# A Provable ID-Based Explicit Authenticated Key Agreement Protocol without Random Oracles

Haibo Tian
Key Lab. on ISN, Xidian University, Xi'an, China
hb_tian@hotmail.com

**Abstract:** This paper gives out an identity-based key agreement protocol and a modified proof model. The protocol can be proved secure in the proof model. The random oracle is never used in the model thanks to an encryption scheme proposed by Gentry in EuroCrypt 2006. Our main idea is the construction of a key agreement protocol from an encryption scheme, which is converse to the traditional construction of an ElGamal encryption scheme from Diffie-Hellman key agreement protocol. The modified model is based on the widely used model proposed by Bellare and Rogaway in 1993. The different is that the model here refines the ability of an adversary and the security goals of a protocol. The refinement captures most security properties and facilitates the proof of reduction to contradiction.
**Keywords:** Identity-based Protocol, Key Agreement, Security Model, Random Oracles

## 1. Introduction

We explain some concepts about explicit authenticated key agreement protocol according to [1]. An explicit authenticated key agreement protocol is a key agreement protocol which provides explicit key authentication. A key agreement protocol or mechanism is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by, or associated with, each of these, (ideally) such that no party can predetermine the resulting value. And key establishment is a process or protocol whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use. Explicit key authentication is the property obtained when both implicit key authentication and key confirmation hold. Implicit key authentication is the property whereby one party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may access to a particular secret key. And key confirmation is the property whereby one party is assured that a second party (possibly unidentified) actually has possession of a particular secret key.

A key agreement protocol is said to be identity-based (ID-based) if identity information of the party involved is used as the party's public key. ID-based protocols need no public key infrastructure, and can obtain explicit key authentication using only key indexed hash functions. After Shamir proposed the idea of identity-based asymmetric key pairs [2], a few identity-based key agreement protocols based on Shamir's idea have been developed, such as [3], [4], [5] etc. However the practical ID-based protocols boomed after appeared the work of [6] and [7] based on paring techniques. Some of the protocols are [8], [9], [10], [11], [12], [13], [14], [15], and [16] etc. The practical protocols enjoy some security properties, such as partially forward security, key control resistance etc.

Usually, some security properties are used to evaluate the security of key agreement protocols, including known session key security, perfect forward secrecy, key-compromise impersonation resilience, unknown key-share resilience, and key control resilience etc. By known session key security, we mean that the compromise of one session key should not compromise the keys established in other sessions. Perfect forward security in the two-party case usually means that if their private keys are compromised, the secrecy of session keys previously established by the two parties should not be affected. If the condition is relaxed to only one principle, it is called partially forward security. If the condition is restricted by adding the loss of the third trusted party's master key in the ID-based scenario, it is called master-key forward security [15]. By key-compromise impersonation resilience, we mean that 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 means that party $A$ should not be able to be coerced into sharing a key with party $C$ when in fact $A$ thinks that she/he is sharing the key with some party $B$. By key control resilience, we mean that one single party should not be able to decide the formation of the session key. It is desirable that the above security properties are captured in a security model, and that a protocol satisfies the security goals defined in the security model, so as to conclude a protocol enjoys the security properties.

To the best of our knowledge, there are some models to prove ID-based protocols, at least including BR model [17], BRP model [18], BCP model [19], CK model [20] etc. Most ID-based protocols are proved in some variant models of BR model, such as protocols in [12], [13], [14], [15], and [16]. Usually, an adversary in a BR style model is powered by some kinds of queries, such as Send, Reveal, Corrupt queries etc. The execution of a protocol is described as oracle responses to the adversary's queries. After polynomial bounded times queries, the adversary is expected to pass a test with a non-negligible probability. If the adversary cannot pass the test and the oracles' *views* or adversary's transcripts satisfy some security properties, it is believed that the protocol is secure in the defined model. Roughly all BR style models are defined and used in the above fashion.

In a BR style model, a security property may be captured according to the definitions and usage of the model. For example, if the Reveal query in a model is defined as session key revealing to an adversary, then a secure protocol in the model enjoys the known session key security. Otherwise, if a key compromise of one session $s$ can compromise another session $t$, the adversary can simply select the session $t$ as the test session and reveal the session $s$ to obtain the answer of the test query. Another example is about perfect forward security which is not captured by some BR style models for ID-based protocols [9], [14], [17] etc. If the Corrupt query in a model is defined as long term private key disclosure and the query is not allowed to corrupt the tested session even after the Test query, the model can not capture the perfect forward security. But if the adversary is relaxed to corrupt any session after the Test query and if the response to Corrupt query is only the long term private key, the perfect forward security property can be captured, similar with the definition 6 in [15].

Another interesting point about current BR style models are the use of random oracles. The random oracle are used in that the distribution of session key is defined as uniform distribution in $\{0,1\}^k$, where $k$ is related to the security parameter. Usually, a session key is an output of a hash function in a protocol, which is modeled as random oracle so as to satisfy the requirement of uniform distribution. Another reason to use random oracle is due to the public key generation phase of ID-based protocols. Usually, a hash function is used to derive public key from ID, which is modeled as random oracle. However random oracle model is criticized for its inability of instantiation. Canetti et al [21] provided a contrived example and Bellare et al [22] provided a more natural example to show that a secure scheme in ROM is not secure in real world.

Our contributions include a security model and an ID-based protocol. The modified model is still a BR style model. However, we refined the model as follows. At first, Extract query is used for private key disclosure. Then Corrupt query serves as a method to obtain oracle's internal variables. This ability reflects the adversary's power to dump target's memory. Next the Extract query can be used to query any session after Test query, which is particularly for the perfect forward security property. And then an authentication condition with explicit adversary power description serves as one security goal, which captures the key compromise impersonation. That goal assures that a secure protocol in our model enjoys authentication property. Finally, session key distribution is demanded uniformly only in the session key sample space and test query is simplified by only demanding an equal priori probability, which enables the direct usage of reduction to contradiction method, and opens the way to reduce the security of protocol directly to mathematical hard problems.

Our ID-based protocol is derived naturally. Gentry in EuroCrypt 2006 proposed an IND-CPA ID-based encryption scheme [23], which can be proved without random oracles with short public parameters. The IND-CPA scheme is similar with the famous ElGamal encryption scheme except that the random part is carried by two group elements. We notice that the random part in the original ElGamal encryption scheme is in fact a Diffie-Hellman public value. So we take the two group elements in Gentry's scheme as a whole serving as a Diffie-Hellman public key. Then we obtain an ID-based Diffie-Hellman key agreement scheme with almost the same property of the original Diffie-Hellman scheme. Then advantage of the ID-based version over the original one is that the ID-based one does not need signature primitive to prevent man in the middle attack. The key indexed hash function is enough. So a three pass ID-based protocol with explicit key authentication property can be obtained using only key indexed hash functions and group operations.

The security model, introduction of bilinear maps, and complexity assumption of our protocol are placed in Section 2. Section 3 is our ID-EAKA protocol. The proof of the protocol is in the Section 4. The last is the Conclusion.

# 2. Preliminaries

Below, we give the security model for an ID-based key agreement protocol. We also review the definition of a bilinear map and discuss the complexity assumption on which the security of our protocol is based.

## 2.1 Security Model

Our security model is based on Bellare and Rogaway [17] security model for key agreement protocols with several modifications. In our model, the protocol determines how parties behave in response to input signals from their environment. Each party may execute the protocol multiple times with the same or different partners. This is modeled by allowing each party to have different instances that execute the protocol. An oracle $\Pi_{i,j}^{s}$ models the behavior of a party with identity $i$ carrying out a protocol session in the belief that it is communicating with a party with identity $j$ for the $s$th time. One oracle instance is used only for one time, which maintains a variable *view* consisting of the oracle's protocol transcripts so far.

An adversary is modeled by a probabilistic polynomial time Turing machine that is assumed to have complete control over all communication links in the network and to interact with parties via oracle accesses. The adversary $A$ is allowed to execute any of the following queries:

- Instantiate ($i, j, s$). The adversary $A$ lets party $i$ to communicate with party $j$ in the $s$th session. The system will sets up a new oracle $\Pi_{i,j}^{s}$ as response.
- Extract ($i$). This allows the adversary to get the long term private key of the party $i$.
- Send ($\Pi_{i,j}^{s}$, $X$). The adversary sends message $X$ to the oracle $\Pi_{i,j}^{s}$. The system will give the output of $\Pi_{i,j}^{s}$ to the adversary as response. If $X = \lambda$, the party $i$ is asked to initiate a session $s$ with party $j$, where $\lambda$ is an empty string.
- Reveal ($\Pi_{i,j}^{s}$). This asks the oracle $\Pi_{i,j}^{s}$ to reveal whatever session key it currently holds.
- Corrupt ($\Pi_{i,j}^{s}$). This allows the adversary obtains all internal variables of the oracle $\Pi_{i,j}^{s}$, such as temporal keys of oracle $\Pi_{i,j}^{s}$.

An oracle $\Pi_{i,j}^{s}$ exists in one of the following several possible states:

- Accepted: an oracle has accepted if it decides to accept, holding a session key, after receipt of properly formulated messages.
- Rejected: an oracle has rejected if it decides not to establish a session key and to abort the protocol.
- Unsettled: an oracle is unsettled if it has not made any decision to accept or reject.
- Opened: an oracle is opened if it has answered a Reveal query.
- Corrupted: an oracle is corrupted if it has answered a Corrupt query.
- Extracted: an oracle is extracted if it has involved in a Extract query

By $\Pi_{j,i}^{s'}$, matching oracle of $\Pi_{i,j}^{s}$, we mean that every message that $\Pi_{i,j}^{s}$ sends out is subsequently delivered to $\Pi_{j,i}^{s'}$, with the response of $\Pi_{j,i}^{s'}$ to this message being returned to the $\Pi_{i,j}^{s}$ as the next message.

The adversary is allowed to make a Test query to receive a value to guess.

- Test ($\Pi_{i,j}^{s}$). If the oracle $\Pi_{i,j}^{s}$ is accepted, not opened and not corrupted, and if the matching oracle (if any) $\Pi_{j,i}^{s}$ is not opened, not corrupted, and $j$ not extracted, the adversary can make a test query to it. The adversary receives either a real session key or a random value as the response with an equal priori probability.

After the test query, the adversary can continue making Instantiate, Extract, Send, Reveal, Corrupt queries to oracles, except that the adversary cannot corrupt the oracles $\Pi_{i,j}^{s}$ and its matching oracle $\Pi_{j,i}^{s}$ (if any).

The adversary is demanded to output one bit to show its advantage in winning the game.

- Output. The adversary output "0" to identify the real session key or "1" to identify the random value. The advantage of adversary in winning the game is

$$Adv_A = |\ \Pr[0 \mid \text{real session key}] - \Pr[0 \mid \text{random value}]\ |.$$

To define an explicit authenticated key agreement protocol, we should prove the protocol satisfying the following goals:

1. If two oracles are matching, then both of them are accepted and have a same session key which is distributed uniformly in the session key sample space.
2. If the oracle $\Pi_{i,j}^{s}$ is accepted, not opened and not corrupted, and if $j$ is not extracted, there is only one oracle $\Pi_{j,i}^{s'}$ whose *view* is identical to the *view* of $\Pi_{i,j}^{s}$ just before the oracle $\Pi_{i,j}^{s}$ accepts if the oracle $\Pi_{j,i}^{s'}$ is not opened and not corrupted.
3. The $Adv_A$ in the defined model is negligible.

*Remark*: Our model has some modifications comparing to current BR style proof models.

Some queries are added and some queries are refined. Instantiate query is added to show the basic principle of passive players and positive adversary about the security model. Extract query is specifically designed for ID-based system. The Extract query in our model has the power to obtain private keys for any identities. Corrupt query is modified to give the internal states to the adversary but not the long term private key. There is one point about the reveal query in our model. If a value is to be a session key only after an oracle accepts, the reveal query may obtain nothing before the oracle accepts. The value can only be obtained by the Corrupt query before the oracle accepts.

The definition of Test query is simplified. We do not define the detail method to produce response to an adversary but to restrict that the two possible responses must be equal priori probability. This flexibility gives us a possible way to lay the security of a protocol directly on a decisional mathematical hard problem. The $Adv_A$ is defined according to the Test query, which is a normal definition for a distinguisher to distinguish two different distributions.

The authentication security goal is somewhat different. The second goal is about authentication, which means that if the party $i$ accepts with intended party $j$, then before that point, the party $j$ must have executed the protocol with intended party $i$ as designed. Firstly, there is no restriction on the role of party $i$ or $j$ in the second goal. Secondly, the adversary restrictions are explicitly addressed in the authentication goal. The restrictions are in fact the same as those in the Test query, so that the conclusion about secrecy property can be used directly in the authentication proof.

## 2.2 Bilinear Maps

Basic notations are as follows.

1. $G$ and $G_T$ are two (multiplicative) cyclic groups of prime order $p$;
2. $g$ is a generator of $G$;
3. $e$: $G \times G \rightarrow G_T$ is a bilinear map.

Let $G$ and $G_T$ be two groups as above. A bilinear map is a map $e$: $G \times G \rightarrow G_T$ with the following properties:

1. Bilinear: for all $u, v \in G$ and $a, b \in Z$, we have $e(u^a, v^b) = e(u, v)^{ab}$;
2. Non-degenerate: $e(g, g) \neq 1$.

We say that $G$ is a bilinear group if the group action in $G$ can be computed efficiently and there exists a group $G_T$ and an efficiently computable bilinear map $e$: $G \times G \rightarrow G_T$ as above. Note that $e(,)$ is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

## 2.3 Complexity Assumptions

The security of our protocol is based on a complexity assumption that is called a truncated version of the decisional augmented bilinear Diffie-Hellman exponent assumption [23] (decisional ABDHE). The problem is defined as follows.

Given a vector of $q$+3 elements

$$(g', g'^{(\alpha^{q+2})}, g, g^{\alpha}, g^{(\alpha^2)}, ..., g^{(\alpha^q)}) \in G^{q+3}$$

as input, outputs $e(g, g')^{(\alpha^{q+1})} \in G_T$. We use $g_i$ and $g_i'$ to denote $g^{(\alpha^i)}$ and $g'^{(\alpha^i)}$ below. An algorithm $A$ has advantage $\varepsilon$ in solving truncated $q$-ABDHE if

$$\Pr[A(g', g'_{q+2}, g, g_1, ..., g_q) = e(g_{q+1}, g')] \geq \varepsilon$$

where the probability is over the random choice of generators $g$, $g'$ in $G$, the random choice of $\alpha$ in $Z_p$, and the random bits used by $A$. The assumption is that there is no such an probability polynomial time (p.p.t) algorithm $A$ has a non-negligible advantage $\varepsilon$.

The decisional version of truncated $q$-ABDHE is defined as one would expect. An algorithm $A$ that outputs $b \in \{0, 1\}$ has advantage $\varepsilon$ in solving truncated decision q-ABDHE if

$$\begin{aligned} &|\Pr[A(g', g'_{q+2}, g, g_1, ..., g_q, e(g_{q+1}, g') = 0] \\ &- \Pr[A(g', g'_{q+2}, g, g_1, ..., g_q, Z) = 0]| \geq \varepsilon \end{aligned}$$

where the probability is over the random choice of generators $g$, $g'$ in $G$, the random choice of $\alpha$ in $Z_p$, the random choice of $Z \in G_T$, and the random bits consumed by $A$.

The truncated (decision) ($t$, $\varepsilon$, $q$)-ABDHE assumption holds in $G$ if no $t$-time algorithm has advantage at least $\varepsilon$ in solving the truncated (decision) $q$-ABDHE problem in $G$.

# 3. The ID-EAKA Protocol

The detail of our protocol is defined as follow.

**Setup**: The PKG picks random generators $g$, $h \in G$ and random $\alpha \in Z_p$. It sets $g_1 = g^{\alpha} \in G$ and $g_T = e(g,g) \in G_T$. The public parameters and private master-key are given by

$$\text{parameters} = (g, g_1, h, g_T, H) \quad \text{master-key} = \alpha$$

where $H$ is a public hash function, $H$: $G_T \times G \times G_T \times G \times G_T \rightarrow \{0,1\}^k$, $k$ is a security parameter.

**Extract**: To generate a private key for identity $ID \in Z_p$, the PKG generates random $r_{ID} \in Z_p$, and outputs the private key

$$d_{ID} = (r_{ID}, h_{ID}), \text{ where } h_{ID} = (hg^{-r_{ID}})^{1/(\alpha - ID)}.$$

If $ID = \alpha$, the PKG aborts.

**Protocol flow**: For two parties Alice and Bob whose identification strings are $ID_A$ and $ID_B$, the algorithm proceeds as follows.

1. Alice selects $x \in_R Z_p$, computes $M_{11} = (g_1 g^{-ID_B})^x$ and $M_{12} = g_T^x$. Alice sends $M_1 = M_{11} \| M_{12}$ to Bob, where symbol "||" denotes concatenation.
2. Bob selects $y \in_R Z_p$, computes $M_{21} = (g_1 g^{-ID_A})^y$, $M_{22} = g_T^y$, $K_{BA} = e(g,h)^{xy}$, $M_{23} = H(K_{BA} \| M_{11} \| M_{12} \| M_{21} \| M_{22})$. Alice sends $M_2 = M_{21} \| M_{22} \| M_{23}$ to Bob. $K_{BA}$ is computed as follows:

$$K_{BA} = e((M_{11})^y, h_{ID_B})((M_{12})^y)^{r_{ID_B}}$$

3. Alice computes $K_{AB} = e((M_{21})^x, h_{ID_A})((M_{22})^x)^{r_{ID_A}}$ and $V\_M_{23} = H(K_{AB} \| M_{11} \| M_{12} \| M_{21} \| M_{22})$. If $M_{23} \neq V\_M_{23}$, Alice rejects and aborts the protocol. Else if $M_{23} = V\_M_{23}$, Alice accepts, sets $K_{AB}$ as the session key, computes $M_3 = H(K_{AB} \| M_{21} \| M_{22} \| M_{11} \| M_{12})$, and sends $M_3$ to Bob.
4. Bob computes $V\_M_3 = H(K_{BA} \| M_{21} \| M_{22} \| M_{11} \| M_{12})$. If $M_3 \neq V\_M_3$, Bob rejects and aborts the protocol. Else Bob accepts and sets $K_{BA}$ as the session key.

# 4. Security analysis

To prove security of the ID-EAKA protocol, we prove our protocol achieves the three security goals in the security model.

**TH 1**: If two oracles are matching, then both of them are accepted and have a same session key which is distributed uniformly at random on the session key space.

Proof. Suppose two oracles $\Pi_{i,j}^{s}$ and $\Pi_{j,i}^{s'}$. Assume the oracle $\Pi_{i,j}^{s}$ receives the Send ($\Pi_{i,j}^{s}$, $\lambda$) query. Then the oracle $\Pi_{i,j}^{s}$ acts as an initiator and $\Pi_{j,i}^{s'}$ as a responder. Before the initiator accepts, the initiator has a *view* ($M_1$, $M_2$) which is identical to the *view* of responder because the initiator and responder are matching. At that point,

$$K_{ji} = e((M_{11})^y, h_j)((M_{12})^y)^{r_i} = e((g_1 g^{-j})^{xy}, h_j)((g_T)^{xy})^{r_j} = e(g,h)^{xy}$$

$$= e((g_1 g^{-i})^{yx}, h_i)((g_T)^{yx})^{r_i} = e((M_{21})^x, h_i)((M_{22})^x)^{r_i} = K_{ij}$$

and the initiator and responder has identical ($M_{11}$|| $M_{12}$|| $M_{21}$|| $M_{22}$), so the equality $M_{23}$ = $V\_M_{23}$ holds. The initiator will accept according to the protocol and give the last message to the responder. Before the responder accepts, the responder has a *view* ($M_1$, $M_2$, $M_3$) which is identical to the *view* of initiator. Obviously, the responder will also accept.

The session key is $e(g,h)^{xy}$, where $e(g,h)$ can be determined by public parameters. The session key is distributed uniformly in $G_T$ since the exponent $x$ and $y$ are selected randomly during the protocol execution. □

**TH 2:** If the truncated decision $q$-ABDHE problem is hard, the $Adv_A$ is negligible with ($q$–1) times Extract queries.

Proof. Let $A$ be an adversary who has non-negligible $Adv_A$ in the defined model. We construct an algorithm $B$ solves the truncated decisional $q$-ABDHE problem.

$B$ takes as input a random truncated decision $q$-ABDHE challenge ($g'$, $g'_{q+2}$, $g$, $g_1$ ,…, $g_q$, $Z$), where $Z$ is either e($g_{q+1}$, $g'$) or a random element of $G_T$. Algorithm $B$ proceeds as follows.

**Setup**: $B$ generates a random polynomial $f(z) \in Z_p[z]$ of degree $q$. It sets $h=g^{f(\alpha)}$, computing $h$ from ($g$, $g_1$, … , $g_q$). Other public parameters $g_T$ and $H$ is defined as the protocol usual definition. The public parameters are ($g$, $g_1$, $h$, $g_T$, $H$). There is no master-key belonging to $B$.

**Queries**:

- Instantiate ($i, j, s$): $B$ sets up a new oracle $\Pi^s_{i,j}$ .
- Extract ($i$): If $i$ = $\alpha$, $B$ uses $\alpha$ to solve truncated decision $q$-ABDHE immediately. Else, let $F_i(z)$ denote the ($q$ –1) degree polynomial $(f(z) - f(i))/(z - i)$. $B$ computes ($r_i$, $h_i$) to be ($f(i)$, $g^{F_i(\alpha)}$). This is a valid private key for $i$, since $g^{F_i(\alpha)}$ = $g^{(f(z) - f(i))/(z - i)}$ = $(hg^{-f(i)})^{1/(\alpha-i)}$ as required. $B$ gives ($r_i$, $h_i$) to the adversary as response. Since the number of Extract queries is less than ($q$ –1) and $f(z)$ is random selected, the generated private key has identical distribution as in a real protocol context.
- Send ($\Pi^s_{i,j}$ , $X$). Suppose that $B$ guesses the oracle $\Pi^s_{I,J}$ is to be tested. The matching oracle of $\Pi^s_{I,J}$ is $\Pi^t_{J,I}$ that receives the first message sent by $\Pi^s_{I,J}$ or sends the first message received by $\Pi^s_{I,J}$ . Generally, suppose that $\Pi^s_{I,J}$ is the initiator. $B$ will compute $M_1$, $M_2$ and $M_3$ as follows for the two oracles when needed.

  Let $f_2(z) = z^{q+2}$ and let $F_{2,J}(z) = (f_2(z) - f_2(J))/(z - J)$, which is a polynomial of degree $q$ +1.

  $M_{11}$ = $g'^{(f_2(\alpha)-f_2(J))\cdot x}$ and $M_{12}$ = $Z^x \cdot e(g', \prod_{l=0}^{q} g^{F_{2,J,l}\alpha^l})^x$ where random value $x$ is selected as in our protocol definition.

$M_{21}$ = $(g_1 g^{-I})^y$, $M_{22}$ = $g_T^y$, $M_{23}$ = $H(K_{JI} \| M_{11} \| M_{12} \| M_{21} \| M_{22})$, where random value $y$ is selected as in our protocol definition and $K_{JI}$ is calculated as follows:

$$\begin{aligned} K_{JI} &= e((M_{11})^y, h_J)((M_{12})^y)^{r_J} \\ &= e(g'^{(f_2(\alpha)-f_2(J))}, g^{(f(x)-f(J))/(x-J)})^{xy}(Z \cdot e(g', \prod_{l=0}^{q} g^{F_{2,J,l}\alpha^l}))^{xyf(J)}. \end{aligned}$$

$M_3 = H(K_{JI} \| M_{21} \| M_{22} \| M_{11} \| M_{12})$ where $K_{JI}$ is obtained from the oracle $\Pi^t_{J,I}$.

$B$ will set the status of oracle $\Pi^s_{I,J}$ as accepted if $B$ checks that the *view* of $\Pi^s_{I,J}$ is exactly the specially produced messages just before the special $M_3$ is sent out. $B$ will set the status of oracle $\Pi^t_{J,I}$ as accepted if $B$ checks that the *view* of $\Pi^t_{J,I}$ is exactly the specially produced messages after the special $M_3$ is received. For any other Send queries that are not related to the above two oracles, $B$ will act exactly according to the protocol specification and model rules.

- Reveal ($\Pi^s_{i,j}$). If the query is to reveal the session key of accepted oracle $\Pi^s_{I,J}$ or $\Pi^t_{J,I}$, the guessed oracle to be tested or its matching oracle if any, $B$ will stop the game with a Fail output. Else, $B$ gives the session key hold by the oracle $\Pi^s_{i,j}$. Note that our protocol sets session key after an accept decision is made. Before the accept decision, the Reveal query will be responded by a $\lambda$ symbol.
- Corrupt ($\Pi^s_{i,j}$). If the query is to corrupt $\Pi^s_{I,J}$ or $\Pi^t_{J,I}$, $B$ will stop the game with a Fail output. Else, $B$ gives all internal variables of $\Pi^s_{i,j}$ to the adversary.
- Test ($\Pi^s_{i,j}$). If $B$ made a wrong guess, $B$ stops the game with a Fail output. Else, $B$ gives the $K_{JI}$ to the adversary.

**Output:**

$B$ will forward the output of our adversary to the truncated decision $q$-ABDHE challenger as a response.

**Analysis:**

If $B$ does not stop before the output event, the simulation is indistinguishable. First, from the viewpoint of the adversary, the only chance to distinguish the simulation is to analyze the messages $M_1$, $M_2$ and $M_3$. The reason is that the output of Extract query has identical distribution as in a real protocol context, and that the outputs of other queries are generated according to the protocol specification or the model rules. Next let's focus on the doubtable messages. Assume $s = (\log_g g')F_{2,J}(\alpha)$, then $M_{11}$ = $g^{xs(\alpha-J)}$. If $Z$ = $e(g_{q+1}, g')$, $M_{12}$ = $g_T^{xs}$ and $K_{JI} = e(g,h)^{xsy}$. So the messages $M_1$, $M_2$ and $M_3$ have identical distribution as in a real protocol context in this case. If $Z \neq e(g_{q+1}, g')$, then the distribution of $M_{12}$, $M_{23}$ and $M_3$ are not the same as in a real protocol context. If the adversary can distinguish the two distributions, then the adversary can distinguish the value $Z$ from $e(g_{q+1}, g')$, which contradicts the truncated decision q-ABDHE assumption.

If the simulation is indistinguishable, the adversary should give a qualified output. So if B does not stop before the output event, the adversary will give "0" to identify the real session key or "1" to identify the random value. Since what the adversary obtained from the Test query is just $K_{JI}$, which is a real session key if $Z = e(g_{q+1}, g')$ or a random value if not, $B$ can use adversary's output to solve the truncated decision q-ABDHE problem with an identical advantage.

Now we calculate the probability that $B$ does not stop. If $B$ made a right guess, then there is no stop event before output because a right guess means that the guessed oracle is accepted, not opened, not corrupted and the matching oracle (if any) $\Pi^s_{j,i}$ is not opened, not corrupted, and not extracted just before the Test query event. Also the guessed oracle and its matching oracle (if any) are limited not to be revealed or corrupted after test query event and before the output event. So the probability that $B$ does not stop equals to the probability of right guess, which is at least $1/q$. The conclusion is that if $Adv_A$ is non-negligible, the probability to solve the $q$-ABDHE problem is at least $Adv_A/q$. □

**TH 3:** If an oracle $\Pi^s_{i,j}$ accepts, not opened and not corrupted, if $j$ has never been extracted, there is only one oracle $\Pi^{s'}_{j,i}$ whose *view* is identical to the *view* of $\Pi^s_{i,j}$ just before the oracle $\Pi^s_{i,j}$ accepts if the oracle $\Pi^{s'}_{j,i}$ is not opened and not corrupted.

Proof. Suppose there are $q_I$ times Instantiate queries at all. We divided the proof into two parts according to oracle roles.

Case 1: Suppose the oracle $\Pi^s_{i,j}$ receives the Send ($\Pi^s_{i,j}$ , $\lambda$) query as an initiator. If the oracle $\Pi^s_{i,j}$ accepts, according to the protocol, the equation $M_{23} = V\_M_{23}$ holds. $V\_M_{23} = H(K_{ij}||M_1||M_{21}||M_{22})$, where $M_1$ is produced by the initiator and $M_{21}||M_{22}$ is received by the initiator. If $M_{23}$ is not computed the same as $V\_M_{23}$, it means a target collision of $H$ function. So $M_{23} = H(K_{ij}||M_1||M_{21}||M_{22})$. The $M_{23}$ producer can access to $K_{ij}$ because of the one-way property of $H$ function. Since TH2 says that the adversary can not distinguish $K_{ij}$ from random values, the value $K_{ij}$ can only be accessed by oracles. The $M_{23}$ producer is not the oracle $\Pi^s_{i,j}$ because there is no $K_{ij}$ in the oracle when $M_{23}$ is created. The $M_{23}$ producer should have received $M_1$ since $K_{ij}=e(g,h)^{xy}$ involves random value $x$ and this value is selected only by oracle $\Pi^s_{i,j}$ with probability $(1-q_I/p)$. The $M_{23}$ producer should be oracle $\Pi^{s'}_{j,X}$ because identity $j$ is included in $M_1$, where $X$ denotes any party. The $M_{21}||M_{22}$ should also be created by oracle $\Pi^{s'}_{j,X}$ since $K_{AB}=e(g,h)^{xy}$ involves random value $y$ and this value is selected only by oracle $\Pi^{s'}_{j,X}$ with probability $(1-q_I/p)$. The oracle $\Pi^{s'}_{j,X}$ should be $\Pi^{s'}_{j,i}$ since identity $i$ is included in $M_2$. The oracle $\Pi^{s'}_{j,i}$ is unique at least with a probability $(1-2q_I/p)$.

Case 2: Suppose oracle $\Pi^{s}_{i,j}$ acts as a responder. If the responder accepts, according to the protocol, the equation $M_3 = V\_M_3$ holds. $V\_M_3 = H(K_{ij} \,||\, M_{21}||\, M_{22}||\, M_1)$, where $M_{21}||\, M_{22}$ is produced by the responder and $M_1$ is received by the responder. As in case 1, $M_3$ should be computed the same as $V\_M_3$, such that $M_3 = H(K_{ij} \,||\, M_{21}||\, M_{22}||\, M_1)$. As in case 1, the $M_3$ producer can access to $K_{ij}$. From TH2, the $K_{ij}$ can be computed only by uncorrupted oracles. The $M_3$ producer is not the responder because the responder only used $K_{ij}$ in the $M_{23}$ computation, which was equal to $M_3$ with probability less than $1/p$ (if $i=j$ and $x=y$). The $M_3$ producer should have received $M_2$ since $K_{ij}=e(g,h)^{xy}$ involves random value $y$ and this value is selected only by the responder with probability $(1-q_I/p)$. The $M_3$ producer should be oracle $\Pi^{s'}_{j,X}$ because identity $j$ is included in $M_2$, where $X$ denotes any party. The $M_1$ should also be created by oracle $\Pi^{s'}_{j,X}$ since $K_{AB}=e(g,h)^{xy}$ involves random value $x$ and this value is selected only by oracle $\Pi^{s'}_{j,X}$ with probability $(1-q_I/p)$. The oracle $\Pi^{s'}_{j,X}$ should be $\Pi^{s'}_{j,i}$ since identity $i$ is included in $M_1$. The oracle $\Pi^{s'}_{j,i}$ is unique at least with a probability $(1-(2q_I+1)/p)$. □

## 5. Security Property

We consider the following security properties.

- Known session keys. The adversary in our model can use Reveal query to obtain session keys before or after a Test query.
- Impersonation attack resistance. If a protocol cannot resist impersonation attack, one party, say Alice, may accepted with a wrong belief that she is communicating with Bob while she is communicating with Charles. In our model, this means that when an oracle $\Pi^{s}_{A,B}$ accepts, it is not the case that there is only one oracle $\Pi^{s'}_{B,A}$ whose *view* is identical to the *view* of $\Pi^{s}_{A,B}$. It may be the case that there is no matching oracle of $\Pi^{s}_{A,B}$. For Example, the attack 11.2 in [24] for authentication only STS protocol can be modeled by adversary extracting Charles's private key. When Bob accepted, there is no matching oracle. It also may be the case that there is a matching oracle $\Pi^{s'}_{B,X}$ ($X$ denotes uncertain party) whose

*view* is identical to the *view* of $\Pi^{s}_{A,B}$. For example, the attack 11.3 in [24] for STS protocol can be modeled in our model by only using Send queries. When Alice accepted, there is a oracle $\Pi^{s'}_{B,X}$ serving as a matching oracle of $\Pi^{s}_{A,B}$.

- Unknown key share. If a protocol can not resist unknown key share attack, one party, say Alice, may accepted with a wrong belief that she shares a key with Bob in one session while she shares it with Charles or with another session of Bob. In our model, it means that an oracle $\Pi^{s}_{A,B}$ accepted, while oracle $\Pi^{s'}_{X,Y}$ ($X$ and $Y$ denotes uncertain parties) held the shared key and didn't match the oracle $\Pi^{s}_{A,B}$. An adversary can obtain advantage by Test($\Pi^{s}_{A,B}$), and Reveal($\Pi^{s'}_{X,Y}$) or Corrupt($\Pi^{s'}_{X,Y}$).
- Key compromise impersonation resilience. The second security goal is enough to capture this property. The argument is similar to the impersonation attack resistance property. Note that the second goal has never demand the private key of *i* should be secret.
- Perfect forward secrecy. This property requires that previously agreed session keys should remain secret, even if both parties' long-term private keys are compromised. The intended allowance of Extract queries after Test query event and the third goal in the model assure this property.
- Key control resilience. It means that one single party should not be able to decide the formation of the session key. It is interesting to note that our model can not capture this property because key formation is neither a property about secrecy nor a property about authentication. Fortunately, the definition of key agreement protocol shows evidence that our protocol is key control resilience.

## 6. Conclusion

We proposed a modified BR style proof model and an ID-based protocol. The modified model can capture more security properties and facilitate the direct reduction to contradictions proof method in protocol security proof. The ID-based protocol is an explicit authenticated key agreement protocol using no signatures with a standard model proof.

## Reference


[1] A.J. Meneaes, P.C. van Oorschot, and S.A. Vanstone. Handbook of Applied Cryptography. CRC Press, 1997.

[2] A. Shamir. Identity-based cryptosystems and signatures schemes. In G.T. Blakey and D. Chaum, editors, Advanced in Cryptography – Proceedings of Crypto'84, LNCS 196, pp. 48-53. Spring-Verlag, 1985.

[3] E. Okamoto. Proposal for identity-based key distribution system. Electronics Letters. Vol.22, pp. 1283-1284. 1986.

[4] M. Girault and J. Paillés. An identity-based scheme providing zero-knowledge authentication and authenticated key exchange. In Proceedings of ESORICS 90, pages 173–184. 1990.
[5] K. Tanaka and E. Okamoto. Key distribution system for mail systems using ID-related information directory. Computers and Security Vol.10, pp. 25-33. 1991.
[6] A. Joux. A one round protocol for tripartite Diffie-Hellman. In proceedings of Algorithmic number theory symposium, ANTS-IV, LNCS 1838, pp. 385-394. 2000.
[7] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In proceedings of 2000 symposium on Cryptography and Information Security,SCIS 2000.
[8] N. P. Smart. Identity-based authenticated key agreement protocol based on Weil pairing. Electronics Letters Vol.38, No.13, pp. 630-632. 2002.
[9] L. Chen and C. Kudla. Identity based authenticated key agreement protocols from pairing. In proceedings of 16th IEEE Security Foundations Workshop, pp. 219-233. IEEE Computer Society Press, 2003.
[10] M. Scott. Authenticated ID-based key exchange and remote log-in with insecure token and PIN number. http://eprint.iacr.org/2002/164.pdf
[11] K. Shim. Efficient ID-based authenticated key agreement protocol based on the Weil pairing. Electronics Letters. Vol.39, No.8, pp. 653-654. 2003.
[12] P. McCullagh and P. Barreto. A new two-party identity-based authenticated key agreement. In Proceedings of CT-RSA 2005, LNCS 3376, pp. 262-274. Springer-Verlag, 2005.
[13] K.-K. R. Choo, C. Boyd, and Y. Hitchcock. On Session Key Construction in Provably-Secure Key Establishment Protocols. First International Conference on Cryptology in Malaysia - Mycrypt 2005, LNCS 3715, pp. 116-131. Springer-Verlag. 2005.
[14] Y. Wang. Efficient Identity-Based and Authenticated Key Agreement Protocol. Cryptology ePrint Archive, Report 2005/108.
[15] Z. Cheng, L. Chen, R. Comley, and T. Tang. Identity-Based Key Agreement with Unilateral Identity Privacy Using Pairings. In 2nd Information Security Practice and Experience Conference - ISPEC 2006, LNCS 3903. Springer-Verlag, 2006.
[16] K. Y. Choi, J. Y. Hwang, D. H. Lee, and I. S. Seo. ID-based Authenticated Key Agreement for Low-Power Mobile Devices. In Tenth Australasian Conference on Information Security and Privacy - ACISP 2005, LNCS 2005, pp. 494-505. Springer-Verlag, 2005.
[17] M. Bellare, and P. Rogaway. Entity Authentication and Key Distribution. In Advances in Cryptology - Crypto 1993, LNCS 773, pp. 110-125. Springer-Verlag, 1994.
[18] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In Advances in Cryptology – Eurocrypt 2000, LNCS 1807, pp. 139 -155. Springer-Verlag, 2000.
[19] E. Bresson, O. Chevassut, and D. Pointcheval. Provably Authenticated Group Diffie–Hellman Key Exchange –The Dynamic Case. In Advances in Cryptology - Asiacrypt 2001, LNCS 2248, pp. 209-223. Springer-Verlag, 2001.
[20] R. Canetti, and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In Advances in Cryptology - Eurocrypt 2001, LNCS 2045, pp. 453-474. Springer-Verlag, 2001.
[21] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In proceedings of the 30th Annual Symposium on the Theory of Computing (STOC'98), pages 209-218. ACM Press, 1998.

[22] M. Bellare, A. Boldyreva, and A. Palacio. A uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In C. Cachin and J. Camenisch, editor, Advance in Cryptology–Proceedings of EUROCRYPT2004, Lecture Notes in Computer Science 3027, pages 171-188. Springer-Verlag, 2004.
[23] C. Gentry. Practical Identity-Based Encryption Without Random Oracles. In S. Vaudenay, editor, proceedings of EUROCRYPT 2006, LNCS 4004, pp. 445-464. Springer-Verlag, 2006.
[24] W. Mao. Modern cryptography: theory and practice. Prentice-Hall PTR. 2003.