter: T
Distribute MPK to all users.

Static Secret-Key Distribution

PKG's Computation for P_A	**PKG's Computation for P_B**
$ssk_A \leftarrow \mathbf{SSKGen}(MSK, ID_A)$	$ssk_B \leftarrow \mathbf{SSKGen}(MSK, ID_B)$
Send ssk_A to P_A via a secret channel.	Send ssk_B to P_B via a secret channel.

Update Information Distribution
PKG's Computation at Certain Intervals
Update RL by $\mathbf{Revoke}(RL, T, rl)$, $ku_T \leftarrow \mathbf{KeyUp}(MSK, T, RL)$
Distribute (ku_T, T) to all users.

Current Secret-Key Generation

P_A's Computation when P_A receives ku_T.	**P_B's Computation when P_B receives ku_T.**
$csk_{A,T} \leftarrow \mathbf{CSKGen}(ID_A, T, ssk_A, ku_T)$	$csk_{B,T} \leftarrow \mathbf{CSKGen}(ID_B, T, ssk_B, ku_T)$
Current secret-key of P_A: $csk_{A,T}$	Current secret-key of P_B: $csk_{B,T}$

Session Key Generation

P_A's Computation when P_A makes a session with P_B.	**P_B's Computation when P_B makes a session with P_A.**
$(esk_A, epk_A) \leftarrow \mathbf{EKGen}(ID_A, ID_B, T, csk_{A,T})$	$(esk_B, epk_B) \leftarrow \mathbf{EKGen}(ID_B, ID_A, T, csk_{B,T})$
Ephemeral secret-key of P_A: esk_A	Ephemeral secret-key of P_B: esk_B
Ephemeral public-key of P_A: epk_A	Ephemeral public-key of P_B: epk_B
Send epk_A to P_B via a public channel.	Send epk_B to P_A via a public channel.
P_A's Computation when P_A receives epk_B.	**P_B's Computation when P_B receives epk_A.**
$SK \leftarrow \mathbf{SKGen}(ID_A, ID_B, T, csk_{A,T}, esk_A, epk_B)$	$SK \leftarrow \mathbf{SKGen}(ID_B, ID_A, T, csk_{B,T}, esk_B, epk_A)$

Session key shared by P_A and P_B: SK

Fig. 1. Behavior of an RIB-AKE protocol

one of P_A. If P_A is the responder, the session is identified by $sid = (\Pi, \mathcal{R}, T, ID_A, ID_B, \cdot, epk_A, s')$ or $(\Pi, \mathcal{R}, T, ID_A, ID_B, epk_B, epk_A, s')$ where s' means that the session is the s'-th initialized one of P_A. It is said that P_A is the *owner* of session sid when the fourth component of sid is ID_A. Also, P_B is said to be a *peer* of session sid when the fifth component of sid is ID_B. A session is *completed* when the session key has been computed in that session.

The matching session of sid $(= (\Pi, \mathcal{I}, T, ID_A, ID_B, epk_A, epk_B, s))$ is a session with $(\Pi, \mathcal{R}, T, ID_B, ID_A, epk_A, epk_B, s')$ and vice versa.

2.4 Adversary

An adversary, $\mathcal{A}$, is modeled as a PPT Turing machine that controls all communication between the parties, including session activation. Let T_{cu} and $RL_{T_{cu}}$ be the time counter and the revoke list maintained by the challenger, respectively. We model the adversary's capability by the following queries:

- $\mathbf{ParGen}(1^\lambda, N)$: The adversary requests the PKG to generate the parameter and obtains the master public-key, MPK.
- $\mathbf{SSKRev}(ID)$: The adversary obtains the static secret-key, ssk_{ID}, of the user with identifier ID.

– **KeyUp**(T): If $T \le T_{cu}$, then the adversary obtains the key update information, ku_T, else obtains $\perp$.
– **CSKRev**(ID, T): If $T \le T_{cu}$, then the adversary obtains the current secret-key, $csk_{ID,T}$, of the user with identifier ID, else obtains $\perp$.
– **ESKRev**(sid): The adversary obtains the ephemeral secret-key, esk, of the session owner.
– **SKRev**(sid): The adversary obtains the session key if the session is completed.
– **MSKRev**$()$: The adversary obtains the master secret-key, MSK.
– **EstablishUser**(U, ID): The query allows the adversary to join a party as the user, P, with the identity, ID, and obtain the static secret-key, ssk_{ID}. If this query establishes a party, then we call the party *dishonest*. If not, we call the party *honest*.
– **Send**$(message)$: $message$ is given in the form $(\Pi, \mathcal{I}, T, ID_A, ID_B)$, $(\Pi, \mathcal{R}, T, ID_A, ID_B)$, $(\Pi, \mathcal{I}, T, ID_A, ID_B, epk_B)$, or $(\Pi, \mathcal{R}, T, ID_A, ID_B, epk_B)$. The adversary obtains the response from the party according to the protocol specification.
– **Revoke**(RL): If $RL_{T_{cu}} \not\subset RL$, return $\perp$. Otherwise, T_{cu} is incremented as $T_{cu} \leftarrow T_{cu} + 1$, update the revoke list as $RL_{T_{cu}} \leftarrow RL$, and return T_{cu}.

2.5 Freshness

Here, we give the definition of *freshness* [6].

Definition 2. *Let $sid^* = (\Pi, \mathcal{I}, T^*, ID_A, ID_B, epk_A, epk_B, s)$ or $(\Pi, \mathcal{R}, T^*, ID_A, ID_B, epk_B, epk_A, s')$ be a completed session between the honest party, P_A, with the identifier, ID_A, and the honest party, P_B, with the identifier, ID_B. When a matching session of sid^* exists, we denote it as $\overline{sid^*}$. We say that sid^* is* fresh *if none of the following conditions are satisfied:*

1. *The adversary, $\mathcal{A}$, issues* **SKRev**(sid^*), *or* **SKRev**$(\overline{sid^*})$ *if $\overline{sid^*}$ exists.*
2. *$\overline{sid^*}$ exists and adversary $\mathcal{A}$ makes either of the following queries:*
   – **ESKRev**(sid^*) *and* **SSKRev**(ID_A) *where $ID_A \notin RL_{T^*}$.*
   – **ESKRev**$(\overline{sid^*})$ *and* **SSKRev**(ID_B) *where $ID_B \notin RL_{T^*}$.*
   – **ESKRev**(sid^*) *and* **CSKRev**(ID_A, T^*) *where $ID_A \notin RL_{T^*}$.*
   – **ESKRev**$(\overline{sid^*})$ *and* **CSKRev**(ID_B, T^*) *where $ID_B \notin RL_{T^*}$.*
3. *$\overline{sid^*}$ does not exist and adversary $\mathcal{A}$ makes either of the following queries:*
   – **ESKRev**(sid^*) *and* **SSKRev**(ID_A) *where $ID_A \notin RL_{T^*}$.*
   – **SSKRev**(ID_B) *where $ID_B \notin RL_{T^*}$.*
   – **ESKRev**(sid^*) *and* **CSKRev**(ID_A, T^*) *where $ID_A \notin RL_{T^*}$.*
   – **CSKRev**(ID_B, T^*) *where $ID_B \notin RL_{T^*}$.*

Note that if the adversary, $\mathcal{A}$, issues **MSKRev**$()$, we regard the adversary, $\mathcal{A}$, as having issue **CSKRev**(ID, T^*), **SSKRev**(ID) for any user, P, identified with ID where $ID \notin RL$.

2.6 Security Experiment

We consider the following security game between the challenger and the adversary. First, the adversary, $\mathcal{A}$, receives an RIB-AKE protocol, Π, a master public-key, MPK, and a set of honest parties from the challenger. The adversary, $\mathcal{A}$, then arbitrarily executes the queries, described in Section 2.4, multiple times to the challenger. Along the way, $\mathcal{A}$ executes the following query only once.

- **Test**(sid^*): The session, sid^*, must be fresh. The challenger randomly selects a bit $b \in \{0,1\}$ and returns the session key for sid^* if $b = 0$, or a randomly generated key if $b = 1$.

The game continues until the adversary, $\mathcal{A}$, outputs a guess, b'. The adversary wins the game when the test session, sid^*, is still fresh and the adversary's guess is correct, i.e., $b' = b$. We define the adversary's advantage as $\mathrm{Adv}_{\Pi}^{\text{RIB-AKE}}(\mathcal{A}) = |2\Pr(\mathcal{A} \text{ wins}) - 1|$. Then, we define the security of RIB-AKE as follows.

Definition 3 (rid-eCK security model [6]). *An RIB-AKE protocol, Π, is said to be* secure in the rid-eCK model *if the advantage,* $\mathrm{Adv}_{\Pi}^{\text{RIB-AKE}}(\mathcal{A})$*, defined above is negligible in λ for any PPT adversary, $\mathcal{A}$.*

2.7 Symmetric-Key Encryption

A symmetric-key encryption scheme, Σ, consists of the following three PPT algorithms:

Gen(1^λ): The key generation algorithm is executed only once by a user. With the security parameter, λ, as input, it outputs a secret-key, k.

Enc(m, k): The encryption key algorithm is executed by a sender. With a message, m, and a secret-key, k, as input, it outputs a ciphertext, c.

Dec(c, k): The decryption algorithm is executed by a receiver. With a ciphertext, c, and a secret-key, k, as input, it outputs a message, m'.

Note that the secret-key is confidential between the sender and receiver.

We require Σ to be IND-CPA secure. However, we omit to describe the definition of the IND-CPA security. See appropriate references, e.g., [1].

3 Protocol modifiedFSUrev

First, we convert protocol FSU [3] to protocol FSUrev, which is an RIB-AKE protocol, by applying a generic technique, invented by Seo and Emura [8].

Next, we modify protocol FSUrev to reduce the communication complexity. Note that the modified protocol is insecure.

Finally, we construct protocol modifiedFSUrev, which is an RIB-AKE protocol, to overcome the insecurity of the modified protocol, and discuss its security.

3.1 Protocol FSUrev

We construction an RIB-AKE protocol, FSUrev, based on protocol FSU [3].

Protocol FSUrev consists of the PPT algorithm shown below:

Let $\Sigma = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ be a symmetric encryption scheme.

- **ParGen**($1^\lambda, N$) $\to$ (MSK, MPK, RL, T):
  1. Generate $z \in_U \mathbb{Z}_q$ and set $Z_j = g_j^z$ $(\in G_j)$ $(j = 1, 2)$.
  2. Output $MSK = z$, $MPK = (Z_1, Z_2)$, $RL = \emptyset$, and $T = 0$.
- **SSKGen**(MSK, ID_i) $\to$ ssk_i:
  1. Set $Q_{i||0,j} = H_j(ID_i||0) = g_j^{q_{i||0,j}}$ $(\in G_j)$ $(j = 1, 2)$ and set $D_{i||0,j} = Q_{i||0,j}^z$ $(\in G_j)$ $(j = 1, 2)$.

2. Generate a secret-key, K_i, using **Gen** in Σ.
3. Output $ssk_i = (D_{i||0,1}, D_{i||0,2}, K_i)$.
   Note that $D_{i||0,1}$ and $D_{i||0,2}$ are used as $csk_{i,0}$ $(= (D_{i||0,1}, D_{i||0,2}))$, i.e., the initial current secret-key for which no revoke has occurred yet.

– **Revoke**$(rl) \to RL$:
  1. Update $RL \leftarrow RL \cup rl$ and $T \leftarrow T + 1$.
– **KeyUp**$(MSK, T, RL) \to ku_T$:
  1. For each user, P_i, not revoked at T, set $Q_{i||T,j} = H_j(ID_i||T) = g_j^{q_{i||T,j}} \in G_j$ $(j = 1, 2)$ and set $D_{i||T,j} = Q_{i||T,j}^z$ $(\in G_j)$ $(j = 1, 2)$.
  2. Compute $C_{i,1} \leftarrow \mathbf{Enc}(K_i, D_{i||T,1})$, $C_{i,2} \leftarrow \mathbf{Enc}(K_i, D_{i||T,2})$.
  3. Output $ku_T = \{(ID_i, C_{i,1}, C_{i,2}) \mid \text{User } P_i \text{ is not revoked at time counter } T.\}$.
– **CSKGen**$(ssk_i, ku_T) \to csk_{i,T}$:
  1. Decrypts $D_{i||T,1} \leftarrow \mathbf{Dec}(K_i, C_{i,1}), D_{i||T,2} \leftarrow \mathbf{Dec}(K_i, C_{i,2})$.
  2. Output $csk_{i,T} = (D_{i||T,1}, D_{i||T,2})$.
– **EKGen**$(csk_{i,T}) \to (esk_{ID}, epk_{ID})$:
  1. Generate $x_i \in_U Z_q$ and set $X_{i,j} = g_j^{x_i}$ $(j = 1, 2)$.
  2. Output $esk_i = x_i$, $epk_i = (X_{i,1}, X_{i,2})$.
– **SKGen**$(ID_i, ID_{i'}, T, csk_{i,T}, esk_i, epk_{i'}) \to SK$.
  1. If the algorithm is executed by an initiator (resp. responder), do as follows:
  2. Parse $csk_{i,T} = (D_{i||T,1}, D_{i||T,2})$, $esk_i = x_i$ and $epk_{i'} = (X_{i',1}, X_{i',2})$.
  3. Compute $Q_{i'||T,2} = H_2(ID_{i'}||T)$ (resp. $Q_{i'||T,1} = H_1(ID_{i'}||T)$).
  4. Let $\sigma_1 = \hat{e}(D_{i||T,1}, Q_{i'||T,2})$ (resp. $\sigma_1 = \hat{e}(Q_{i'||T,1}, D_{i||T,2})$).
  5. Let $\sigma_2 = \hat{e}(D_{i||T,1}Z_1^{x_i}, Q_{i'||T,2}X_{i',2})$ (resp. $\sigma_2 = \hat{e}(Q_{i'||T,1}X_{i',1}, D_{i||T,2}Z_2^{x_i})$).
  6. Let $\sigma_3 = X_{i',1}^{x_i}$ (resp. $\sigma_3 = X_{i',1}^{x_i}$).
  7. Let $\sigma_4 = X_{i',2}^{x_i}$ (resp. $\sigma_4 = X_{i',2}^{x_i}$).
  8. Let $ST = (\Pi, T, ID_i, ID_{i'}, epk_i, epk_{i'})$ (resp. $ST = (\Pi, T, ID_{i'}, ID_i, epk_{i'}, epk_i)$).
  9. Output $SK = H(ST, \sigma_1, \sigma_2, \sigma_3, \sigma_4)$.

3.2 Insecure Protocol

Let us modify protocol FSUrev where the initiator sends an ephemeral public-key in G_1 and the responder returns an ephemeral public-key in G_2 (see Fig. 2).

P_A (ID_A)	T	P_B (ID_B)
	$Z_1 = g_1^z$, $Z_2 = g_2^z$	
$Q_{A\|\|T,1} = H_1(ID_A\|\|T)$		$Q_{B\|\|T,1} = H_1(ID_B\|\|T)$
$Q_{A\|\|T,2} = H_2(ID_A\|\|T)$		$Q_{B\|\|T,2} = H_2(ID_B\|\|T)$
$D_{A\|\|T,1} = Q_{A\|\|T,1}^z$		$D_{B\|\|T,1} = Q_{B\|\|T,1}^z$
$D_{A\|\|T,2} = Q_{A\|\|T,2}^z$		$D_{B\|\|T,2} = Q_{B\|\|T,2}^z$
$X_{A,1} = g_1^{x_A}$		$X_{B,2} = g_2^{x_B}$
$\sigma_1 = \hat{e}(D_{A\|\|T,1}, Q_{B\|\|T,2})$		$\sigma_1 = \hat{e}(Q_{A\|\|T,1}, D_{B\|\|T,2})$
$\sigma_2 = \hat{e}(D_{A\|\|T,1}Z_1^{x_A}, Q_{B\|\|T,2}X_{B,2})$		$\sigma_2 = \hat{e}(Q_{A\|\|T,1}X_{A,1}, D_{B\|\|T,2}Z_2^{x_B})$
$\sigma_3 = \hat{e}(Z_1^{x_A}, X_{B,2})$		$\sigma_3 = \hat{e}(X_{A,1}, Z_2^{x_B})$
	$ST = (\Pi, T, ID_A, ID_B, epk_A, epk_B)$	
	$SK = H(ST, \sigma_1, \sigma_2, \sigma_3)$	

Fig. 2. Outline of the insecure protocol

However, this modified protocol is not rid-eCK secure as an adversary, $\mathcal{A}$, who knows the master secret-key, z, can break the protocol. In other words, $\mathcal{A}$ can compute the shared values

$$\sigma_1 = \hat{e}(H_1(ID_A||T), H_2(ID_B||T))^z \ (= g_T^{zq_{A||T,1}q_{B||T,2}})$$
$$\sigma_2 = \hat{e}(H_1(ID_A||T)X_{A,1}, H_2(ID_B||T)X_{B,2})^z \ (= g_T^{z(q_{A||T,1}+x_A)(q_{B||T,2}+x_B)})$$
$$\sigma_3 = \hat{e}(X_{A,1}, X_{B,2})^z \ (= g_T^{zx_Ax_B})$$

from the public values, ID_A, ID_B, $X_{A,1}$, $X_{B,2}$, and T.

3.3 Protocol modifiedFSUrev

To avoid the situation, we consider a further modification where the PKG generates two master secret-keys, z and y, computes master public-keys, and erases y (see Fig. 3).

P_A (ID_A)	T	P_B (ID_B)
	$Z_1 = g_1^z,\ Z_2 = g_2^z$	
	$Y_1 = g_1^y,\ Y_2 = g_2^y$	
$Q_{A\|\|T,1} = H_1(ID_A\|\|T)$		$Q_{B\|\|T,1} = H_1(ID_B\|\|T)$
$Q_{A\|\|T,2} = H_2(ID_A\|\|T)$		$Q_{B\|\|T,2} = H_2(ID_B\|\|T)$
$D_{A\|\|T,1} = Q_{A\|\|T,1}^z$		$D_{B\|\|T,1} = Q_{B\|\|T,1}^z$
$D_{A\|\|T,2} = Q_{A\|\|T,2}^z$		$D_{B\|\|T,2} = Q_{B\|\|T,2}^z$
$X_{A,1} = g_1^{x_A}$		$X_{B,2} = g_2^{x_B}$
$\sigma_1 = \hat{e}(D_{A\|\|T,1}, Q_{B\|\|T,2})$		$\sigma_1 = \hat{e}(Q_{A\|\|T,1}, D_{B\|\|T,2})$
$\sigma_2 = \hat{e}(D_{A\|\|T,1}Z_1^{x_A}, Q_{B\|\|T,2}X_{B,2})$		$\sigma_2 = \hat{e}(Q_{A\|\|T,1}X_{A,1}, D_{B\|\|T,2}Z_2^{x_B})$
$\sigma_3 = \hat{e}((Z_1Y_1)^{x_A}, X_{B,2})$		$\sigma_3 = \hat{e}(X_{A,1}, (Z_2Y_2)^{x_B})$
	$ST = (\Pi, T, ID_A, ID_B, epk_A, epk_B)$	
	$SK = H(ST, \sigma_1, \sigma_2, \sigma_3)$	

Fig. 3. Outline of protocol modifiedFSUrev

We name the resultant protocol modifiedFSUrev. A description is given below:
Let $\Sigma = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ be a symmetric encryption scheme.

- $\mathbf{ParGen}(1^\lambda, N) \to (MSK, MPK, RL, T)$:
  1. Generate $z, y \in_U \mathbb{Z}_q$ and set $Z_j = g_j^z, Y_j = g_j^y$ $(\in G_j)$ $(j = 1, 2)$.
  2. Output $MSK = z$, $MPK = (Z_1, Z_2, Y_1, Y_2)$, $RL = \emptyset$, and $T = 0$.
     Note that y is erased after generating Y_1 and Y_2.
- $\mathbf{SSKGen}(MSK, ID_i) \to ssk_i$:
  1. Set $Q_{i||0,j} = H_j(ID_i||0) = g_j^{q_{i||0,j}}$ $(\in G_j)$ $(j = 1, 2)$ and set $D_{i||0,j} = Q_{i||0,j}^z$ $(\in G_j)$ $(j = 1, 2)$.
  2. Generate a secret-key, K_i, as $K_i \leftarrow \mathbf{Gen}(1^\lambda)$.
  3. Output $ssk_i = (D_{i||0,1}, D_{i||0,2}, K_i)$.
     Note that $D_{i||0,1}$ and $D_{i||0,2}$ are used as $csk_{i,0}$ $(= (D_{i||0,1}, D_{i||0,2}))$, i.e., the initial current secret-key for which no revoke has occurred yet.
- $\mathbf{Revoke}(RL, T, rl) \to RL$:
  1. Update $RL \leftarrow RL \cup rl$ and $T \leftarrow T + 1$.
- $\mathbf{KeyUp}(MSK, T, RL) \to ku_T$:

1. For each user, P_i, not revoked at T, set $Q_{i||T,j} = H_j(ID_i||T) = g_j^{q_{i||T,j}}$ $(\in G_j)$ $(j = 1, 2)$ and set $D_{i||T,j} = Q_{i||T,j}^z$ $(\in G_j)$ $(j = 1, 2)$.
2. Compute $C_{i,1} \leftarrow \mathbf{Enc}(K_i, D_{i||T,1}), C_{i,2} \leftarrow \mathbf{Enc}(K_i, D_{i||T,2})$.
3. Output $ku_T = \{(ID_i, C_{i,1}, C_{i,2}) \mid \text{User } P_i \text{ is not revoked at time counter } T.\}$.

– $\mathbf{CSKGen}(ID_i, T, ssk_i, ku_T) \rightarrow csk_{i,T}$:
  1. Decrypts $D_{i||T,1} \leftarrow \mathbf{Dec}(K_i, C_{i,1}), D_{i||T,2} \leftarrow \mathbf{Dec}(K_i, C_{i,2})$.
  2. Output $csk_{i,T} = (D_{i||T,1}, D_{i||T,2})$.

– $\mathbf{EKGen}(ID_i, ID_{i'}, T, csk_{i,T}) \rightarrow (esk_i, epk_i)$:
  1. If the algorithm is executed by an initiator (resp. responder), do as follows:
  2. Generate $x_i \in_U Z_q$ and set $X_{i,1} = g_1^{x_i}$ (resp. $X_{i,2} = g_2^{x_i}$).
  3. Output $esk_i = x_i, epk_i = X_{i,1}$ (resp. $esk_i = x_i, epk_i = X_{i,2}$).

– $\mathbf{SKGen}(ID_i, ID_{i'}, T, csk_{i,T}, esk_i, epk_{i'}) \rightarrow SK$.
  1. If the algorithm is executed by an initiator (resp. responder), do as follows:
  2. Parse $csk_{i,T} = (D_{i||T,1}, D_{i||T,2})$, $esk_i = x_i$ and $epk_{i'} = X_{i',2}$ (resp. $epk_{i'} = X_{i',1}$).
  3. Compute $Q_{i'||T,2} = H_2(ID_{i'}||T)$ (resp. $Q_{i'||T,1} = H_1(ID_{i'}||T)$).
  4. Let $\sigma_1 = \hat{e}(D_{i||T,1}, Q_{i'||T,2})$ (resp. $\sigma_1 = \hat{e}(Q_{i'||T,1}, D_{i||T,2})$).
  5. Let $\sigma_2 = \hat{e}(D_{i||T,1}Z_1^{x_i}, Q_{i'||T,2}X_{i',2})$ (resp. $\sigma_2 = \hat{e}(Q_{i'||T,1}X_{i',1}, D_{i||T,2}Z_2^{x_i})$).
  6. Let $\sigma_3 = \hat{e}((Z_1Y_1)^{x_i}, X_{i',2})$ (resp. $\sigma_3 = \hat{e}(X_{i',1}, (Z_2Y_2)^{x_i})$).
  7. Let $ST = (\Pi, T, ID_i, ID_{i'}, epk_i, epk_{i'})$ (resp. $ST = (\Pi, T, ID_{i'}, ID_i, epk_{i'}, epk_i)$).
  8. Output $SK = H(ST, \sigma_1, \sigma_2, \sigma_3)$.

Two parties, P_A and P_B, compute the same shared values,

$$\sigma_1 = g_T^{zq_{A||T,1}q_{B||T,2}},\ \sigma_2 = g_T^{z(q_{A||T,1}+x_A)(q_{B||T,2}+x_B)},\ \sigma_3 = g_T^{(z+y)x_Ax_B},$$

and, therefore, have the same session key, SK.

3.4 Security of Protocol modifiedFSUrev

Regarding the security of protocol modifiedFSUrev, we have the following theorem.

Theorem 1. *If G_1, G_2, and G_T are cyclic groups where the asymmetric GBDH assumption holds, H, H_1, and H_2 are random oracles, and Σ is IND-CPA secure, protocol* modifiedFSUrev *is rid-eCK secure.*

A detailed proof is shown in Appendix A.

4 Conclusions

We examined whether a revocation function can be added to a protocol, protocol FSU, that has been adopted as an international standard, ISO/IEC11770-3.

To make protocol FSU revocable, a generic technique converting an identity-based encryption scheme to a revocable identity-based encryption scheme is applied to protocol FSU.

In addition, to make the converted RIB-AKE protocol efficient, we reduced ephemeral information exchanged in the protocol, and introduced an additional parameter to the master public-key where the secret information of the additional parameter is not needed to include in the master secret-key.

We discussed the security of the resultant protocol, and proved that it is rid-eCK secure under the asymmetric GBDH assumption.

Acknowledgement.

We would like to thank Koutarou Suzuki, Junichi Tomida, Taroh Sasaki, Koki Iwai, and Takeru Kawaguchi for their valuable comments on the earlier works of this paper.

References

1. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: IEEE FOCS '97. pp. 394–403. IEEE Computer Society (1997)
2. Emura, K., Takayasu, A., Watanabe, Y.: Generic constructions of revocable hierarchical identity-based encryption. Cryptology ePrint Archive, Report 2021/515 (2021), https://eprint.iacr.org/2021/515
3. Fujioka, A., Hoshino, F., Kobayashi, T., Suzuki, K., Ustaoglu, B., Yoneyama, K.: id-eCK secure ID-based authenticated key exchange on symmetric and asymmetric pairing. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **96-A**(6), 1139–1155 (2013)
4. Fujioka, A., Suzuki, K., Ustaoglu, B.: Ephemeral key leakage resilient and efficient ID-AKEs that can share identities, private and master keys. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. Lecture Notes in Computer Science, vol. 6487, pp. 187–205. Springer (2010)
5. ISO/IEC 11770-3:2021 Information security – Key management Part 3: Mechanisms using asymmetric techniques (2021)
6. Nakagawa, K., Fujioka, A., Nagai, A., Tomida, J., Xagawa, K., Yasuda, K.: Making the identity-based Diffie-Hellman key exchange efficiently revocable. In: Aly, A., Tibouchi, M. (eds.) LATINCRYPT 2023. Lecture Notes in Computer Science, vol. 14168, pp. 171–191. Springer (2023)
7. Okano, Y., Tomida, J., Nagai, A., Yoneyama, K., Fujioka, A., Suzuki, K.: Revocable hierarchical identity-based authenticated key exchange. In: Park, J.H., Seo, S. (eds.) ICISC 2021. Lecture Notes in Computer Science, vol. 13218, pp. 3–27. Springer (2021)
8. Seo, J.H., Emura, K.: Revocable identity-based encryption revisited: Security model and construction. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. Lecture Notes in Computer Science, vol. 7778, pp. 216–234. Springer (2013)
9. Seo, J.H., Emura, K.: Revocable hierarchical identity-based encryption. Theoretical Computer Science **542**, 44–62 (2014)

A Security Proof of Theorem 1

We need the asymmetric GBDH assumption in the pairing groups G_1, G_2, and G_T, with generators of g_1, g_2 and g_T, respectively, where the orders of all g_i are q.

Let the inputs of the asymmetric GBDH problem be $U_1 := g_1^u, U_2 := g_2^u, V_1 := g_1^v, V_2 := g_2^v, W_1 := g_1^w$ and $W_2 := g_2^w$. Also, let $S := \hat{e}(g_1, g_2)^{uvw}$, the solution to the asymmetric GBDH problem.

Assume that a PPT adversary, $\mathcal{A}$, exists that breaks the rid-eCK security of our RIB-AKE protocol. We construct the asymmetric GBDH solver, $\mathcal{S}$, that simulates the protocol's environment using the DBDH oracle and can solve the asymmetric GBDH problem with non-negligible probability. $\mathcal{A}$ is said to be successful with non-negligible probability if $\mathcal{A}$ wins the distinguishing game with probability $\frac{1}{2} + f(\lambda)$, where $f(\lambda)$ is non-negligible, and event M denotes that $\mathcal{A}$ is successful.

Let the test session be $sid^* = (\Pi, \mathcal{I}, T, ID_A, ID_B, epk_A, epk_B, t)$ or $(\Pi, \mathcal{R}, T, ID_B, ID_A, epk_A, epk_B, t')$, which is a completed session between honest users, P_A and P_B, where user P_A is the initiator and user P_B is the responder of the test session, sid^*. Let H^* be the event that adversary $\mathcal{A}$ queries $(ST, \sigma_1, \sigma_2, \sigma_3)$ to H. Let $\overline{H^*}$ be the complement of event H^*. 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 $f(\lambda) \leq \Pr(M \wedge H^*)$. Henceforth, the event $M \wedge H^*$ is denoted by M^*.

Assume that adversary $\mathcal{A}$ succeeds in an environment with n_u $(\leq N)$ users and activates at most n_s sessions within a user.

Note that Y_1 and Y_2 may be revealed to $\mathcal{A}$ however the secret value of them, y, is not as y is erased after computation of Y_1 and Y_2.

Before going into the details of the proof, one expression is defined. We say that a shared value is *correctly formed* w.r.t. the static and ephemeral public-keys in a session when the shared value can be computed from the static and the ephemeral public-keys together with their secret-keys. Note that the correctness of σ_i can be checked with procedure `Check` described later.

Hereafter, we consider the following eight cases, which reflect the restrictions in freshness' definition in Section 2.5. See Table 1 for which keys are leaked to the adversary:

- Case 1: The owner and the peer of session sid^*, P_A and P_B, are non-revoked. The matching session, $\overline{sid^*}$, exists.
  (a) $\mathcal{A}$ queries $\mathbf{SSKRev}(ID_i^*)$ and $\mathbf{CSKRev}(ID_i^*, T^*)$ for $i = A, B$ ($\mathbf{MSKRev}$ is also possible.)
  (b) $\mathcal{A}$ makes queries $\mathbf{SSKRev}(ID_A^*)$, $\mathbf{CSKRev}(ID_A^*, T^*)$, and $\mathbf{ESKRev}(\overline{sid^*})$.
  (c) $\mathcal{A}$ makes queries $\mathbf{SSKRev}(ID_B^*)$, $\mathbf{CSKRev}(ID_B^*, T^*)$, and $\mathbf{ESKRev}(sid^*)$.
  (d) $\mathcal{A}$ queries $\mathbf{ESKRev}(sid^*)$ and $\mathbf{ESKRev}(\overline{sid^*})$.
- Case 2: The owner and the peer of session sid^*, P_A and P_B, are non-revoked. The matching session, $\overline{sid^*}$, does not exist.
  (a) $\mathcal{A}$ queries $\mathbf{SSKRev}(ID_A^*)$ and $\mathbf{CSKRev}(ID_B^*, T^*)$.
  (b) $\mathcal{A}$ issues $\mathbf{ESKRev}(sid^*)$.
- Case 3: The peer of the session sid^* is revoked. The matching session $\overline{sid^*}$ does not exist.
  (a) $\mathcal{A}$ makes queries $\mathbf{SSKRev}(ID_A^*)$, $\mathbf{CSKRev}(ID_A^*, T^*)$, and $\mathbf{SSKRev}(ID_B^*)$.
  (b) $\mathcal{A}$ queries $\mathbf{ESKRev}(sid^*)$ and $\mathbf{SSKRev}(ID_B^*)$.

Here, users P_A and P_B are the initiator and responder of the test session sid^*, respectively. Table 1 classifies events, named E_{1a}, E_{1b}, E_{1c}, E_{1d}, E_{2a}, E_{2b}, E_{3a}, and E_{3b}. In these tables, "ok" means that the secret-key is not revealed. "r" means that the secret-key may be revealed. "n" means that no matching session exists or the peer of the session, sid^*, is revoked. The "instance embedding" row shows how the simulator embeds an instance of the asymmetric GBDH problem.

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

Event $E_{1a} \wedge M^*$. $\mathcal{S}$ generate MSK $(= (z, y) \in_U \mathbb{Z}_q^2)$ and define $MPK := (Z_1, Z_2, Y_1, Y_2)$ $(= (g_1^z, g_2^z, U_1, U_2))$ First, $\mathcal{S}$ guesses the test session, sid^*, owned by two users, P_A and P_B. $\mathcal{S}$ selects P_A and P_B from n_u users. $\mathcal{S}$ guesses the test session with probability $1/n_u^2 n_s^2$, i.e., the test session is the t-th session initialized by **Send** query as the initiator and owned by P_A, and the matching session of the test session is the t'-th session initialized by **Send** query as the responder and owned by peer P_B. $\mathcal{S}$ sets the ephemeral public-key

Table 1. Classification of events.

	MSK	ssk_A	csk_A	esk_A	ssk_B	csk_B	esk_B	instance embedding
E_{1a}	r	r	r	ok	r	r	ok	$Y_1 := U_1,\ Y_2 := U_2,\ X_{A,1} := V_1,\ X_{B,2} := W_2$
E_{1b}	ok	r	r	ok	ok	ok	r	$Z_1 := U_1,\ Z_2 := U_2,\ X_{A,1} := V_1,\ Q_{B\|\|T,2} := W_2$
E_{1c}	ok	ok	ok	r	r	r	ok	$Z_1 := U_1,\ Z_2 := U_2,\ Q_{A\|\|T,1} := V_1,\ X_{B,2} := W_2$
E_{1d}	ok	ok	ok	r	ok	ok	r	$Z_1 := U_1,\ Z_2 := U_2,\ Q_{A\|\|T,1} := V_1,\ Q_{B\|\|T,2} := W_2$
E_{2a}	ok	r	r	ok	ok	ok	n	$Z_1 := U_1,\ Z_2 := U_2,\ X_{A,1} := V_1,\ Q_{B\|\|T,2} := W_2$
E_{2b}	ok	ok	ok	r	ok	ok	n	$Z_1 := U_1,\ Z_2 := U_2,\ Q_{A\|\|T,1} := V_1,\ Q_{B\|\|T,2} := W_2$
E_{3a}	ok	r	r	ok	r	n	n	$Z_1 := U_1,\ Z_2 := U_2,\ X_{A,1} := V_1,\ Q_{B\|\|T,2} := W_2$
E_{3b}	ok	ok	ok	r	r	n	n	$Z_1 := U_1,\ Z_2 := U_2,\ Q_{A\|\|T,1} := V_1,\ Q_{B\|\|T,2} := W_2$

"ok" means that the secret-key is not revealed. "r" means that the secret-key may be revealed. "n" means that no matching session exists or the peer of the session sid^* is revoked. The "instance embedding" row shows how the simulator embeds an instance of the asymmetric GBDH problem.

of t-th session of user P_A as $epk_A^* := V_1$. $\mathcal{S}$ sets the ephemeral public-key of t'-th session of user P_B as $epk_B^* := W_2$.

Note that esk_A^* $(= v)$ and esk_B^* $(= w)$ cannot be calculated; however, they are not asked in event E_{1a}.

$\mathcal{S}$ activates adversary $\mathcal{A}$ on this set of users and awaits the actions of $\mathcal{A}$.

$\mathcal{S}$ maintains a list, L_H, that contains queries and answers of H oracle, and a list, L_S, that contains queries and answers of **SKRev**. $\mathcal{S}$ maintains a list, L_{SSK}, that contains identifies and static secret-keys. $\mathcal{S}$ simulates oracle queries as follows:

1. **Send**$(\Pi, \mathcal{I}, T, ID_i, ID_{i'})$: If $i = A$, $i' = B$, and it is the t-th initialized session of A, i.e., the test session, records $(\Pi,\ \mathcal{I},\ T,\ ID_A,\ ID_B,\ epk_A^*,\ \cdot,\ *,\ t)$ in list $L_{\mathbf{Send}}$, and returns V_1. Otherwise, $\mathcal{S}$ computes epk_i, esk_i honestly, records $(\Pi,\ \mathcal{I},\ T,\ ID_i,\ ID_{i'},\ epk_i,\ \cdot,\ esk_i,\ s)$ in list $L_{\mathbf{Send}}$ when it is the s-th initialized session of P_i, and returns epk_i.
2. **Send**$(\Pi, \mathcal{R}, T, ID_i, ID_{i'})$: If $i = B$, $i' = A$, and it is the t'-th initialized session of B, i.e., the test session, i.e.,, $\mathcal{S}$ records $(\Pi,\ \mathcal{R},\ T,\ ID_B,\ ID_A,\ epk_B^*,\ \cdot,\ *,\ t')$ in list $L_{\mathbf{Send}}$, and returns W_2. Otherwise, $\mathcal{S}$ computes epk_i, esk_i honestly, records $(\Pi,\ \mathcal{R},\ T,\ ID_i,\ ID_{i'},\ epk_i,\ \cdot,\ esk_i,\ s')$ in list $L_{\mathbf{Send}}$ when it is the s'-th initialized session of P_i, and returns epk_i.
3. **Send**$(\Pi, \mathcal{I}, T, ID_i, ID_{i'}, epk_{i'})$: If it is the response corresponding to $(\Pi,\ \mathcal{I},\ T,\ ID_i,\ ID_{i'},\ epk_i,\ \cdot,\ esk_i,\ s)$, $\mathcal{S}$ records $(\Pi,\ \mathcal{I},\ T,\ ID_i,\ ID_{i'},\ epk_i,\ epk_{i'}\ esk_i,\ s)$ as completed in list $L_{\mathbf{Send}}$. Otherwise, $\mathcal{S}$ returns an error.
4. **Send**$(\Pi, \mathcal{R}, T, ID_i, ID_{i'}, epk_{i'})$: If it is the response corresponding to $(\Pi,\ \mathcal{R},\ T,\ ID_i,\ ID_{i'},\ \cdot,\ epk_i,\ esk_i,\ s')$, $\mathcal{S}$ records $(\Pi,\ \mathcal{R},\ T,\ ID_i,\ ID_{i'},\ epk_{i'},\ epk_i\ esk_i,\ s')$ as completed in list $L_{\mathbf{Send}}$. Otherwise, $\mathcal{S}$ returns an error.
5. $H(ST, \sigma_1, \sigma_2, \sigma_3)$:
   (a) If $(ST, \sigma_1, \sigma_2, \sigma_3)$ is recorded in list L_H, then $\mathcal{S}$ returns the value, SK, recorded in list L_H.
   (b) Else if there exists a session, $(\Pi,\ \mathcal{I},\ T,\ ID_i,\ ID_{i'}, epk_i,\ epk_{i'},\ s)$ or $(\Pi,\ \mathcal{R},\ ID_{i'},\ ID_i,\ epk_i,\ epk_{i'},\ s')$, recorded in list L_S where σ_1, σ_2, and σ_3 are correctly formed w.r.t. the static and ephemeral public-keys in the session, and the session is the test session, i.e., $i = A$, $i' = B$, and $s = t$ or $s' = t'$, then $\mathcal{S}$ computes the answer of the asymmetric GBDH instance by procedure `Extract` described below, and is successful by outputting the answer.

(c) Else if there exists a session, $(\Pi, \mathcal{I}, T, ID_i, ID_{i'}, epk_i, epk_{i'}, s)$ or $(\Pi, \mathcal{R}, ID_{i'}, ID_i, epk_i, epk_{i'}, s')$, recorded in list L_S where σ_1, σ_2, and σ_3 are correctly formed w.r.t. the static and ephemeral public-keys in the session, and the session is not the test session, then $\mathcal{S}$ returns the value, SK, recorded in list L_S and records SK and $(ST, \sigma_1, \sigma_2, \sigma_3)$ in list L_H.

(d) Otherwise, $\mathcal{S}$ returns a random value, SK, and records SK and $(ST, \sigma_1, \sigma_2, \sigma_3)$ in list L_H.

6. **SKRev**$((\Pi, \mathcal{I}, T, ID_i, ID_{i'}, epk_i, epk_{i'}, s)$ or $(\Pi, \mathcal{R}, T, ID_{i'}, ID_i, epk_i, epk_{i'}, s')$:

   (a) If $(\Pi, \mathcal{I}, T, ID_i, ID_{i'}, epk_i, epk_{i'}\ esk_i, s)$ or $(\Pi, \mathcal{R}, T, ID_{i'}, ID_i, epk_i, epk_{i'}\ esk_{i'}, s')$ is not recorded in $L_{\mathbf{Send}}$, $\mathcal{S}$ returns an error.

   (b) Else if $(\Pi, \mathcal{I}, T, ID_i, ID_{i'}, epk_i, epk_{i'}, s)$ or $(\Pi, \mathcal{R}, T, ID_{i'}, ID_i, epk_i, epk_{i'}, s')$ $(= sid))$ is recorded in list L_S, then $\mathcal{S}$ returns the value, SK, recorded in list L_S.

   (c) Else if there exists $(ST, \sigma_1, \sigma_2, \sigma_3)$ recorded in list L_H where σ_1, σ_2, and σ_3 are correctly formed w.r.t. the static and ephemeral public-keys in the session, then $\mathcal{S}$ returns the value, SK, recorded in list L_H and records sid and SK in list L_S.

   (d) Otherwise, $\mathcal{S}$ returns a random value, SK, and records sid and SK in list L_S.

7. **ESKRev**$((\Pi, \mathcal{I}, T, ID_i, ID_{i'}, epk_i, epk_{i'}, s)$ or $(\Pi, \mathcal{R}, T, ID_{i'}, ID_i, epk_i, epk_{i'}, s')$: If $(\Pi, \mathcal{I}, T, ID_i, ID_{i'}, epk_i, epk_{i'}, s)$ or $(\Pi, \mathcal{R}, T, ID_{i'}, ID_i, epk_i, epk_{i'}, s')$ $(= sid)$ is the test session, i.e., $i = A$, $i' = B$, $epk_i = V_1$, $epk_{i'} = W_2$, and $s = t$ or $s' = t'$, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ picks esk_i in list $L_{\mathbf{Send}}$, and returns it.
8. **SSKRev**(ID_i): If $(ID_i, D_{i,1}, D_{i,2}, K_i)$ is recorded in list L_{SSK}, $\mathcal{S}$ returns $D_{i,1}$, $D_{i,2}$, and K_i. Else if $(ID_i, q_{i,j}, Q_{i,j})$ is recorded in list L_{H_j} $(j = 1, 2)$, $\mathcal{S}$ sets $D_{i,j} = Q_{i,j}^z$ $(\in G_j)$ $(j = 1, 2)$, chooses a secret-key, K_i, of Σ, records $(ID_i, D_{i,1}, D_{i,2}, K_i)$ in L_{SSK}, and returns $D_{i,1}$, $D_{i,2}$, and K_i. Otherwise, $\mathcal{S}$ sets $Q_{i,j} = H_j(ID_i)$ $(= g_j^{q_{i,j}})$ $(\in G_j)$ $(j = 1, 2)$, sets $D_{i,j} = Q_{i,j}^z$ $(\in G_j)$ $(j = 1, 2)$, chooses a secret-key, K_i, of Σ, records $(ID_i, D_{i,1}, D_{i,2}, K_i)$ in L_{SSK}, records $(ID_i, q_{i,j}, Q_{i,j})$ in list L_{H_j} $(j = 1, 2)$, and returns $D_{i,1}$, $D_{i,2}$, and K_i.
9. **KeyUp**(T): If $T > T_{cu}$ where T_{cu} is the current time counter which $\mathcal{S}$ manages, $\mathcal{S}$ returns $\perp$. Else if (T, ku_T) is recorded in list L_{KU}, $\mathcal{S}$ returns ku_T.
   Otherwise, $\mathcal{S}$ returns ku_T as follows:
   - $\mathcal{S}$ sets $ku_T = \emptyset$.
   - For each user, P_i, not revoked at T, $\mathcal{S}$ sets $Q_{i||T,j} = H_j(ID_i||T) = g_j^{q_{i||T,j}}$ $(\in G_j)$ $(j = 1, 2)$ and sets $D_{i||T,j} = Q_{i||T,j}^z$ $(\in G_j)$ $(j = 1, 2)$.
   - If $(ID_i, D_{i,1}, D_{i,2}, K_i)$ is recorded in list L_{SSK}, $\mathcal{S}$ computes $C_{i,1} \leftarrow \mathbf{Enc}(K_i, D_{i||T,1})$, $C_{i,2} \leftarrow \mathbf{Enc}(K_i, D_{i||T,2})$, and adds $(ID_i, C_{i,1}, C_{i,2})$ to ku_T.
   - Else if $(ID_i, q_{i,j}, Q_{i,j})$ is recorded in list L_{H_j} $(j = 1, 2)$, $\mathcal{S}$ sets $D_{i,j} = Q_{i,j}^z$ $(\in G_j)$ $(j = 1, 2)$, chooses a secret-key, K_i, of Σ, and records $(ID_i, D_{i,1}, D_{i,2}, K_i)$ in L_{SSK}. $\mathcal{S}$ computes $C_{i,1} \leftarrow \mathbf{Enc}(K_i, D_{i||T,1})$, $C_{i,2} \leftarrow \mathbf{Enc}(K_i, D_{i||T,2})$, and adds $(ID_i, C_{i,1}, C_{i,2})$ to ku_T.
   - Otherwise, $\mathcal{S}$ sets $Q_{i,j} = H_j(ID_i) = g_j^{q_{i,j}}$ $(\in G_j)$ $(j = 1, 2)$, sets $D_{i,j} = Q_{i,j}^z$ $(\in G_j)$ $(j = 1, 2)$, chooses a secret-key, K_i, of Σ, records $(ID_i, D_{i,1}, D_{i,2}, K_i)$ in L_{SSK}, and records $(ID_i, q_{i,j}, Q_{i,j})$ in list L_{H_j} $(j = 1, 2)$, and $\mathcal{S}$ computes $C_{i,1} \leftarrow \mathbf{Enc}(K_i, D_{i||T,1})$, $C_{i,2} \leftarrow \mathbf{Enc}(K_i, D_{i||T,2})$, and adds $(ID_i, C_{i,1}, C_{i,2})$ to ku_T.
   - Finally, $\mathcal{S}$ adds (T, ku_T) to L_{KU} and returns ku_T.

10. **CSKRev**(ID, T): If $T > T_{cu}$ or $ID \in RL$, $\mathcal{S}$ returns $\perp$. Else if (T, ku_T) is recorded in list L_{KU}, $\mathcal{S}$ computes $D_{i||T,1} \leftarrow \mathbf{Dec}(K_i, C_{i,1}), D_{i||T,2} \leftarrow \mathbf{Dec}(K_i, C_{i,2})$, sets $csk_{i,T} = (D_{i||T,1}, D_{i||T,2})$, and returns $csk_{i,T}$. Otherwise, $\mathcal{S}$ generates ku_T as Step 9, sets $csk_{i,T} = (D_{i||T,1}, D_{i||T,2})$, and returns $csk_{i,T}$.
11. **MSKRev**(): $\mathcal{S}$ returns z.
12. **Test**(sid): If sid is not t-th session of P_A, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ responds to the query faithfully.
13. If $\mathcal{A}$ outputs a guess, $\mathcal{S}$ aborts with failure.
14. $H_j(ID_i)$ $(j = 1, 2)$: If $(ID_i,\ q_{i,j},\ Q_{i,j})$ is recorded in list L_{H_j}, then $\mathcal{S}$ returns hash value $Q_{i,j}$. Otherwise, $\mathcal{S}$ selects random $q_{i,j}$, sets $Q_{i,j} = H_t(ID_i) = g^{q_{i,j}}$, records $(ID_i,\ q_{i,j},\ Q_{i,j})$ in list L_{H_j}, and returns $Q_{i,j}$.

`Extract`. This procedure computes the answer to the asymmetric GBDH instance as follows: $\sigma_3' = \sigma_3 / \hat{e}(X_{A,1}, X_{B,2})^z = g_T^{yx_Ax_B} = g_T^{uvw}$

`Check`. This procedure checks whether the shared secrets are correctly formed w.r.t. the static and ephemeral public-keys, and can consistently simulate **SKRev** and H queries. More precisely, in the simulation of the $H(ST,\ \sigma_1,\ \sigma_2,\ \sigma_3)$ query, solver $\mathcal{S}$ needs to check that the shared secrets, σ_1, σ_2, and σ_3, are correctly formed, and if so, $\mathcal{S}$ returns session key SK being consistent with the previously answered **SKRev**$(\Pi,\ \mathcal{I},\ T,\ ID_A,\ ID_B,\ epk_A,\ epk_B,\ s)$ and **SKRev**$(\Pi,\ \mathcal{R},\ T,\ ID_B,\ ID_A,\ epk_A,\ epk_B,\ s')$ queries. The solver, $\mathcal{S}$, can check if shared secrets, σ_1, σ_2, and σ_3, are correctly formed w.r.t. the static and ephemeral public-keys by asking $\mathrm{DBDH}^{1,1,2}$ oracle as

$$\begin{aligned} \mathrm{DBDH}^{1,1,2}(Z_1, Q_{A||T,1}, Q_{B||T,2}, \sigma_1) &= 1, \\ \mathrm{DBDH}^{1,1,2}(Z_1, Q_{A||T,1}X_{A,1}, Q_{B||T,2}X_{B,2}, \sigma_2) &= 1, \\ \mathrm{DBDH}^{1,1,2}(Z_1Y_1, X_{A,1}, X_{B,2}, \sigma_3) &= 1, \end{aligned}$$

and this implies that σ_1, σ_2, and σ_3 are correctly formed.

Notice that, in other cases in Table 1, the solver, $\mathcal{S}$, can check whether shared secrets, σ_1, σ_2, and σ_3, are correctly formed or not with the same procedure.

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 P_A is initiator, P_B is responder, $X_{A,1}$ is V_1, and $X_{B,2}$ is W_2 as the test session, sid^*, is at least $\frac{1}{n_u^2 n_s^2}$. Suppose this is indeed the case, solver $\mathcal{S}$ does not abort in Step 12.

Suppose event E_{1a} occurs, solver $\mathcal{S}$ does not abort in Steps 7.

Suppose event M^* occurs, adversary $\mathcal{A}$ queries correctly formed σ_1, σ_2, and σ_3 to H. Therefore, solver $\mathcal{S}$ is successful as described in Step 5c since σ_1, σ_2, and σ_3 are correctly formed, and does not abort as in Step 13.

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

Event $E_{1b} \wedge M^*$. The reduction to the asymmetric GBDH assumption is similar to event $E_{1a} \wedge M^*$ except for the following points:

$\mathcal{S}$ sets $Z_1 = U_1$, $Z_2 = U_2$, $X_{A,1} = V_1$, and $Q_{B||T,2} = W_2$. $\mathcal{S}$ obtains the solution $g_T^{zx_Aq_{B||T,2}}$ as $\sigma_2 \cdot (\hat{e}(X_{A,1}, Z_2)^{x_B})^{-1} \cdot (\hat{e}(Z_1, Q_{B||T,2}X_{B,2})^{q_{A||T,1}})^{-1} = g_T^{zx_Aq_{B||T,2}}$.

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

Event $E_{1c} \wedge M^*$. The reduction to the asymmetric GBDH assumption is similar to event $E_{1a} \wedge M^*$ except for the following points:

$\mathcal{S}$ sets $Z_1 = U_1$, $Z_2 = U_2$, $Q_{A||T,1} = V_1$, and $X_{B,2} = W_2$. $\mathcal{S}$ obtains the solution $g_T^{zq_{A||T,2}x_B}$ as $\sigma_2 \cdot (\hat{e}(Q_{A||T,1}, Z_2)^{q_{B||T,2}})^{-1} \cdot (\hat{e}(Z_1, Q_{B||T,2}X_{B,2})^{x_A})^{-1} = g_T^{zq_{A||T,2}x_B}$.

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

Event $E_{1d} \wedge M^*$. The reduction to the asymmetric GBDH assumption is similar to event $E_{1a} \wedge M^*$ except for the following points:

$\mathcal{S}$ sets $Z_1 = U_1$, $Z_2 = U_2$, $Q_{A||T,1} = V_1$, and $Q_{B||T,2} = W_2$. $\mathcal{S}$ obtains the solution $g_T^{zq_{A||T,1}q_{B||T,2}}$ as outputting σ_1.

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

Event $E_{2a} \wedge M^*$. The reduction to the asymmetric GBDH assumption is similar to event $E_{1a} \wedge M^*$ except for the following points:

$\mathcal{S}$ sets $Z_1 = U_1$, $Z_2 = U_2$, $X_{A,1} = V_1$, and $Q_{B||T,2} = W_2$. $\mathcal{S}$ obtains the solution $g_T^{zx_A q_{B||T,2}}$ as $\sigma_2 \cdot (\sigma_3 \cdot (\hat{e}(X_{A,1}, X_{B,2})^y)^{-1}) \cdot (\hat{e}(Z_1, Q_{B||T,2}X_{B,2})^{q_{A||T,1}})^{-1} = g_T^{zx_A q_{B||T,2}}$.

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

Event $E_{2b} \wedge M^*$. The reduction to the asymmetric GBDH assumption is similar to event $E_{1a} \wedge M^*$ except for the following points:

$\mathcal{S}$ sets $Z_1 = U_1$, $Z_2 = U_2$, $Q_{A||T,1} = V_1$, and $Q_{B||T,2} = W_2$. $\mathcal{S}$ obtains the solution $g_T^{zq_{A||T,1}q_{B||T,2}}$ as outputting σ_1.

$\mathcal{S}$ is successful with probability $\Pr(S) \geq \frac{p_{2b}}{n_u^2}$, where p_{2b} is probability that $E_{2b} \wedge M^*$ occurs.

Event $E_{3a} \wedge M^*$. The reduction to the asymmetric GBDH assumption is similar to event $E_{1a} \wedge M^*$ except for the following points:

$\mathcal{S}$ sets $Z_1 = U_1$, $Z_2 = U_2$, $X_{A,1} = V_1$, and $Q_{B||T,2} = W_2$. $\mathcal{S}$ obtains the solution $g_T^{zx_A q_{B||T,2}}$ as $\sigma_2 \cdot (\sigma_3 \cdot (\hat{e}(X_{A,1}, X_{B,2})^y)^{-1}) \cdot (\hat{e}(Z_1, Q_{B||T,2}X_{B,2})^{q_{A||T,1}})^{-1} = g_T^{zx_A q_{B||T,2}}$.

$\mathcal{S}$ is successful with probability $\Pr(S) \geq \frac{p_{3a}}{n_u^2 n_s}$, where p_{3a} is probability that $E_{3a} \wedge M^*$ occurs.

Event $E_{3b} \wedge M^*$. The reduction to the asymmetric GBDH assumption is similar to event $E_{1a} \wedge M^*$ except for the following points:

$\mathcal{S}$ sets $Z_1 = U_1$, $Z_2 = U_2$, $Q_{A||T,1} = V_1$, and $Q_{B||T,2} = W_2$. $\mathcal{S}$ obtains the solution $g_T^{zq_{A||T,1}q_{B||T,2}}$ as outputting σ_1.

$\mathcal{S}$ is successful with probability $\Pr(S) \geq \frac{p_{3b}}{n_u^2}$, where p_{3b} is probability that $E_{3b} \wedge M^*$ occurs.