Identity-Based Key Agreement with Unilateral Identity Privacy Using Pairings

Zhaohui Cheng[1], Liqun Chen[2], Richard Comley[1], and Qiang Tang[3]

[1] School of Computing Science, Middlesex University
The Burroughs Hendon, London NW4 4BT, UK
{m.z.cheng,r.comley}@mdx.ac.uk
[2] Hewlett-Packard Laboratories, Bristol, UK
liqun.chen@hp.com
[3] Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
qiang.tang@rhul.ac.uk

Abstract. In most of the existing identity-based key agreement schemes, it is usually assumed that either the communicated parties know each other's identifier before the protocol starts or their identifiers are transferred along with the protocol messages. However, these schemes are not suitable for use in many real-world applications aimed to achieve unilateral identity privacy, which means that one communicating party does not want to expose his identifier to an outsider while his partner cannot know his identifier in advance. In this paper, we propose an efficient identity-based two-party key agreement scheme with unilateral identity privacy using pairing, and formally analyze its security in a modified Bellare-Rogaway key agreement security model.

1 Introduction

Two-party key agreement has been well investigated since the Diffie-Hellman (DH) key exchange was proposed [20]. Because the key agreement schemes are usually supposed to be used in open networks, where the communication is fully controlled by adversaries, most of the research efforts have been devoted to the authenticated key agreement schemes. An authenticated key agreement (AK for short) scheme can establish a secret between (among) the involved parties and guarantee no other party aside from the intended one can gain access to the established session secret key. Moreover, an authenticated key agreement with key confirmation (AKC for short) [27] further guarantees that if the protocol successfully ends then every party can be assured that the others actually possess the same session key. Besides the basis security properties required by AK and AKC, the following general security properties (some of them are discussed in [11, 25]) might also be required depending on the deployment environment.

– *Known session key security (KnSK).* The compromise of one session key should not compromise the keys established in other sessions.

- *Forward secrecy (FwS).* If one party's private key is compromised, the secrecy of session keys previously established by the party should not be affected. *Perfect forward secrecy (PFS)* requires that the session key is still secure even when both parties' private keys are compromised. In identity-based schemes, there is a master key which could be compromised as well. The *master-key forward secrecy* requires that the session keys generated before are still secure even when the master key is compromised.
- *Key-compromise impersonation resilience (KCI).* The compromise of party A's long-term private key should not enable the adversary to impersonate other parties to A.
- *Unknown key-share resilience (UknKSh).* Party A should not be able to be coerced into sharing a key with party C when in fact A thinks that he is sharing the key with some party B.
- *Key control (KContl).* One single party should not be able to decide the formation of the session key.

Apart from these general security properties, other special security properties might also be required in certain environment. In this paper we are especially interested in preserving one user's *identity privacy (IP)* during every (key agreement) protocol execution, which means that no outsider adversary can determine the user's identity. User identity privacy is very important in some environments such as the mobile networks. In particular, the privacy of user identity over the air interface of mobile networks is essential to guarantee the user's location privacy and prevent an adversary from associating a user with his activities [6, 22]. Even in general network environments, some key agreement protocols such as the Internet Key Exchange protocol [21], identity privacy is also a desirable property.

Authenticated key agreement protocols can be built based on both symmetric cryptography and asymmetric cryptography. If symmetric cryptography is used then either a symmetric key or a password should be distributed before the key agreement protocol is executed. If traditional (certificate-based) asymmetric cryptography is used, normally a public key infrastructure (PKI) will be required to be deployed. Shamir proposed the concept of identity-based cryptosystem [32] in which each party's public key is the party's identity which can be an arbitrary string. Hence, in the identity-based systems no certificate is required so that the management of public keys is greatly simplified. Following Shamir's work many identity-based two-party key agreement schemes using pairings have been proposed (e.g., [35, 34, 15, 31, 19, 26, 38, 36]).

More or less, these protocols also consider the general security properties of key agreement protocols. Based on the assumption that two participants know each other's identity in advance, some of these schemes also achieve user identity privacy. However, such assumption that two parties know each other's identity in advance might not hold in many practical environments. For example, in the mobile networks of *infrastructure* mode, a mobile station could know the identifier of an access point (or a base station in the telecommunication networks) by some means, such as through a broadcast message from the access point, but the access point cannot know the mobile station's identifier *a prior*. Similarly in the remote log-in services, a server cannot know a log-in user's identifier before hand.

Achieving identity privacy in a key establishment protocol is not a new challenge. A lot of research has been done to address this issue, especially in the mobile networks, e.g., [6, 1, 22]. However, these solutions rely on either the symmetric cryptography or traditional asymmetric cryptography so that it might cause problems when practically deployed. In this paper we mainly focus on the identity-based (using pairings) two-party key agreement schemes aimed to achieve user identity privacy.

In the literature of identity-based schemes, some identity-based encryption schemes, such as the signcryption schemes [2, 16] and the authenticated encryption scheme [12], are closely related to one-pass authenticated key establishment schemes. These schemes in [2, 16, 12] achieve the basic *ciphertext unforgeability* property, i.e. without both the sender and the recipient private key, the adversary cannot forge a ciphertext which can be accepted by the recipient, and *message confidentiality* property, i.e. the adversary cannot decrypt a ciphertext even if it has compromised the sender's private key. Moreover, these schemes achieve *ciphertext anonymity*, i.e. the ciphertext hides both the sender and recipient identifiers to any outsider. However, it is clear that these schemes ([2, 16, 12]) cannot be directly used as an AK or AKC protocol and those in [2, 16] will have high computation complexity if they are adjusted and used as key establishment protocols. Some other pairing-based cryptographic primitives also consider the identity privacy, such as the secret-handshaking scheme [7] and the ad-hoc anonymous identification scheme [28]. However, these primitives do not fit the requirement of authenticated key agreement either.

The rest of the paper is organised as follows. In Section 2 we briefly introduce pairings, the related assumptions, and some other necessary primitives. In Section 3 we describe the security model used to evaluate a two-party key agreement protocol with identity privacy. In Section 4 we propose an efficient authenticated key agreement with unilateral user identity privacy, investigate its security, and compare its security attributes and computation complexity with some existing protocols. Finally, we conclude this paper.

2 Preliminary

2.1 Bilinear Pairings and Assumptions

In this subsection we briefly introduce some background knowledge of the pairings.

Definition 1 [4] *A pairing is a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q, which has the following properties:*

1. *Bilinear: $\hat{e}(sP, tR) = \hat{e}(P, R)^{st}$ for all $P, R \in \mathbb{G}_1$ and $s, t \in \mathbb{Z}_q$.*
2. *Non-degenerate: $\hat{e}(P, P) \neq 1_{\mathbb{G}_2}$ for $P \in \mathbb{G}_1^*$.*

3. Computable: $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}_1$ is polynomial-time computable.

In the literature the security of many the identity-based schemes are based on the following assumptions.

Assumption 1 (Bilinear Diffie-Hellman (BDH) [4]) *For $x, y, z \in_R \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, given $\langle P, xP, yP, zP\rangle$, computing $\hat{e}(P, P)^{xyz}$ is hard.*

Assumption 2 (Decisional Bilinear Diffie-Hellman (DBDH)) *For $x, y, z, r \in_R \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, distinguishing between the distributions $\langle P,\ xP,\ yP,\ zP,\ \hat{e}(P, P)^{xyz}\rangle$ and $\langle P,\ xP,\ yP,\ zP,\ \hat{e}(P, P)^r\rangle$ is hard.*

Assumption 3 (Gap Bilinear Diffie-Hellman (GBDH) [30]) *With the help of an oracle to solve the DBDH problem, solving the BDH problem is still hard.*

2.2 Some Primitives

In order to hide one user's identity, we need to employ an one-time symmetric-key encryption scheme specified below.

Definition 2 [18] *An one-time symmetric-key encryption $\mathcal{E}$ consists of two deterministic polynomial-time algorithms:*

- **Encrypt** $\mathbb{E}(EK, m)$*: Given a secret key $EK \in \mathcal{K}$ (the secret key space) and a message m from message space $\mathcal{M}$, the algorithm outputs the ciphertext c.*
- **Decrypt** $\mathbb{D}(EK, c)$*: Given a secret key $EK \in \mathcal{K}$ and a cipertext c, the algorithm outputs the plaintext m.*

which satisfy $\mathbb{D}(EK, \mathbb{E}(EK, m)) = m$.

A one-time symmetric-key encryption secure against passive attacks requires that for any passive adversary who possesses a pair of PPT algorithms (A_1, A_2), the following function $\epsilon(k)$ is negligible.

$$\epsilon(k) = \Pr[b = b' \mid \begin{array}{c} ((m_1, m_2), \sigma) \leftarrow A_1(1^k);\ b \leftarrow \{0, 1\};\ EK \leftarrow \mathcal{K}; \\ c^* \leftarrow \mathbb{E}(EK, m_b);\ b' \leftarrow A_2(1^k, \sigma, c^*) \end{array}]$$

Note that, m_1 and m_2 could be with equal length of any polynomial of k from the message space $\mathcal{M}$. A passive secure one-time symmetric-key encryption achieves the ciphertext indistinguishability property with no encryption or decryption oracle access for the adversary (i.e. the IND-P0-C0 security [29]). As suggested in [18], this type of encryption scheme can be built from standard encryption techniques, such as using a pseudo-random bit generator (PRG) to extend EK and then employing "one-time pad" operation. In this paper, we specifically use such *"one-time pad" with pseudorandom bit stream* scheme.

To provide key confirmation for AKC, we shall use a secure message authentication code (MAC) specified as follows.

Definition 3 [3] *A message authentication code (MAC) is a pair of polynomial algorithms (G, M):*

- *On input 1^k, algorithm G uniformly chooses a bit string s from its range.*
- *For every s in the range of $G(1^k)$ and for every $m \in \{0,1\}^*$, the deterministic algorithm M computes $\alpha = M(s, m)$. α is called the tag of message m under $M(s, \cdot)$. We shorthand $M(s, \cdot)$ as $M_s(\cdot)$.*

For a probabilistic oracle machine F^{M_s}, which has the access to the MAC oracle M_s, we denote by $Q_F^{M_s}(x)$ the set of the queries made by F^{M_s}. A MAC scheme is secure *if for every probabilistic polynomial oracle machine $F^{M_{(\cdot)}}$, the function $\epsilon(k)$ defined by:*

$$\epsilon(k) = \Pr[M_s(m) = \alpha \wedge m \notin Q_F^{M_s}(1^k) \mid s \leftarrow G(1^k) \text{ and } (m, \alpha) \leftarrow F^{M_s}(1^k)]$$

is negligible, where the probability is taken over the coin tosses of algorithms G and $F^{M_{(\cdot)}}$.

3 Security Model of Key Agreement with Identity Privacy

In this paper we use a modified Blake-Wilson et al.'s security model [11] which extends the Bellare-Rogaway model [10] (referred to as the BR model) to the public key setting.

In the BR model [10], each party involved in a session is treated as an oracle, and an adversary can access the oracle by issuing some specified queries (defined below).An oracle $\Pi_{i,j}^s$ denotes the s-th instance of party i involved with a partner party j in a session (note that an oracle $\Pi_{i,j}^s$ can be uniquely identified by i and s). The oracle $\Pi_{i,[j]}^s$ executes the prescribed protocol Π and produces the output as $\Pi(1^k, i, [j], S_i, P_i, [P_j], conv_{i,[j]}^s, r_{i,[j]}^s, x)$= $(m, \delta_{i,[j]}^s, \sigma_{i,j}^s, [j])$ where x is the input message; m is the outgoing message; S_i and P_i are the private/public key pair of party i; $\delta_{i,[j]}^s$ is the decision of the oracle (accept or reject the session or no decision yet); $\sigma_{i,j}^s$ is the generated session key and P_j is the public key of the intended partner j (see [10, 11] for more details). After the response is generated, the conversation transcript $conv_{i,[j]}^s$ is updated as $conv_{i,[j]}^s.x.m$ (where "$a.b$" denotes the result of the concatenation of two strings, a and b).

There are some differences between the protocol formulation described here and those in [10, 11] because we need to consider the identity privacy property. An oracle may not know its partner identity during some stage of the protocol so that the partner's identifer and its public key are presented as an optional input (denoted by $[z]$) to the protocol's algorithm Π. If an oracle does not know its partner's identifier, it cannot use this information when generating the response, but at the end of the protocol it should be able to output the intended partner's identifier, if it accepts the session.

There is an obstacle in the BR model to simulate the situation that an adversary only passively eavesdrops the conversation between two parties when the protocol has identity privacy, because in the BR model, the adversary fully controls the network and schedules the messages among the parties. Without knowing the identity of oracles, the adversary cannot dispatch the messages. In [14], Canetti et al. introduced an extra identifier "destination address" to simulate the "post-specified peer setting" which is a related notion to identity privacy. In the model of [14], a session is activated by the adversary using (i, t, d) where i is the party identifier, t is the session identifier, and d is the destination address. However, although using destination address d can simulate the situation that the party sends out a message to an unknown party, such activation still allows the adversary to know the source identifier of the message.

Instead of introducing further complexity into the model, we consider identity privacy only when simulating the behavior of an honest party (who strictly follows the protocol specification and is modeled as an oracle) and allow the adversary to know the (source and possibly destination) identifiers of each message generated by the honest party (we can think that in the attack the honest parties reveal to the adversary the hidden identifiers in messages). While, it is not required that the adversary knows the identifiers of a message faked by itself. Hence the adversary's ability is not restricted, but the model is only used to test the common security attributes, such as mutual authentication and session key security. Identity privacy has to be scrutinized separately. This is also the strategy used in [14].

The security of a protocol is tested by a game with two phases. In the first phase, an adversary E is allowed to issue the following queries in any order.

1. Send a message with knowing partner: $Send(\Pi^s_{i,j}, x)$. Oracle $\Pi^s_{i,j}$ executes $\Pi(1^k, i, j, S_i, P_i, P_j, conv^s_{i,j}, r^s_{i,j}, x)$ and responds with m and $\delta^s_{i,j}$. If the oracle $\Pi^s_{i,j}$ does not exist, it will be created. Message x can be λ in the query which causes an oracle to be generated as an initiator, otherwise as a responder. This is a normal query formalized in [10, 11]. We note that even in a protocol with identity privacy, one party may know its intended partner before receiving any message.
2. Send a message without knowing partner: $Send(\Pi^s_{i,*}, x)$. Oracle $\Pi^s_{i,*}$ who does not know its partner so far, follows the protocol $\Pi(1^k, i, S_i, P_i, conv^s_{i,*}, r^s_{i,*}, x)$ to generate response. If the oracle recovers a partner identifier j through some algorithm $\mathcal{F}$: $j = \mathcal{F}(1^k, i, S_i, P_i, conv^s_{i,*}, r^s_{i,*}, x)$, it replaces the unknown partner identifier $*$ with j and retrieves j's public key P_j. This is a new query in the model, but a similar formulation has been used in [14].
3. Reveal a session's agreed session key: $Reveal(\Pi^s_{i,*})$. If the oracle has not accepted, it returns $\perp$; otherwise, it must have known its partner j, and then $\Pi^s_{i,j}$ reveals the session's private output $\sigma^s_{i,j}$. Note that the adversary may not know an oracle's partner. Here, the oracle is not required to disclose the partner's identity even if it has accepted the session. The adversary's capacity can be further strengthened by obtaining the partner's identifier information of an oracle when issuing the query.

4. Corrupt a party: $Corrupt(i)$. The party i responds with the private key S_i. Here, the adversary is not allowed to replace a party's private key because the attack is impossible in the identity-based schemes.

Once the adversary decides that the first phase is over, it starts the second phase by choosing a *fresh oracle* $\Pi_{i,*}^s$ and issuing a $Test(\Pi_{i,*}^s)$ query, where the *fresh oracle* $\Pi_{i,*}^s$ and $Test(\Pi_{i,*}^s)$ query are defined as follows.

Definition 4 (fresh oracle) *An oracle $\Pi_{i,*}^s$ is fresh if (1) $\Pi_{i,*}^s$ has accepted (hence it knows the partner j); (2) $\Pi_{i,j}^s$ is unopened (not being issued the* Reveal *query); (3) j is not corrupted (not being issued the* Corrupt *query); (4) there is no opened oracle $\Pi_{j,i}^t$, which has had a matching conversation to $\Pi_{i,j}^s$.*

It should be noted that this concept of fresh oracle is particularly defined to address the key-compromise impersonation resilience property [17] since it implies that the user i could have been issued a *Corrupt* query.

5. $Test(\Pi_{i,*}^s)$. Oracle $\Pi_{i,*}^s$ which is fresh, so knows its partner j, as a challenger, randomly chooses $b \in \{0, 1\}$ and responds with $\sigma_{i,j}^s$, if $b = 0$; otherwise it returns a random sample generated according to the distribution of the session secret $\sigma_{i,j}^s$.

If the adversary guesses the correct b, we say that it wins. the adversary's advantage is defined as

$$Advantage^E(k) = \max\{0, \Pr[E \text{ wins}] - \tfrac{1}{2}\}.$$

We use the session ID, which can be the concatenation of the messages in a session (see [9]), to define matching conversations, i.e. two oracles $\Pi_{i,j}^s$ and $\Pi_{j,i}^t$ have the *matching conversations* if they derive the same session ID from their own conversation transcripts.

A secure authenticated key (AK) agreement protocol is defined as follows.

Definition 5 [11] *Protocol Π is a secure AK if:*

1. *In the presence of the benign adversary, which faithfully conveys messages, on $\Pi_{i,j}^s$ and $\Pi_{j,i}^t$, both oracles always accept holding the same session key σ, and this key is distributed uniformly at random on $\{0,1\}^k$;*

and if for every adversary E:

2. *If two oracles $\Pi_{i,j}^s$ and $\Pi_{j,i}^t$ with matching conversations have accepted* [4] *and both i and j are uncorrupted, then both hold the same session key σ;*

[4] In [11], this condition is not required. Instead, it is required that two oracles with matching conversation should both accept. While in a protocol with identity privacy, a party may only know its partner when it has accepted the session. We note that only when the partner is determined, a meaningful matching conversation can be defined. In [14], it is defined that an unfinished oracle has a matching conversation to any other oracle.

3. *Advantage$^E(k)$ is negligible.*

As demonstrated in [11, 17] that if a protocol is secure regarding the above formulation, it achieves the session key authentication and the general security properties: known session key security, key-compromise impersonation resilience and unknown key-share resilience. As in principle, there is no public-key based DH style key agreement without signature that can achieve PFS, in this paper we adopt a weaker definition of PFS [17].

Definition 6 *An AK protocol is said to be forward secure if the adversary wins the game with negligible advantage when it chooses as the challenger (i.e. in place of the fresh oracle) an* unopened *oracle $\Pi^s_{i,j}$ which has a matching conversation to an* unopened *oracle $\Pi^t_{j,i}$ and both oracles accepted. If both i and j can be corrupted then the protocol achieves perfect forward secrecy.*

Apart from the requirements in Definition 5, an AKC *additionally* requires that the following *No-Matching* event can only happen with a negligible probability.

Definition 7 *No-matching$^E(k)$ is the event that there exist i, j, s such that an oracle $\Pi^s_{i,j}$ accepted and there is no accepted oracle $\Pi^t_{j,i}$ which has engaged in a matching conversation to $\Pi^s_{i,j}$ and party j is uncorrupted.*

4 A Key Agreement Protocol with Identity Privacy

In this section we propose an efficient identity-based two-party key agreement protocol with client identity privacy for use in the client-server environment, and discuss its security properties. It should be noted that it can also be used in other environments with the same security requirements.

4.1 Description of the Scheme

In the system there is a Key Generation Center (KGC) which with the given security parameter k generates the system parameters as follows:

1. Generate two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Pick a random generator $P \in \mathbb{G}_1^*$.
2. Pick a random $s \in \mathbb{Z}_q^*$ as the **master key** and compute $P_{pub} = sP$.
3. Pick six cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$, $H_3 : \mathbb{G}_2 \rightarrow \{0,1\}^n$, $H_4 : \mathbb{G}_2 \rightarrow \{0,1\}^l$, $H_5 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \{0,1\}^w$, and $H_6 : \mathbb{G}_2 \rightarrow \{0,1\}^w$ for some integers $n, w, l > 0$. In the practice, we can set $n = 128$, $l = 160$ and $w = 128$ for instance.

The KGC keeps the **master key** as a secret and publishes other parameters. For any user with an identity $ID \in \{0,1\}^*$, the KGC computes $Q_{ID} = H_1(ID)$, $d_{ID} = sQ_{ID}$ and passes d_{ID} as the private key to this party via some secure channel.

We suppose that the client possesses the identity A and the server possesses the identity B. Without loss of generality, we assume that there is always an association procedure (such as a TCP connection procedure or a physical connection establishment phase) between the client and the server before starting the key agreement protocol.

Once the association has been established, A and B proceed as follows (this process is also depicted in Fig. 1. The operations included by [ ] are specified for the AKC version.).

1. The server B randomly chooses $r_B \in \mathbb{Z}_q^*$ and sends (B, r_BQ_B) to the client, where $Q_B = H_1(B)$. It should be noted that the server B does not know the identity of the client.
2. The client A responds as follows. 1) randomly choose $r_A \in \mathbb{Z}_q^*$; 2) compute $h = H_2(r_AQ_A, r_BQ_B)$; 3) compute the agreed secret $K = \hat{e}(d_A, r_BQ_B + hQ_B)^{r_A}$; 4) generate an encryption key $EK = H_3(K)$; 5) use a (passively) secure symmetric-key encryption $\mathcal{E}$, specifically the "one-time pad" with pseudorandom bit stream scheme defined in Section 2.2, to encrypt message $m = (r_A, A)$ with EK as $\{m\}_{EK}$; 6) send r_AQ_A and the ciphertext $\{m\}_{EK}$ to the server B.

$$
\begin{array}{rcl}
(Client)A & & B(Server) \\
\hline
 & \xleftarrow{B, r_BQ_B} & r_B \in_R \mathbb{Z}_q^*. \\
r_A \in_R \mathbb{Z}_q^*; & r_AQ_A, \{m\}_{EK}, & \\
h = H_2(r_AQ_A, r_BQ_B); & \xrightarrow{[\mathtt{MAC}_1]} & \\
K = \hat{e}(d_A, r_BQ_B + hQ_B)^{r_A}; & & h = H_2(r_AQ_A, r_BQ_B); \\
EK = H_3(K);\ m = r_A, A; & & K = \hat{e}(r_AQ_A, d_B)^{r_B+h}; \\
[MK = H_4(K);\ \mathtt{MAC}_1 = & & EK = H_3(K);\ m' = r_C, ID_C; \\
\mathtt{MAC}_{MK}(A, B, r_AQ_A, r_BQ_B)]. & & [MK = H_4(K);\ \text{check } \mathtt{MAC}_1;] \\
 & & r_CH_1(ID_C) \overset{?}{=} r_AQ_A. \\
[\text{check } \mathtt{MAC}_2.] & \xleftarrow{[\mathtt{MAC}_2]} & [\mathtt{MAC}_2 = \\
 & & \mathtt{MAC}_{MK}(B, A, r_BQ_B, r_AQ_A).] \\
SK = H_5(A, B, r_AQ_A, r_BQ_B, K) & & SK = H_5(A, B, r_AQ_A, r_BQ_B, K) \\
[SK' = H_6(K)] & & [SK' = H_6(K)] \\
\hline
\end{array}
$$

Fig. 1. Protocol 1

3. The server B processes the incoming message as follows. 1) compute $h = H_2(r_AQ_A, r_BQ_B)$; 2) compute the agreed secret $K = \hat{e}(r_AQ_A, d_B)^{r_B+h}$; 3) generate a decryption key $EK = H_3(K)$; 4) decrypt $\{m\}_{EK}$ with EK to recover (r_C, ID_C); 5) check if the equation $r_CH_1(ID_C) = r_AQ_A$ holds. If the equation does not hold, B aborts the protocol.

4. On completion of the AK protocol, both parties compute the session key SK as specified in Fig. 1.
5. To provide key confirmation, we can extend the protocol using the standard MAC methodology by using a secure MAC scheme defined in Definition 3. In the second flow, the client A computes a MAC key $MK = H_4(K)$ and generates a tag of message $(A, B, r_A Q_A, r_B Q_B)$ using MK as the secret. The tag is sent to the server as the third component of the message. Once the server B receives the message, it proceeds as in step 3 of AK, but verifies the MAC tag before checking the equation $r_C H_1(ID_C) \overset{?}{=} r_A Q_A$ (this would slightly help the server against the deny-of-service attack). If both checks succeed, the server B sends the third message including a MAC tag generated as in Fig. 1 to the client A. The client A then has to verify the validity of the third message (the tag MAC_2). If the check succeeds, the session key is generated as SK' in Fig. 1.

In this work, we assume that at least one input of the used pairing should be checked in the specified group in general, otherwise the computation of the pairing should fail. Such check is important because the behavior of a pairing is not defined when an input is not from the specified group, and the output could be with much smaller order. Only one input needs to be checked in the pairing for the proposed protocol, because the other is either from the specified group or checked in the protocol explicitly.

For the specified protocol in Fig. 1, the second input should be checked in the specified group in the pairing ($r_A Q_A$ is checked in the protocol mandatorily). While we can simply exchange the inputs of the pairing, if the check on the first input is available. Such check may be trivial for some pairing instances, such as the Tate pairing, because the Tate pairing allows the second input to be any point on the curve.

4.2 Security Analysis

In the proposed protocol, the agreed secret K and EK are dynamically generated in every session, furthermore, there is no decryption oracle for any fixed EK to help the adversary in the running of the protocol. Hence, we only require the one-time symmetric-key encryption scheme to be passively secure as defined in Definition 2. While, for the simplicity of security analysis, we specifically require that $\mathcal{E}$ is the "one time pad" with pseudorandom bit stream scheme, which is an efficient and passively secure one-time encryption scheme [18].

In addition, there is another feature we should stress before the formal analysis. In the second flow of the protocol, we organise the plaintext m by concatenating random r_A and identifier A, because field r_A has fixed length. This could simplify the implementation. While, in the security model, we assume that the identifier A should be reviewed to the adversary, hence we require that when employing the encryption scheme (the "one-time pad" with pseudo-random bit stream) the message should be encrypted in the reverse order, i.e. the last bit of the plaintext should be encrypted first. As a result, we can directly make use of the next bit unpredictability property of a PRG [5].

Next, we firstly investigate the security of AK version of the proposed protocol.

As defined in Definition 4, for a chosen fresh oracle $\Pi^s_{i,j}$ in the game, party i can be corrupted (this is particularly used to address the known-key impersonation resilience property). Because the protocol differentiates the role of parties (some are clients and others are servers. This can be done by requiring that the client and the server identifers are in two separate sets, say (ID^C, ID^S)), we consider the security of the AK protocol in the following two cases:

(1) *Server authentication.* We consider the known-(client)-key attack that the adversary tries to impersonate a server to a client whose private key is known to the adversary, i.e. the chosen fresh oracle $\Pi^s_{i,j}$ is with $i \in ID^C$ and $j \in ID^S$.

Theorem 1 *The protocol is secure against the known-client-key attack, provided the BDH assumption is sound, the hash functions are modelled as random oracles.*

Proof is presented in Appendix.

(2) *Client authentication.* Now we consider the known-(server)-key attack that the adversary tries to impersonate a client to a server whose private key is known to the adversary, i.e. the chosen fresh oracle $\Pi^s_{i,j}$ is with $i \in ID^S$ and $j \in ID^C$.

Theorem 2 *The protocol is secure against the known-server-key attack, provided the GBDH assumption is sound, the hash functions are modelled as random oracles.*

Proof is presented in Appendix.

Theorem 1, 2 show that the AK protocol possesses the following security property: session key authentication, known session key security, key-compromise impersonation resilience and unknown key-share resilience. Based on these results and the fact that the BDH assumption implies the corresponding GBDH assumption, it is straightforward that the following theorem holds.

Theorem 3 *The protocol is a secure AK, provided the GBDH assumption is sound and the hash functions are modelled as random oracles.*

Now let's look at the the forward secrecy (note that a protocol satisfies Definition 5 does not necessary achieve the forward secrecy). The AK protocol in Fig. 1 achieves PFS.

Theorem 4 *The AK protocol has PFS regarding Definition 6, provided that the GBDH assumption is sound and the hash functions are modelled as random oracles.*

The proof is sketched in Appendix.

It is clear that, if the adversary knows the master key s then it can compute K, so that the scheme does not achieve master-key forward secrecy. However, this property may not be a defect since provides an possible way for legal interception which is also important in the mobile communications.

User identity privacy is achieved for the following reasons. (1) We note that r_A is randomly sampled from $\mathbb{Z}_q^*$, hence $r_A Q_A$ is evenly distributed in message space $\mathbb{G}_1^*$ which could be sampled by any other client with the same distribution. So, $r_A Q_A$ for client A is indistinguishable from $r_X Q_X$ for any other client X. (2) The client's identity is hidden by the symmetric-key encryption applied in the second flow. The used key is the output from a hash function on the agreed secret so that it is evenly distributed in the key space of the used encryption if we assume the hash function can be modeled as a random oracle. Hence, the ciphertext of one plaintext should be indistinguishable from those of other plaintexts because the used $\mathcal{E}$ possesses the indistinguishability security property.

The AKC version of the protocol employs the standard mechanism (MAC) to provide key confirmation. The security can be proved using the similar methods as in Theorem 1, 2.

4.3 A Strengthened Variant with Master-key Forward Secrecy

As analysed in the previous subsection, the protocol does not achieve master-key forward secrecy. There are two possible implications of this "weakness". (1) Anyone who records the execution of the protocol and then compromises the KGC's master key can recover any session key. (2) The "curious" KGC can recover any session key and detect the client identity by merely eavesdropping on a session of the protocol. Below we show a method to strengthen the protocol with reasonable cost to remove this weakness. The strengthened protocol (**Protocol 2**) proceeds as follows:

1. Identical with step 1 of Protocol 1
2. Apart from the operations of step 2 in Protocol 1, the client A also chooses another $r'_A \in \mathbb{Z}_q^*$ and computes $K' = r'_A r_B Q_B$ and $r'_A Q_B$. But this time, EK is computed as $EK = H'_3(K, K')$ where H'_3 is a hash function with $H'_3 : \mathbb{G}_2 \times \mathbb{G}_1 \to \{0,1\}^n$. A sends the message $(r_A Q_A, r'_A Q_B, \{m\}_{EK})$ to the server.
3. The server works as in Protocol 1, but also computes $K' = r_B r'_A Q_B$, and so EK is computed as the above step.
4. The session key of the AK protocol is computed by $SK = H'_5(A, B, r_A Q_A, r_B Q_B, K, K')$ where we use a hash function $H'_5 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \to \{0,1\}^w$.
5. The AKC protocol still uses the MAC mechanism to provide key confirmation. MK is computed as $MK = H'_4(K, K')$ where we employ a hash function $H'_4 : \mathbb{G}_2 \times \mathbb{G}_1 \to \{0,1\}^l$, and the extra component $r'_A Q_B$ is given as part of the authenticated message. The session key is generated as $SK' = H'_6(K, K')$ with a hash function $H'_6 : \mathbb{G}_2 \times \mathbb{G}_1 \to \{0,1\}^w$.

It is straightforward to show that the security proofs for Protocol 1 can be easily adapted for Protocol 2. While, as the formation of EK and SK (SK') uses $r'_A r_B Q_B$ as part of the contribution. Without r'_A or r_B, EK and SK (SK') cannot be recovered if the DH problem is hard in $\mathbb{G}_1$ which is implied by the BDH assumption. Hence, Protocol 2 has the master-key forward secrecy and hides the client identity to the KGC as well.

4.4 Evaluation of Security and Complexity

Here we summarise the security properties and computation complexity of the proposed protocol and some other schemes in the literature in Table 1 (we only count the heavy operations: Pairing, Scalar and Exponentiation). From the table, we can find that the proposed protocols have strong security with good computation efficiency. Moreover, the proposed protocol can work in both modes with and without user identity privacy.

	KnSK	FwS	UknKSh	KCI	IP	KContl	Comp Complexity
Proposal 1	✓	✓^{*1}	✓	✓	✓	A and B	1P+2S+1E
Proposal 2	✓	✓^{*2}	✓	✓	✓	A and B	1P+4S+1E^{*3}
Smart Protocol[34]	✓	✓^{*4}	✓	✓	χ	A and B	2P+3S
CK Protocol[15]	✓	✓^{*1}	✓	✓	χ	A and B	1P+2S

^{*1}: PFS is achieved.
^{*2}: The master-key forward secrecy is achieved.
^{*3}: The server has less computation complexity with 1P+3S+1E.
^{*4}: By using the key derivation function proposed in [15], the maser-key forward secrecy is achieved.

Table 1. Computation Complexity and Security Property

5 Conclusion

User identity privacy is an important security property for protocols used in several environments such as mobile networks. In this paper we investigate the user identity privacy in the protocols employing identity-based primitives from pairings. By considering both security strength and computation efficiency, we present an identity-based protocols using pairing which preserves both user identity privacy and high efficiency.

6 Acknowledgement

We would like to thank Steven Galbraith for helpful discussions on pairings and Kim-Kwang Raymond Choo for comments on the draft of the paper.

References

1. M. Abadi. Private Authentication. *Privacy Enhancing Technologies 2002*, LNCS 2482, pp. 27-40.
2. X. Boyen. Multipurpose Identity-Based Signcryption: A Swiss Army Knife for Identity-Based Cryptography. *Advances in Cryptology - Crypto 2003*, LNCS 2729, Springer-Verlag (2003), pp. 382-398.
3. M. Bellare, R. Canetti and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology - Crypto 96*, LNCS 1109, 1996.
4. D. Boneh and M. Franklin. Identity Based Encryption from The Weil Pairing. *Advances in Cryptology - Crypto 2001*, LNCS 2139, 2001.
5. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Computing*, 13(4):850-863, November 1984.
6. C. Boyd and A. Mathuria. Key Establishment Protocols for Secure Mobile Communications: A Selective Survey. *ACISP'98*, LNCS 1438, pp. 344-355.
7. D. Balfanz, G. Durfee, N. Shankar, D.K. Smetters, J. Staddon and H.C. Wong. Secret Handshakes from Pairing-Based Key Agreements. *IEEE Symposium on Security and Privacy (Proceedings)*, pp. 180-196, 2003.
8. C. Boyd, W. Mao and K. Paterson. Key agreement using statically keyed authenticators. *Applied Cryptography and Network Security – ACNS'2004*, LNCS 3089, Springer-Verlag (2004).
9. M. Bellare, D. Pointcheval and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Proceedings of Advances in Cryptology - Eurocrypt 2000*, LNCS 1807, Springer-Verlag, 2000.
10. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proceedings of Advances in Cryptology - Crypto '93*, LNCS 773, pp. 232-249, Springer-Verlag, 1993.
11. S. Blake-Wilson, D. Johnson and A. Menezes. Key agreement protocols and their security analysis. In *Proceedings of the Sixth IMA International Conference on Cryptography and Coding*, Springer-Verlag, 1997.
12. Z. Cheng and R. Comley. Efficient Certificateless Public Key Encryption. Cryptology ePrint Archive, Report 2005/012.
13. Z. Cheng and L. Chen. On Security Proof of McCullagh-Barreto's Key Agreement Protocol and its Variants. Cryptology ePrint Archive, Report 2005/201.
14. R. Canetti and H. Krawczyk. Security Analysis of IKE's Signature-based Key-Exchange Protocol. *Advances in Cryptology - Crypto 2002.*
15. L. Chen and C. Kudla. Identity Based Authenticated Key Agreement from Pairings. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop*, pp. 219-233, June 2003. See also Cryptology ePrint Archive, Report 2002/184.
16. L. Chen and J. Malone-Lee. Improved Identity-Based Signcryption. In *Proceedings of Public Key Cryptography - PKC 2005.* LNCS 3386, pp. 362-379. See also Cryptology ePrint Archive, Report 2004/114.
17. Z. Cheng, M. Nistazakis, R. Comley and L. Vasiu. On The Indistinguishability-Based Security Model of Key Agreement Protocols-Simple Cases. In *Proc. of ACNS 2004*. Full version avaible on Cryptology ePrint Archive, Report 2005/129.
18. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33, 167C226, 2003.
19. R. Dupont and A. Enge. Practical Non-Interactive Key Distribution Based on Pairings. In *Proceedings of the International Workshop on Coding and Cryptography (WCC)*, Versailles, 2003. See also Cryptology ePrint Archive, Report 2002/136.

20. W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22 (6), pp. 644-654,1976
21. D. Harkins and D. Carrel. The Internet Key Exchange Protocol (IKE). IETF RFC 2409, Nov. 1998.
22. G. Horn, K. Martin and C. Mitchell. Authentication Protocols for Mobile Network Environment Value-added Serivices. *IEEE Transactions on Vehicular Technology*, 51(2):383-392, 2002.
23. M. Jakobsson and D. Pointcheval. Mutal authentication for low power mobile devices. *Financial Cryptography*, LNCS 2339, pp. 178-195, 2001.
24. V. Miller. Short Programs for Functions on Curves. unpublished manuscript, 1986.
25. L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone. An Efficient Protocol for Authenticated Key Agreement. *Designs, Codes and Cryptography*, 28, pp. 119-134, 2003.
26. N. McCullagh and P.S.L.M. Barreto. A New Two-Party Identity-Based Authenticated Key Agreement. *CT-RSA 2005*, LNCS 3376. See also Cryptology ePrint Archive, Report 2004/122.
27. A. Menezes, P. van Oorschot and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.
28. L. Nguyen. Accumulators from bilinear pairings and applications. *CT-RSA 2005*, LNCS 3376, pp. 275-292.
29. J. Katz and M. Yung. Characterization of Security Notions for Probabilistic Private-Key Encryption. To appear in *Journal of Cryptology.*
30. T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In *Proceedings of Public Key Cryptography - PKC 2001*, LNCS 1992, PP. 104-118, Springer-Verlag, 2001.
31. M. Scott. Authenticated ID-based Key Exchange and remote log-in with insecure token and PIN number. Cryptology ePrint Archive, Report 2002/164.
32. A. Shamir. Identity-Based Cryptosystems and Signature Schemes. *Advances in Cryptology-Crypto '84*, LNCS 196, 1984.
33. J. Silverman. The Arithmetic of Elliptic Curve. Springer-Verlag, 1986.
34. N.P. Smart. An Identity Based Authenticated Key Agreement Protocol Based on the Weil Pairing. *Electronics Letters 38*, pp. 630-632, 2002.
35. R. Sakai, K. Ohgishi and M. Kasahara. Cryptosystems Based on Pairing. The *2000 Symposium on Cryptography and Information Security*, Okinawa, Japan, 2000.
36. Y. Wang. Efficient Identity-Based and Authenticated Key Agreement Protocol. Cryptology ePrint Archive, Report 2005/108.
37. D. Wong and A. Chan. Efficient and mutally authenticated key exchange for low power computing devices. *Advances in Cryptology - Asiacrypt 2001*, LNCS 2248, pp. 272-289, 2001.
38. G. Xie. An ID-Based Key Agreement Scheme from pairing. Cryptology ePrint Archive, Report 2005/093.
39. F. Zhang and X. Chen. Cryptanalysis and improvement of an ID-based ad-hoc anonymous identification scheme at CT-RSA 05. Cryptology ePrint Archive, Report 2005/103.

Appendix

Before we prove the theorems, we describe a game which eases the proving.

Interactive game with a BDH challenger. An adversary A with a pair of probabilistic polynomial-time (PPT) algorithms $(A_1(r_1;\ \cdots), A_2(r_2;\ \cdots))$ where r_i is used by A_i as the random tape, engages with a challenger in the following game:

```
Interactive BDH game
$(P, aP, bP, cP, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, q) \leftarrow \mathcal{G}(1^k)$;
$(X, \sigma) \leftarrow A_1(r_1;\ P, aP, bP, cP, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, q)$;
$h \leftarrow \mathbb{Z}_q^*$;
$K \leftarrow A_2(r_2;\ h, \sigma)$.
```

where $\mathcal{G}$ randomly chooses $a, b, c \in \mathbb{Z}_q^*, P \in \mathbb{G}_1^*$; σ is the state information passed from A_1 to A_2. We say that A wins the game if it computes $K = \hat{e}(aP, X+hbP)^c$. Define the advantage of the adversary as the function of k as

$$\mathrm{Adv}_A(k) = \Pr[\text{A wins}].$$

Theorem 5 *If the BDH assumption is sound, any adversary participating the interactive BDH game can only have negligible advantage.*

Proof: Given a BDH instance $(P, aP, bP, cP, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, q)$, we construct an algorithm in the following way:

```
    $(X, \sigma) \leftarrow A_1(r_1;\ P, aP, bP, cP, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, q)$;
    $h \leftarrow \mathbb{Z}_q^*$;
    $K_1 \leftarrow A_2(r_2;\ h, \sigma)$;
$\rhd_1$
    $K_2 \leftarrow A_2(r_2;\ h-1, \sigma)$;
$\rhd_2$
    output $K_1/K_2$.
```

Claim 1 *For any adversary with PPT algorithms (A_1, A_2) with advantage $\epsilon(k)$ to win the interactive BDH game, the above algorithm outputs $\hat{e}(P,P)^{abc}$ with probability $\epsilon^2(k)$.*

Proof: Obviously, at point $\rhd_1$, the algorithm computes correct $K_1 = \hat{e}(aP, X + hbP)^c$ with probability $\epsilon(k)$. Because h is chosen independently from X and σ, two execution of A_2 should have the same probability to output the correct answer. Hence at point $\rhd_2$, the event that $K_1 = \hat{e}(aP, X + hbP)^c$ and at the same time $K_2 = \hat{e}(aP, X + (h-1)bP)^c$ happens with probability $\epsilon^2(k)$. The claim follows.

The theorem follows from the claim. □

Proof of Theorem 1

Proof: We define the session ID as the concatenation of $r_B Q_B \| r_A Q_A$. Obviously this session ID can uniquely identify a session between party A and B because

r_i in the protocol should be a random integer varying on each session. The first two conditions are trivial to prove. Now we prove that the protocol meets the third condition.

We prove the theorem by constructing an algorithm A using the adversary B against the protocol as a subroutine to win the interactive BDH game with non-negligible advantage. The game proceeds in the following way. After $\mathcal{G}$ outputs the challenge $(P, sP, bP, aP, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, q)$, A simulates the system setup to adversary B as follow. The system parameters are set as $(P, sP, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, q, H_1, H_2, H_3, H_5, \mathcal{E})$ where $P_{pub} = sP$ (i.e. the master key is s which A does not known); H_1, H_2, H_3, H_5 are random oracles controlled by A.

Assume that adversary B queries q_1 *server* identifiers on H_1 and creates q_C *client* oracles. Recall that in the protocol, it is required that the client and server identifers are in two separate sets (ID^C, ID^S), which is the way to differentiate the roles of parties. In this proof, we use two types of **Send** query to reflect the behavior of the protocol.

- **Send**^{S}**(**$\Pi^t_{i,*}$**, M)**: This query is sending a message to a server oracle (the t-th instance of server party i). Message M is in the form of $(X, \{m\}_{EK})$ or λ. If $M = \lambda$, a server oracle will be created. We use $*$ to represent the sender because the receiver of M does not know the sender before it decrypts the message in the real world. In the simulation, A cannot make use of knowledge of its partner's identifier before it decrypts the incoming message.
- **Send**^{C}**(**$\Pi^t_{i,*}$**, M)**: This query is sending a message to a client oracle. Message M is in the form of (j, X). This query will trigger the creation of a client oracle. While in this proof, we slightly abuse the notation. We use $\Pi^t_{i,*}$ to denote the t-th client oracle among all the client oracles created in the game, instead of the t-th instance of party i. This abuse will not affect the strength of the model, because originally t is only used to uniquely identify an instance of a party.

Algorithm A randomly chooses $1 \leq I \leq q_1$ and $1 \leq J \leq q_C$ and starts simulating the real world where the adversary B launches the attack.

- $H_1(ID_i)$: A maintains an initially empty list H_1^{list} with entries of the form (ID_i, Q_i, x_i). A responds to the query in the following way.
  - If ID_i already appears on H_1^{list} in a tuple (ID_i, Q_i, x_i), then A responds with $H_1(ID_i) = Q_i$.
  - If ID_i is the I-th unique *server* identifier query, then A inserts $(ID_i, bP, \perp)$ into the list and returns bP.
  - Otherwise, A randomly chooses $x \in \mathbb{Z}_q^*$, inserts (ID_i, xP, x) into the list and returns xP.
- $H_2(X_i, Y_i)$: A maintains an initially empty list H_2^{list} with entries of the form (X_i, Y_i, z_i) indexed by (X_i, Y_i). A responds to the query in the following way.
  - If a tuple indexed by (X_i, Y_i) is on the list, then A responds with z_i.
  - Otherwise, A randomly chooses $z \in \mathbb{Z}_q^*$, inserts (X_i, Y_i, z) into the list and returns z.

- $H_3(K_i)$: A maintains an initially empty list H_3^{list} with entries of the form (K_i, k_i) indexed by K_i. A responds to the query in the following way.
  - If a tuple indexed by K_i is on the list, then A responds with k_i.
  - Otherwise, A randomly chooses $k \in \{0,1\}^n$, inserts (K_i, k) into the list and returns k.
- $H_5(ID_i, ID_j, Q_i, Q_j, K_i)$: A maintains an initially empty list H_5^{list} with entries of the form $(ID_i, ID_j, Q_i, Q_j, K_i, \ell_i)$ indexed by $(ID_i, ID_j, Q_i, Q_j, K_i)$. A responds to the query in the following way.
  - If a tuple indexed by $(ID_i, ID_j, Q_i, Q_j, K_i)$ is on the list, then A responds with ℓ_i.
  - Otherwise, A randomly chooses $\ell \in \{0,1\}^w$, inserts $(ID_i, ID_j, Q_i, Q_j, K_i, \ell)$ into the list and returns ℓ.
- **Corrupt**(ID_i): A looks through list H_1^{list}. If ID_i is not on the list, A queries $H_1(ID_i)$. A checks the value of x_i: if $x_i \neq \perp$, then A responds with $x_i sP$; otherwise, A aborts the game (**Event 1**).
- **Send**^{S}**(**$\Pi_{i,*}^t$**, M)**: A maintains a list Ω for every (client or server) oracle of the form $(\Pi_{i,j}^t, r_{i,j}^t, tran_{i,j}^t, K_{i,j}^t, SK_{i,j}^t)$ where $r_{i,j}^t$ is the random string used to generate message; $tran_{i,j}^t$ is the transcript of the oracle so far, and $K_{i,j}^t$ and $SK_{i,j}^t$ are set $\perp$ initially. A proceeds in the following way:
  - If $M = \lambda$, A randomly chooses $r \in \mathbb{Z}_q^*$ and responds with (i, rQ_i) where Q_i is found from H_1^{list} with identifier i (i.e. $r_{i,*}^t = r$).
  - Otherwise ($M = (X, \{m\}_{EK})$), A proceeds in the following way:
    * If x_i on H_1^{list} corresponding to i is not $\perp$, then compute $K = \hat{e}(X, x_i sP)^{r_{i,*}^t + z}$ where $z = H_2(X, r_{i,*}^t Q_i)$. Use $EK = H_3(K)$ to decrypt $\{m\}_{EK}$ to recover (r_C, ID_C). If $X = r_C H_1(ID_C)$, then set the partner of the oracle as ID_C, $K_{i,ID_C}^t = K$, and $SK_{i,ID_C}^t = H_5(ID_C, i, X, r_{i,ID_C}^t Q_i, K_{i,ID_C}^t)$. Otherwise, continue.
    * ($x_i = \perp$) Try every $k_\ell = H_3(K_\ell)$ on H_3^{list} as EK to decrypt $\{m\}_{EK}$ to recover (r_C, ID_C) and test $X = r_C H_1(ID_C)$. If the equation holds, store (r_C, ID_C) into an initially empty list $\mathcal{L}$.
    * If $\mathcal{L}$ is empty, reject the message. (**Rejection 1**)
    * Otherwise, for any tuple (r_C, ID_C) in $\mathcal{L}$, compute $K = \hat{e}(x_j sP, r_{i,*}^t Q_i + zQ_i)^{r_C} = \hat{e}(X, sbP)^{r_{i,*}^t + z}$, where x_j is from H_1^{list} for identifier ID_C, and $z = H_2(X, r_{i,*}^t Q_i)$. Note that there is only one K for the given $X, r_{i,*}^t$ and all the pairs (r_C, ID_C) with $r_C H(ID_C) = X$.
    * Use $EK = H_3(K)$ to decrypt $\{m\}_{EK}$ again to recover (r^*, ID^*) and test if equation $r^* H_1(ID^*) = X$ holds.
    * If the equation holds, set the partner of the oracle as ID^*, $K_{i,ID^*}^t = K$, and $SK_{i,ID^*}^t = H_5(ID^*, i, X, r_{i,ID^*}^t Q_i, K_{i,ID^*}^t)$.
    * Otherwise, reject the message. (**Rejection 2**)
- **Send**^{C}**(**$\Pi_{i,*}^t$**, M)**: (Message M is in the form of (j, X)). A proceeds in the following way:
  - Set the partner of the oracle as j.
  - If $t = J$ and $x_j \neq \perp$ which is found from H_1^{list} with identifier j, A aborts the game (**Event 2**).

- Otherwise, if $t = J$, A randomly chooses $u \in \mathbb{Z}_q^*$ and checks if (uaP, X) has been queried on H_2, until one such pair is not found on H_2^{list}. Such u can always be found, because A is a PPT algorithm of k. All the instructions executed before this point belong to the algorithm A_1 of A. A dumps the content on all the maintained lists and system parameters to the tape σ, and outputs (X, σ). The interactive BDH challenger returns $h \in_R \mathbb{Z}_q^*$. After this point, all the instructions belong to algorithm A_2 of A. A reconstructs all the lists and system setup from σ and immediately inserts (uaP, X, h) into H_2^{list}. Now A continues to respond to B's queries. A randomly chooses $EK \in \{0,1\}^n, r \in \mathbb{Z}_q^*$ and generates $\{r, i\}_{EK}$, and then responds with $(uaP, \{r, i\}_{EK})$. If B rejects the message (**Event 3**), A randomly chooses K_ℓ from list H_3^{list} and returns $K_\ell^{1/u}$ to the interactive BDH challenger.
- Otherwise, A randomly chooses $r \in \mathbb{Z}_q^*$ and computes $K_{i,j}^t = \hat{e}(x_i sP, X + zQ_j)^r$, where x_i and Q_j are found from H_1^{list} with identifier i and j; $z = H_2(rQ_i, X)$. A computes $SK_{i,j}^t = H_5(i, j, rQ_i, X, K_{i,j}^t)$ and responds with $(rQ_i, \{r, i\}_{EK})$ where $EK = H_3(K_{i,j}^t)$.

– **Reveal**($\Pi_{i,*}^t$): The oracle must have accepted and so knows its partner j. Otherwise $\perp$ should be returned. If $i \in ID^C$ and $t = J$, or $i \in ID^S$ but $\Pi_{i,j}^t$ has a matching conversation with $\Pi_{u,v}^J$ with $u \in ID^C$, A aborts the game (**Event 4**). Otherwise, A returns $SK_{i,j}^t$.

– **Test**($\Pi_{i,*}^t$): The oracle should be *fresh*, so must have accepted and knows its partner j. If $i \notin ID^C$ or $t \neq J$, A aborts the game (**Event 5**). Otherwise, A randomly chooses a number $\zeta \in \{0,1\}^w$ and gives it to B as the response. After B responds, A randomly chooses a tuple from H_5^{list} with the value K_ℓ. A computes $K_\ell^{1/u}$ and returns it to the interactive BDH challenger. Note that if the game does not abort, for the challenge oracle $K_{i,j}^t = \hat{e}(x_i sP, X + hbP)^{ua/x_i}$.

Let **Event 6** be that, in the attack, adversary B indeed chose to impersonate a server, whose identifier was queried on H_1 as the I-th distinct server identifier query, to the J-th client oracle. Then following the rules of the BR game, it's clear that **Event 1, 2, 4, 5** would not happen, and so the game would not abort before A can answer the interactive BDH challenge.

Claim 2 *The Reveal query does not help the adversary to win the game.*

Proof: Suppose the chosen fresh oracle $\Pi_{i,j}^J$ with $i \in ID^C$ had the session ID $r_B Q_j \| r_A Q_i$ when **Event 6** happened. Then $SK_{i,j}^J = H_5(i, j, r_A Q_i, r_B Q_j, K_{i,j}^J)$. Because H_5 is a random oracle, the adversary had to at least query the session key of oracle $\Pi_{i,j}^v$ or $\Pi_{j,i}^u$ to find the possible same session key of $\Pi_{i,j}^J$. However, because of the way of defining session ID, the queried oracle $\Pi_{j,i}^u$ could not have $r_B Q_j$ and $r_A Q_i$ in the exchanged message. Hence this query does not help the adversary. To reveal the session key of $\Pi_{i,j}^v$ with $v \neq J$ does not help either, because the oracle would generate $r_A Q_i$ in the second message as oracle $\Pi_{i,j}^J$ with only negligible probability. The claim follows.

Claim 3 **Event 3** *only happened with negligible probability if* $H_3(\hat{e}(sP, X + zbP)^{ua})$ *where* $z = H_2(uaP, X)$ *was not queried when* **Event 6** *happened.*

Proof: When **Event 6** happened, B could not know the decryption key EK if $H_3(\hat{e}(sP, X + zbP)^{ua})$ was not queried, because H_3 is a random oracle. On the other hand, $\mathcal{E}$ has the indistinguishability security. Hence, B who simulated the server oracle, could only with negligible probability differentiate the corrected ciphertext from the faked one in the simulation without knowing EK.

Claim 4 *If* EK *is not known by the adversary, then a message* $(X, \{m\}_{EK})$ *will be accepted by the server in the real world with only negligible probability.*

Proof: The message will be accepted by the server, only if the recovered message (r_C, ID_C) from $\{m\}_{EK}$ meets the equation $X = r_C H_1(ID_C)$. Define a relation $R = \{(r, ID, X) \mid rH_1(ID) = X, r \in \mathbb{Z}_q^*, H_1 \text{ is a hash function}, H_1(ID) \in \mathbb{G}_1^*\}$ If EK is not known by the adversary B, the situation can be described by the following game.

$(X, c) \leftarrow B^{H_1}(1^k, R)$;
$EK \leftarrow \mathcal{K}$;
$(r_C, ID_C) = \mathbb{D}(EK, c)$;
If $(r_C, ID_C, X) \in R$, accept, else reject.

where $\mathcal{E}$ is the used encryption scheme.

If $H_1(ID_C)$ has not been queried, then the equation holds with negligible probability because H_1 is modelled as a random oracle. Suppose, in the attack, q_C distinct client identifiers were queried on H_1. Then for a chosen X there are at most q_C pairs could possibly meet the equation (i.e. $\|R\| = q_C$ in the real attack). Suppose the length of message $m = (r, ID)$ (resp. EK) is f (resp. n). Because the used $\mathcal{E}$ is the "one-time pad" with pseudorandom bit stream scheme, we can assume that the decryption of a given ciphertext c under a random decryption key should have an even distribution among 2^n possibilities randomly sampled from 2^f messages. Then the probability that for a given ciphertext c generated before EK is randomly chosen, the decrypted message $\mathbb{D}(EK, c)$ is one of q_C pairs (i.e. $(\mathbb{D}(EK, c), X) \in R$), is $\frac{q_C}{2^n}$. Note that in the game both q_C and n are polynomials of security parameter k. Hence the probability is negligible.

When the protocol proceeds without identity privacy, i.e. identifier A is sent in the second flow with plaintext, instead of being encrypted with EK, the claim follows from a similar argument as above, but the accept probability is $1/2^n$.

Claim 5 *If* A *did not terminate the game,* B *cannot notice any inconsistence between the simulation and the real world.*

Proof: It is clear that if A did not terminate the game, then the responses to queries including H_1, H_2, H_3, H_5, Corrupt, Send^C, Reveal and Test, are consistent with the one in the real world. Now let's take a close look at the response to query **Send**^{S}**(**$\mathit{\Pi}_{i,*}^t$**, M)**. Except for the two rejections, the response from other part of A's behavior honestly follows the protocol. **Rejection 1** only happens when B did not query $EK = H_3(\hat{e}(X, sbP)^{r+z})$. Hence B would not know EK, because H_3 is a random oracle. From Claim 4, in the real world, the server accepts the message with only negligible probability. **Rejection 2** happens if $\{m\}_{EK}$ is not valid. This could happen in the attack by a naughty adversary which generates $X = r_C H(ID_C)$, but uses $EK = H_3(T)$ for some other $T \neq K$ to encrypt $m = \{r_C, ID_C\}$. If the server accepts the message, it will be quite assured that it is in a simulation, instead of being in the real world.

Let $\mathcal{H}$ be the event that $\hat{e}(sP, X + hbP)^{ua}$ has been queried to H_5 or H_3. Let $\mathcal{F}$ be the event that A did not abort the game. Let $\mathcal{W}$ be the event that A finds the correct $\hat{e}(sP, X + hbP)^{ua}$ on the list H_5^{list} or H_3^{list}. Suppose H_5 (resp. H_3) has been queried for q_5 (resp. q_3) times and B has the advantage $\epsilon(k)$ against the protocol. We have

$$\Pr[A \text{ wins}] = \Pr[\mathcal{F} \wedge \mathcal{H} \wedge \mathcal{W}] = \Pr[\text{Event } 6 \wedge \mathcal{H} \wedge \mathcal{W}] \geq \frac{1}{q_1 \cdot q_C} \cdot \epsilon(k) \cdot \frac{1}{\max\{q_3, q_5\}}.$$

Combine Theorem 5, the theorem follows. □

Proof of Theorem 2

Proof: Define session ID as in the proof of Theorem 1.

Given a GBDH problem instance $(P, sP, aP, bP, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, q)$ and a DBDH oracle $\mathcal{O}_{DBDH}$ which given (xP, yP, zP, T) returns 1 if $\hat{e}(P, P)^{xyz} = T$; otherwise returns 0, we construct an algorithm A using the adversary B against the protocol to solve the GBDH problem.

A simulates the system setup to adversary B as follow. The system parameters are set as $(P, sP, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, q, H_1, H_2, H_3, H_5, \mathcal{E})$ where $P_{pub} = sP$ (i.e. the master key is s which A does not known); H_1, H_2, H_3, H_5 are random oracles controlled by A.

As in Theorem 1, we use two types of **Send** query. While, this time we abuse the notation of $\Pi^t_{i,*}$ used in $\mathbf{Send}^S$. We use $\Pi^t_{i,*}$ to denote the t-th server oracle among all the server oracles created in the game.

Assume that adversary B queries q_1 *client* identifiers on H_1 and creates q_S *server* oracles. Algorithm A randomly chooses $1 \leq I \leq q_1$ and $1 \leq J \leq q_S$ and starts simulating the real world where the adversary B launches the attack.

- $H_1(ID_i)$: A maintains an initially empty list H_1^{list} with entries of the form (ID_i, Q_i, x_i). A responds to the query in the following way.
  - If ID_i already appears on H_1^{list} in a tuple (ID_i, Q_i, x_i), then A responds with $H_1(ID_i) = Q_i$.
  - If ID_i is the I-th unique *client* identifier query, then A inserts $(ID_i, aP, \perp)$ into the list and returns aP.
  - Otherwise, A randomly chooses $x \in \mathbb{Z}_q^*$, inserts (ID_i, xP, x) into the list and returns xP.

- $H_2(X_i, Y_i)$: A maintains an initially empty list H_2^{list} with entries of the form (X_i, Y_i, z_i) indexed by (X_i, Y_i). A responds to the query in the following way.
  - If a tuple indexed by (X_i, Y_i) is on the list, then A responds with z_i.
  - Otherwise, A randomly chooses $z \in \mathbb{Z}_q^*$, inserts (X_i, Y_i, z) into the list and returns z.
- $H_3(K_\ell)$: A maintains an initially empty list H_3^{list} with entries of the form (K_i, k_i) indexed by K_i. A responds to the query in the following way.
  - If a tuple indexed by K_ℓ is on the list, then A responds with the corresponding k_i.
  - Otherwise, for every tuple $(\Pi_{i,j}^t, u_w, z_w, x_w, X_w, k_w)$ on list $\mathcal{L}$ which is maintained in the **Send**^{C} routine,
    * Compute $K = \frac{K_\ell^{1/u_w}}{\hat{e}(aP, z_w x_w sP)}$.
    * Access oracle $\mathcal{O}_{DBDH}(sP, aP, X_w, K)$.
    * If $\mathcal{O}_{DBDH}$ returns 1, A removes the entry from $\mathcal{L}$ and sets $K_{i,j}^t = K_\ell$ and $SK_{i,j}^t = H_5(i, j, u_w aP,\ X_w, K_{i,j}^t)$. Then A inserts (K_ℓ, k_w) into list H_3^{list} and responds with k_w.
    * Otherwise, continue.
  - Otherwise, A randomly chooses $k \in \{0,1\}^n$, inserts (K_ℓ, k) into the list and returns k.
- $H_5(ID_1, ID_2, Q_1, Q_2, K_v)$: A maintains an initially empty list H_5^{list} with entries of the form $(ID_i, ID_j, Q_i, Q_j, K_i, \ell_i)$ indexed by $(ID_i, ID_j, Q_i, Q_j, K_i)$. A responds to the query in the following way.
  - If a tuple indexed by $(ID_1, ID_2, Q_1, Q_2, K_v)$ is on the list, then A responds with the corresponding ℓ_i.
  - Otherwise, for every tuple $(\Pi_{i,j}^t, u_w, z_w, x_w, X_w, k_w)$ on list $\mathcal{L}$,
    * Compute $K = \frac{K_v^{1/u_w}}{\hat{e}(aP, z_w x_w sP)}$.
    * Access oracle $\mathcal{O}_{DBDH}(sP, aP, X_w, K)$.
    * If $\mathcal{O}_{DBDH}$ returns 1, A removes the entry from $\mathcal{L}$ and sets $K_{i,j}^t = K_v$ and $SK_{i,j}^t = H_5(i, j, u_w aP, X_w, K_{i,j}^t)$. A inserts (K_v, k_w) into list H_3^{list}. If $ID_1 = i, ID_2 = j, Q_1 = u_w aP$, and $Q_2 = X_w$, then A responds with $SK_{i,j}^t$.
    * Otherwise, continue.
  - Otherwise, A randomly chooses $\ell \in \{0,1\}^w$ and inserts $(ID_1, ID_2, Q_1, Q_2, K_v, \ell)$ into the list and returns ℓ.
- **Corrupt**(ID_i): A looks through list H_1^{list}. If ID_i is not on the list, A queries $H_1(ID_i)$. A checks the value of x_i: if $x_i \neq \perp$, then A responds with $x_i sP$; otherwise, A aborts the game (**Event 1**).
- **Send**^{S}**(**$\Pi_{i,*}^t$**, M)**: A maintains a list Ω for every (client or server) oracle of the form $(\Pi_{i,j}^t, tran_{i,j}^t, r_{i,j}^t,\ K_{i,j}^t, SK_{i,j}^t)$ where $tran_{i,j}^t$ is the transcript of the oracle so far; $r_{i,j}^t$ is the random integer used by the oracle to generate message, and $K_{i,j}^t$ and $SK_{i,j}^t$ are set $\perp$ initially. A proceeds in the following way:
  - If $t = J$,
    * If $M = \lambda$, A responds with (i, bP) and set $r_{i,*}^t = \perp$.

    * Otherwise (i.e. $M = (X, \{m\}_{EK})$), A tries every k_w on H_3^{list} as EK to decrypt $\{m\}_{EK}$ to recover (r_C, ID_C) and tests $X = r_C H_1(ID_C)$. If no pair meets the equation, or if for all the recovered messages (r_C, ID_C) decrypted using some k_w's, x on H_1^{list} corresponding to ID_C is not $\perp$, A aborts the game (**Event 2**). Otherwise (i.e. there is at least a $k_w = H_3(K_w)$ decrypting a valid ID_C whose corresponding x on H_1^{list} is $\perp$), A proceeds in the following way. For all the valid K_w,
        · Compute $K = (\frac{K_w}{\hat{e}(r_C aP, x_i sP)^z})^{1/r_C}$.
        · Access oracle $\mathcal{O}_{DBDH}(sP, aP, bP, K)$.
        · If $\mathcal{O}_{DBDH}$ returns 1, return K as the answer to the GBDH problem. Note that $K_{i,ID_C}^t = \hat{e}(r_C aP, x_i sP)^{r_{i,ID_C}^t + z}$ with $r_{i,ID_C}^t = b/x_i$.

      If for all K_w, the test fails, A aborts the game (**Event 3**).
  - Otherwise,
    * If $M = \lambda$, A randomly chooses $r_{i,*}^t \in \mathbb{Z}_q^*$ and responds with $(i, r_{i,*}^t Q_i)$ where Q_i is found from H_1^{list} with identifier i;
    * Otherwise ($M = (X, \{m\}_{EK})$), A computes $K = \hat{e}(X, x_i sP)^{r_{i,*}^t + z}$, where x_i is from H_1^{list} for identifier i; $r_{i,*}^t$ is from list Ω for oracle $\Pi_{i,*}^t$ and $z = H_2(X, r_{i,*}^t Q_i)$. A decrypts $\{m\}_{EK}$ to recover (r_C, ID_C) using $EK = H_3(K)$ and tests $X = r_C H_1(ID_C)$. If the equation does not hold, A rejects the message; otherwise, A sets the partner identifier as ID_C, $K_{i,ID_C}^t = K$ and computes $SK_{i,ID_C}^t = H_5(ID_C, i, X, r_{i,ID_C}^t Q_i, K_{i,ID_C}^t)$.

– **Send**^{C}**(**$\Pi_{i,*}^t$**, M)**: (Message M is in the form of (j, X)). A maintains an initially empty list $\mathcal{L}$ with entries of the form $(\Pi^w, u_w, z_w, x_w, X_w, k_w)$. A proceeds in the following way:
  - Set the partner of the oracle as j.
  - If $x_i = \perp$ from H_1^{list} with identifier i, A proceeds in the following way.
    * Randomly chooses $u \in \mathbb{Z}_q^*$ and computes $z = H_2(uaP, X)$.
    * Find x_j from H_1^{list} with identifier j.
    * For every (K_ℓ, k_ℓ) on H_3^{list},
        · Compute $K = \frac{K_\ell^{1/u}}{\hat{e}(aP, zx_j sP)}$.
        · Access oracle $\mathcal{O}_{DBDH}(sP, aP, X, K)$.
        · If $\mathcal{O}_{DBDH}$ returns 1, A sets $K_{i,j}^t = K_\ell$ and $SK_{i,j}^t = H_5(i, j, uaP, X, K_{i,j}^t)$. Then A uses k_ℓ as EK to encrypt $\{u, i\}_{EK}$ and responds with $(uaP, \{u, i\}_{EK})$.
        · Otherwise, continue.
    * For every $(\cdots, K_v, \ell_v)$ on H_5^{list},
        · Compute $K = \frac{K_v^{1/u}}{\hat{e}(aP, zx_j sP)}$.
        · Access oracle $\mathcal{O}_{DBDH}(sP, aP, X, K)$.
        · If $\mathcal{O}_{DBDH}$ returns 1, A sets $K_{i,j}^t = K_v$. Then A uses $k = H_3(K_{i,j}^t)$ as EK to encrypt $\{u, i\}_{EK}$ and responds with $(uaP, \{u, i\}_{EK})$.
        · Otherwise, continue.

    - $*$ It is highly unlikely that the above two searches can find proper response, because $K^t_{i,j}$ is depending on z and u which are just generated randomly. So, the searches above can be omitted. Instead, A directly randomly chooses $k \in \mathbb{Z}^*_q$ and stores the tuple $(\Pi^t_{i,j}, u, z, x, X, k)$ into list $\mathcal{L}$. A uses k as EK to encrypt $\{u, i\}_{EK}$ and responds with $(uaP, \{u, i\}_{EK})$.
  - Otherwise, A randomly chooses $r \in \mathbb{Z}^*_q$ and computes $K^t_{i,j} = \hat{e}(x_i sP, X + zQ_j)^r$, where x_i and Q_j are found from H_1^{list} with identifier i and j; $z = H_2(rQ_i, X)$. A computes $SK^t_{i,j} = H_5(i, j, rQ_i, X, K^t_{i,j})$ and responds with $(rQ_i, \{r, i\}_{EK})$ where $EK = H_3(K^t_{i,j})$.
- **Reveal**$(\Pi^t_{i,*})$: The oracle must have accepted and so knows its partner j. Otherwise $\perp$ should be returned. If $i \in ID^S$ and $t = J$, or $i \in ID^C$ but has a matching conversation with $\Pi^J_{u,v}$ with $u \in ID^S$, A aborts the game (**Event 4**). Otherwise, A returns $SK^t_{i,j}$.
- **Test**$(\Pi^t_{i,*})$: The oracle should be *fresh*, so must have accepted and knows its partner j. If $i \notin ID^S$ or $t \neq J$, A aborts the game (**Event 5**). If B indeed chose the J-th server oracle as the challenge, then the game stopped at the **Send**^{S} routine.

Claim 6 *If A did not abort the game, B could not find inconsistence between the simulation and the real world.*

Proof: The response to queries are indistinguishable from the one in the real world. In particular, to respond to queries on H_3 and H_5, A significantly uses the access to $\mathcal{O}_{DBDH}$ and the programmability of a random oracle to make sure that the response is consistent with the one in Send^C.

Let **Event 6** be that, in the attack, adversary B indeed chose to impersonate a client, whose identifier was queried on H_1 as the I-th distinct client identifier query, to the J-th server oracle. Then following the rules of the BR game, it's clear that **Event 1, 4, 5** would not happen.

Claim 7 **Event 2** *happened with negligible probability if* **Event 6** *happened*

Proof: There are two possibilities that **Event 2** could happen. 1) B impersonated a client whose identifier ID^* was not queried on H_1 as the I-th one, to the J-th server oracle. While this could not happen if **Event 6** happened. 2) B did not query $EK = H_3(\hat{e}(X, sP)^{b+z})$, and so did not know the used symmetric key EK because H_3 is a random oracle, but still was able to impersonate the targeted client whose identifier ID^* is the I-th distinct client identifier queried on H_1. In this case A should generate a valid ciphertext c for a message $m = (r^*, ID^*)$ with $X = r^* H_1(ID^*)$. However we show if the adversary can generate such valid ciphertext to make the server accept the message, it can break the used $\mathcal{E}$ scheme. In the one-time symmetric-key encryption game, B randomly chooses another m' with equal length of m and uses (m, m') as the finding message pair. Once it has received the challenge ciphertext c^*, B checks if $c = c^*$. Because the encryption algorithm $\mathbb{E}$ in $\mathcal{E}$ is a deterministic algorithm, B can always win the game.

Claim 8 **Event 3** *happened with negligible probability if* **Event 6** *happened.*

Proof: This event implies that $EK = H_3(\hat{e}(X, sP)^{b+z})$ was not queried on H_3. As proved in Claim 7, this event can only happen with negligible probability when **Event 6** happened.

Claim 9 *The Reveal query does not help the adversary to win the game.*

Proof: The proof is similar to Claim 2. Omitted.

So, let $\mathcal{F}$ be the event that A did not abort the game. We have,

$$\Pr[\mathcal{F}] = \Pr[\text{Event } 6] \geq \frac{1}{q_1 \cdot q_S}.$$

Let $\mathcal{H}$ be the event that $\hat{e}(saP, bP + zx_j sP)^{r_C}$ has been queried to H_3 where r_C is recovered from $\{m\}_{EK}$ in the message to oracle $\Pi^J_{i,j}$ with $i \in ID^S$ and $z = H_2(r_C aP, bP)$. We have

$$Pr[A \text{ wins}] = Pr[\mathcal{F} \wedge \mathcal{H}] \geq \frac{1}{q_1 \cdot q_S} \cdot \epsilon(k).$$

□

Proof Sketch of Theorem 4

Proof (sketch): If the adversary B can win the game by choosing some session between party I and J, given (aP, bP, sP) we construct an algorithm A for the GBDH problem as follows. A sets $d_I = sxP$ and $d_J = syP$ for some $x, y \in \mathbb{Z}_q^*$. In the attacking session, the algorithm sets $r_I Q_I = aP$ and $r_J Q_J = bP$ (i.e. $r_I = a/x$ and $r_J = b/y$ by noticing that $Q_I = xP$ and $Q_J = yP$. Note that according to the rules of the game, the adversary should not tamper with the messages in the challenge session). Then the algorithm computes $T = \hat{e}(P, P)^{sab} = \hat{e}(Q_I, Q_J)^{sr_I r_J} = \frac{K}{\hat{e}(r_I Q_I, d_J)^h}$, where K is the established secret of the session which is computed by the adversary subroutine. Note that B has to compute K to win the game with non-negligible advantage because H_5 is a random oracle. While in the proof, there is a pitfall: To simulate the challenge session, A has to generate a ciphertext which it does not know both the full plaintext m and the encryption key EK. This problem can be solved by making use of the access to the DBDH oracle, the programmability of random oracles, and the ciphertext indistinguishability of the used $\mathcal{E}$. These strategies have been used in proofs of Theorem 1, 2. □