On the Security of Wang's Provably Secure Identity-based Key Agreement Protocol

Maurizio Adriano Strangio

University of Rome "Roma Tre", ROME, ITALY
strangio@mat.uniroma3.it

Abstract. In a 2005 IACR report, Wang published an efficient identity-based key agreement protocol (IDAK) suitable for resource constrained devices.
The author shows that the IDAK key agreement protocol is secure in the Bellare-Rogaway model with random oracles and also provides separate ad-hoc security proofs claiming that the IDAK protocol is not vulnerable to Key Compromise Impersonation attacks and also enjoys Perfect Forward Secrecy (PFS).
In this report, we review the security properties of the protocol and point out that it is vulnerable to Unknown Key Share attacks. Although such attacks are often difficult to setup in a real world environment they are nevertheless interesting from a theoretical point of view so we provide a version of the protocol that fixes the problem in a standard way. We also provide a security proof of the IDAK protocol based on the Gap Bilinear Diffie Hellman and random oracle assumptions in the stronger extended Canetti-Krawczyk security model of distributed computing.

1 Introduction

In a 2005 IACR report ([9] and also [10]), Wang proposed a novel identity-based key agreement protocol (IDAK) using the Weil/Tate pairing and also provided a security proof in the Bellare-Rogaway model [1].

The model does not capture Key Compromise Impersonation (KCI) attacks or Perfect Forward Secrecy (PFS); for the later security features the author provides separate ad-hoc proofs. However, the protocol is vulnerable to Unknown Key Share (UKS) attacks. Although such attacks are often difficult to setup in a real world environment they are nevertheless interesting from a theoretical point of view so we provide a version of the protocol that fixes the problem in a standard way.

We also show that the IDAK protocol is provably secure in the extended Canetti-Krawczyck (eCK) model [7] under the Gap Bilinear Diffie Hellman and random oracle assumptions.

2 Notation and mathematical background

To make the paper self-contained, we briefly recall the underlying mathematical concepts and notation. Let us consider two multiplicative cyclic groups G and G_1 of order q with g a generator of G. The bilinear map $\hat{e} : G \times G \rightarrow G_1$ has the following three properties:

1. bilinearity, for all $g_1, g_2 \in G$ and $x, y \in Z : \hat{e}(g_1^x, g_2^y) = \hat{e}(g_1, g_2)^{xy} = \hat{e}(g_1^y, g_2^x)$;
2. non-degeneracy, for all $g \in G$, $\hat{e}(g, g) \neq 1$ is a generator in G_1;
3. computability, for $g_1, g_2 \in G : \hat{e}(g_1, g_2) \in G_1$ is computable in polynomial time.

The modified Weil and Tate pairings associated with supersingular elliptic curves are examples of admissible pairings [5], [3].

If X is a finite set then $x \stackrel{R}{\leftarrow} X$ or $x \in_R X$ denote the sampling of an element uniformly at random from X. If α is neither an algorithm nor a set $x \leftarrow \alpha$ represents a simple assignment statement.

The (computational) Bilinear Diffie-Hellman assumption (BDH) holds in the group G if for random elements $x, y, z \in Z_q^*$ it is computationally hard to compute $\hat{e}(g, g)^{xyz}$.

Assumption 1 (BDH) *The group G satisfies the Bilinear Diffie-Hellman assumption if for all PPT algorithms we have:*

$$x \stackrel{R}{\leftarrow} Z_q^*; y \stackrel{R}{\leftarrow} Z_q^*; z \stackrel{R}{\leftarrow} Z_q^*; X \leftarrow g^x; Y \leftarrow g^y; Z \leftarrow g^z :$$
$$\Pr\left[\mathcal{A}(X, Y, Z) = \hat{e}(g, g)^{xyz}\right] < \epsilon$$

where the probability is taken over the coin tosses of $\mathcal{A}$ (and random choices of x, y, z) and ϵ is a negligible function.

The Decisional Bilinear Diffie-Hellman assumption (DBDH) holds in the group G if for random elements $x, y, z, r \in Z_q^*$ it is computationally hard to distinguish the distributions $\langle g^x, g^y, g^z, g^r \rangle$ and $\langle g^x, g^y, g^z, \hat{e}(g, g)^{xyz} \rangle$.

Assumption 2 (DBDH) *The group G satisfies the Decisional Bilinear Diffie-Hellman Assumption if for all PPT algorithms we have:*

$$x \stackrel{R}{\leftarrow} Z_q^*; y \stackrel{R}{\leftarrow} Z_q^*; z \stackrel{R}{\leftarrow} Z_q^*; r \stackrel{R}{\leftarrow} G; X \leftarrow g^x; Y \leftarrow g^y; Z \leftarrow g^z :$$
$$\Pr[\mathcal{A}(X, Y, Z, r) = 1] - \Pr[\mathcal{A}(X, Y, Z, \hat{e}(g, g)^{xyz}) = 1] < \epsilon$$

where the probability is taken over the coin tosses of $\mathcal{A}$ (and random choices of x, y, z, r) and ϵ is a negligible function.

The Gap Bilinear Diffie-Hellman assumption (GBDH) holds in the group G if for random elements $x, y, z \in Z_q^*$ it is computationally hard to solve the BDH problem even with access to a DBDH oracle.

Assumption 3 (GBDH) *The group G satisfies the Gap Bilinear Diffie-Hellman Assumption if for all PPT algorithms we have:*

$$x \stackrel{R}{\leftarrow} Z_q^*; y \stackrel{R}{\leftarrow} Z_q^*; z \stackrel{R}{\leftarrow} Z_q^*; X \leftarrow g^x; Y \leftarrow g^y; Z \leftarrow g^z :$$
$$\text{DBDH}(g, g^a, g^b, g^c, h) \leftarrow 1 \text{ iif } h = e(g, g)^{abc}; a \stackrel{R}{\leftarrow} Z_q^*; b \stackrel{R}{\leftarrow} Z_q^*; c \stackrel{R}{\leftarrow} Z_q^*;$$
$$\Pr\left[\mathcal{A}(X, Y, Z) = \hat{e}(g, g)^{xyz}\right] < \epsilon$$

where the probability is taken over the coin tosses of $\mathcal{A}$ (and random choices of x, y, z) and ϵ is a negligible function.

3 Review of the IDAK protocol

In this section we review the IDAK identity-based key agreement protocol. The protocol is completely specified by three algorithms Setup, Extract, Exchange [9]:

- **Setup**, for input the security parameter k:
  1. Generate a bilinear group $G_\rho = \{G, G_1, \hat{e}\}$ with the groups G and G_1 of prime order q. Define h as the co-factor of the group order q for G;
  2. Choose a generator $g \in G$;
  3. Choose a random master secret key $\alpha \in_R Z_q^*$;
  4. Choose the cryptographic hash functions $H : \{0,1\}^* \rightarrow G$ and $\pi : G \times G \rightarrow Z_q^*$. In the security analysis of protocol IDAK, H and π are simulated as random oracles.

  The system parameters are $(hq, h, g, G, G_1, \hat{e}, H, \pi)$ and the master secret key is α.
- **Extract**, For a given identification string $ID \in \{0,1\}^*$, the algorithm computes $g_{ID} = H(ID) \in G$ and returns the private key $d_{ID} = g_{ID}^{\alpha}$;
- **Exchange**, For two peers A and B with identities ID_A and ID_B respectively, the algorithm proceeds as follows:
  1. A selects $x \in_R Z_q^*$, computes $R_A = g_{ID_A}^x$ and sends R_A to B;
  2. B selects $y \in_R Z_q^*$, computes $R_B = g_{ID_B}^y$ and sends R_B to A;
  3. On receipt of R_B, A computes $s_A = \pi(R_A, R_B), s_B = \pi(R_B, R_A)$ and the shared secret sk_{AB} as $\hat{e}(g_{ID_A}, g_{ID_B})^{(x+s_A)(y+s_B)h\alpha} = \hat{e}(g_{ID_B}^{s_B} \cdot R_B, g_{ID_A}^{(x+s_A)h\alpha})$;
  4. On receipt of R_A, B computes $s_A = \pi(R_A, R_B), s_B = \pi(R_B, R_A)$ and the shared secret sk_{BA} as $\hat{e}(g_{ID_A}, g_{ID_B})^{(x+s_A)(y+s_B)h\alpha} = \hat{e}(g_{ID_A}^{s_A} \cdot R_A, g_{ID_B}^{(x+s_B)h\alpha})$;

The main result of [9] is Theorem 5.2 which proves that IDAK is a secure key agreement protocol in the Bellare-Rogaway model under the DBDH and random oracle assumptions. The author also presents ad ad-hoc security proofs claiming that the protocol enjoys PFS (Theorem 6.1) and is not vulnerable to KCI attacks (Theorem 7.1).

4 The security features of the IDAK protocol

In this section we review the security properties of the IDAK protocol.

4.1 Entity Impersonation

When an honest party A is corrupted then a malicious adversary can easily impersonate him in a protocol execution with another party B since the adversary has total control of any session initiated with identity A. The adversary can also register a new identity (say C) with the KGC and successfully conduct an impersonation attack against B.

4.2 Forward Secrecy

A key agreement protocol has perfect forward secrecy (PFS), if after a communication session between two peers is completed, the adversary cannot learn the session key even if it corrupts both parties involved in that session. In other words, learning the long term private keying material of parties should not compromise the security of past completed sessions.

A weaker notion of forward secrecy (weak perfect forward secrecy − wPFS) considers only completed sessions in which the adversary was passive (i.e. did not modify the messages transcripts exchanged by the parties).

For identity protocols "'master key PFS"' (mkPFS) refers to the consequences of KGC corruption. The IDAK protocol does not possess mkPFS since knowledge of the secret master key α would allow even a passive adversary to trivially break the protocol since the session key of two peers A, B is calculated as $\hat{e}(g_{ID_B}^{s_B} \cdot R_B, R_A \cdot g_{ID_A}^{s_A})^{\alpha}$. To make the IDAK protocol mPFS Wang proposes the exchange of a an additional Diffie-Hellmann ephemeral key that should be included in the computation of the session key (see Section 6 in [9]).

4.3 Key Compromise Impersonation

In a KCI attack, the opponent has learned the long-term private key of an honest party (say A) and attempts to establish a valid session key with A by masquerading as another legitimate principal (say B). This attack represents a subtle threat that is often underestimated and difficult to counter [8].

Suppose a malicious adversary $\mathcal{A}$ has learned the long term private key d_{ID_A} of principal A; she is now able to set up a man-in-the-middle attack during a run of the protocol between A and B. The attack should work as follows. $\mathcal{A}$ lets message R_A reach its intended destination (B) but replaces B's response R_B to A with a transcript X. On receipt of X, A calculates its session key as follows:

$$\begin{aligned} sk_{AB} &= \hat{e}(g_{ID_B}^{s_B} \cdot X, d_{ID_A}^{(x+s_A)}) \\ &= \hat{e}(g_{ID_B}^{s_B} \cdot X, (g_{ID_A}^{x} \cdot g_{ID_A}^{s_A})^{h\alpha}) \\ &= \hat{e}(d_{ID_B}^{s_B} \cdot X^{\alpha}, R_A \cdot g_{ID_A}^{s_A}) \end{aligned}$$

To succeed in the attack $\mathcal{A}$ must choose a suitable message X possibly exploiting the algebraic properties of the underlying groups in order to have A accept a known session key. To eliminate the dependency from α the only possible choice is to set $X = g_{ID_B}$ so that $X^{\alpha} = d_{ID_B}$, however, the adversary would have to corrupt B also to succeed.

A more powerful adversary, with the possibility of corrupting B's session state in order to obtain the value of y, may succeed in breaking the protocol. The scenario wherein the adversary has additional information is not technically a KCI attack; such capabilities are considered in the security models of Canetti and Krawczyk [4] and Lamacchia *et al.* [7].

4.4 Unknown Key Share

The first Unknown Key Share (UKS) attack against a key agreement protocol was described by Diffie *et al.* [6] who showed how a dishonest entity E can setup an attack whereby an honest entity A finishes a protocol execution believing she shares a key with B, but B mistakenly believes the key is shared with E (while the later does not necessarily know the session key held by B).

It is possible to mount a UKS attack against the IDAK protocol:

1. Adversary $\mathcal{A}$ register identity E with the KGC and obtains the private key d_{ID_E};
2. After A negotiating a communication session with B, she sends message R_A to B;
3. $\mathcal{A}$ intercepts message R_A, and initiates a new communication session with B as the legitimate peer E by sending $R_E = g_{ID_E}^t$ for $t \in_R Z_q^*$, thus replacing message R_A;
4. On receipt of R_E, B cannot detect the difference between R_E and R_A because they are indistinguishable and repsonds with message R_B;
5. $\mathcal{A}$ relays R_B to A. As the result, A accepts believing that she is communicating with B (and indeed they calculate the same session key) while the latter peer also accepts but is convinced she is connected to E. In this case the adversary cannor compute the key shared by A, B but nevertheless tha attack is successful.

In most cases countermeasures are easily implemented to avoid this type of attacks, so it is not often a major concern in practice. In particular, a common solution is to add the peer identities in the calculation of the key as illustrated in Figure 1.

$$
\begin{array}{rl}
 & A : x \xleftarrow{R} Z_q^* \\
 & \quad R_A \leftarrow g_{ID_A}^x \\
A \rightarrow B : & R_A \\
 & B : y \xleftarrow{R} Z_q^* \\
 & \quad R_B \leftarrow g_{ID_B}^y \\
B \rightarrow A : & R_B \\
 & A : s_A \leftarrow \pi(A, B, R_A, R_B),\ s_B \leftarrow \pi(B, A, R_B, R_A) \\
 & \quad \sigma \leftarrow \hat{e}(g_{ID_B}^{s_B} \cdot R_B, d_{ID_A}^{(x+s_A)}) \\
 & \quad sk_{AB} \leftarrow H_1(A, B, R_A, R_B, \sigma) \\
 & B : s_A \leftarrow \pi(A, B, R_A, R_B),\ s_B \leftarrow \pi(B, A, R_B, R_A) \\
 & \quad \sigma \leftarrow \hat{e}(g_{ID_A}^{s_A} \cdot R_A, d_{ID_B}^{(y+s_B)}) \\
 & \quad sk_{BA} \leftarrow H_1(A, B, R_A, R_B, \sigma)
\end{array}
$$

Fig. 1. Protocol IDAK

5 A formal security proof of the IDAK protocol in the Extended Canetti-Krawczyk model

In this section we review the extended Canetti-Krawczyk (eCK) model of distributed computing of Lamacchia *et al*. [7] and then prove that the IDAK protocol is eCK-secure.

5.1 The eCK model

Parties and the adversary are modeled as probabilistic Turing machines. The network is totally under the control of the adversary who can delete, modify or inject messages that are exchanged by any two honest parties running the protocol.
Sessions. Instances of a protocol run by two any parties A and B are called *sessions*. A Session Identifier (SID) consists of the identities of the two participants and the messages transcripts they exchanged during the communication. For any two parties A and B, we denote by sid the SID of the session owned by A and by sid^* the SID of the session owned by its intended peer B.
Adversary. To capture all types of attacks resulting from ephemeral and long-term data compromise the adversary is allowed to ask the following queries

a) `EphemeralKeyReveal`(sid) to obtain *all* short-term secret information used by a party in session sid;
b) `PrivateKeyReveal`(ID) which returns long-term private keys of the principal with identity ID;
c) `SessionKeyReveal`(sid) for exposing the session key of session sid;
d) `Extract`(ID) allows the adversary to register a new identity ID and receive the private keying material relative to this identity.

In the eCK security experiment the adversary can ask two additional queries

a) `Test`(sid) when the adversary asks this query (only once) a random bit b is used to decide whether to return the real session key of sid or a random key;
c) `Guess`(b') when the adversary terminates it outputs bit b' as its guess of b, the query returns 1 if the guess is correct 0 otherwise.

Test sessions are classified as either "passive" or "active". In a passive session the adversary is limited only to observing the communication transcripts exchanged by two honest participants while in active sessions the adversary can also tamper with them (e.g. cancel or modify communication messages). As the result, in passive sessions two parties may complete matching sessions (i.e. the conversation of the initiator matches the conversation of the responder and both have accepted [2]); on the other hand, active sessions are those where matching sessions are not necessarily established. The adversary succeeds in the eCK experiment if the selected test session is *fresh* and `Guess` returns 1. Set out below is the definition of a *fresh* session.

Definition 1 (fresh session) *Let sid denote the SID of a completed session run by party A with a peer B. Let sid^* denote the SID of the matching session of sid owned by B, if it exists. Session sid is* fresh *if none of the following conditions hold:*

1. *the adversary issues a* `SessionKeyReveal`*(sid) query or a* `SessionKeyReveal`*(sid^*) query (if sid^* exists);*
2. *sid^* exists and the adversary makes either of the following queries: (a)* `PrivateKeyReveal`*(A) and* `EphemeralKeyReveal`*(sid) or (b)* `PrivateKeyReveal`*(B) and* `EphemeralKeyReveal`*(sid^*);*

3. *sid^* does not exist and the adversary makes either of the following queries:*
   (a) `PrivateKeyReveal`*(A) and* `EphemeralKeyReveal`*(sid) or*
   (b) `PrivateKeyReveal`*(B).*

In the eCK experiment the adversary is allowed to continue interacting with the parties even after issuing the test query with the restriction that the test session must remain fresh.

Theorem 1 (eCK-security) *A key agreement protocol is* eCK-secure *if the following conditions hold:*

1. *If two honest parties complete matching sessions then, except with negligible probability, they both compute the same session key;*
2. *No polynomially bounded adversary can distinguish the session key of a fresh session from a randomly chosen session key, with probability greater than 1/2 plus a negligible function (in the security parameter).*

5.2 Proof of eCK security of protocol IDAK

The following theorem proves that the two-pass IDAK key agreement protocol is secure in the eCK model under the Gap Bilinear Diffie Hellman and random oracle assumptions.

Theorem 2 (IDAK eCK-security) *If $\mathcal{H}, \mathcal{H}_1, \pi$ are random oracles, and the GBDH assumption holds in the group G (from hereon G will stand for G_1), then IDAK is an eCK-secure key agreement protocol. Concretely, there exists a GBDH solver S_D and a DLOG solver S_L in G such that*

$$\mathtt{Adv}^{eCK}_{IDAK}(E) \leq c_1 \cdot \mathtt{Adv}^{DLOG}(S_L) + c_2 \cdot \mathtt{Adv}^{GBDH}(S_D) + \epsilon$$

where ℓ is the security parameter, $s(\ell)$ the number of activated sessions, $N(\ell)$ the number of principals, $c_1 = c_1(\ell) = s(\ell)^2$, $c_2 = c_2(\ell) = s(\ell)N(\ell)$ and $\epsilon = \epsilon(\ell) = \mathrm{O}(s(\ell)^2/2^\ell)$.

Proof. Let E denote a polynomially bounded adversary in the security parameter ℓ. The adversary E defeats protocol IDAK with non-negligible probability if it succeeds in the experiment described in Section 5.1 with probability $1/2 + \epsilon(\ell)$, where $\epsilon(\ell)$ is non-negligible. Guessing the answer of the Test query succeeds with probability $1/2$. Since $\mathcal{H}_1$ is a random oracle, E can only distinguish a session key from a random string with probability significantly greater than $1/2$ if the adversary E succeeds in one of the following attacks:

A1: (Replicate) E forces a session to accept the same key of the test session even though the two sessions are distinct and non-matching and obtains the secret key from the target session (by asking a `SessionKeyReveal` query);

A2: (Forge) for two parties (say A, B) E computes $\sigma = sk_{AB} = sk_{BA}$ and queries $\mathcal{H}_1$ with (A, B, R_A, R_B, σ) .

For A1 to occur the adversary E must query the random oracle $\mathcal{H}_1$ with the same 5-tuple used to compute the key of the test session. The attack succeeds if the random oracle produces collisions, since distinct sessions have distinct 5-tuples such collisions occur with negligible probability $O(s(\ell)^2/2^\ell)$. Therefore, to succeed in the eCK experiment the adversary E must perform a forging attack (A2). To this end, we show that if an adversary E is able to win the eCK experiment then we may construct a GBDH solver in the underlying group G which uses E as a subroutine. Algorithm S simulates the eCK experiment in such a way that E's view is indistinguishable from the real one. We consider the two cases described below.

`Case 1)` (matching sessions) The adversary E is substantially passive (limiting itself only to eavesdropping the messages exchanged by any two parties) and chooses a test session with a matching session. Let `match` denote the event that the adversary chooses either one of the matching sessions sid or sid^* owned respectively by A, B as its test session (this event is conditioned on A2). The description of algorithm S follows:

1. S receives in input a GBDH challenge $(g, X = g^x, Y = g^y, Z = g^z)$ where $x, y, z \in_R Z_q^*$;
2. S establishes the identities ID_i and the long-term keys (d_{ID_i}, g_{ID_i}) of $N(\ell)$ principals by calling algorithm `Extract`;
3. S runs E as a subroutine answering its queries as follows: a) `send` queries are simulated as usual, except for sessions sid and sid^* defined above; in this case assuming that $\text{ID}_i \equiv A$ and $\text{ID}_j \equiv B$ (for some i, j) S selects the ephemeral keys $x, y \in_R Z_q^*$ for A, B (respectively) and sets the messages $R_A = X, R_B = Y$. The session keys sk_{AB}, sk_{AB} are set equal to a random value G so E's view of the protocol execution is the same as the real one because $sk_{AB}, sk_{AB}, X, Y \in G$ and $\mathcal{H}_1$ is a random oracle; b) all other queries are answered according to the protocol specification;
4. When E terminates, algorithm S also terminates and outputs the same bit b' as E (as the input to the `Guess` query).

We claim that if E succeeds in the eCK experiment then S can solve the GBDH challenge. Indeed, if event `match` occurs, to succeed in the eCK experiment E must query the random oracle $\mathcal{H}_1$ with the tuple (A, B, R_A, R_B, σ). To correctly compute σ, E must obtain x, y and to do so she can use any combination of queries that will maintain the test session fresh (recall that E cannot reveal both x, d_{ID_A} or y, d_{ID_B}). Therefore, E must ask `PrivateKeyReveal` queries of both A and B (to obtain d_{ID_A} and d_{ID_B}) and then the only way she can obtain x (or y) is by computing $DLOG_G(R_A)$ (or $DLOG_G(R_B)$). As a result, the probability that S succeeds is at least $\Pr[\texttt{match}|\text{A2}]/s(\ell)^2$ since the simulation is perfect until event `match` occurs. This implies that S can solve the discrete logarithm in G (i.e. algorithm S_L).

`Case 2)` The adversary E chooses a test session owned by A that does not have a matching session; we denote this event by `nomatch`. S receives in input a GBDH challenge $(g, X = g^x, Y = g^y, Z = g^z)$where $x, y, z \in_R Z_q^*$. To setup the simulation, S establishes the identities ID_i and the long-term keys (d_{ID_i}, g_{ID_i}) of $N(\ell) - 1$ honest principals except for a randomly chosen party A for which she sets $d_A = X$. Now S can correctly simulate all sessions (involving honest parties) invoked by E during the

eCK experiment except for those owned by A. When E activates A, S simulates E's queries as follows:

1. `Send` queries, E activates a session sid owned by A and a peer B who owns session sid^*. Regardless of A's role in the communication (i.e. as either initiator or responder), S selects $\hat{x} \in_R Z_q^*$, runs `Send`$(B, R_A = g^{\hat{x}}_{ID_A})$ and sets the session key of A to $sk_{AB} \in_R G$. The value of $\hat{x}$ and sk_{AB} are indistinguishable from the real values (in particular because H_1 is a random oracle). Note also that S is perfectly aware that B may be a fake party because she can handle any queries of the type `Extract`(B) (for $B \neq$A).
2. `EphemeralKeyReveal`(sid) queries, for the sid of a session owned by A the answer is $\hat{x}$ (generated for A by S, see above);
3. `PrivateKeyReveal`(A) queries, S aborts;
4. `SessionKeyReveal`(sid) queries, S returns the key sk_{AB} generated for sid (see above);
5. queries of the random oracles and `Test` queries are simulated in the standard way.

The simulator S described above may fail either during the simulation of `Send` queries (or if E asks the query `PrivateKeyReveal`(A) as seen above). Indeed, assuming A is the initiator (without loss of generality) and event `nomatch` has occurred, E can distinguish between the real and simulated worlds if she computes the session key sk_{BA} by querying $\mathcal{H}_1$ with the tuple $(A, B, R_A, R_B = g^{\hat{y}}_{ID_B}, \sigma)$ where $\sigma = \hat{e}(g^{s_A}_{ID_A} \cdot R_A, d^{(\hat{y}+s_B)}_{ID_B})$ and $\hat{y} \in_R Z_q^*$; by choosing sid as the test session and asking the query `SessionKeyReveal`(sid) E is able to determine that $sk_{AB} \neq sk_{BA}$ (this implies that session sid is not fresh and thus E cannot choose it as the test session). However, since event A2 has occurred, S can intercept E's queries of the random oracle $\mathcal{H}_1$ having as arguments the tuple (A, B, R_A, R_B, σ) and respond with sk_{AB} if DBDH$((X, g^{\hat{x}+s_A}_{ID_B}, g^{\hat{y}+s_B}_{ID_B}, \sigma)$=1. Therefore, the only way E can detect that it is running in a simulated experiment is by asking `PrivateKeyReveal` queries of both A and B (to obtain $\hat{x}$ and $\hat{y}$) and $DLOG_G(X)$; this event occurs with probability at least Pr[`nomatch`|A2]/$s(\ell)N(\ell)$ thus allowing S to solve the GBDH problem in G (i.e. algorithm S_D).

We note that in case A is the responder then E can choose sid as the test session but cannot ask a `PrivateKeyReveal`(A) query. Therefore again to detect the simulation E must ask a query of the type $DLOG_G(X)$).

6 Conclusions

The IDAK protocol exhibits an excellent design, interesting security features and high computational efficiency, which make it a good candidate for resource constrained devices.

In this report we have proven that the IDAK protocol is secure in the extended Canetti-Krawczyk model, the later is stronger than the Bellare-Rogaway model since it intrinsically accounts for KCI attacks and other types of possible threats.

As the result, the protocol can be used to secure communications betweens peers in insecure networks (e.g. the Internet) that are totally under control of powerful active adversaries.

References

1. M. Bellare and P. Rogaway. Entity authentication and key distribution. *In Proceedings of CRYPTO 1993*, LNCS 773:232–249, 1993.
2. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *In 1st Conference on Computer and Communications Security*, pages 62–73, 1993.
3. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. *Proceedings of Crypto01, Springer-Verlag, New York*, LNCS 2139:213–229, 2001.
4. R. Canetti and H. Krawczyk. Analysis of key exchange protocols and their use for building secure channels. *Advances in Cryptology-EUROCRYPT 2001*, LNCS 2045:453–474, 2001.
5. L. Chen and C. Kudla. Identity based authenticated key agreement protocols from pairings. *Cryptology ePrint Archive, Report 2002/184*, http://eprint.iacr.org/2002/184.pdf, 2002.
6. W. Diffie, P. van Oorschot, and M. Wiener. Authentication and Authenticated Key Exchange. *Designs, Codes and Cryptography*, 2:107–125, 1992.
7. B. L. et al. Stronger Security of Authenticated Key Exchange. *LNCS*, 4784:1–16, 2007.
8. M. A. Strangio. On the Resilience of Key Agreement Protocols to Key Compromise Impersonation. *Cryptology ePrint Archive, Report 2006/252*, http://eprint.iacr.org/2006/252.pdf, 2006.
9. Y. Wang. Efficient Identity-Based and Authenticated Key Agreement Protocol. *Cryptology ePrint Archive, Report 2005/108*, http://eprint.iacr.org/2005/108.pdf, 2005.
10. Y. Wang. Efficient Identity-Based and Authenticated Key Agreement Protocol. *CoRR*, abs/1207.5438, 2012.