IBAKE: Identity-Based Authenticated Key Exchange Protocol

Vladimir Kolesnikov
Alcatel-Lucent Bell Labs Research
600 Mountain Ave. Murray Hill, NJ 07974, USA
kolesnikov@research.bell-labs.com

Ganapathy S. Sundaram
Alcatel-Lucent
600 Mountain Ave. Murray Hill, NJ 07974, USA
ganesh.sundaram@alcatel-lucent.com

Abstract

The past decade has witnessed a surge in exploration of cryptographic concepts based on pairings over Elliptic Curves. In particular, identity-based cryptographic protocols have received a lot of attention, motivated mainly by the desire to eliminate the need for large-scale public key infrastructure.

We follow this trend in this work, by introducing a new Identity-Based Authenticated Key Exchange (IBAKE) protocol, and providing its formal proof of security. IBAKE provides mutually-authenticated Key Exchange (AKE) using identities as public credentials.

One identity-based AKE subtlety that we address in this work is the resilience to the man-in-the-middle attacks by the Key Management Service. For efficiency, we employ two Elliptic Curves with differing properties. Specifically, we use a combination of a super-singular and non-super-singular curves, where the super-singular curve is used as an identity-based encryption "wrapper" to achieve mutual authentication, and the resulting session key is based on a Diffie-Hellman key exchange in the non-super-singular curve.

We provide a detailed proof of security of the resulting protocol with respect to (our own natural adaptation and simplification of) the AKE definitions of Kolesnikov and Rackoff.

1 Introduction

Since the discovery of identity-based encryption [BF03] (conference version [BF01]), much attention has turned to the exploration of cryptographic concepts based on pairings over Elliptic Curves, and in particular, their applications to general identity-based cryptographic protocols. Following this trend, we re-visit the problem of securing sessions between two parties over an insecure network. Specifically, we address the problem of two-party mutually-Authenticated Key Exchange (AKE) in the identity-based public key cryptographic setting.

The AKE problem has a rich history and research legacy. Based on the type of credential used, there are two categories of protocols - symmetric and asymmetric. Our approach falls into the latter category; in addition, instead of relying on more conventional public key tools such as RSA (and the required costly public-key infrastructure), we construct an efficient AKE protocol in an the identity-based cryptographic setting.

In our approach, we use two different Elliptic Curves, for authentication and session key generation respectively. Specifically, we use a combination of a super-singular and non-super-singular curves, wherein the super-singular curve is used as an identity-based encryption "wrapper" to achieve mutual authentication, and the resulting session key is based on a Diffie-Hellman key exchange in the non-super-singular curve. We provide a detailed proof of security of the resulting protocol. The definitions of security we employ are natural adaptations and simplifications of the definitions proposed recently by Kolesnikov and Rackoff [KR06, KR08].

1.1 Identity-Based Cryptography

The concept of Identity-Based Cryptography, wherein the "identity" (e.g., username, domain name, etc.) is used as a public key, and the corresponding private key is derived from the identity and a master secret, owes its origins to Shamir [Sha85]. The main attraction of this approach is the avoidance of the expensive public-key infrastructure (PKI), needed to certify the binding of a public key to an identity.

The idea of identity-based cryptography remained unrealized until 2001, when Boneh and Franklin [BF01] proposed an implementation based on Weil pairing, soon followed by Joux [Jou02]. These seminal works have catalyzed new areas of exploration, and new public key encryption systems that avoid the use of large-scale Certificate Authorities (CA) (and related PKI) have been proposed.

The riddance of the CAs does not come for free. In ID-based setting, players' private keys cannot be generated by the players themselves, since the generation involves a global master secret. Instead, private keys are computed by a Key Management Service (KMS), and transmitted to players upon identity verification. Thus, in contrast with CAs in the traditional public-key setting, where players can generate their keys themselves, the KMS knows all the keys in the ecosystem, and must be trusted not to abuse this knowledge. In essense, the KMS must be treated as a key escrow entity.

1.2 IBAKE and Our Contribution

The design of our protocol can be explained using an "inside out" approach. Consider two parties exchanging a key in an insecure environment. The natural protocol to consider in this setting is the Diffie-Hellman protocol [DH76] over a group. In our protocol, we rely on the Elliptic Curve Diffie-Hellman protocol as proposed by Koblitz [Kob87] and Miller [Mil86][1]. However, Diffie-Hellman suffers from a man-in-the-middle attack, as it does not authenticate communicating entities. In order to address this, we employ identity-based

[1]This choice was made with a view towards reducing the complexity of the key exchange computation.

encryption to "wrap" the DH messages, protecting their privacy, and ensuring that only the designated player is able to decrypt them and perform the following steps of the protocol.

Our Contribution. IBAKE follows a natural approach to AKE design. Our contribution is the first, to our knowledge, AKE protocol with detailed proof of security, which explicitly and efficiently combines the identity-based credentials with the the resilience against corrupt KMS/Key Escrow, by using two curves with different properties. Our detailed formal proof of security is based on our AKE definition, derived from one recently proposed by Kolesnikov and Rackoff [KR06, KR08].

We note that our protocol is secure even if the two participants are attached to two different KMSs. We optimize the performance of our protocol; a report on the implementation, its efficiency and a comparison with existing protocols is forthcoming.

1.3 Related Work

We now briefly compare our protocol to existing AKE solutions. Firstly, we note that the immense body of work on symmetric, password-based, and public-key-based AKE protocols does not achieve our goals, due to their inapplicability in identity-based setting. Namely, symmetric and password-based protocols require key setup among each pair of players, quickly overwhelming the storage resources in many settings. Further, as argued above in Section 1.1, traditional public-key-based solutions, such as [MQV95], require the use of costly CAs, which also does not scale in all settings.

Prior work on identity-based AKE, e.g., [Sma02, CCS06], of course, resolves the CA issue. However, identity-based AKE protocols often do not address the key escrow problem discussed above in Section 1.1. That is, the trusted KMS who knows private keys of all players, can *post factum* compute all the session keys in the system, and break into all communication. By using a DH exchange inside the identity-based IKE, we ensure protection against this threat.

1.4 Organization

The rest of this paper is organized as follows. In Section 2 we present the definitions of security relevant to our construction and present the construction and security proofs in Section 3.

We note that Section 3 starts with the protocol presentation, some additional intuition for the design, security claim and proof intuition. Thus, the beginning of Section 3 could be a good starting point for the reader looking to get straight to the protocol, its precise security claim, and high-level intuition.

2 Definition of Security of KE

In this section, we present the definition of security we use in our proofs. This definition is a natural adaptation (actually, mostly simplification) of the definitions of Kolesnikov and Rackoff [KR06, KR08]. The latter definitions consider a substantially more general setting of multi-factor authenticated KE, where parties possess both long and short keys (e.g. PSK

and passwords). The definitions of [KR06, KR08] have the graceful degradation property, that is, a compromise of some of the keys results in the security level accorded by the remaining key(s). Naturally (also indirectly implied in [KR06, KR08]), omitting the use of short keys results in the definition for our setting.

The main difference in our setting is that we consider peer-to-peer, while [KR06, KR08] considers client-server setting. This implies the corresponding amendment in the definition. Namely, instead of creating identities of honest client and a server, here we create identities of two honest players (we still allow a polynomial number of malicious identities).

While one can use one of several KE definitions, we found game-based definition to be the simplest to use. Further, specifically for the reduction of the IBAKE protocol, they are most convenient, since the security of the IBE (identity-based encryption) primitives is defined as a game [BF03] (presented, for convenience, in Appendix A).

For completeness and the formalization of discussion, we now present the adapted definition that we will use.

We denote by $\mathcal{P}_{k,i}^{\mathcal{P}_n}$ the i-th instance of player $\mathcal{P}_k$ who wants to talk to (some instance of) player $\mathcal{P}_n$.

KE definitions rely on the notion of *partners* to specify what constitutes an attack. Informally, two instances of players are partners, if they establish a secure channel. Syntactically, we define partners as follows.

Definition 1 *We say that an instance $\mathcal{P}_{k,i}^{\mathcal{P}_n}$ of a player $\mathcal{P}_k$ who wants to talk to an instance of $\mathcal{P}_n$ and an instance $\mathcal{P}_{n,j}^{\mathcal{P}_k}$ of a player $\mathcal{P}_n$ who wants to talk to an instance of $\mathcal{P}_k$ are* partners*, if they have output the same session id sid.*

Session id *sid* is an additional (somewhat artificial) output of KE, which need not be used in real execution, but which is needed for syntactic manipulations in the proof. We omit the detailed discussion of the need of *sid* and refer the interested reader to literature, e.g., [KR06, KR08] for additional information.

We start by presenting the KE game, which model attacks of a real-life adversary.

Game KE. *First Key Management Service (KMS) is initialized, and its* `setup` *algorithm is run. Then Adv runs players by executing steps 1-5 multiple times, in any order:*

1. *Adv creates an honest player $\mathcal{P}_k$. Adv is allowed to pick any unused name $N_{\mathcal{P}_k}$ for $\mathcal{P}_k$. $\mathcal{P}_k$ is registered with KMS, and a private key is set up and associated with it based on its name.*

2. *Adv creates a corrupt player $\mathcal{B}_i$. Adv is allowed to pick any unused name $N_{\mathcal{B}_i}$ for $\mathcal{B}_i$. $\mathcal{B}_i$ is registered with KMS, and a private key is set up and associated with it based on its name. The generated private key is given to Adv.*

3. *Adv creates an instance $\mathcal{P}_{k,i}^{\mathcal{P}_n}$ of the honest player $\mathcal{P}_k$ who wants to talk to (good or bad) player $\mathcal{P}_n$. $\mathcal{P}_{k,i}$ is given (secretly from Adv) as input his private key and the name of his intended partner $N_{\mathcal{P}_n}$.*

4. *Adv delivers a message m to an honest party instance. The instance immediately responds with a reply (by giving it to Adv) and/or terminates and outputs the result (a sid and either the session key, or the* failure symbol $\perp$*) according to the protocol. Adv learns only the sid part of the output.*

5. *Adv "opens" any successfully completed and checked honest instance – then he is given the session key output of that instance.*

Then Adv asks for a challenge on an instance of an honest player.

The challenge of instance $\mathcal{P}_{k,i}^{\mathcal{P}_n}$ is handled as follows. $\mathcal{P}_{k,i}^{\mathcal{P}_n}$, who has been instantiated to talk to another honest player $\mathcal{P}_n$, must have completed and output a session key. The challenge is, equiprobably, either the key output by $\mathcal{P}_{k,i}^{\mathcal{P}_n}$ or a random string of the same length. Adv must not have opened $\mathcal{P}_{k,i}^{\mathcal{P}_n}$ or his partner, and is not allowed to do it in the future.

Then Adv continues to run the game as before (execute steps 2-5). Finally, Adv outputs a single bit b which denotes Adv's guess at whether the challenge string was random. Adv wins if he makes a correct guess, and loses otherwise. Adv cannot "withdraw" from a challenge, and must produce his guess.

The above game is almost sufficient for security definition. The only remaining technical aspect is the enforcement of non-triviality. We need to prevent improper partnering (e.g. players unnecessarily outputting same *sid*). Recall, *Adv* is not allowed to challenge parties whose partner has been opened; SID ensures that *Adv* is not unfairly restricted. We handle this by introducing the following game

Game SID *is derived from game KE by adjusting the win condition, as follows (and otherwise is identical). Adv does not ask for (nor answers) the challenge. Adv wins if any two honest partners output different session keys.*

Note, SID allows for one (or both) of the partners to output a failure symbol. *Adv* only wins if two successfully completed parties output different session keys.

We are now ready to present the definition.

Definition 2 *We say that a key exchange protocol Π is secure, if for every polytime adversaries Adv_1, Adv_{sid} playing games KE and SID, their probabilities of winning (over the randomness used by the adversaries, all players and generation algorithms) is at most only negligibly (in security parameter n) better than:*

- $1/2$*, for KE,*
- 0*, for SID.*

Resilience against corrupt KMS. We do not formally introduce the property of resilience to KMS/key escrow service corruption into our definition. A natural way to do so would be to add the requirement that the following two distributions D_1, D_2 are indistinguishable. D_1 includes the master key, a completed AKE execution transcript and the session key; D_2 is defined exactly as D_1, with the exception that instead of the session key, it includes a random string of appropriate length. We note that a single transcript here

is sufficient. That is, a protocol cannot be constructed in a way that a distinguisher wins if he has access to two AKE transcripts, but not when he has access to only one. This is because any number of additional transcripts can be generated by the distinguisher himself, when given the master key.

We will show that our protocol has this corrupt KMS resilience property in Section 3.1.

3 IBAKE Essential Exchange and its Proof of Security

We now formally state and prove the security theorem of our proposed IBAKE Protocol.

For simplicity, we consider only the core of the protocols, leaving out the non-security-essential messages. It is clear that the following Protocol 1 represents the full IBAKE Protocol [CS11], and the latter is secure if so is the former. We will thus consider and prove security of the following protocol.

Protocol 1 *(Essential message exchange of the IBAKE Protocol)*

A		B
given curve, point p		*given curve, point p*
select a random point a		
	$IBEnc_B(A, B, ap) \rightarrow$	
		select a random point b
	$\leftarrow IBEnc_A(B, A, ap, bp)$	
verify received ap		
if fail, abort		
	$IBEnc_B(A, B, bp) \rightarrow$	
		verify received ap
		if fail, abort
set $sk = KDF(abp)$		*set* $sk = KDF(abp)$
set $sid = (ap, bp)$,		*set* $sid = (ap, bp)$,
Output (sk, sid)		*Output* (sk, sid)

Theorem 1 *Let IBEnc be a CCA2 secure encryption scheme according to the definition of [BF03]. Then, protocol 1 is a secure KE protocol, according to the Definition 2, in the Random Oracle (RO) model, where KDF is a RO.*

As mentioned above, the proof of Theorem 1 implies the security of the IBAKE Protocol [CS11].

Similarity to [BR94]. Firstly, observe that our protocol closely resembles the Bellare-Rogaway (BR) KE protocol [BR94]. There are several differences. First, the BR protocol is based on symmetric keys. This, in particular, implies that MitM *Adv* cannot create new encrypted and MAC'ed messages, and only can forward messages that he already saw in previous executions. A consequence of this is that we need to encrypt the first message

from the initiator (BR didn't), for the following reason. If the message was not encrypted, then MitM *Adv* could easily forge the response and successfully assume any identity.

Further, we use the DH KE inside the exchange protected by IBE. This is to prevent a KMS from *post-factum* decrypting the message exchange, deriving the exchanged keys and breaking into the session. We will make an (easy) argument that our technique is sufficient for this goal in Section 3.1.

Proof intuition: It is relatively easy to see at a high level that IBAKE is a secure KE protocol. Importantly, we use a strong CCA2-secure IBE encryption, thus preventing *Adv* from being capable of generating encryptions, under keys of honest players, of anything related to what was generated by honest players. In particular, after having seen and captured the message $IBEnc_A(B, A, ap, bp)$ (e.g., sent by an instance of B), *Adv* provably cannot generate a message $IBEnc_A(B, A, \hat{a}p, bp)$ for any $\hat{a}$. Therefore, inclusion of ap in the second message implies that the intended player B generated the second message in response to the first message, which B must have decrypted, and thus the included bp came from B. Similarly, receipt of the encryption bp in the third message confirms to player A that the earlier message was obtained and decrypted by the "right" instance of his partner B.

Our formal proof relies on game-based definitions of security. We show that no adversary *Adv* can win in the game where he is challenged to distinguish from random the session key output by an honest party. We do this by constructing an IBE encryption adversary[2] who uses the KE adversary and wins "often" if the KE adversary wins "often". That is, IBE adversary translates the IBE game and challenge into the KE game and challenge, has KE Adv solve it, and then translates the solution back to the CCA game to win it.

For simplicity, we actually prove the theorem with respect to the slightly modified IBE game G, defined as follows. It is easy to see (and we will later argue) that the IBE game and the game G describe the same class of secure encryption functions.

IBE Game G. *(A variant of the IBE game of [BF03], cf. Appendix A.)*
Denote the slightly modified CCA2-security game of IBE as G. G is the same as the CCA2 IBE game, except in the the challenge query, instead of one tuple of $\langle ID, M_0, M_1\rangle$, G takes two such tuples $\langle ID_1, M_0^1, M_1^1\rangle, \langle ID_2, M_0^2, M_1^2\rangle$ (possibly with two different identities, and, correspondingly returns two encryptions, corresponding to the same randomly chosen bit b. Of course, further `decrypt` *queries are restricted to not accept either of the two produced encryptions (with corresponding ID's.*

It is easy to see (via a standard hybrid argument) that the game G guarantees the same security properties as the CCA2 IBE game.

Observation 1 *Jumping ahead, we note that we introduce game G to make clearer the following aspect of the proof. We will use message* 2 *of the protocol, sent by the responder, to translate the IBE game challenge to the KE game challenge, which we will then ask Adv to solve. However, initiator is expected to respond to this message, which we can't simulate statistically, since we can't call the IBE game's* `decrypt` *query on this message.*

[2]See definitions in [BF03], also included in Appendix A for convenience.

Our solution is to get the correct initiator's response from the (modified) IBE game G, by allowing to pass two pairs of ciphertexts in the `challenge` *query.*

Lemma 1 *Let IBEnc be an encryption scheme secure according to the above game G. Then, protocol 1 is a secure KE protocol, according to the Definition 2, in the Random Oracle (RO) model, where KDF is a RO.*

Proof of Lemma 1:

Suppose the contrary, that is that the above protocol is not secure, and there exists a polytime *Adv* who gains a non-negligible advantage in game KE.

We present a distinguisher *Dist* and show that it gains a corresponding non-negligible advantage in G. *Dist* simulates an environment (i.e. KE players and their actions), in which he runs *Adv*, answers *Adv*'s queries and uses *Adv*'s decisions to make decisions in G. We say "*Dist* stops", meaning "*Dist* finishes processing *Adv*'s request and returns control to *Adv*", and "*Dist* sends (outputs) m", meaning "*Dist* simulates the given player sending (outputting) m, by giving m to *Adv*".

Dist starts up *Adv* and gives him the public key of KMS, which he received from G. *Dist* then runs *Adv* and satisfies its requests for information as follows. Note that a player must have been created to create its instances. We also note that while we carefully handle how honest players respond to messages and responses (allegedly) coming from the honest players – this is how we reduce the KE game to G – *Dist* mostly simply executes the protocol to handle messages (allegedly) coming from corrupt players. Overall, *Adv*'s view of the interaction with *Dist* will be computationally close to *Adv*'s view of a real KE game. It is important to ensure that this is indeed so, as otherwise a clever *Adv* may "refuse to help *Dist* in his playing G".

Informally, in our "translation" of *Adv*'s ability to successfully answer a KE Game challenge into the setting of IBE security game G, we will have *Dist* play G, select a challenge based on a message that a KE instance may send, and "hope" that *Adv* will choose the same instance for his challenge. If he does, as we will see, *Dist* wins in G essentially whenever *Adv* wins KE. If *Adv* chooses another instance for his challenge, we cannot take advantage of our setup, and *Dist* simply makes a random guess for G. Because *Adv* can not infer which instance is associated with a challenge, he will be forced to choose the G-challenge-related instance appropriately often. That is, if we bound the total number of honest instances that *Adv* is allowed to create by q, then the probability of *Adv* challenging the "right" instance is at least $1/q$.

Variables notation. In the following, we will use indexed variables. We will adhere to the following notation. Messages received by players will have indices indicating the alleged origin, and also will have a "hat" to denote that they have been possibly modified in transit. For example, an encryption E_1 sent by instance $\mathcal{P}_{k,i}^{\mathcal{P}_n}$ will be denoted by $E_{1,\mathcal{P}_{k,i}^{\mathcal{P}_n}}$. Upon delivery to instance $\mathcal{P}_{n,j}^{\mathcal{P}_m}$ the (same or possibly different) string will be denoted $\hat{E}_{1,\mathcal{P}_{n,j}^{\mathcal{P}_m}}$, since it is claimed to be

1. *Adv creates a bad player B_i:*

Dist runs the extraction query of game G (of Phase 1) by giving B^i's name to G. *Dist* obtains the corresponding private key from G, stores it, and gives it to *Adv*.

2. *Adv creates an honest player $\mathcal{P}_{k,i}^{\mathcal{P}_n}$:*
   No action necessary (player's public key is his name). We can't run extraction query of G, as this player's keys may be the selected G's challenge.

3. *Adv creates an initiator instance $\mathcal{P}_{k,i}^{B_j}$ of honest player $\mathcal{P}_k$ talking to corrupt player B_j and starts the protocol:*
   Dist follows the protocol. That is, *Dist* selects a random point a, and generates and gives to *Adv* the encryption $E_{1,\mathcal{P}_{k,i}^{B_j}} = Enc_{B_j}(\mathcal{P}_k, B_j, ap)$.

4. *Adv creates an initiator instance $\mathcal{P}_{k,i}^{\mathcal{P}_n}$ of $\mathcal{P}_k$ talking to honest $\mathcal{P}_k$ and starts the protocol:*
   Dist generates and gives to *Adv* the encryption of the first message, according to the protocol.

5. *Adv creates a responder instance $\mathcal{P}_{k,i}$ of the honest player $\mathcal{P}_k$.*
   No action needed from *Dist*.

6. *Adv delivers a first message $m_{\mathcal{P}_{n,i}} = (\hat{E}_1)$ to an instance $\mathcal{P}_{n,i}^{\mathcal{P}_k}$ of honest responder $\mathcal{P}_n$ (allegedly) from honest player $\mathcal{P}_k$:*
   If *Dist* has not seen this specific encryption $\hat{E}_1$ or if he has generated it himself (but not via a `challenge` query with identity $\mathcal{P}_n$ – see below for how and when `challenge` queries are executed), *Dist* decrypts $m_{\mathcal{P}_{n,i}}$ by executing the decryption query of G with parameters $\langle m_{\mathcal{P}_{n,i}}, \mathcal{P}_n \rangle$[3].

   In either of the above cases, *Dist* continues according to protocol. It parses the decrypted value, verifies the player names and extracts $\hat{ap}$. It also generates random b, r from the appropriate group. If next message is *not* selected for the challenge, *Dist* generates and gives to *Adv* the encryption $E_{2,\mathcal{P}_{n,i}^{\mathcal{P}_k}} = Enc_{\mathcal{P}_k}(\mathcal{P}_n, \mathcal{P}_k, \hat{ap}, bp)$.

   However, if the next message is selected for the challenge, *Dist* executes the challenge query of G by passing the two tuples. The first tuple is $M_0^1 = \langle \mathcal{P}_n, \mathcal{P}_k, \hat{ap}, bp \rangle, M_1^1 = \langle \mathcal{P}_n, \mathcal{P}_k, \hat{ap}, rp \rangle$ and the challenge identity $\mathcal{P}_k$. The second tuple is $M_0^2 = \langle \mathcal{P}_k, \mathcal{P}_n, bp \rangle, M_1^2 = \langle \mathcal{P}_k, \mathcal{P}_n, rp \rangle$ and the challenge identity $\mathcal{P}_n$. *Dist* receives two challenge encryptions ce_1, ce_2 under the keys $\mathcal{P}_k$ and $\mathcal{P}_n$ respectively, of one of the two messages from each tuple. *Dist* gives ce_1 to *Adv* at this time.

   In the final option, if *Dist* has obtained $\hat{E}_1$ from a `challenge` query which was associated with $\mathcal{P}_n$, `decrypt` is not allowed by G. In this case, *Dist* might not know the exact content of the plaintext, namely the encrypted group element. However, he does know that the group element is one of the two options he gave as input to `challenge`). Further, *Dist* knows the identity/public-key used to produce the encryption, and the

[3] We can avoid this `decrypt` query if *Dist* had generated $\hat{E}_1$ himself with the public key $\mathcal{P}_n$, since we know the answer already; however this has no effect on the proof.

player identities of the plaintext (this is because they were the same in plaintexts M_0^i, M_1^i submitted inside the challenge tuple). Thus, *Dist* can verify whether the message should result in the protocol outputting failure or an encryption message. *Dist* generates failure simulation in the natural way. In case a response (and not a failure output) is expected, *Dist* is not able to simulate it statistically close, since he does not know which of the group elements is inside $\hat{E}_1$. However, he can simulate the response computationally, as follows. He simply responds with a random encryption $E_{2,\mathcal{P}_{n,i}^{\mathcal{P}_k}} = Enc_{\mathcal{P}_k}(\mathcal{P}_n, \mathcal{P}_k, r_1 p, r_2 p)$. We note that the plaintext of this encryption is computationally indistinguishable from encryption of any other plaintext of the same size. Further, when submitted to any instance of any player, including instances of player $\mathcal{P}_k$, to be indistinguishable from real execution, this message should always trigger failure output, which is easy to simulate. (This is because this message should only be accepted by an instance of $\mathcal{P}_k$, which happened to randomly choose the same ap, an event of negligible probability.)

7. *Adv delivers a first message $m_{\mathcal{P}_{n,i}} = (\hat{E}_1)$ to an instance $\mathcal{P}_{n,i}^{B_j}$ of honest responder $\mathcal{P}_n$ (allegedly) from player B_j:*
   If this message was generated by `challenge` query with identity $\mathcal{P}_n$, we cannot call `decrypt`. However, if so, the identities under encryption will not verify (`challenge` is never called with identity B_j, either as initiator or a responder), so *Dist* outputs failure in this case.

   Otherwise, *Dist* decrypts the message by issuing a `decrypt` query and proceeds according to the protocol. That is, if the message decrypts with appropriate identities, he responds with the encryption of identities and ap, bp. If the message fails to decrypt, he responds with the appropriate error message, according to the protocol.

8. *Adv delivers (the only) message $m_{\mathcal{P}_{k,i}} = (\hat{E}_2)$ to an instance $\mathcal{P}_{k,i}^{\mathcal{P}_n}$ of honest initiator $\mathcal{P}_k$ (allegedly) from player $\mathcal{P}_n$:*
   If it was the challenge encryption ce_1, generated with identity $\mathcal{P}_k$ that was delivered, *Dist* cannot issue `decrypt` query. In this case, he confirms that ce_1 was delivered to the "right" instance. I.e., *Dist* confirms that the player names match and that this instance had generated and sent the corresponding ap, that is that $ap = \hat{ap}$. If this is not the case, *Dist* proceeds to output failure according to the protocol. Otherwise, *Dist* gives ce_2 to *Adv* and outputs sid =transcript of the messages this instance saw.

   Challenge encryption ce_2 should always be rejected at this point, so if ce_2 was delivered, *Dist* simulates failure.

   Otherwise, *Dist* is allowed to `decrypt`, and he does so by executing the decryption query of G with parameters $\langle m_{\mathcal{P}_{k,i}}, \mathcal{P}_k \rangle$. Then, *Dist* continues according to protocol. It parses the decrypted value, verifies the player names and extracts $\hat{ap}, \hat{bp}$. He confirms that $ap = \hat{ap}$. He then responds with the encryption $Enc_{\mathcal{P}_n}(\mathcal{P}_k, \mathcal{P}_n, \hat{bp}$ and outputs sid =transcript of the messages this instance saw.

9. *Adv delivers (the only) message $m_{\mathcal{P}_{k,i}} = (\hat{E}_2)$ to an instance $\mathcal{P}_{k,i}^{B_j}$ of honest initiator*

$\mathcal{P}_k$ (allegedly) from corrupt player B_j:
If this message was generated by `challenge` query with identity $\mathcal{P}_k$, we cannot call `decrypt`. However, if so, the identities under encryption will not successfully verify (`challenge` is never called by *Dist* with identity B_j under encryption), so *Dist* outputs failure in this case.

Otherwise, *Dist* decrypts the message by issuing a `decrypt` query. In either case, he proceeds according to the protocol. If the message decrypts, *Dist* responds to *Adv* and outputs *sid* according to the protocol. Otherwise, he simply simulates error according to the protocol.

10. *Adv delivers the second message $m_{\mathcal{P}_{n,i}} = (\hat{E}_3)$ to an instance $\mathcal{P}_{n,i}^{\mathcal{P}_k}$ of honest responder $\mathcal{P}_n$ (allegedly) from honest player $\mathcal{P}_k$:*
If it was the challenge encryption ce_2 that was delivered, *Dist* confirms that it was delivered to the "right" instance. I.e., *Dist* confirms that the player names match and that this instance had generated and sent the corresponding ce_1. If this is not the case, *Dist* proceeds to output failure according to the protocol. Otherwise, *Dist* outputs *sid* =transcript of the messages this instance saw.

If it was the challenge encryption ce_1 that was delivered, *Dist* simulates failure.

Otherwise, *Dist* decrypts the message by issuing a `decrypt` query and proceeds according to the protocol.

11. *Adv delivers the second message $m_{\mathcal{P}_{k,i}} = (\hat{E}_3)$ to an instance $\mathcal{P}_{k,i}^{B_j}$ of honest responder $\mathcal{P}_k$ (allegedly) from player B_j:*
If this message was generated by `challenge` query with identity $\mathcal{P}_k$, we cannot call `decrypt`. However, if so, the identities under encryption will not successfully verify (`challenge` is never called by *Dist* with identity B_j under encryption), so *Dist* outputs failure in this case.

Otherwise, *Dist* decrypts the message by issuing a `decrypt` query and proceeds according to the protocol.

12. *Adv sends an* open *request on a (completed and not failed or challenged) honest player instance $\mathcal{P}_{k,i}^{\mathcal{P}_n}$ of $\mathcal{P}_k$:*
If this instance completed and has been delivered either ce_1 or ce_2 by *Adv*, then give up and randomly choose the answer to G's challenge. This is because *Adv* can't challenge this instance anymore, nor can he challenge its partner, where the other challenge encryption was delivered.

Otherwise, *Dist* computes *sk* according to the protocol and gives it to *Adv*. Note that *Dist* always has enough information to compute *sk*.

13. *Adv sends a* challenge *request on a (completed and not failed or opened) honest player instance $\mathcal{P}_{k,i}^{\mathcal{P}_n}$ of $\mathcal{P}_k$, whose partner was not opened:*
If this instance has been delivered either ce_1 or ce_2 by *Adv*, then we will translate *Adv*'s response into G.

At this time *Dist* outputs to *Adv* secret key *sk* consistent with G choice[4] $b = 0$, i.e. $sk = KDF(abp)$. Note, if this is guessed correctly, this would correspond to the KE game providing the true session key at the challenge. If guessed incorrectly, and the output should have been $sk = KDF(arp)$, this would correspond to the KE game providing a random string[5].

14. *Adv answers challenge by providing a bit b:*
If *Adv* thinks (i.e. responds) that true session key was provided, *Dist* answers G's challenge with $b = 0$; otherwise, with $b = 1$. As argued above in 13, if *Adv* is correct, then *Dist* is also correct.

First, it is easy to verify (and we provided arguments with the descriptions of *Dist*'s actions) that all calls to G's oracles will be legal requests in G.

Further, as discussed above, the view of *Adv* is indistinguishable from that of a real KE game, so his advantage there can be transferred into winning G. In particular, *Adv* cannot detect that he is being "interrogated" by *Dist*, and eliminate his KE-winning advantage, e.g., by shutting down or giving wrong answers.

By assumption of the theorem, *Adv* wins with probability non-negligibly more than 1/2. We shown that *Dist* wins whenever *Adv* wins on the right challenge (except for a negligible fraction of the time), and breaks even in other cases. Therefore, the constructed *Dist* wins the game G with probability non-negligibly more than 1/2.

□

It remains to show that if there exists an adversary who gains a non-negligible advantage in the game G, there exist an adversary doing same in the IBE game. The proof is done by a standard hybrid argument and is omitted. □

Lemma 1 combined with the above observation constitutes the proof of Theorem 1. □.

3.1 Forward Secrecy with respect to Corrupt KMS

One consideration in all identity-based systems is the possession of all of the players' secret keys by KMS, and its capability to decrypt any message. We address this threat by executing a Diffie-Hellman exchange inside the encrypted messages. Namely, players generate *ap* and *bp*, and set keying material to *abp*, which, based on the DH assumption, is difficult to compute without knowing either *a* or *b*.

It is now clear that, while KMS or its agent can play an *active* man-in-the-middle, it cannot compute the exchanged keys *post-factum*, after the KE has completed. We note that employing DH exchange inside a KE is a standard KE technique to achieve *perfect forward secrecy* [DOW92] – security of the completed sessions even in the case of future compromise of the long-term credentials. Our contribution here is explicit inclusion of this technique, and the performance optimization of its parameters.

[4]Our choice of what *sk* to present to *Adv* is immaterial, as it is effectively random due to the random choice of G's challenge.

[5]This is where modeling KDF as a RO comes into play – we need this assumption as *abp* or *arp* are not distributed as uniform strings.

References

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Berlin, Germany.

[BF03] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Berlin, Germany.

[CCS06] L. Chen, Z. Cheng, and N.P. Smart. Identity-based key agreement protocols from pairings. Cryptology ePrint Archive, Report 2006/199, 2006. `http://eprint.iacr.org/`.

[CS11] V. Cakulev and G. Sundaram. IBAKE: Identity-Based Authenticated Key Agreement. IETF Network Working Group Internet-Draft, 2011. `http://tools.ietf.org/html/draft-cakulev-ibake-03`.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography, 1976.

[DOW92] Whitfield Diffie, Paul C. Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[Jou02] Antoine Joux. The Weil and Tate pairings as building blocks for public key cryptosystems. In *Algorithmic Number Theory, 5th International Symposium*, pages 20–32, 2002.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[KR06] Vladimir Kolesnikov and Charles Rackoff. Key exchange using passwords and long keys. In *Theory of Cryptography, TCC 2006*, volume 3876 of *LNCS*, pages 100–119. Springer, 2006.

[KR08] Vladimir Kolesnikov and Charles Rackoff. Password mistyping in two-factor-authenticated key exchange. In *ICALP (2)*, pages 702–714, 2008.

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO'85*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426, Santa Barbara, CA, USA, August 18–22, 1986. Springer, Berlin, Germany.

[MQV95] A. Menezes, M. Qu, and S. Vanstone. Some new key agreement protocols providing mutual implicit authentication. In *Second Workshop on Selected Areas in Cryptography*, 1995.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1985. Springer, Berlin, Germany.

[Sma02] N.P. Smart. An identity based authenticated key agreement protocol based on the weil pairing. *Electronics Letters*, 38(13):630–632, 2002.

A Definition of IBE security – CCA2 version [BF03]

For completeness, we include the definition of security used in [BF03].

We say that an identity-based encryption scheme E is semantically secure against an adaptive chosen ciphertext attack (IND-ID-CCA) if no polynomially bounded adversary A has a non-negligible advantage against the Challenger in the following IND-ID-CCA game:

- **Setup:** The challenger takes a security parameter k and runs the `Setup` algorithm. It gives the adversary the resulting system parameters `params`. It keeps the master-key to itself.
- **Phase 1:** The adversary issues queries $q_1, ..., q_m$ where query q_i is one of:
  - Extraction query $\langle ID_i \rangle$. The challenger responds by running algorithm `Extract` to generate the private key d_i corresponding to the public key ID_i. It sends d_i to the adversary.
  - Decryption query $\langle ID_i, C_i \rangle$. The challenger responds by running algorithm `Extract` to generate the private key d_i corresponding to ID_i. It then runs algorithm `Decrypt` to decrypt the ciphertext C_i using the private key d_i. It sends the resulting plaintext to the adversary.

  These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, ..., q_{i-1}$.
- **Challenge:** Once the adversary decides that Phase 1 is over it outputs two equal length plaintexts $M_0, M_1 \in M$ and an identity ID on which it wishes to be challenged. The only constraint is that ID did not appear in any private key extraction query in Phase 1. The challenger picks a random bit $b \in \{0, 1\}$ and sets $C =$`Encrypt`(`params`,ID, M_b). It sends C as the challenge to the adversary.
- **Phase 2:** The adversary issues more queries $q_{m+1}, ..., q_n$ where query q_i is one of:
  - Extraction query $\langle ID_i \rangle$ where $ID_i \neq ID$. Challenger responds as in Phase 1.

– Decryption query $\langle ID_i, C_i \rangle \neq \langle ID, C_i \rangle$. Challenger responds as in Phase 1.

These queries may be asked adaptively as in Phase 1.

- **Guess:** Finally, the adversary outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

  We refer to such an adversary A as an IND-ID-CCA adversary. We define adversary A's advantage in attacking the scheme E as the following function of the security parameter k (k is given as input to the challenger): $Adv_{E,A}(k) = |Pr[b = b'] - 1/2|$. The probability is over the random bits used by the challenger and the adversary.

Using the IND-ID-CCA game we can define chosen ciphertext security for IBE schemes. As usual, we say that a function $g : R \to R$ is *negligible* if for any $d > 0$ we have $|g(k)| < 1/k^d$ for sufficiently large k.

Definition 3 *We say that the IBE system E is semantically secure against an adaptive chosen ciphertext attack if for any polynomial time IND-ID-CCA adversary A the function $Adv_{E,A}(k)$ is negligible. As shorthand, we say that E is IND-ID-CCA secure.*