Efficient Identity-Based and Authenticated Key Agreement Protocol

Yongge Wang
UNC Charlotte (USA), yonwang@uncc.edu

Abstract

Several identity based and authenticated key agreement protocols have been proposed in recent years and all of them have been shown to be non-secure. It remains an open question to design secure identity based and authenticated key agreement protocols. In this paper, we propose an efficient identity-based and authenticated key agreement protocol IDAK using Weil/Tate pairing. A security model for identity based key agreement protocol is established and the security properties of IDAK are proved in this model with random oracle. In particular, it is shown that the IDAK protocol possesses all characteristics that a secure key agreement should have.

1 Introduction

Key establishment protocols are one of the most important cryptographic primitives that have been used in our society. The first unauthenticated key agreement protocol based on asymmetric cryptographic techniques were proposed by Diffie and Hellman [10]. Since this seminal result, many authenticated key agreement protocols have been proposed and the security properties of key agreement protocols have been extensively studied. In order to implement these authenticated key agreement protocols, one needs to get the corresponding party's authenticated public key. For example, in order for Alice and Bob to execute the NIST recommended MQV key agreement protocol [15, 19], Alice needs to get an authenticated public key g^b for Bob and Bob needs to get an authenticated public key g^a for Alice first, where a and b are Alice and Bob's private keys respectively. One potential approach for implementing these schemes is to deploy a public key infrastructure (PKI) system, which has proven to be difficult. Thus it is preferred to design easy to deploy authenticated key agreement systems. Identity based key agreement system is such an example.

In 1984, Shamir [25] proposed identity based cryptosystems where user's identities (such as email address, phone numbers, office locations, etc.) could be used as the public keys. Several identity based key agreement protocols (see, e.g., [7, 12, 16, 20, 23, 24, 26, 29, 31]) have been proposed since then. Most of them are not practical or do not have all required security properties. Joux [13] proposed a one-round tripartite non-identity based key agreement protocol using Weil pairing. Then a feasible identity based encryption scheme IBE [6] based on Weil or Tate paring was designed.

Based on Weil and Tate pairing techniques, Smart [29], Chen-Kudla [7], Scott [24], Shim [26], and McCullagh-Barreto [16] designed identity based and authenticated key agreement protocols. However, none of these protocols is secure (details could be found in Section §10 of this paper).

For example, all these protocols are insecure against key revealing attacks and some of them do not have perfect forward secrecy property for session keys if both parties long term private keys are corrupted. Indeed, several of these protocols were "PROVED" to be secure in the Bellare-Rogaway security model for key agreement protocols and the proofs were found to be flawed later. For example, Chen and Kudla [7] proved that their protocol is secure in the Bellare-Rogaway [3] secure key agreement model. However, Cheng et al. [8] pointed out that the proof in [7] is flawed and their protocol is not secure against the key revealing attacks (the fundamental component in Bellare-Rogaway model).

Thus it remains to be an open problem to design efficient secure identity based and authenticated key agreement protocols. In this paper, we propose an efficient identity based and authenticated key agreement protocol achieving all security properties that an authenticated key agreement protocol should have. In addition, our protocol is designed for efficient implementation with pre-computations. Without pre-computation, our protocol is as efficient as (or more efficient than) existing identity based key agreement protocols.

The advantage of identity based key agreement is that non-PKI system is required. The only prerequisite for executing identity based key agreement protocols is the deployment of authenticated system-wide parameters. Thus, it is easy to implement these protocols in relatively closed environments such as government organizations and commercial entities.

There is an extensive literature on the security of key agreement protocols. Bellare and Rogaway [3] provided formalizations for certain symmetric-key cases. They introduced the model of an adversary in control over all communications, modelled session key revealing attacks, and suggested that the session key should be strongly secure in the sense of semantic security. Fiat and Shamir [11] introduced the random oracle model to analyze the security of cryptographic protocols. The random oracle model has been further enhanced by Bellare and Rogaway [2]. We will show that in random oracle model, our IDAK is a secure authenticated key agreement protocol in a security model based on Bellare-Rogaway model [3]. In a summary, our contributions of this paper include: (1) An efficient identity based and authenticated key agreement protocol. Without pre-computation, our protocol is at least as efficient as existing (including the non-secure ones) identity based key agreement protocols. With pre-computation, our protocol is very efficient and is suitable for resource constraint devices. (2) A security model for identity based key agreement protocols which is used to prove security properties for our IDAK protocol. (3) Practical considerations and application domain discussions of identity based key agreement protocols.

The remainder of this paper is organized as follows. In §2 we briefly describe bilinear maps, bilinear Diffie-Hellman problem, and its variants. In §3, we describe our identity based and authenticated key agreement protocol IDAK. §4 describes a security model for identity based key agreement. In section §5, we prove the security of IDAK key agreement protocol. In sections §6 and §7, we discuss key compromise impersonation resilience and perfect forward secrecy properties of IDAK key agreement protocol, and in section §8, we describe IDAK key agreement protocol with key confirmation and we prove its security. In section §9, we discuss implementation issues and applications. We conclude our paper with a discussion on related protocols and their insecurity in §10.

2 Bilinear maps and the bilinear Diffie-Hellman assumptions

2.1 Bilinear maps

In the following, we briefly describe the bilinear maps and bilinear map groups. The details could be found in Joux [13] and Boneh and Franklin [6].

1. G and G_1 are two (multiplicative) cyclic groups of prime order q.
2. g is a generator of G.
3. $\hat{e} : G \times G \to G_1$ is a bilinear map.

A bilinear map is a map $\hat{e} : G \times G \to G_1$ with the following properties:

1. bilinear: for all $g_1, g_2 \in G$, and $x, y \in Z$, we have $\hat{e}(g_1^x, g_2^y) = \hat{e}(g_1, g_2)^{xy}$.
2. non-degenerate: $\hat{e}(g, g) \neq 1$.

We say that G is a bilinear group if the group action in G can be computed efficiently and there exists a group G_1 and an efficiently computable bilinear map $\hat{e} : G \times G \to G_1$ as above. Concrete examples of bilinear groups are given in [13, 6]. For example, let G be a subgroup of the additive group of the points of an elliptic curve $E_{a,b}/F_p$ and G_1 be a subgroup of the multiplicative group of a finite field $F^*_{p^2}$. Then the Weil pairing (respectively, Tate pairing) could be used to construct bilinear maps between these two groups. For convenience, throughout the paper, we view both G and G_1 as multiplicative groups though the concrete implementation of G could be additive elliptic curve groups.

2.2 Complexity assumptions

Throughout the paper *efficient* means probabilistic polynomial-time, *negligible* refers to a function ε_k which is smaller than $1/k^c$ for all $c > 0$ and sufficiently large k, and *overwhelming* refers to a function $1 - \varepsilon_k$ for some negligible ε_k. Consequently, a function δ_k is *non-negligible* if there exists a constant c and there are infinitely many k such that $\delta_k > 1/k^c$. We first formally define the notion of a bilinear group family and computational indistinguishable distributions (some of our terminologies are adapted from Boneh [5]).

Bilinear group families A *bilinear group family* $\mathcal{G}$ is a set $\mathcal{G} = \{G_\rho\}$ of bilinear groups $G_\rho = \langle G, G_1, \hat{e}\rangle$ where ρ ranges over an infinite index set, G and G_1 are two groups of prime order q_ρ, and $\hat{e} : G \times G \to G_1$ is a bilinear map. We denote by $|\rho|$ the length of the binary representation of ρ. We assume that group and bilinear operations in $G_\rho = \langle G, G_1, \hat{e}\rangle$ are efficient in $|\rho|$. Unless specified otherwise, we will abuse our notations by using q as the group order instead of q_ρ in the remaining part of this paper.

Instance generator An *Instance Generator, $\mathcal{IG}$,* for a bilinear group family $\mathcal{G}$ is a randomized algorithm that given an integer k (in unary, that is, 1^k), runs in polynomial-time in k and outputs some random index ρ for $G_\rho = \langle G, G_1, \hat{e}\rangle$, and a generator g of G, where G and G_1 are groups of prime order q. Note that for each k, the Instance Generator induces a distribution on the set of indices ρ.

The following Bilinear Diffie-Hellman Assumption (BDH) has been used by Boneh and Franklin [6] to show security of their identity-based encryption scheme.

Bilinear Diffie-Hellman Problem Let $\mathcal{G} = \{G_\rho\}$ be a bilinear group family and g be a generator for G, where $G_\rho = \langle G, G_1, \hat{e} \rangle$. The BDH problem in $\mathcal{G}$ is as follows: given $\langle g, g^x, g^y, g^z \rangle$ for some $x, y, z \in Z_q^*$, compute $\hat{e}(g,g)^{xyz} \in G_1$. A CBDH algorithm $\mathcal{C}$ for $\mathcal{G}$ is a probabilistic polynomial-time algorithm that can compute the function $\mathrm{BDH}_g(g^x, g^y, g^z) = \hat{e}(g,g)^{xyz}$ in G_ρ with a non-negligible probability. That is, for some fixed c we have

$$\Pr\left[\mathcal{C}(\rho, g, g^x, g^y, g^z) = \hat{e}(g,g)^{xyz}\right] \geq \frac{1}{k^c} \tag{1}$$

where the probability is over the random choices of x, y, z in Z_q^*, the index ρ, the random choice of $g \in G$, and the random bits of $\mathcal{A}$.

CBDH Assumption. The bilinear group family $\mathcal{G} = \{G_\rho\}$ *satisfies* the CBDH-Assumption if there is no CBDH algorithm for $\mathcal{G}$. A perfect-CBDH algorithm $\mathcal{C}$ for $\mathcal{G}$ is a probabilistic polynomial-time algorithm that can compute the function $\mathrm{BDH}_g(g^x, g^y, g^z) = \hat{e}(g,g)^{xyz}$ in G_ρ with overwhelming probability. $\mathcal{G}$ *satisfies* the perfect-CBDH-Assumption if there is no perfect-CBDH algorithm for $\mathcal{G}$.

Theorem 2.1 *A bilinear group family $\mathcal{G}$ satisfies the CBDH-Assumption if and only if it satisfies the perfect-CBDH-Assumption.*

Proof. The fact that the CBDH-Assumption implies the perfect-CBDH-Assumption is trivial. The converse is proved by the self-random-reduction technique (see [4, 17]). Let $\mathcal{O}$ be a CBDH oracle. That is, there exists a $c > 0$ such that (1) holds with $\mathcal{C}$ replaced with $\mathcal{O}$. We construct a perfect-CBDH algorithm $\mathcal{C}$ which makes use of the oracle $\mathcal{O}$. Given $g, g^x, g^y, g^z \in G$, algorithm $\mathcal{C}$ must compute $\hat{e}(g,g)^{xyz}$ with overwhelming probability. Consider the following algorithm: select $a, b, c \in_R Z_q$ and output

$$I_{x,y,z,a,b,c} = \mathcal{O}(g, g^{x+a}, g^{y+b}, g^{z+c}) \cdot \hat{e}(g,g)^{-(abz+abc+ayz+ayc+xbz+xbc+xyc)}.$$

One can easily verify that if $\mathcal{O}(\rho, g, g^{x+a}, g^{y+b}, g^{z+c}) = \hat{e}(g,g)^{(x+a)(y+b)(z+c)}$, then $I_{x,y,z,a,b,c} = \hat{e}(g,g)^{xyz}$. Consequently, standard amplification techniques can be used to construct the algorithm $\mathcal{C}$. The details are omitted. $\square$

Consider Joux's tripartite key agreement protocol [13]: Alice, Bob, and Carol fix a bilinear group $\langle G, G_1, \hat{e} \rangle$. They select $x, y, z \in_R Z_q^*$ and exchange g^x, g^y, and g^z. Their shared secret is $\hat{e}(g,g)^{xyz}$. To *totally break* the protocol a passive eavesdropper, Eve, must compute the BDH function: $\mathrm{BDH}_g(g^x, g^y, g^z) = \hat{e}(g,g)^{xyz}$.

CBDH-Assumption by itself is not sufficient to prove that Joux's protocol is useful for practical cryptographic purposes. Even though Eve may be unable to recover the entire secret, she may still be able to predict quite a few bits (less than $c \log k$ bits for some constant c; Otherwise, CBDH assumption is violated) of information for $\hat{e}(g,g)^{xyz}$ with some confidence. If $\hat{e}(g,g)^{xyx}$ is to be the basis of a shared secret key, one must bound the amount of information Eve is able to deduce about it, given g^x, g^y, and g^z. This is formally captured by the, much stronger, Decisional Bilinear Diffie-Hellman assumption (DBDH-Assumption)

Definition 2.2 *Let $\{\mathcal{X}_\rho\}$ and $\{\mathcal{Y}_\rho\}$ be two ensembles of probability distributions, where for each ρ both $\mathcal{X}_\rho$ and $\mathcal{Y}_\rho$ are defined over the same domain. We say that the two ensembles are* computationally indistinguishable *if for any probabilistic polynomial-time algorithm $\mathcal{D}$, and any $c > 0$ we have*

$$\left|\Pr\left[\mathcal{D}\left(\mathcal{X}_\rho\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathcal{Y}_\rho\right) = 1\right]\right| < \frac{1}{k^c}$$

for all sufficiently large k, where the probability is taken over all $\mathcal{X}_\rho$, $\mathcal{Y}_\rho$, and internal coin tosses of $\mathcal{D}$.

In the remainder of the paper, we will say in short that the two distributions $\mathcal{X}_\rho$ and $\mathcal{Y}_\rho$ are computationally indistinguishable.

Let $\mathcal{G} = \{G_\rho\}$ be a bilinear group family. We consider the following two ensembles of distributions:

- $\{\mathcal{X}_\rho\}$ of random tuples $\langle \rho, g, g^x, g^y, g^z, \hat{e}(g,g)^t\rangle$, where g is a random generator of G ($G_\rho = \langle G, G_1, \hat{e}\rangle$) and x, y, z, t are randomly chosen from Z_q.
- $\{\mathcal{Y}_\rho\}$ of tuples $\langle \rho, g, g^x, g^y, g^z, \hat{e}(g,g)^{xyz}\rangle$, where g is a random generator of G and x, y, z are randomly chosen from Z_q.

An algorithm that solves the Bilinear Diffie-Hellman decision problem is a polynomial time probabilistic algorithm that can effectively distinguish these two distributions. That is, given a tuple coming from one of the two distributions, it should output 0 or 1, and there should be a non-negligible difference between (a) the probability that it outputs a 1 given an input from $\{\mathcal{X}_\rho\}$, and (b) the probability that it outputs a 1 given an input from $\{\mathcal{Y}_\rho\}$. The bilinear group family $\mathcal{G}$ *satisfies the DBDH-Assumption* if the two distributions are computationally indistinguishable.

Remark. The DBDH-Assumption is implied by a slightly weaker assumption: *perfect*-DBDH-Assumption. A perfect-DBDH statistical test for $\mathcal{G}$ distinguishes the inputs from the above $\{\mathcal{X}_\rho\}$ and $\{\mathcal{Y}_\rho\}$ with overwhelming probability. The bilinear group family $\mathcal{G}$ *satisfies the perfect-DBDH-Assumption* if there is no such probabilistic polynomial-time statistical test.

3 The scheme IDAK

In this section, we describe our identity-based and authenticated key agreement scheme IDAK. Let k be the security parameter given to the setup algorithm and $\mathcal{IG}$ be a bilinear group parameter generator. We present the scheme by describing the three algorithms: **Setup**, **Extract**, and **Exchange**.

Setup: For the input $k \in Z^+$, the algorithm proceeds as follows:

1. Run $\mathcal{IG}$ on k to generate a bilinear group $G_\rho = \{G, G_1, \hat{e}\}$ and the prime order q of the two groups G and G_1. Let h be the co-factor of the group order q for G (that is, the order of the basing elliptic curve group for G is qh). If G is not an elliptic curve group, then h could be defined similarly. Choose a random generator $g \in G$.
2. Pick a random master secret $\alpha \in Z_q^*$.
3. Choose cryptographic hash functions $H : \{0,1\}^* \rightarrow G$ and $\pi : G \times G \rightarrow Z_q^*$. In the security analysis, we view H and π as random oracles.

The system parameter is $\langle q, h, g, G, G_1, \hat{e}, H, \pi\rangle$ and the master secret key is α.

Extract: For a given identification string $\mathrm{ID} \in \{0,1\}^*$, the algorithm computes a generator $g_{\mathrm{ID}} = H(\mathrm{ID}) \in G$, and sets the private key $d_{\mathrm{ID}} = g_{\mathrm{ID}}^\alpha$ where α is the master secret key.

Exchange: For two participants Alice and Bob whose identification strings are ID_A and ID_B respectively, the algorithm proceeds as follows.

1. Alice selects $x \in_R Z_q^*$, computes $R_A = g_{\mathrm{ID}_A}^x$, and sends it to Bob.
2. Bob selects $y \in_R Z_q^*$, computes $R_B = g_{\mathrm{ID}_B}^y$, and sends it to Alice.
3. Alice computes $s_A = \pi(R_A, R_B)$, $s_B = \pi(R_B, R_A)$, and the shared secret sk_{AB} as
$$\hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^{(x+s_A)(y+s_B)h\alpha} = \hat{e}\left(g_{\mathrm{ID}_B}^{s_B} \cdot R_B, g_{\mathrm{ID}_A}^{(x+s_A)h\alpha}\right).$$
4. Bob computes $s_A = \pi(R_A, R_B)$, $s_B = \pi(R_B, R_A)$, and the shared secret sk_{BA} as
$$\hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^{(x+s_A)(y+s_B)h\alpha} = \hat{e}\left(g_{\mathrm{ID}_A}^{s_A} \cdot R_A, g_{\mathrm{ID}_B}^{(y+s_B)h\alpha}\right).$$

A simple analysis shows that in the IDAK protocol, each party needs to compute three (or two if π takes values from $[1, 2^{\lceil \log q/2 \rceil}]$) exponentiations and one pairing. However, if each party could do some pre-computation, then only one pairing and two (or one) exponentiation are required during the key agreement session. In particular, if Alice selects the random value x and computes the value of g_A^x off-line before the key agreement session, then she only needs to carry out the computation of $g_B^{s_B} \cdot R_B$, $g_A^x \cdot g_A^{s_A}$, and $\hat{e}\left(g_B^{s_B} \cdot R_B, g_A^{(x+s_A)h\alpha}\right)$ during the key agreement session. This improves the performance of the protocol implementation.

In the following sections, we describe a security model for identity based and authenticated key agreement protocol. Our model is based on Bellare and Rogaway [3] secure key agreement model. We then show that our IDAK protocol is secure in this model with random oracle plus DBDH-Assumption. In particular, our protocol achieves perfect forward secrecy property and security against key revealing attacks. In a summary, our protocol is more efficient compared with existing protocols and has better security properties.

We conclude this section with a theorem which says that the shared secret established by the IDAK key agreement protocol is computationally indistinguishable from a random value. In another word, if we assume that the adversary is passive and forward all messages exactly in the way it receives, then the agreed keys by entities achieve semantic security.

Theorem 3.1 *Let $\mathcal{G} = \{G_\rho\}$ be a bilinear group family, $G_\rho = \langle G, G_1, \hat{e}\rangle$, and g_1, g_2 be random generators of G. Assume that DBDH-Assumption holds for $\mathcal{G}$. Then the distributions $\langle g_1, g_2, g_1^x, g_2^y, \hat{e}(g_1, g_2)^{(x+\pi(g_1^x, g_2^y))(y+\pi(g_2^y, g_1^x))h\alpha}\rangle$ and $\langle g_1, g_2, g_1^x, g_2^y, \hat{e}(g_1, g_2)^{zh}\rangle$ are computationally indistinguishable, where α, x, y, z are selected from Z_q^* uniformly.*

Before we give a proof for Theorem 3.1, we first prove two lemmas that will be used in the proof of the Theorem.

Lemma 3.2 *(Naor and Reingold [17]) Let $\mathcal{G} = \{G_\rho\}$ be a bilinear group family, $G_\rho = \langle G, G_1, \hat{e}\rangle$, m be a constant, g be a random generator of G, and $\hat{g} = \hat{e}(g, g)$. Assume that the DBDH-Assumption holds for G_ρ. Then the two distributions $\langle \mathcal{R}, (\hat{g}^{x_i y_j z_l} : i, j, l \leq m)\rangle$ and $\langle \mathcal{R}, (\hat{g}^{u_{ijl}} : i, j, l \leq m)\rangle$ are computationally indistinguishable. Here $\mathcal{R}$ denotes the tuple $(g, (g^{x_i}, g^{y_j}, g^{z_l} : i, j, l \leq m))$ and x_i, y_j, z_l, u_{ijl} are randomly chosen from Z_q.*

Proof. Using a random reduction, Naor and Reingold [17, Lemma 4.4] (see also Shoup [28, §5.3.2] showed that the two distributions $\langle \mathcal{R}, (g^{x_i y_j} : i, j \leq m)\rangle$ and $\langle \mathcal{R}, (g^{u_{ij}} : i, j \leq m)\rangle$ are computationally indistinguishable. The proof can be directly modified to obtain a proof for this Lemma. The details are omitted. □

Lemma 3.3 *Let $\mathcal{G} = \{G_\rho\}$ be a bilinear group family, $G_\rho = \langle G, G_1, \hat{e}\rangle$, g be a random generator of G, $\hat{g} = \hat{e}(g,g)$, and f_1 and f_2 be two polynomial-time computable functions. If the two distributions $\mathcal{X}_1 = \langle \mathcal{R}, \hat{g}^{f_1(\mathbf{x})}, \hat{g}^{f_2(\mathbf{x})}\rangle$ and $\mathcal{Y}_1 = \langle \mathcal{R}, \hat{g}^{z_1}, \hat{g}^{z_2}\rangle$ are computationally indistinguishable, then the two distributions $\mathcal{X}_2 = \langle \mathcal{R}_1, \hat{g}^{f_1(\mathbf{x})+f_2(\mathbf{x})}\rangle$ and $\mathcal{Y}_2 = \langle \mathcal{R}_2, \hat{g}^{z}\rangle$ are computationally indistinguishable, where $\mathcal{R} = (g, (g^{x_i} : 1 \le i \le m))$, $\mathbf{x} = (x_1, \ldots, x_m)$, and x_i, z_1, z_2, z are randomly chosen from Z_q.*

Proof. For a contradiction, assume that there is a probabilistic polynomial-time algorithm $\mathcal{D}$ that distinguishes the two distributions $\mathcal{X}_2$ and $\mathcal{Y}_2$ with non-negligible probability δ_k. In the following we construct a probabilistic polynomial-time algorithm $\mathcal{D}'$ to distinguish the two distributions $\mathcal{X}_1$ and $\mathcal{Y}_1$. $\mathcal{D}'$ is defined by letting $\mathcal{D}'(\mathcal{R}, X, Y) = \mathcal{D}(\mathcal{R}, X\cdot Y)$ for all $\mathcal{R}$, and $X, Y \in G_1$. By this definition, we have $\Pr[\mathcal{D}'_r(\mathcal{X}_1) = 1|\mathcal{R}, r] = \Pr[\mathcal{D}_r(\mathcal{X}_2) = 1|\mathcal{R}, r]$, for any fixed internal coin tosses r of $\mathcal{D}$ and $\mathcal{D}'$.

Let $D^{\mathcal{D}}_{\mathcal{R},r} = \{X : \mathcal{D}_r(\mathcal{R}, X) = 1\}$ and $D^{\mathcal{D}'}_{\mathcal{R},r} = \{(X,Y) : \mathcal{D}'_r(\mathcal{R}, X, Y) = 1\}$. By definition of $\mathcal{D}'$, we have $D^{\mathcal{D}'}_{\mathcal{R},r} = \{(X,Y) : X\cdot Y \in D^{\mathcal{D}}_{\mathcal{R},r}\}$. It follows that $|D^{\mathcal{D}'}_{\mathcal{R},r}| = q|D^{\mathcal{D}}_{\mathcal{R},r}|$ and $\Pr[\mathcal{D}'_r(\mathcal{Y}_1) = 1|\mathcal{R}, r] = |D^{\mathcal{D}'}_{\mathcal{R},r}|/q^2 = |D^{\mathcal{D}}_{\mathcal{R},r}|/q = \Pr[\mathcal{D}_r(\mathcal{Y}_2) = 1|\mathcal{R}, r]$. Thus we have

$$
\begin{aligned}
& |\Pr[\mathcal{D}'(\mathcal{X}_1) = 1] - \Pr[\mathcal{D}'(\mathcal{Y}_1) = 1]| \\
= \ & \left|\textstyle\sum_{\mathcal{R},r} \Pr[\mathcal{R}, r]\cdot(\Pr[\mathcal{D}'_r(\mathcal{X}_1) = 1|\mathcal{R}, r] - \Pr[\mathcal{D}'_r(\mathcal{Y}_1) = 1|\mathcal{R}, r])\right| \\
= \ & \left|\textstyle\sum_{\mathcal{R},r} \Pr[\mathcal{R}, r]\cdot(\Pr[\mathcal{D}_r(\mathcal{X}_2) = 1|\mathcal{R}, r] - \Pr[\mathcal{D}_r(\mathcal{Y}_2) = 1|\mathcal{R}, r])\right| \\
= \ & |\Pr[\mathcal{D}(\mathcal{X}_2) = 1] - \Pr[\mathcal{D}(\mathcal{Y}_2) = 1]| \\
> \ & \delta_k.
\end{aligned}
$$

Hence, $\mathcal{D}'$ distinguishes the distributions $\mathcal{X}_1$ and $\mathcal{Y}_1$ with non-negligible probability δ_k. This contradicts the assumption of the Lemma. □

Proof of Theorem 3.1 Let $\hat{g} = \hat{e}(g,g)$.By Lemma 3.2, the two distributions

$$
\begin{aligned}
&\mathcal{X} = \langle g, g^\alpha, g^x, g^y, \hat{g}^{h\alpha xy}, \hat{g}^{h\alpha x\pi(g^y,g^x)}, \hat{g}^{h\alpha y\pi(g^x,g^y)}, \hat{g}^{h\alpha\pi(g^x,g^y)\pi(g^y,g^x)}\rangle \quad \text{and} \\
&\mathcal{Y} = \langle g, g^\alpha, g^x, g^y, \hat{g}^{hz'_1}, \hat{g}^{hz'_2\pi(,g^y,g^x)}, \hat{g}^{hz'_3\pi(g^x,g^y)}, \hat{g}^{hz'_4\pi(g^x,g^y)\pi(g^y,g^x)}\rangle
\end{aligned}
$$

are computationally indistinguishable assuming that DBDH-Assumption holds for $\mathcal{G}$, where g is a random generator of G_ρ and $\alpha, x, y, z'_1, z'_2, z'_3, z'_4 \in_R Z_q$. Since π is a fixed function from G to Z^*_q and q is a prime, it is straightforward to verify that for any $\alpha, x, y \in Z_q$, $\hat{g}^{hz'_2\pi(g^y,g^x)}$, $\hat{g}^{hz'_3\pi(g^x,g^y)}$, and $\hat{g}^{hz'_4\pi(g^x,g^y)\pi(g^y,g^x)}$ are uniformly (and independently of each other) distributed over G. It follows that the distribution

$$\mathcal{Z} = \langle g, g^\alpha, g^x, g^y, \hat{g}^{hz_1}, \hat{g}^{hz_2}, \hat{g}^{hz_3}, \hat{g}^{hz_4})\rangle$$

is computationally indistinguishable from the distribution $\mathcal{Y}$, where $z_1, z_2, z_3, z_4 \in_R Z_q$. Thus $\mathcal{X}$ and $\mathcal{Z}$ are computationally indistinguishable. The Theorem now follows from Lemma 3.3. □

4 The security model

Our security model is based on Bellare and Rogaway [3] security models for key agreement protocols with several modifications. In our model, we assume that we have at most $m \le \text{poly}(k)$

protocol participants (principals): $\mathrm{ID}_1, \ldots, \mathrm{ID}_m$, where k is the security parameter. The protocol determines how principles behave in response to input signals from their environment. Each principle may execute the protocol multiple times with the same or different partners. This is modelled by allowing each principle to have different instances that execute the protocol. An oracle $\Pi^s_{i,j}$ models the behavior of the principle ID_i carrying out a protocol session in the belief that it is communicating with the principle ID_j for the sth time. One given instance is used only for one time. Each $\Pi^s_{i,j}$ maintains a variable *view* (or *transcript*) consisting of the protocol run transcripts so far.

The adversary is modelled 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 the principles via oracle accesses to $\Pi^s_{i,j}$. The adversary is allowed to execute any of the following queries:

- $\mathbf{Extract}(\mathrm{ID})$. This allows the adversary to get the long term private key for a new principle whose identity string is ID.
- $\mathbf{Send}(\Pi^s_{i,j}, X)$. This sends message X to the oracle $\Pi^s_{i,j}$. The output of $\Pi^s_{i,j}$ is given to the adversary. The adversary can ask the principle ID_i to initiate a session with ID_j by a query $\mathbf{Send}(\Pi^s_{i,j}, \lambda)$ where λ is the empty string.
- $\mathbf{Reveal}(\Pi^s_{i,j})$. This asks the oracle to reveal whatever session key it currently holds.
- $\mathbf{Corrupt}(i)$. This asks ID_i to reveal the long term private key d_{ID_i}.

The difference between the queries **Extract** and **Corrupt** is that the adversary can use **Extract** to get the private key for an identity string of her choice while **Corrupt** can only be used to get the private key of existing principles.

Let Π^s_{ij} be an initiator oracle (that is, it has received a λ message at the beginning) and $\Pi^{s'}_{ji}$ be a responder oracle. If every message that Π^s_{ij} sends out is subsequently delivered to $\Pi^{s'}_{ji}$, with the response to this message being returned to Π^s_{ij} as the next message on its transcript, then we say the oracle $\Pi^{s'}_{ji}$ matches Π^s_{ij}. Similarly, if every message that $\Pi^{s'}_{ji}$ receives was previously generated by Π^s_{ij}, and each message that $\Pi^{s'}_{ji}$ sends out is subsequently delivered to Π^s_{ij}, with the response to this message being returned to $\Pi^{s'}_{ji}$ as the next message on its transcript, then we say the oracle Π^s_{ij} matches $\Pi^{s'}_{ji}$. The details for an exact definition of matching oracles could be found in [2].

For the definition of matching oracles, the reader should be aware the following scenarios: Even though the oracle Π^s_{ij} thinks that its matching oracle is $\Pi^{s'}_{ji}$, the real matching oracle for Π^s_{ij} could be $\Pi^{t'}_{ji}$. For example, if Π^s_{ij} sends a message X to $\Pi^{s'}_{ji}$ and $\Pi^{s'}_{ji}$ replies with Y. The adversary decides not to forward the message Y to Π^s_{ij}. Instead, the adversary sends the message X to initiate another oracle $\Pi^{t'}_{ji}$ and ID_i does not know the existence of this new oracle $\Pi^{t'}_{ji}$. The oracle $\Pi^{t'}_{ji}$ replies with Y' and the adversary forwards this Y' to Π^s_{ij} as the responding message for X. In this case, the transcript of Π^s_{ij} matches the transcript of $\Pi^{t'}_{ji}$. Thus we consider Π^s_{ij} and $\Pi^{t'}_{ji}$ as matching oracles. In another word, the matching oracles are mainly based the message transcripts.

In order to define the notion of a secure session key exchange, the adversary is given an additional experiment. That is, in addition to the above regular queries, the adversary can choose, at any time during its run, a $\mathbf{Test}(\Pi^s_{i,j})$ query to a completed oracle $\Pi^s_{i,j}$ with the following properties:

- The adversary has never issued, at any time during its run, the query $\mathbf{Extract}(\mathrm{ID}_i)$ or $\mathbf{Extract}(\mathrm{ID}_j)$.
- The adversary has never issued, at any time during its run, the query $\mathbf{Corrupt}(i)$ or $\mathbf{Corrupt}(j)$.
- The adversary has never issued, at any time during its run, the query $\mathbf{Reveal}(\Pi_{i,j}^s)$.
- The adversary has never issued, at any time during its run, the query $\mathbf{Reveal}(\Pi_{j,i}^{s'})$ if the matching oracle $\Pi_{j,i}^{s'}$ for $\Pi_{i,j}^s$ exists (note that such an oracle may not exist if the adversary is impersonating the ID_j to the oracle $\Pi_{i,j}^s$). The value of s may be different from the value of s' since the adversary may run fake sessions to impersonate any principles without victims' knowledge.

Let $sk_{i,j}^s$ be the value of the session key held by the oracle $\Pi_{i,j}^s$ that has been established between ID_i and ID_j. The oracle $\Pi_{i,j}^s$ tosses a coin $b \leftarrow_R \{0,1\}$. If $b = 1$, the adversary is given $sk_{i,j}^s$. Otherwise, the adversary is given a value r randomly chosen from the probability distribution of keys generated by the protocol. In the end, the attacker outputs a bit b'. The advantage that the adversary has for the above guess is defined as

$$\mathrm{Adv}^{\mathcal{A}}(k) = \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

Now we are ready to give the exact definition for a secure key agreement protocol.

Definition 4.1 *A key agreement protocol Π is secure if the following properties are satisfied for any adversary:*

1. *If two uncorrupted oracles Π_{ij}^s and $\Pi_{ji}^{s'}$ have matching conversations (e.g., the adversary is passive) and both of them are complete according to the protocol Π, then both oracles will always accept and hold the same session key which is uniformly distributed over the key space.*
2. $\mathrm{Adv}^{\mathcal{A}}(k)$ *is negligible.*

In the following, we briefly discuss the attributes that a secure key agreement protocol in the above model achieves.

- **Known session keys**. The adversary may use **Reveal**$(\Pi_{i,j}^{s'})$ query before or after the query **Test**$(\Pi_{i,j}^s)$. Thus in a secure key agreement model, the adversary learns zero information about a fresh key for session s even if she has learnt keys for other sessions s'.
- **Impersonation attack**. If the adversary impersonates ID_j to ID_i, then she still learns zero information about the session key that the oracle Π_{ij}^s holds for this impersonated ID_j since there is no matching oracle for Π_{ij}^s in this scenario. Thus $\mathcal{A}$ can use **Test** query to test this session key that Π_{ij}^s holds.
- **Unknown key share**. If ID_i establishes a session key with ID_l though he believes that he is talking to ID_j, then there is an oracle Π_{ij}^s that holds this session key sk_{ij}. At the same time, there is an oracle $\Pi_{li'}^{s'}$ that holds this session key sk_{ij}, for some i' (normally $i' = i$).

During an unknown key share attack, the user ID_j may not know this session key. Since Π_{ij}^{s} and $\Pi_{li'}^{s'}$ are not matching oracles, the adversary can make the query $\mathbf{Reveal}(\Pi_{li'}^{s'})$ to learn this session key before the query $\mathbf{Test}(\Pi_{ij}^{s})$. Thus the adversary will succeed for this **Test** query challenge if the unknown key share attack is possible.

- **Key compromise impersonation resilience**. If the entity A's long term private key is compromised, then the adversary could impersonate A to others, but it should not be able to impersonate others to A. Similar to other security models [3] for key agreement protocols, our model does not capture this property. However, we will give a separate proof that the IDAK key agreement protocol indeed has this property.

- **Perfect forward secrecy**. This property requires that previously agreed session keys should remain secret, even if both parties' long-term private key materials are compromised. Similar to other security models [3] for key agreement protocols, our model does not capture this property. However, we will give a separate proof that the IDAK key agreement protocol indeed has this property.

5 The security of IDAK

Before we present the security proof for the IDAK key agreement protocol, we first prove some preliminary results that will be used in the security proof.

Lemma 5.1 *Let $\mathcal{G} = \{G_\rho\}$ be a bilinear group family, $G_\rho = \langle G, G_1, \hat{e}\rangle$, g be a random generator of G, and $\pi : G \times G \to Z_q$ be a random oracle. Assume DBDH-Assumption holds for $\mathcal{G}$ and let $\mathcal{X}$ and $\mathcal{Y}$ be two distributions defined as*

$$\begin{aligned} & \mathcal{X} = \langle \mathcal{R}, g^{\beta x_0}, g^{\gamma y_0}, \hat{e}(g,g)^{(x_0+\pi(g^{\beta x_0}, g^{\gamma y_0}))(y_0+\pi(g^{\gamma y_0}, g^{\beta x_0}))\alpha\beta\gamma}, \hat{e}(g,g)^{\alpha\beta\gamma}\rangle \\ \textit{and} \quad & \mathcal{Y} = \langle \mathcal{R}, g^{\beta x_0}, g^{\gamma y_0}, \hat{e}(g,g)^{(x_0+\pi(g^{\beta x_0}, g^{\gamma y_0}))(y_0+\pi(g^{\gamma y_0}, g^{\beta x_0}))t}, \hat{e}(g,g)^{t}\rangle \end{aligned}$$

Then we have

1. *The two distributions $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable if $\mathcal{R}$ is defined as*

$$\mathcal{R} = \left(g, g^\alpha, g^\beta, g^\gamma, g^x, g^r, g_\mathcal{A}, \hat{e}\left(g^{x+\beta\pi(g^x, g_\mathcal{A})}, g_\mathcal{A} \cdot g^{r\pi(g_\mathcal{A}, g^x)}\right)^\alpha\right),$$

   $\alpha, \beta, \gamma, x, t, x_0$ are chosen from Z_q^* uniformly, $g^r = g^\gamma$ or r is either chosen from Z_q^* uniformly, $g_\mathcal{A}$ and $g^{\gamma y_0}$ are chosen from G according to any probabilistic polynomial time computable ensembles of distribution $\mathcal{G}(g^x, g^r, g^\alpha, g^\beta, g^\gamma, g^{\beta x_0})$. Note that the distributions for $g_\mathcal{A}$ and $g^{\gamma y_0}$ could be different.

2. *For any constant $m \le poly(k)$, the two distributions $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable if $\mathcal{R}$ is defined as:*

$$(g, g^\alpha, g^\beta, g^\gamma, (g^{x_i}, g^{r_j}, g_{\mathcal{A},l})_{i,j,l\le m}, (\hat{e}(g^{x_i+\beta\pi(g^{x_i}, g_{\mathcal{A},l})}, g_{\mathcal{A},l} \cdot g^{r_j\pi(g_{\mathcal{A},l}, g^{x_i})})^\alpha : i,j,l \le m))$$

   where $\alpha, \beta, \gamma, x_i$ are uniformly chosen from Z_q^*, r_j are either chosen from Z_q^* uniformly or $g^{r_j} = g^\gamma$, and $g_{\mathcal{A},l}$ is chosen according to a polynomial time computable ensemble of distribution $\mathcal{G}(g^{x_i}, g^{r_j}, g^\alpha, g^\beta, g^\gamma, g^{\beta x_0} : i,j,l \le m)$.

3. *For any constant $m \leq poly(k)$, the two distributions $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable if $\mathcal{R} = (\mathcal{R}_1, \mathcal{R}_2)$, where $\mathcal{R}_1$ is defined as the $\mathcal{R}$ in the item 2, and $\mathcal{R}_2$ is defined as:*

$$((g_{\mathcal{A},i}, g^{r_j}, g_{\mathcal{A},l})_{i,j,l \leq m}, (\hat{e}(g_{\mathcal{A},i} \cdot g^{\beta\pi(g_{\mathcal{A},i}, g_{\mathcal{A},l})}, g_{\mathcal{A},l} \cdot g^{r_j \pi(g_{\mathcal{A},l}, g_{\mathcal{A},i})})^{\alpha} : i, j, l \leq m))$$

where r_j are either chosen from Z_q^* uniformly or $g^{r_j} = g^{\gamma}$, $g_{\mathcal{A},i}$ and $g_{\mathcal{A},l}$ are chosen according to polynomial time computable ensembles of distribution $\mathcal{G}(g^{x_i}, g^{r_j}, g^{\alpha}, g^{\beta}, g^{\gamma}, g^{\beta x_0}, g^{\gamma y_0} : i, j, l \leq m)$ with the condition that "$g_{\mathcal{A},i} \neq g^{\beta x_0}$ or $g_{\mathcal{A},l} \neq g^{\gamma y_0}$". Note that $g_{\mathcal{A},i}$ and $g_{\mathcal{A},l}$ could have different distributions.

Proof. The Lemma could be proved using complicated version of the Splitting lemma by Pointcheval-Stern [21] (see the proof of Theorem 7). In the following, we use the random reduction to prove the lemma.

1. For a contradiction, assume that there is a polynomial time probabilistic algorithm $\mathcal{D}$ that distinguishes $\mathcal{X}$ and $\mathcal{Y}$. We construct a polynomial time probabilistic algorithm $\mathcal{A}$ that distinguishes $\langle g, g^u, g^v, g^w, \hat{e}(g,g)^a \rangle$ and $\langle g, g^u, g^v, g^w, \hat{e}(g,g)^{uvw} \rangle$ with δ_k, where u, v, w, a are uniformly at random in Z_q.

Let the input of $\mathcal{A}$ be $\langle g, g^u, g^v, g^w, \hat{e}(g,g)^{\tilde{a}} \rangle$, where $\tilde{a}$ is either uvw or uniformly at random in Z_q. $\mathcal{A}$ chooses uniformly at random $c_1, c_2, c_3, x, x_0 \in Z_q$, sets $g^{\alpha} = g^{c_1 u + c_2}$, $g^{\beta} = g^{v + c_3}$, $g^{\gamma} = g^{w + c_4}$, chooses uniformly at random $r \in Z_q$ or lets $g^r = g^{\beta}$, chooses $g^{\gamma y_0}, g_{\mathcal{A}} \in G$ according to the distributions $\mathcal{G}(g^x, g^r, g^{\alpha}, g^{\beta}, g^{\gamma}, g^{\beta x_0})$ (the distributions for $g_{\mathcal{A}} \in G$ and $g^{\gamma y_0}$ could be different). Since g^x and $g^{\beta x_0}$ are uniformly chosen from G, we may assume that the values of $\pi(g^x, g_{\mathcal{A}})$ and $\pi(g^{\gamma y_0}, g^{\beta x_0})$ are unknown yet. Without loss of generality, we may assume that $x + \beta\pi(g^x, g_{\mathcal{A}})$ and $y_0 + \pi(g^{\gamma y_0}, g^{\beta x_0})$ take values c_5 and c_6 respectively, where c_5 and c_6 are uniformly chosen from Z_q. In a summary, the value of $\mathcal{R}$ could be computed from $g^u, g^v, g^w, c_1, c_2, c_3, c_4, c_5$ efficiently. $\mathcal{A}$ then sets

$$\hat{e}(g,g)^{\tilde{t}} = \hat{e}(g,g)^{c_1 \tilde{a} + c_4(c_1 u + c_2)(v + c_3) + w(c_1 u c_3 + c_1 v + c_2 c_3)}.$$

$\mathcal{A}$ can compute $\hat{e}(g,g)^{(x_0 + \pi(g^{\beta x_0}, g^{\gamma y_0}))(y_0 + \pi(g^{\gamma y_0}, g^{\beta x_0}))\tilde{t}}$ using the values of $\hat{e}(g,g)^{\tilde{t}}$, x_0, $\pi(g^{\beta x_0}, g^{\gamma y_0})$, c_6. Let $\mathcal{A}\left(g, g^u, g^v, g^w, \hat{e}(g,g)^{\tilde{a}}\right) = \mathcal{D}(\tilde{\mathcal{X}})$, where $\tilde{\mathcal{X}}$ is obtained from $\mathcal{Y}$ by replacing t with $\tilde{t}$ and taking the remaining values as defined above.

Note that if $\tilde{a} = uvw$, then $\tilde{t} = \alpha\beta\gamma$, and $\tilde{\mathcal{X}}$ is distributed according to the distribution $\mathcal{X}$. That is, $\alpha, \beta, \gamma, x, x_0$ are uniform in Z_q and independent of each other and of (u, v, w), $(r,\ g_{\mathcal{A}},\ g^{\gamma y_0})$ is chosen according to the specified distributions. Otherwise, $\tilde{\mathcal{X}}$ is distributed according to the distribution $\mathcal{X}$, and $\tilde{t}$ is uniform in Z_q and independent of $\alpha, \beta, \gamma, x, x_0, r, u, v, w, g_{\mathcal{A}}, g^{\gamma y_0}$. Therefore, by definitions,

$$\begin{array}{rl} & \Pr\left[\mathcal{A}\left(g, g^u, g^v, g^w, \hat{e}(g,g)^{uvw}\right) = 1\right] = \Pr\left[\mathcal{D}(\mathcal{X}) = 1\right] \\ \text{and} & \Pr\left[\mathcal{A}\left(g, g^u, g^v, g^w, \hat{e}(g,g)^{a}\right) = 1\right] = \Pr\left[\mathcal{D}(\mathcal{Y}) = 1\right] \end{array}$$

Thus $\mathcal{A}$ distinguishes $\langle g, g^u, g^v, g^w, \hat{e}(g,g)^a \rangle$ and $\langle g, g^u, g^v, g^w, \hat{e}(g,g)^{uvw} \rangle$ with δ_k, where a is uniform at random in Z_q. This is a contradiction.

2. This part of the Lemma could be proved in the same way. The details are omitted.

3. Since "$g_{\mathcal{A},i} \neq g^{\beta x_0}$ or $g_{\mathcal{A},l} \neq g^{\gamma y_0}$", we may assume that the values of $\pi(g_{\mathcal{A},i}, g_{\mathcal{A},l})$ and $\pi(g_{\mathcal{A},l}, g_{\mathcal{A},i})$ are unknown yet. By the random oracle property of π, this part of the Lemma could be proved in the same way as in item 1. The details are omitted. □

Theorem 5.2 *Suppose that the functions H and π are random oracles and the bilinear group family $\mathcal{G}$ satisfies DBDH-Assumption. Then the IDAK scheme is a secure key agreement protocol.*

Proof. By Theorem 3.1, the condition 1 in the Definition 4.1 is satisfied for the IDAK key agreement protocol. In the following, we show that the condition 2 is also satisfied.

For a contradiction, assume that the adversary $\mathcal{A}$ has non-negligible advantage $\delta_k = \mathrm{Adv}^{\mathcal{A}}(k)$ in guessing the value of b after the **Test** query. We show how to construct a simulator $\mathcal{S}$ that uses $\mathcal{A}$ as an oracle to distinguish the distributions $\mathcal{X}$ and $\mathcal{Y}$ in the item 3 of Lemma 5.1 with non-negligible advantage $2\delta_k(q_E-2)^2/q_E^4$, where q_E denotes the number of distinct H-**queries** that the algorithm $\mathcal{A}$ has made. The game between the challenger and the simulator $\mathcal{S}$ starts with the challenger first generating bilinear groups $G_\rho = \langle G, G_1, \hat{e}\rangle$ by running the algorithm **Instance Generator**. The challenger then randomly selects $\alpha, \beta, \gamma, t \in Z_q$ and $b \in \{0,1\}$. The challenger gives the tuple $\langle \rho, g, g^\alpha, g^\beta, g^\gamma, \hat{e}(g,g)^{\tilde{t}}\rangle$ to the algorithm $\mathcal{S}$ where $\tilde{t} = \alpha\beta\gamma$ if $b=1$ and $\tilde{t} = t$ otherwise. During the simulation, the algorithm $\mathcal{S}$ can ask the challenger to provide randomly chosen g^{x_i}. $\mathcal{S}$ may then choose (with the help of $\mathcal{A}$ perhaps) $g_{\mathcal{A},l}$ according to a polynomial time computable distribution $\mathcal{G}(g^{x_i}, g^{r_j}, g^\alpha, g^\beta, g^\gamma, g^{\alpha x_0} : i,j,l \le m)$ and sends $g_{\mathcal{A},l}$ to the challenger. The challenger responds with $\hat{e}(g^{x_i+\beta\pi(g^{x_i}, g_{\mathcal{A},l})}, g_{\mathcal{A},l}\cdot g^{r_j\pi(g_{\mathcal{A},l}, g^{x_i})})^\alpha$. At the end of the simulation, the algorithm $\mathcal{S}$ is supposed to output its guess $b' \in \{0,1\}$ for b.

The algorithm $\mathcal{S}$ selects two integers $I, J \le q_E$ randomly and works by interacting with $\mathcal{A}$ as follows:

Setup: Algorithm $\mathcal{S}$ gives $\mathcal{A}$ the IDAK system parameters $\langle q, h, G, G_1, \hat{e}, H, \pi\rangle$ where $q, G, G_1, \hat{e}$ are parameters from the challenger, H and π are random oracles controlled by $\mathcal{S}$ as follows.

H-**queries**: At any time algorithm $\mathcal{A}$ can query the random oracle H using the queries $\mathbf{Extract}(\mathrm{ID}_i)$ or $\mathbf{GetID}(\mathrm{ID}_i) = H(\mathrm{ID}_i)$. To respond to these queries algorithm $\mathcal{S}$ maintains an H^{list} that contains a list of tuples $\langle \mathrm{ID}_i, g_{\mathrm{ID}_i}\rangle$. The list is initially empty. When $\mathcal{A}$ queries the oracle H at a point ID_i, $\mathcal{S}$ responds as follows:

1. If the query ID_i appears on the H^{list} in a tuple $\langle \mathrm{ID}_i, g_{\mathrm{ID}_i}\rangle$, then $\mathcal{S}$ responds with $H(\mathrm{ID}_i) = g_{\mathrm{ID}_i}$.
2. Otherwise, if this is the I-th new query of the random oracle H, $\mathcal{S}$ responds with $g_{\mathrm{ID}_i} = H(\mathrm{ID}_i) = g^\beta$, and adds the tuple $\langle \mathrm{ID}_i, g^\beta\rangle$ to the H^{list}. If this is the J-th new query of the random oracle, $\mathcal{S}$ responds with $g_{\mathrm{ID}_i} = H(\mathrm{ID}_i) = g^\gamma$, and adds the tuple $\langle \mathrm{ID}_i, g^\gamma\rangle$ to the H^{list}.
3. In the remaining case, $\mathcal{S}$ selects a random $r_i \in Z_q$, responds with $g_{\mathrm{ID}_i} = H(\mathrm{ID}_i) = g^{r_i}$, and adds the tuple $\langle \mathrm{ID}_i, g^{r_i}\rangle$ to the H^{list}.

π-**queries**: At any time the challenger, the algorithm $\mathcal{A}$, and the algorithm $\mathcal{S}$ can query the random oracle π. To respond to these queries algorithm $\mathcal{S}$ maintains a π^{list} that contains a list of tuples $\langle g_1, g_2, \pi(g_1, g_2)\rangle$. The list is initially empty. When $\mathcal{A}$ queries the oracle π at a point (g_1, g_2), $\mathcal{S}$ responds as follows: If the query (g_1, g_2) appears on the π^{list} in a tuple $\langle (g_1, g_2), \pi(g_1, g_2)\rangle$, then $\mathcal{S}$ responds with $\pi(g_1, g_2)$. Otherwise, $\mathcal{S}$ selects a random $v_i \in Z_q$, responds with $\pi(g_1, g_2) = v_i$, and adds the tuple $\langle (g_1, g_2), v_i\rangle$ to the π^{list}. Technically, the random oracle π could be held by an independent third party to avoid the confusion that the challenger also needs to access this random oracle also.

Query phase: $\mathcal{S}$ responds to $\mathcal{A}$'s queries as follows.

For a $\mathbf{GetID}(\mathrm{ID}_i)$ query, $\mathcal{S}$ runs the H-**queries** to obtain a g_{ID_i} such that $H(\mathrm{ID}_i) = g_{\mathrm{ID}_i}$, and responds with g_{ID_i}.

For an $\mathbf{Extract}(\mathrm{ID}_i)$ query for the long term private key, if $i = I$ or $i = J$, then $\mathcal{S}$ reports failure and terminates. Otherwise, $\mathcal{S}$ runs the H-**queries** to obtain $g_{\mathrm{ID}_i} = H(\mathrm{ID}_i) = g^{r_i}$, and responds $d_{\mathrm{ID}_i} = (g^\alpha)^{r_i} = g^\alpha_{\mathrm{ID}_i}$.

For a $\mathbf{Send}(\Pi^s_{i,j}, X)$ query, we distinguish the following three cases:

1. $X = \lambda$. If $i = I$ or J, $\mathcal{S}$ asks the challenger for a random $R_i \in G$ (note that $\mathcal{S}$ does not know the discrete logarithm of R_i with base g_{ID_i}), otherwise $\mathcal{S}$ chooses a random $u_i \in Z^*_q$ and sets $R_i = g^{u_i}_{\mathrm{ID}_i}$. $\mathcal{S}$ lets $\Pi^s_{i,j}$ reply with R_i. That is, we assume that ID_i is carrying out an IDAK key agreement protocol with ID_j and ID_i sends the first message R_i to ID_j.

2. $X \neq \lambda$ and the transcript of the oracle $\Pi^s_{i,j}$ is empty. In this case, $\Pi^s_{i,j}$ is the responder to the protocol and has not sent out any message yet. If $i = I$ or J, $\mathcal{S}$ asks the challenger for a random $R_i \in G$, otherwise $\mathcal{S}$ chooses a random $u_i \in Z^*_q$ and sets $R_i = g^{u_i}_{\mathrm{ID}_i}$. $\mathcal{S}$ lets $\Pi^s_{i,j}$ reply with R_i and marks the oracle $\Pi^s_{i,j}$ as completed.

3. $X \neq \lambda$ and the transcript of the oracle $\Pi^s_{i,j}$ is not empty. In this case, $\Pi^s_{i,j}$ is the protocol initiator and should have sent out the first message already. Thus $\Pi^s_{i,j}$ does not need to respond anything. After processing the query $\mathbf{Send}(\Pi^s_{i,j}, X)$, $\mathcal{S}$ marks the oracle $\Pi^s_{i,j}$ as completed.

For a $\mathbf{Reveal}(\Pi^s_{i,j})$ query, if $i \neq I$ and $i \neq J$, $\mathcal{S}$ computes the session key $sk_{ij} = \hat{e}(g^{\pi(R_j,R_i)}_{\mathrm{ID}_j} \cdot R_j, d^{(u_i+\pi(R_i,R_j))h}_{\mathrm{ID}_i})$ and responds with sk_{ij}, here R_j is the message received by $\Pi^s_{i,j}$. Note that the message R_j may not necessarily be sent by the oracle $\Pi^{s'}_{j,i}$ for some s' since it could have been a bogus message from $\mathcal{A}$. Otherwise, $i = I$ or $i = J$. Without loss of generality, we assume that $i = I$. In this case, the oracle $\Pi^s_{I,j}$ dose not know its private key $g^{\beta\alpha}$. Thus it needs help from the challenger to compute the shared session key. Let R_I and R_j be the messages that $\Pi^s_{I,j}$ has sent out and received respectively. $\Pi^s_{I,j}$ gives these two values to the challenger and the challenger computes the shared session key $sk_{Ij} = \hat{e}\left(g^{\pi(R_j,R_i)}_{\mathrm{ID}_j} \cdot R_j, R^{\alpha h}_I g^{\pi(R_I,R_j)\alpha\beta h}\right)$. $\Pi^s_{I,j}$ then responds with k_{Ij}.

For a $\mathbf{Corrupt}(i)$ query, if $i = I$ or $i = J$, then $\mathcal{S}$ reports failure and terminates. Otherwise, $\mathcal{S}$ responds with $d_{\mathrm{ID}_i} = (g^\alpha)^{r_i} = g^\alpha_{\mathrm{ID}_i}$.

For the **Test**$(\Pi^s_{i,j})$ query, if $i \neq I$ or $j \neq J$, then $\mathcal{S}$ reports failure and terminates. Otherwise, assume that $i = I$ and $j = J$. Let $R_I = g^{u_I}_{\mathrm{ID}_I}$ be the message that $\Pi^s_{i,j}$ sends out (note that the challenger generated this message) and $R_J = g^{u_J}_{\mathrm{ID}_J}$ be the message that $\Pi^s_{i,j}$ receives (note that R_J could be the message that the challenger generated or could be generated by the algorithm $\mathcal{A}$). $\mathcal{S}$ gives the messages R_I and R_J to the challenger. The challenger computes $X = \hat{e}(g,g)^{(u_I+\pi(R_I,R_J))(u_J+\pi(R_J,R_I))\tilde{t}h}$ and gives X to $\mathcal{S}$. $\mathcal{S}$ responds with X. Note that if $\tilde{t} = \alpha\beta\gamma$, then X is the session key. Otherwise, X is a uniformly distributed group element.

Guess: After the **Test**$(\Pi^s_{i,j})$ query, the algorithm $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. Algorithm $\mathcal{S}$ outputs b' as its guess to the challenger.

Claim: If $\mathcal{S}$ does not abort during the simulation then $\mathcal{A}$'s view is identical to its view in the real attack. Furthermore, if $\mathcal{S}$ does not abort, then $\left|\Pr[b = b'] - \frac{1}{2}\right| > \delta_k$, where the probability is over all random coins used by $\mathcal{S}$ and $\mathcal{A}$.

Proof of Claim: The responses to H**-queries** and π**-queries** are the same as in the real attack since the response is uniformly distributed. All responses to the getID queries, private key extract queries, message delivery queries, reveal queries, and corrupt queries are valid. It remains to show that the response to the test query is valid also. When $\tilde{t}$ is uniformly distributed over Z_q, then Theorem 3.1 shows that $X = \hat{e}(g,g)^{(u_I+\pi(R_I,R_J))(u_J+\pi(R_J,R_I))\tilde{t}h}$ is uniformly distributed over G and is computationally indistinguishable from a random value before $\mathcal{A}$'s view. Therefore, by definition of the algorithm $\mathcal{A}$, we have $\left|\Pr[b=b'] - \frac{1}{2}\right| > \delta_k$. $\square$

Suppose $\mathcal{A}$ makes a total of q_E H-queries. We next calculate the probability that $\mathcal{S}$ does not abort during the simulation. The probability that $\mathcal{S}$ does not abort for **Extract** queries is $(q_E-2)/q_E$. The probability that $\mathcal{S}$ does not abort for **Corrupt** queries is $(q_E-2)/q_E$. The probability that $\mathcal{S}$ does not abort for **Test** queries is $2/q_E^2$. Therefore, the probability that $\mathcal{S}$ does not abort during the simulation is $2(q_E-2)^2/q_E^4$. This shows that $\mathcal{S}$'s advantage in distinguishing the distributions $\mathcal{X}$ and $\mathcal{Y}$ in Lemma 5.1 is at least $2\delta_k(q_E-2)^2/q_E^4$ which is non-negligible.

To complete the proof of Theorem 5.2, it remains to show that the communications between $\mathcal{S}$ and the challenger are carried out according to the distributions $\mathcal{X}$ and $\mathcal{Y}$ of Lemma 5.1. For a $\mathbf{Reveal}(\Pi^s_{I,j})$ query, the challenger outputs $\hat{e}\left(g_{\mathrm{ID}_j}^{\pi(R_j,R_I)} \cdot R_j, R_I^{\alpha h} g^{\pi(R_I,R_j)\alpha\beta h}\right)$ to the algorithm $\mathcal{S}$. Let $R_I = g^x$, $R_j = g_{\mathcal{A}}$, and $g_{\mathrm{ID}_j} = g^r$. Then x is chosen uniform at random from Z_q, r is chosen uniform at random from Z_q^* when $j \neq J$ or $r = \gamma$ when $j = J$, and the value of $g_{\mathcal{A}}$ is chosen by the algorithm $\mathcal{A}$ or by the algorithm $\mathcal{S}$ or by the challenger in probabilistic polynomial time according to the current views. For example, if $g_{\mathcal{A}}$ is chosen by the algorithm $\mathcal{A}$, then $\mathcal{A}$ may generate $g_{\mathcal{A}}$ as the combination (e.g., multiplication) of some previously observed messages/values or generate it randomly. Thus, ignoring the co-factor h, the communication between the challenger and the algorithm $\mathcal{S}$ during $\mathbf{Reveal}(\Pi^s_{I,j})$ queries is carried out according to the distributions $\mathcal{X}$ and $\mathcal{Y}$ of Lemma 5.1. The case for $\mathbf{Reveal}(\Pi^s_{J,j})$ queries is the same.

For the **Test**$(\Pi^s_{I,J})$ query, the challenger outputs $X = \hat{e}(g,g)^{(u_I+\pi(R_I,R_J))(u_J+\pi(R_J,R_I))\tilde{t}h}$ to the algorithm $\mathcal{S}$, where $R_I = g^{\beta u_I}$ and $R_J = g^{\gamma u_J}$. Let $x_0 = u_I$ and $y_0 = u_J$. Then x_0 is chosen uniform at random from Z_q and the value of $g^{\gamma y_0}$ is chosen by the algorithm $\mathcal{A}$ or by the challenger in probabilistic polynomial time according to the current views. Similarly, $\mathcal{A}$ may choose $g^{\gamma y_0}$ as the combination (e.g., multiplication) of some previously observed messages/values. Ignoring the co-factor h, the communication between the challenger and the algorithm $\mathcal{S}$ during the $\mathbf{Test}(\Pi^s_{I,J})$ query is carried out according to the distributions $\mathcal{X}$ and $\mathcal{Y}$ of Lemma 5.1.

It should be noted that after the **Test**$(\Pi^s_{I,J})$ query, the adversary may create bogus oracles for the participants ID_I and ID_J and send bogus messages that may depend on all existing communicated messages (including messages held by the oracle $\Pi^s_{I,J}$) and then reveal session keys from these oracles. In particular, the adversary may play a man in the middle attack by modifying the messages sent from $\Pi^s_{I,J}$ to $\Pi^{s'}_{J,I}$ and modifying the messages sent from $\Pi^{s'}_{J,I}$ to $\Pi^s_{I,J}$. Then the oracles $\Pi^{s'}_{J,I}$ and $\Pi^s_{I,J}$ are not matching oracles. Thus $\mathcal{A}$ can reveal the session key held by the oracle $\Pi^{s'}_{J,I}$ before the guess. In the $\mathcal{R}_2$ part in the distributions $\mathcal{X}$ and $\mathcal{Y}$ of Lemma 5.1, we have the condition "$g_{\mathcal{A},i} \neq g^{\beta x_0}$ or $g_{\mathcal{A},l} \neq g^{\gamma y_0}$" (this condition holds since the algorithm $\mathcal{A}$ has not revealed the matching oracles for $\Pi^s_{I,J}$). If both $g_{\mathcal{A},i} \neq g^{\beta x_0}$ and $g_{\mathcal{A},l} \neq g^{\gamma y_0}$, then the oracle $\Pi^{s'}_{J,I}$ is a matching oracle for $\Pi^s_{I,J}$ and $\mathcal{A}$ is not allowed to reveal the session key held by the oracle $\Pi^{s'}_{J,I}$. Thus, Ignoring the co-factor h, the communication between the challenger and the algorithm $\mathcal{S}$ during these $\mathbf{Test}(\Pi^s_{I,J})$ query is carried out according to the distributions $\mathcal{X}$ and $\mathcal{Y}$ of Lemma 5.1.

In the summary, all communications between the challenger and $\mathcal{S}$ are carried out according to the distributions $\mathcal{X}$ and $\mathcal{Y}$ of Lemma 5.1. This completes the proof of the Theorem. $\Box$

6 Perfect forward secrecy

In this section, we show that the protocol IDAK has the additional perfect forward secrecy property when both parties long term private keys are corrupted. That is, even if Alice and Bob lose their private keys $d_A = g_{ID_A}^{\alpha}$ and $d_B = g_{\mathrm{ID}_B}^{\alpha}$, the session keys established by Alice and Bob in the previous sessions are still secure. In order to show this, it is sufficient to show that the two distributions $\left(\mathcal{R}, \hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^{(x+\pi(g_{\mathrm{ID}_A}^x, g_{\mathrm{ID}_B}^y))(y+\pi(g_{\mathrm{ID}_B}^y, g_{\mathrm{ID}_A}^x))\alpha}\right)$ and $(\mathcal{R}, \hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^z)$ are computationally indistinguishable for $\mathcal{R} = (g_{\mathrm{ID}_A}^{\alpha}, g_{\mathrm{ID}_B}^{\alpha}, g_{\mathrm{ID}_A}^{x}, g_{\mathrm{ID}_B}^{y})$ and uniform at random chosen $g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B}, x, y, z, \alpha$. Consequently, it is sufficient to prove the following theorem.

Theorem 6.1 *Let $\mathcal{G} = \{G_\rho\}$ be a bilinear group family, $G_\rho = \langle G, G_1, \hat{e}\rangle$. Assume that DBDH-Assumption holds for $\mathcal{G}$. Then the two distributions*

$$\begin{array}{rl} & \mathcal{X} = (g_1, g_2, g_1^{\alpha}, g_2^{\alpha}, g_1^{x}, g_2^{y}, \hat{e}(g_1, g_2)^{xy\alpha}) \\ \textit{and} & \mathcal{Y} = (g_1, g_2, g_1^{\alpha}, g_2^{\alpha}, g_1^{x}, g_2^{y}, \hat{e}(g_1, g_2)^{z}) \end{array}$$

are computationally indistinguishable for random chosen $g_1, g_2, x, y, z, \alpha$.

Proof. We use a random reduction. For a contradiction, assume that there is a polynomial time probabilistic algorithm $\mathcal{D}$ that distinguishes $\mathcal{X}$ and $\mathcal{Y}$ with a non-negligible probability δ_k. We construct a polynomial time probabilistic algorithm $\mathcal{A}$ that distinguishes $(\mathcal{R}, \hat{e}(g,g)^t)$ and $(\mathcal{R}, \hat{e}(g,g)^{uvw})$ with δ_k, where $\mathcal{R} = (g, g^u, g^v, g^w)$ and u, v, w, t are uniformly at random in Z_q. Let the input of $\mathcal{A}$ be $(\mathcal{R}, \hat{e}(g,g)^{\tilde{t}})$, where $\tilde{t}$ is either uvw or uniformly at random in Z_q. We construct $\mathcal{A}$ as follows. $\mathcal{A}$ chooses random $c_1, c_2, c_3, c_4, c_5 \in Z_q$ and sets $g_1 = g^{c_1}$, $g_2 = g^{c_2}$, $g_1^{\alpha} = g^{uc_1c_3}$, $g_2^{\alpha} = g^{uc_2c_3}$, $g_1^{x} = g^{vc_1c_4}$, $g_2^{y} = g^{wc_2c_5}$, and $\hat{e}(g_1, g_2)^{\tilde{z}} = \hat{e}(g,g)^{\tilde{t}c_1c_2c_3c_4c_5}$. Let $\mathcal{A}\left(\mathcal{R}, \hat{e}(g,g)^{\tilde{t}}\right) = \mathcal{D}\left(g_1, g_2, g_1^{\alpha}, g_2^{\alpha}, g_1^{x}, g_2^{y}, \hat{e}(g_1, g_2)^{\tilde{z}}\right)$. Note that if $\tilde{t} = uvw$, then c_1, c_2, α, x, y are uniform in Z_q (and independent of each other and of u, v, w) and $xy\alpha = \tilde{z}$. Otherwise, c_1, c_2, α, x, y are uniform in Z_q and independent of each other and of u, v, w. Therefore, by the definitions,

$$\begin{array}{rl} & \Pr\left[\mathcal{A}\left(\mathcal{R}, \hat{e}(g,g)^{uvw}\right) = 1\right] = \Pr\left[\mathcal{D}(\mathcal{X}) = 1\right] \\ \text{and} & \Pr\left[\mathcal{A}\left(\mathcal{R}, \hat{e}(g,g)^{t}\right) = 1\right] = \Pr\left[\mathcal{D}(\mathcal{Y}) = 1\right] \end{array}$$

Thus $\mathcal{A}$ distinguishes $\langle g, g^u, g^v, g^w, \hat{e}(g,g)^t\rangle$ and $\langle g, g^u, g^v, g^w, \hat{e}(g,g)^{uvw}\rangle$ with δ_k. This is a contradiction. $\Box$

Though Theorem 6.1 shows that the protocol IDAK achieves perfect forward secrecy even if both participating parties' long term private keys were corrupted, IDAK does not have perfect forward secrecy when the master secret α were leaked. The perfect forward secrecy against the corruption of α could be achieved by requiring Bob (the responder in the IDAK protocol) to send $g_{\mathrm{ID}_A}^{y}$ in addition to the value $R_B = g_{\mathrm{ID}_B}^{y}$ and by requiring both parties to compute the shared secret as $H(g_{\mathrm{ID}_A}^{xy}||sk_{AB})$ where sk_{AB} is the shared secret established by the IDAK protocol.

7 Key compromise impersonation resilience

In this section, we show that the protocol IDAK has the key compromise impersonation resilience property. That is, if Alice loses her private key $d_A = g^{\alpha}_{ID_A}$, then the adversary still could not impersonate Bob to Alice. In order to show this, it is sufficient to show that the two distributions $\left(\mathcal{R}, \hat{e}\left(g^{x}_{\mathrm{ID}_A} \cdot g_{\mathrm{ID}_A}^{\pi(g^{x}_{\mathrm{ID}_A}, R_B)}, R_B \cdot g_{\mathrm{ID}_B}^{\pi(R_B, g^{x}_{\mathrm{ID}_A})}\right)^{\alpha}\right)$ and $(\mathcal{R}, \hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^z)$ are computationally indistinguishable for $\mathcal{R} = (g^{\alpha}_{\mathrm{ID}_A}, g^{x}_{\mathrm{ID}_A}, R_B)$, where $g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B}, x, z, \alpha$ are chosen uniform at random, and R_B is chosen according to some probabilistic polynomial time distribution. Since the value $\hat{e}\left(g_{\mathrm{ID}_A}^{\pi(g^{x}_{\mathrm{ID}_A}, R_B)}, R_B \cdot g_{\mathrm{ID}_B}^{\pi(R_B, g^{x}_{\mathrm{ID}_A})}\right)^{\alpha}$ is known, it is sufficient to prove the following theorem.

Theorem 7.1 *Let $\mathcal{G} = \{G_\rho\}$ be a bilinear group family, $G_\rho = \langle G, G_1, \hat{e}\rangle$. Assume that DBDH-Assumption holds for $\mathcal{G}$. Then the two distributions*

$$\begin{array}{rl} & \mathcal{X} = \left(g_1, g_2, g_1^{\alpha}, g_1^{x}, R_B, \hat{e}\left(g_1^{x}, R_B \cdot g_2^{\pi(R_B, g_1^x)}\right)^{\alpha}\right) \\ \textit{and} & \mathcal{Y} = (g_1, g_2, g_1^{\alpha}, g_1^{x}, R_B, \hat{e}(g_1, g_2)^z) \end{array}$$

are computationally indistinguishable for random chosen g_1, g_2, x, z, α, where R_B is chosen according to some probabilistic polynomial time distribution.

Proof. Since g_1^x is chosen uniform at random, and π is a random oracle, we may assume that $R_B \cdot g_2^{\pi(R_B, g_1^x)}$ is uniformly distributed over G when R_B is chosen according to any probabilistic polynomial time distribution. Thus the proof is similar to the proof of Theorem 6.1 and the details are omitted. The theorem could also be proved using the Splitting lemma [21] which was used to prove the fork lemma. Briefly, the Splitting lemma translates the fact that when a subset A is "large" in a product space $X \times Y$, it has many large sections. Using the Splitting lemma, one can show that if $\mathcal{D}$ can distinguish $\mathcal{X}$ and $\mathcal{Y}$, then by replaying $\mathcal{D}$ with different random oracle π, one can get sufficient many tuples $(g_1, g_2, g_1^{\alpha}, g_1^{x}, R_B, \pi_1, \pi_2)$ such that (1) $\pi_1(R_B, g_1^x) \not= \pi_2(R_B, g_1^x)$; (2) $\mathcal{D}$ distinguishes $\mathcal{X}_1$ and $\mathcal{Y}$ (respectively $\mathcal{X}_2$ and $\mathcal{Y}$) when z is uniformly chosen but other values takes the values from the above tuple with π_1 (respectively π_2). Since $\hat{e}\left(g_1^x, R_B \cdot g_2^{\pi_1(R_B, g_1^x)}\right)^{\alpha} / \hat{e}\left(g_1^x, R_B \cdot g_2^{\pi_2(R_B, g_1^x)}\right)^{\alpha} = \hat{e}\left(g_1, g_2\right)^{x\alpha(\pi_1(R_B, g_1^x) - \pi_2(R_B, g_1^x))}$. Thus, for the above tuple, we can distinguish $\hat{e}\left(g_1, g_2\right)^{x\alpha}$ from $\hat{e}\left(g, g\right)^{z}$ for random chosen z. This is a contradiction with the DBDH-Assumption. □

8 IDAK with key confirmation

The security Definition 4.1 in Section §4 for key agreement protocols does not provide the following assurance to a user ID_i during a key agreement protocol: one oracle Π^s_{ij} has been engaged in a conversation and has successfully finished the protocol with a session key output. However, there may be no matching oracle $\Pi^{s'}_{ji}$ existing at all (though according to the definition, the adversary learns zero information about the session key held by Π^s_{ij}). In order to provide assurance against the above scenario, we study secure key agreement protocols with key confirmation in this section. First we slightly modify our matching oracle definition from Section §4. The definition of matching oracles in Section §4 does require all messages that Π^s_{ij} sends out should reach its matching oracle $\Pi_{ji}^{s'}$ and vice versa. In this section, when we talk about matching oracles, we do not require the last message of the protocol to reach its destination. Indeed, in any protocol, the party who sends the last message flow cannot "know" whether or not its last message was received by its partner (see [3]).

Let **No-Matching**$^E(k)$ denote the event that, during the protocol execution against the adversary, there exists an oracle Π_{ij}^{s} with the following properties:

1. Π_{ij}^{s} has been engaged in a conversation and has successfully finished the protocol with a session key output.
2. There is no matching oracle $\Pi_{ji}^{s'}$ for Π_{ij}^{s} existing.
3. The adversary has not compromised the long term keys for ID_i and ID_j.

Definition 8.1 *A protocol* Π *is a secure key agreement protocol with key confirmation if* Π *is a secure key agreement protocol and the probability of* **No-Matching**$^E(k)$ *is negligible.*

It is straightforward to observe that IDAK is not a secure in the sense of Definition 8.1. In this section, we design a secure key agreement scheme with key confirmation. We first briefly describe message authentication code. A *Message Authentication Code* (see, e.g., [1]) is a deterministic polynomial time algorithm $\mathrm{MAC}_{(\cdot)}(\cdot)$. To authenticate a message m with a key K, one computes the authenticated message pair $(m, a) = (m, \mathrm{MAC}_K(m))$, where $a = \mathrm{MAC}_K(m)$ is called the tag on m. A MAC scheme is secure if the probability for an adversary to forge a tag a for a (not authenticated yet) message m of the adversary's choice under a randomly chosen key K is negligible. The adversary is allowed to make adaptive-message attacks. That is, the adversary can choose messages m' (different from the target message) and ask the MAC oracle to generate the authentication tag on m' under the target key K. In the following, we describe the IDAK protocol with key confirmation and show that it is secure according to Definition 8.1.

The **Setup** algorithm is the same as that in IDAK protocol, in addition, we also need to choose two additional random oracles $\mathcal{H}_1$ and $\mathcal{H}_2$ (both will be used as key derivation functions), and a secure message authentication function $\mathrm{MAC}_{(\cdot)}(\cdot)$ (see, e.g., [1]).

The **Extract** algorithm for IDAKC is the same as that in IDAK protocol.

The **Exchange** algorithm for IDAKC proceeds as follows:

Exchange For two participants Alice and Bob whose identification strings are ID_A and ID_B respectively, the algorithm proceeds as follows.

1. Alice selects $x \in_R Z_q^*$, computes $R_A = g_{\mathrm{ID}_A}^x$, and sends it to Bob.
2. (a) Bob selects $y \in_R Z_q^*$, computes $R_B = g_{\mathrm{ID}_B}^y$.
   (b) Bob computes $s_A = \pi(R_A, R_B)$, $s_B = \pi(R_B, R_A)$, and the shared secret sk_{IDAK} as
   $$\hat{e}(g_{\mathrm{ID}_B}, g_{\mathrm{ID}_A})^{(x+s_A)(y+s_B)h\alpha} = \hat{e}\left(g_{\mathrm{ID}_A}^{s_A} \cdot R_A, g_{\mathrm{ID}_B}^{(y+s_B)h\alpha}\right).$$
   (c) Bob computes $K_1 = \mathcal{H}_1(sk_{\mathrm{IDAK}})$ and $K_2 = \mathcal{H}_2(sk_{\mathrm{IDAK}})$.
   (d) Bob computes $\mathrm{MAC}_{K_2}(\mathrm{ID}_B, \mathrm{ID}_A, R_B, R_A)$ and sends this together with R_B to Alice.
3. (a) Alice computes $s_A = \pi(R_A, R_B)$, $s_B = \pi(R_B, R_A)$, and the shared secret sk_{IDAK} as
   $$\hat{e}(g_{\mathrm{ID}_B}, g_{\mathrm{ID}_A})^{(x+s_A)(y+s_B)h\alpha} = \hat{e}\left(g_{\mathrm{ID}_B}^{s_B} \cdot R_B, g_{\mathrm{ID}_A}^{(x+s_A)h\alpha}\right).$$

(b) Alice computes $K_1 = \mathcal{H}_1(sk_{\mathrm{IDAK}})$ and $K_2 = \mathcal{H}_2(sk_{\mathrm{IDAK}})$.

(c) Alice computes $\mathrm{MAC}_{K_2}(\mathrm{ID}_A, \mathrm{ID}_B, R_A, R_B)$ and sends this to Bob.

Theorem 8.2 *Assume that H, π, $\mathcal{H}_1$ and $\mathcal{H}_2$ are independent random oracles, MAC is a secure message authentication function, and the group family $\mathcal{G}$ satisfies DBDH-Assumption. Then IDAKC is a secure key agreement protocol with key confirmation.*

Proof. By Theorem 5.2, IDAKC is a secure key agreement protocol. Thus we only need to show that the probability of **No-Matching**$^E(k) = \varepsilon_k$ is negligible.

For a contradiction, assume that the adversary has a non-negligible advantage ε_k such that there exists an oracle Π^s_{IJ} that has been engaged in a conversation and has successfully finished the protocol with a session key output, but there is no matching oracle $\Pi^{s'}_{JI}$ existing. We show how to construct a simulator $\mathcal{S}$ that uses $\mathcal{A}$ as an oracle to forge an authentication tag on an un-authenticated message m under an unknown random key with non-negligible advantage $\varepsilon_k(2^{2k}-1)(1-\delta_k)(q_E-2)(q_E^2 q_N - 2)^2/q_E^7 q_N^3 2^{2k}$, where q_E is the number of H-**queries** that the simulation makes, q_N is the maximum number of IDAKC key agreement sessions that the algorithm $\mathcal{A}$ initiates for each participant, δ_k is the probability that the adversary can compute the session key of an un-revealed oracle. The game between the challenger and the simulator $\mathcal{S}$ starts with the challenger first choose a random key $\mathcal{K}$ for the MAC scheme. During the simulation, $\mathcal{S}$ can present messages m to the challenger to get the MAC tag on m under this key $\mathcal{K}$ (but the adversary $\mathcal{A}$ is not allowed to ask the challenger for MAC tags). At the end of the simulation, the algorithm $\mathcal{S}$ is supposed to output a message m and its guess a for the MAC tag on m under the key $\mathcal{K}$. The algorithm $\mathcal{S}$ works by interacting with $\mathcal{A}$ as follows:

Setup: Algorithm $\mathcal{S}$ selects uniformly at random system parameters $\langle q, h, G, G_1, \hat{e}, H, \mathcal{H}_1, \mathcal{H}_2, \pi\rangle$ and gives it to $\mathcal{A}$, where $H, \mathcal{H}_1, \mathcal{H}_2$, and π are random oracles controlled by $\mathcal{S}$ as follows. These random oracles could be queried by $\mathcal{S}$ or $\mathcal{A}$ during the simulation. Meanwhile, $\mathcal{S}$ keeps the master secret key α in secret.

H-**queries**, π-**queries**, $\mathcal{H}_1$-**queries**, and $\mathcal{H}_2$-**queries**: They are the same as the π-**queries** in the proof of Theorem 5.2. That is, $\mathcal{S}$ answers all distinct queries independently and randomly. Note that H-**queries** defined here is different from that in the proof of Theorem 5.2.

Query phase (MAC forgery phase): $\mathcal{S}$ chooses three integers $I, J \le n$ and $s_0 \le q_N$, and responds to $\mathcal{A}$'s queries as follows.

For an $\mathbf{Extract}(\mathrm{ID}_i)$ query, $\mathcal{S}$ runs the H-**queries** to obtain $g_{\mathrm{ID}_i} = H(\mathrm{ID}_i)$ and responds with $d_{\mathrm{ID}_i} = g^{\alpha}_{\mathrm{ID}_i}$.

For a $\mathbf{Send}(\Pi^s_{i,j}, X)$ query, we distinguish the following three cases:

1. $X = \lambda$. In this case, $\Pi^s_{i,j}$ is the protocol initiator. $\mathcal{S}$ chooses a random $x_i \in Z_q$ and sets $R_i = g^{x_i}_{\mathrm{ID}_i}$. $\mathcal{S}$ lets $\Pi^s_{i,j}$ reply with R_i. That is, we assume that ID_i is carrying out an IDAKC key agreement protocol with ID_j and ID_i sends the first message R_i to ID_j.

2. $X \neq \lambda$ and the transcript of the oracle $\Pi^s_{i,j}$ is empty. In this case, $\Pi^s_{i,j}$ is the protocol responder and has not sent out any message yet. $\mathcal{S}$ chooses a random $x_i \in Z_q$ and sets $R_i = g^{x_i}_{\mathrm{ID}_i}$. $\mathcal{S}$ then distinguishes the following two cases:

   (a) $i = I$ and $j = J$ and $s = s_0$. Instead of running the $\mathcal{H}_2$-**queries** to obtain $K_2^{i,j}$, $\mathcal{S}$ asks the challenger to generate the MAC tag $a^s_{i,j}$ for the message $m = (\mathrm{ID}_i, \mathrm{ID}_j, R_i, R_j)$

where R_j is the random component received from the other oracle. $\mathcal{S}$ lets $\Pi_{i,j}^s$ reply with $(R_i, a_{i,j}^s)$.

(b) $i \neq I$ or $j \neq J$ or $s \neq s_0$. $\mathcal{S}$ computes the session keying material sk_{IDAK} and runs the $\mathcal{H}_2$-**queries** to obtain $K_2^{i,j} = \mathcal{H}_1(sk_{\text{IDAK}})$. $\mathcal{S}$ computes $a_{i,j}^s = \text{MAC}_{K_2^{i,j}}(\text{ID}_i, \text{ID}_j, R_i, R_j)$ and lets $\Pi_{i,j}^s$ reply with $(R_i, a_{i,j}^s)$, where R_j is the random component received from the other oracle.

3. $X \neq \lambda$ and the transcript of the oracle $\Pi_{i,j}^s$ is not empty. In this case, $\Pi_{i,j}^s$ is the protocol initiator or responder and should have sent out the first message already. $\mathcal{S}$ then distinguishes the following two cases:

(a) $i = I$ and $j = J$ and $s = s_0$. If there is a matching oracle $\Pi_{J,I}^{s_0'}$ for $\Pi_{I,J}^{s_0}$, then $\mathcal{S}$ aborts the simulation with failure. Otherwise, let $a_{j,i}^s$ be the received MAC tag for the message $m = (\text{ID}_j, \text{ID}_i, R_j, R_i)$. $\mathcal{S}$ outputs $a_{j,i}^s$ as the guessed MAC tag for the message $m = (\text{ID}_j, \text{ID}_i, R_j, R_i)$ ($\mathcal{S}$ can terminate the simulation now. However, for easy analysis of the probability, we continue the simulation). $\mathcal{S}$ then asks the challenger whether this MAC tag is valid. If the challenger's answer is yes, $\mathcal{S}$ marks $\Pi_{i,j}^s$ as completed/accepted and terminate the simulation. If the challenger's answer is no, $\mathcal{S}$ marks $\Pi_{i,j}^s$ completed/rejected. Note that, according to the IDAKC protocol, if the oracle $\Pi_{i,j}^s$ is the protocol initiator, then it should send the message authentication tag to the responder as the last message. However, by the new definition matching oracles, this message does not matter.

(b) $i \neq I$ or $j \neq J$ or $s \neq s_0$. If $\Pi_{i,j}^s$ is the protocol responder, then $\mathcal{S}$ should have computed the shared secret $K_2^{i,j}$ already. $\mathcal{S}$ computes the MAC tag $a_{j,i}^s = \text{MAC}_{K_2^{i,j}}(\text{ID}_j, \text{ID}_i, R_j, R_i)$ where R_j is the random component received from the other oracle and compares this tag with the received tag. $\mathcal{S}$ marks $\Pi_{i,j}^s$ as completed/accepted if the two tags are the same, and marks it completed/rejected if the two tags are different. For the case that $\Pi_{i,j}^s$ is the protocol initiator, $\mathcal{S}$ computes the session keying material sk_{IDAK} and runs the $\mathcal{H}_2$-**queries** to obtain $K_2^{i,j} = \mathcal{H}_1(sk_{\text{IDAK}})$. $\mathcal{S}$ computes $a_{i,j}^s = \text{MAC}_{K_2^{i,j}}(\text{ID}_i, \text{ID}_j, R_i, R_j)$ and lets $\Pi_{i,j}^s$ reply with $a_{i,j}^s$, where R_j is the random component received from the other oracle.

For a $\mathbf{Reveal}(\Pi_{i,j}^s)$ query, if "$i = I$ and $j = J$ and $s = s_0$" or "$\Pi_{i,j}^s$ is the matching oracle for $\Pi_{I,J}^{s_0}$" then $\mathcal{S}$ aborts the simulation. Otherwise, $\mathcal{S}$ computes the session keying material sk_{IDAK}, runs the $\mathcal{H}_1$-**queries** to get $K_1^{i,j} = \mathcal{H}_1(sk_{\text{IDAK}})$, and responds with $K_1^{i,j}$. For a $\mathbf{Corrupt}(i)$ query, if $i = I$ or $i = J$, then $\mathcal{S}$ aborts the simulation. Otherwise, $\mathcal{S}$ responds with $d_{\text{ID}_i} = g_{\text{ID}_i}^{\alpha}$.

Claim: If $\mathcal{S}$ does not abort the simulation, then $\mathcal{A}$'s view is identical to its view in the real attack.

Proof of Claim: It is straightforward. □

Suppose that the simulation process makes at most q_E H-**queries** and q_N be the maximum number of IDAKC key agreement sessions that the algorithm $\mathcal{A}$ initiates for each participant. We next calculate the probability that $\mathcal{S}$ succeeds in forging an MAC tag on a message that the challenger has not authenticated.

We first calculate the probability that $\mathcal{S}$ does not abort the simulation. The probability that $\mathcal{S}$ does not abort for **Send** queries is $(q_E^2 q_N - 2)/q_E^2 q_N$. The probability that $\mathcal{S}$ does not abort for **Reveal** queries is $(q_E^2 q_N - 2)/q_E^2 q_N$. The probability that $\mathcal{S}$ does not abort for **Corrupt** queries is $(q_E-2)/q_E$. Therefore, the probability that $\mathcal{S}$ does not abort during the simulation is $(q_E-2)(q_E^2q_N-2)^2/q_E^5q_N^2$.

If the algorithm $\mathcal{A}$ is successful during that simulation (the probability is at least ε_k), then there is a completed/accepted oracle $\Pi_{i,j}^s$ that has no matching oracle. Since there are at most $q_E^2q_N$ oracles during the simulation, the probability for this oracle to be the oracle $\Pi_{I,J}^{s_0}$ is $1/q_E^2q_N$. Thus the probability that the oracle $\Pi_{I,J}^{s_0}$ is marked as completed/accepted is at least

$$\left((q_E-2)(q_E^2q_N-2)^2/q_E^5q_N^2\right)\cdot\varepsilon_k\cdot\left(1/q_E^2q_N\right)=\varepsilon_k(q_E-2)(q_E^2q_N-2)^2/q_E^7q_N^3.$$

If the oracle $\Pi_{I,J}^{s_0}$ is marked as completed/accepted, then $\mathcal{S}$ output a guessed valid MAC tag $a_{J,I}^s$ for the message $m=(\mathrm{ID}_J,\mathrm{ID}_I,R_J,R_I)$. We next calculate the probability that the challenger has never been asked for the MAC tag on this message and the probability that $\mathcal{A}$ does not guess correctly about the keying materials held by the oracle $\Pi_{I,J}^{s_0}$ (that is, the probability that the MAC tag is generated without knowing the secret key or asking the challenger to generate it). Since there is no matching oracle and $\mathcal{A}$ is not allowed to ask the challenger for MAC tags, $\mathcal{A}$ generates this tag $a_{J,I}^s$ by one of the following three approaches: (1). $\mathcal{S}$ asked the challenger to generate the MAC tag for the message $m=(\mathrm{ID}_J,\mathrm{ID}_I,R_J,R_I)$ for another oracle $\Pi_{J,I}^{s'}$. Since $\Pi_{J,I}^{s'}$ is not the matching oracle for $\Pi_{I,J}^{s_0}$, the event in this case happens only with probability $1/2^{2k}$. Here we assume that the messages R_I and R_J are all k bits long. (2). $\mathcal{A}$ guessed correctly about the session keying material sk_{IDAK} for the oracle $\Pi_{I,J}^{s_0}$ and computed the MAC tag $a_{J,I}^s$ by herself. By Theorem 5.2, this probability is bounded by some negligible value δ_k. (3). $\mathcal{A}$ generated the MAC tag $a_{J,I}^s$ by random choice or by using other techniques (e.g., by using flaws in the MAC scheme). According to the security definition of MAC schemes, the forgery on the MAC tag is successful when the events in case (3) happens. Thus, by excluding the probabilities for the cases (1) and (2), the probability that MAC forgery experiment is successful under the condition that the oracle $\Pi_{I,J}^{s_0}$ is marked as completed/accepted is at least $(1-(1/2^{2k}))(1-\delta_k)=(2^{2k}-1)(1-\delta_k)/2^{2k}$. In a summary, the probability that $\mathcal{S}$ successfully forged the MAC code on the un-authenticated message $m=(\mathrm{ID}_J,\mathrm{ID}_I,R_J,R_I)$ is at least

$$\varepsilon_k(2^{2k}-1)(1-\delta_k)(q_E-2)(q_E^2q_N-2)^2/q_E^7q_N^32^{2k}$$

which is non-negligible since ε_k is non-negligible and δ_k is negligible. This completes the proof of the Theorem. □

9 Practical considerations and applications

9.1 The function π

Though in the security proof of IDAK key agreement protocol, π is considered as a random oracle. In practice, we can use following simplified π functions.

- π is a random oracle (secure hash function) from $G\times G$ to $Z^*_{2^{\lceil\log q\rceil/c}}$ (e.g., $c=2$).
- If $g_1=(x_{g_1},y_{g_1}),g_2=(x_{g_2},y_{g_2})\in G$ are points on an elliptic curve, then let $\pi(g_1,g_2)=\bar{x}_g \bmod 2^{|x_g|/2}$ where $\bar{x}_g=x_{g_1}\oplus x_{g_2}$. That is, $\pi(g_1,g_2)$ is the exclusive-or of the second half parts of the first coordinates of the elliptic curve points g_1 and g_2.

- π is a random oracle that the output only depends on the the first input variable or any of the above function restricted in such a way that the output only depends on the the first input variable. In another word, $\pi : G \to Z_q^*$.

It should be noted any π function, for which Lemma 5.1 holds, can be used in the IDAK protocol. Though we do not know whether Lemma 5.1 holds for π functions that we have listed above, we have strong evidence that this is true. First, if we assume that the group G_2 is a generic group in the sense of Nechaev [18] and Shoup [27]. Then we can prove that Lemma 5.1 holds for the above π functions. Secondly, if the distribution $\mathcal{G}(g^x, g^r, g^\alpha, g^\beta, g^\gamma, g^{\beta x_0})$ in Lemma 5.1 is restricted to the distribution:

$$\{g^{f(x,r,\alpha,\beta,\gamma,\beta x_0,\mathbf{y})} : f \text{ is a linear function, } \mathbf{y} \text{ is a tuple of uniformly random values from } Z_q\}.$$

Then we can prove that Lemma 5.1 holds for the above π functions. We may conjecture that the adversary algorithm $\mathcal{A}$ can only generate $g_{\mathcal{A}}$ and $g^{\gamma y_0}$ according to the above distribution unless CDH-Assumption fails for G. Thus, under this conjecture (without the condition that G_2 is a generic group), the above list of π functions can be used in IDAK protocol securely.

9.2 Escrow

In the IDAK key agreement protocol, one has to trust the trusted authority TA since TA has sufficient capability to impersonate any participants and to compute the agreed secrets for any key agreement sessions unless the shared secret is computed as $H(g_{\mathrm{ID}_A}^{xy}||sk_{AB})$ as described in section §6. As mentioned in [6], identity-based systems have the natural property to be escrowed. For example, we assume that there are m TAs in the systems, each of them holding a partial master secret α_i, and the master secret is $\alpha_1 + \ldots + \alpha_m$. Each participant could get her partial private key $g_{\mathrm{ID}}^{\alpha_i}$ from TA_i and compute her private key as $g_{\mathrm{ID}}^{\alpha_1} \cdots g_{\mathrm{ID}}^{\alpha_m}$.

9.3 Applications

IDAK key agreement protocol could be used in all these environments that identity-based public parameters are deployed (e.g., these environments discussed in [6]). One of the most promising applications could be the VoIP environments. VoIP systems are become more and more popular. However, Internet environment is generally not as secure as the traditional phone networks. Eavesdropping is dramatically easy in Internet environments than in traditional phone networks. Though VPN could be one of the potential tools that could be used to protect the VoIP systems, recent experiments show that there are many disadvantages for VPN based VoIP (the most important one is the delays in several routers which could worsen VoIP quality). On the other hand, we really do not expect each VoIP phone will get a public key certificate and each time when we make a phone call, we need to import the certificate for the target phone first. Identity based key agreement protocol provides a promising solution for VoIP systems. The public key for each phone could be based on its identity (e.g., the phone number). Each time, when we make a phone call, the two phones will use the IDAK protocol to establish a session key for conversation encryption/authentication. The public key for each phone could be "permanent" (e.g., based on the phone number) or temporary (e.g., based on the identity consisting of phone number and time-stamps).

10 Related protocols and their insecurity

10.1 Smart protocol

Smart [29] proposed an identity-based and authenticated key agreement protocol without security proofs. Briefly, Smart's protocol works as follows: The trusted authority needs to publish the public key g^α first (note that our protocol does not require a public key) and distributes the private keys $g^\alpha_{\mathrm{ID}_A}$ and $g^\alpha_{\mathrm{ID}_B}$ to Alice and Bob respectively. During the key agreement session, Alice selects $x \in_R Z_q^*$ and sends g^x to Bob, Bob selects $y \in_R Z_q^*$ and sends g^y to Alice. Then both parties compute the shared secret $sk_{NS} = \hat{e}(g^x_{\mathrm{ID}_B} \cdot g^y_{\mathrm{ID}_A}, g^\alpha) = \hat{e}(g^x_{\mathrm{ID}_B}, g^\alpha) \cdot \hat{e}(g^y_{\mathrm{ID}_A}, g^\alpha) = \hat{e}(g^x_{\mathrm{ID}_B}, g^\alpha) \cdot \hat{e}(g^\alpha_{\mathrm{ID}_A}, g^y) = \hat{e}(g^\alpha_{\mathrm{ID}_B}, g^x) \cdot \hat{e}(g^y_{\mathrm{ID}_A}, g^\alpha)$. A simple analysis shows that Smart's protocol requires the computation of two exponentiations and two pairings for each party. Meanwhile, the only pre-computation that each party could do is to select the random value x (respectively, y) and compute the value of g^x (respectively, g^y). Thus with pre-computation, Smart's protocol still requires one exponentiation and two pairings for each party. It is straightforward to show that Smart's protocol is not secure against key revealing attacks and does not have perfect forward secrecy if both parties' private keys were leaked.

10.2 Chen and Kudla protocol

Chen and Kudla [7] proposed an efficient identity-based and authenticated key agreement protocol. Briefly, Chen-Kudla's protocol works as follows: The trusted authority distributes the private keys $g^\alpha_{\mathrm{ID}_A}$ and $g^\alpha_{\mathrm{ID}_B}$ to Alice and Bob respectively (similar to our protocol, no public key is required). Alice selects $x \in_R Z_q^*$ and sends $g^x_{\mathrm{ID}_A}$ to Bob, Bob selects $y \in_R Z_q^*$ and sends $g^y_{\mathrm{ID}_B}$ to Alice. Then both parties compute the shared secret $sk_{CK} = \hat{e}(g_{\mathrm{ID}_B}, g_{\mathrm{ID}_A})^{(x+y)\alpha} = \hat{e}(g^x_{\mathrm{ID}_B} \cdot g^y_{\mathrm{ID}_B}, g^\alpha_{\mathrm{ID}_A}) = \hat{e}(g^\alpha_{\mathrm{ID}_B}, g^x_{\mathrm{ID}_A} \cdot g^y_{\mathrm{ID}_A})$. Analysis shows that this protocol requires the computation of two exponentiations and one pairing for each party. Meanwhile, the only pre-computation that each party could do is to select the random value x (respectively, y) and compute the value of $g^x_{\mathrm{ID}_A}$ (respectively, $g^y_{\mathrm{ID}_B}$). Thus with pre-computation, this protocol still requires one exponentiation and one pairing for each party.

In the random oracle model, Chen and Kudla [7] described a randomized reduction from the exact computation problem of shared secret sk_{CK} in the Chen-Kudla protocol to the problem of computational bilinear Diffie-Hellman problem. Indeed, Chen and Kudla showed that if an additional hash function H' (a second independent random oracle) is applied to the shared secret sk_{CK} to get the keying material $sk'_{CK} = H'(sk_{CK})$, then sk'_{CK} is computationally indistinguishable from a random string. In another word, the security is showed in the random oracle plus CBDH-Assumption model. In the literature [5, 17, 28], it is also preferred to show that a key agreement protocol is secure in the decisional DH-Assumption model without the additional random oracle. One disadvantage of Chen-Kudla protocol is that this protocol does not have the perfect forward secrecy property. That is, if the private keys of Alice and Bob are corrupted at some time, then the adversary can compute all past session keys used between Alice and Bob. Another serious disadvantage of Chen-Kudla protocol is that its security is indeed unproved. Chen and Kudla [7] proved that their protocol is secure in the Bellare-Rogaway [3] secure key agreement model. However, Cheng et al. [8] pointed out that the proof in [7] is flawed and their protocol is not secure against key revealing attacks. Since the key revealing attack is the fundamental property in Bellare-Rogaway model [3], a security model for key agreement protocol without modelling key revealing attacks has limited value. For example, in such a limited model, it is impossible to infer whether the key agreement protocol is secure against important attacks such as known session key attacks and unknown key share attacks.

10.3 Scott protocol

Scott [24] proposed a key exchange protocol with password authentications for the private key. Briefly, Scott's protocol works as follows: The trusted authority needs to choose a master secret α and distributes the private keys $g^{\alpha}_{\mathrm{ID}_A}$ and $g^{\alpha}_{\mathrm{ID}_B}$ to Alice and Bob respectively. Alice may choose a password a to store her private key as: $g^{\alpha-a}_{\mathrm{ID}_A}$. In the following discussion, we will omit the password protection part. During the key agreement session, Alice selects $x \in_R Z_q^*$ and sends $\hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^{\alpha x}$ to Bob. Bob selects $y \in_R Z_q^*$ and sends $\hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^{\alpha y}$ to Alice. The shared secret is $\hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^{\alpha xy}$. This protocol is not secure according to Definition 4.1. The adversary may choose a random number c and change the message from Alice to Bob to $\hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^{\alpha xc}$ and change the message from Bob to Alice to $\hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^{\alpha yc}$. Both Alice and Bob will then compute the shared secret $\hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^{\alpha xyc}$. Since the oracle at Alice side is not a matching oracle for at Bob's oracle, the adversary could reveal Bob's session key before testing Alice's session key. Thus the adversary will succeed in the testing query.

10.4 Other protocols

Shim [26] proposed an ID-based key agreement protocol as follows. During the key agreement session, Alice selects $x \in_R Z_q^*$ and sends g^x to Bob. Bob selects $y \in_R Z_q^*$ and sends g^y to Alice. The shared secret is computed as: $\hat{e}(g^{\alpha}, g^{x}_{\mathrm{ID}_B} \cdot g^{y}_{\mathrm{ID}_A} \cdot g^{xy})\hat{e}(g_{\mathrm{ID}_A}, g_{\mathrm{ID}_B})^{\alpha}$, where g^{α} is the system-wide public key. Sun and Hsieh [30] showed that Shim's protocol is insecure against key compromise impersonation attacks and man in the middle attacks.

McCullagh and Barreto [16] proposed an ID-based key agreement protocol as follows. Assume that the system wide master secret is α, Alice's identity is mapped to an integer $a_A \in Z_q^*$, and Bob's identity is mapped to an integer $a_B \in Z_q^*$. Then Alice and Bob's public keys are $g^{\alpha+a_A}$ and $g^{\alpha+a_B}$ respectively. Their secret keys are $g^{(\alpha+a_A)^{-1}}$ and $g^{(\alpha+a_B)^{-1}}$ respectively. During the key agreement session, Alice selects $x \in_R Z_q^*$ and sends $g^{x(\alpha+a_B)}$ to Bob. Bob selects $y \in_R Z_q^*$ and sends $g^{y(\alpha+a_A)}$ to Alice. The shared secret is computed as $\hat{e}(g, g)^{xy}$. Although this protocol is *"proved"* to be secure [16] in Bellare-Rogaway model. Kwang and Choo [14] pointed out that this protocol is not secure against key revealing attacks.

Acknowledgement

The author would like to thank Zhaohui (Michael) Cheng [9] for many useful discussions and for his comments on an earlier version of this paper. In particular, thanks Zhaohui for the comments on the definition of matching oracles and the key compromise impersonation resilience property.

References

[1] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In: *Advances in Cryptology, Crypto 96*, pages 1–15, 1996.

[2] M. Bellare and P. Rogaway. Random oracles are practical: a paradigms for designing efficient protocols. In: *Proceedings of the 1st ACM Conference on Computer Communication Security*, pages 62–73, ACM Press, 1993.

[3] M. Bellare and P. Rogaway. Entity authentication and key distribution. In: *Advances in Cryptology, Crypto 93*, LNCS 773 (1993), 232–249.

[4] M. Blum and S.Micali. How to generate cryptographically strong sequence of pseudo-random bits. *SIAM J. Comput.* **13**:850–864, 1984.

[5] D. Boneh. The decision Diffie-Hellman problem. In: *ANTS-III*, LNCS 1423 (1998), 48–63.

[6] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing* **32**(3):586–615, 2003.

[7] L. Chen and C. Kudla. Identity based authenticated key agreement protocols from pairing. In: *Proc. 16th IEEE Security Foundations Workshop*, pages 219–233. IEEE Computer Society Press, 2003.

[8] Z. Cheng, M. Nistazakis, R. Comley, and L. Vasiu. On indistinguishability-based security model of key agreement protocols-simple cases. In *Proc. of ACNS 04*, June 2004.

[9] Z. Cheng. Private communications 2005.

[10] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, **6**(1976), 644–654.

[11] A. Fiat and A. Shamir. How to prove yourself: practical solutions of identification and signature problems. In: *Advances in Cryptology, Crypto 86*, LNCS 263 (1987), 186–194.

[12] M. Girault and J. Pailles. An identity-based scheme providing zero-knowledge authentication and authenticated key exchange. In: *Proc. ESORICS 90*, pages 173–184.

[13] A. Joux. A one round protocol for tripartite Diffie-Hellman. In: *Algorithmic number theory symposium, ANTS-IV*, LNCS 1838, pages 385–394, 2000.

[14] K. Kwang and R. Choo. Revisit of McCullagh-Barreto two party ID-based authentication key agreement protocols. http://eprint.iacr.org/2004/343.pdf

[15] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, **28**(2):119–134.

[16] P. McCullagh and P. Barreto. A new two-party identity-based authenticated key agreement. *Proc. of CT-RSA 2005*, pages 262-274, LNCS 3376, Springer Verlag, 2005.

[17] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In: *38th Annual Symposium on Foundations of Computer Science*, IEEE Press, 1998.

[18] V. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, **55**(1994), 165–172.

[19] NIST Special Publication 800-56: Recommendation on key establishment schemes, draft 2.0, 2003. http://csrc.nist.gov/CryptoToolkit/kms/keyschemes-Jan03.pdf.

[20] E. Okamoto. Proposal for identity-based key distribution system. *Electronics Letters* **22**:1283–1284, 1986.

[21] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology* **13**(3):361–396, 2000.

[22] E. Ryu, E. Yoon, and K. Yoo. An efficient ID-based authenticated key agreement protocol from pairing. In: *Networking 2004*, pages 1458–1463, LNCS 3042, Springer Verlag, 2004.

[23] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In: *2000 Symp. on Cryptography and Information Security (SCIS 2000)*, Okinawa, Japan 2000.

[24] M. Scott. Authenticated ID-based key exchange and remote log-in with insecure token and PIN number. http://eprint.iacr.org/2002/164.pdf

[25] A. Shamir. Identity-based cryptosystems and signature schemes. In: *Advances in Cryptology, Crypto 84*, LNCS 196, pages 47–53, Springer Verlag 1984.

[26] K. Shim. Efficient ID-based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters* **39**(8):653–654, 2003.

[27] V. Shoup. Lower bounds for discrete logarithms and related problems. In: *Advances in Cryptology, Eurocrypt 97*, LNCS 1233 (1997), 256–266.

[28] V. Shoup. On formal models for secure key exchange. IBM Technical Report RZ 3120, 1999.

[29] N. P. Smart. Identity-based authenticated key agreement protocol based on Weil pairing. *Electronics Letters* **38**(13):630–632, 2002.

[30] S. Sun and B. Hsieh. Security analysis of Shim's authenticated key agreement protocols from pairing. http://eprint.iacr.org/2003/113.pdf

[31] K. Tanaka and E. Okamoto. Key distribution system for mail systems using ID-related information directory. *Computers and Security* **10**:25–33, 1991.