An extended abstract of this paper appears in ProvSec 2012. This is the full version.

Strongly Authenticated Key Exchange Protocol from Bilinear Groups without Random Oracles

Zheng Yang and Jörg Schwenk

Horst Görtz Institute for IT Security
Chair for Network- and Data Security
Ruhr-University Bochum, Germany
{zheng.yang,joerg.schwenk}@rub.de

November 7, 2012

Abstract

Malicious insider security of authenticated key exchange (AKE) protocol addresses the situation that an AKE protocol is secure even with existing dishonest parties established by adversary in corresponding security experiment. In the eCK model, the EstablishParty query is used to model the malicious insider setting. However such strong query is not clearly formalized so far. We show that the proof of possession assumptions for registering public keys are of prime importance to malicious insider security. In contrast to previous schemes, we present an eCK secure protocol in the standard model, without assuming impractical, strong, concurrent zero-knowledge proofs of knowledge of secret keys done to the CA at key registration. The security proof of our scheme is based on standard pairing assumption, collision resistant hash functions, bilinear decision Diffie-Hellman (BDDH) and decision linear Diffie-Hellman (DLIN) assumptions, and pseudo-random functions with pairwise independent random source πPRF [14].

Keywords: one-round authenticated key exchange, pairing, insider security

1 Introduction

Many critical applications rely on the existence of a confidential channel established by authenticated Key Exchange (AKE) protocols over open networks. In contrast to the most prominent key exchange protocol is the Diffie-Hellman protocol [9] which is vulnerable to the existence of an active adversary (i.e. man-in-the-middle attacks), a secure AKE should be secure against an active adversaries. Over the last decade, the security of AKE against active attacks has been developed increasingly in stronger models. In this paper, we consider PKI-based two party AKE protocol in presence of adversary with strong capabilities. LaMacchia, Lauter and Mityagin [10] recently presented strong security definitions for two-pass key exchange protocol, which is referred as eCK security model. Since the introducing of eCK model, many protocols (e.g. [14, 20, 12, 13]) have been proposed to provide eCK security. But most of those protocols are proven under random oracle model.

Public Key Registration and EstablishParty Query. In the original eCK model [10], the public key registration was considered from three situations: (i) honest key registration, (ii) proof of knowledge (POK) key registration, and (iii) arbitrary key registration. In the security experiment, the above cases are simulated differently by the challenger. As for the honest key registration, all public keys are generated honestly by challenger, and for the other two cases the public keys might be chosen by adversary. In the latter literatures, the EstablishParty query was introduced to model such chosen public key attacks, that might relate to attacks like unknown key share (UKS) attacks [6], etc. In the security experiment, each registered corrupted party by EstablishParty query is controlled by the adversary, which can be used to interact with honest parties in sessions.

We notice that the EstablishParty query has not been clearly formalized so far, where no POK assumption for key registration is addressed by this query. In particular, different POK assumptions would result in different type of adversaries in the security experiment, that would impact the proof simulation, in particular for the proof without random oracles. General speaking, there are two major POK assumptions: knowledge of secret key (KOSK) assumption and plain public key (PPK) assumption. The KOSK assumption (e.g. used in [11]), that requires each party provides the certification authority (CA) with a proof of knowledge of its secret key before the CA certifies the corresponding public key. While implementing the (KOSK) assumption, it is assumed that there exists either efficient knowledge extractor (satisfying requirement in [1]), or the adversary simply hands the challenger corresponding secret keys. The another assumption is the plain public key (PPK) assumption (following the real-world standards PKCS#10 [15]) that nothing more is required than in any usage of public-key cryptography, where the proof of possession might be implemented by having the user send the CA a signature (under the public key it is attempting to get certified) of some message that includes the public key and the user identity. On the contrary, the private keys, of dishonest parties registered under PPK assumption, might be only known by adversary, nor by the challenger. As pointed out by Mihir Bellare and Gregory Neven in [2], the KOSK assumption can't be implemented by the proof based on plain public key (PPK) assumption, and the PPK assumption is much cheaper and more realistic than KOSK assumption.

While designing and analysing eCK protocol against chosen public key attacks, corresponding POK assumption should be explicitly modeled by EstablishParty query. Recently Moriyama and Okamoto (MO) presented an eCK-secure key exchange protocol [13] in the standard model. However, the MO protocol can't be proven secure without KOSK assumption. Since under PPK assumption, if the long-term keys of test oracle (e.g. owned by party $\hat{A}$) are not corrupted and set in terms of a DDH challenge instance, then the challenger is unable to simulate the session key of other oracles of $\hat{A}$ which have dishonest peer (e.g. party $\hat{C}$) established by adversary. Because computing the long-term shared key involving parties $\hat{A}$ and $\hat{C}$ is a CDH hard problem for the challenger. Also, it still left out the task of formally justifying a claim on how to implement the abstract KOSK assumption for MO protocol. Therefore we are motivated to clearly formalize the EstablishParty query and strive to seek eCK secure protocol against chosen public key attacks without KOSK assumption and NAXOS tricks in the standard model.

Potential Threat on leakage of Secret Exponent. Besides the leakage of long-term and ephemeral private keys modeled by eCK model, a 'well' designed protocol should resist with the compromise of other session key related secret information, even though such compromise is not normally expected. A noteworthy instance is the leakage of ephemeral intermediate exponent (e.g., the $a_1 + a_3\alpha$ in MO protocol), due to the up-to-date side-channel attacks. Such kind of leakage has been studied by Sarr et al. [17, 16] based on HMQV protocol. In particular, as pointed by Yoneyama et al. [22], the leakage of intermediate exponent of Okamoto protocol [14] and MO protocol (in two different sessions) would result in exposure of long-term keys. Therefore one should take care of those intermediate exponents while designing protocols, even though it is hard to prove the security on resilience of such leakage (as claimed in [22]). Moreover, the ephemeral secrets that can be revealed in the eCK model, should be clearly specified by each protocol based on appropriate implementation scenario. Note that if the protocol is executed in a computer infected with malware, then all secret session states (including those intermediate exponents mentioned above) might be exposed.

1.1 Contribution

In this paper, we clarify the EstablishParty query in terms of different type of POK assumptions. We present an eCK secure AKE protocol in the standard model, that is able to resist with chosen public key attacks based on only plain public key registration assumption and without NAXOS trick. The security of proposed protocol is based on standard pairing assumption, collision resistant hash functions, bilinear decision Diffie-Hellman (BDDH) and decision linear Diffie-Hellman (DLIN) assumptions, and pseudo-random functions with pairwise independent random source $\pi\mathsf{PRF}$ [14]. Not surprisingly, one must pay a small price for added security with one paring operation. However our protocol can be implemented in a group where DDH problem is easy.

We show that the internal computation algorithm really matters for the security of a protocol. From our construction approach, we illustrate an example on how to mitigate the threat due to leakage of intermediate exponents, for which exponents involve only long-term secrets. In order to relieve the consequences of such leakage, we adapt a generic strategy: first blind those intermediate exponents using uniform random value (e.g. the ephemeral private keys) and next remove the random value after completing corresponding exponential operation.[1] Our approach can also be applied to improve the MO protocol [13] or Okamoto protocol [14] in a similar way.

1.2 Related Work

In the first eCK security model introduced by LaMacchia, Lauter and Mityagin [10], they model the insider security by allowing adversary to register arbitrary public keys without proving knowledge of the corresponding secret key, which was formalized by EstablishParty query in later literatures.

Since then many eCK secure protocols, e.g. [10, 14, 12, 17, 16], have been correctly proven under the malicious insider setting. But most of them are only provable secure with the help of random oracles. Although the protocol [14] by Okamoto is eCK secure in the standard model without KOSK assumption, this protocol heavily relies on the NAXOS trick. Even though the NAXOS trick hides the exponent of the ephemeral public key, it might be leaked because of the up-to-date side-channel attacks. Therefore, a lot of works [13, 20] are motivated to propose eCK-secure key exchange protocols without the NAXOS tricks.

Sarr et al. [17], recently described some potential threats on HMQV due to the leakage of secret intermediate exponent (i.e. the $x + aD$, where $D = H(\hat{A}, \hat{B}, X)$). Namely, if such intermediate exponents in different sessions are identical, the adversary can obtain the secret signature in the target session. In the later, Sarr et al. [16] strengthened the eCK model by allowing the adversary

[1] This would mitigate the attacks described in [16, 22], when the secret intermediate exponent is exposed somehow.

to learn certain intermediate results while computing the session key, under specific implementation environment wherein a tamper-proof device is involved to store long-term keys while session keys are used on an untrusted host machine. The seCK model was further studied by Yoneyama et al., in recent work [22]. They pointed out errors in the security proofs of SMQV and FHMQV [16] on leakage of intermediate computations. Unfortunately, their results also showed that there is no scheme has been provably secure in the seCK model.

2 Preliminaries

Notations. We let κ denote the security parameter and 1^κ the string that consists of κ ones. Each party has a long-term authentication key which is used to prove the identity of the party in an AKE protocol. We let a 'hat' on top of a capital letter denotes an identifier of a participant, without the hat the letter denotes the public key of that party, and the same letter in lower case denotes a private key. For example, a party $\hat{A}$ is supposed to register its public key $A = g^a$ at certificate authority (CA) and keeps corresponding long-term secret key $sk_A = a$ privately. Let $[n] = \{1, \ldots, n\} \subset \mathbb{N}$ be the set of integers between 1 and n. If S is a set, then $a \in_R S$ denotes the action of sampling a uniformly random element from S.

2.1 Collision-Resistant Hashing

Definition 1 (Collision-resistant Hash Function)**.** Let $\mathcal{H}_k$ for $k \in \mathbb{N}$ be a collection of functions of the form $h : \{0,1\}^* \to \{0,1\}^k$. Let $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$. $\mathcal{H}$ is called (t_h, ϵ_h)-collision resistant if for all t_h-time adversaries $\mathcal{A}$ it holds that

$$\Pr\left[h \in_R \mathcal{H}_k,\ (m, m') \leftarrow \mathcal{A}(h),\ m \neq m',\ m, m' \in \{0,1\}^*,\ h(m) = h(m')\right] \leq \epsilon_h = \epsilon_h(\kappa),$$

where the probability is over the random bits of $\mathcal{A}$.

2.2 Pseudo-Random Functions

A *pseudo-random function* is an algorithm PRF that implements a deterministic function $z = \mathsf{PRF}(k, x)$, taking as input a key (seed) $k \in \mathcal{K}$ and some bit string $x \in \mathcal{D}$, and returning a string $z \in \mathcal{R}$, where $\mathcal{K}$ is the key space, $\mathcal{D}$ is the domain and $\mathcal{R}$ is the range of PRF for security parameter κ. Let $\mathcal{A}$ be an adversary that is given oracle access to either $\mathsf{PRF}(k, \cdot)$ for $k \in_R \mathcal{K}$ or a truly random function $\mathsf{RF}(\cdot)$ with the same domain and range as the pseudo-random function PRF.

Definition 2. We say that PRF is a $(t, \epsilon_{\mathsf{PRF}})$-secure pseudo-random function, if any adversary $\mathcal{A}$ running in probabilistic polynomial time t has at most an advantage of ϵ_{PRF} to distinguish the pseudo-random function PRF from a truly random function RF, i.e.

$$\left|\Pr[\mathcal{A}^{\mathsf{PRF}(k,\cdot)}(1^\kappa) = 1] - \Pr[\mathcal{A}^{\mathsf{RF}(\cdot)}(1^\kappa) = 1]\right| \leq \epsilon_{\mathsf{PRF}},$$

where ϵ_{PRF} is a negligible function in κ.

2.3 Pseudo-Random Functions with Pairwise Independent Random Sources (πPRF)

This is a specific class of PRF introduced by Okamoto [14]. The πPRF family associated with key (seed) space $\mathcal{K}$, domain $\mathcal{D}$ and range $\mathcal{R}$ in the security parameter κ, states that if a specific variable $k_{i_0} \in_R \mathcal{K}$ is pairwise independent from other variable, then on input value $m \in \mathcal{D}$ the output $h \in \mathcal{R}$ of this function indexed by k_{i_0} is indistinguishable from random.

Suppose that function $f_\Sigma : I_\Sigma \to X_\Sigma$ is a deterministic polynomial-time algorithm, where X_Σ is a set of random variables over $\Sigma \in_R \mathcal{K}$ and I_Σ is a set of indices regarding Σ, then this algorithm outputs $k_i \in X_\Sigma$ from $i \in I_\Sigma$. Let $(k_{i_0}, k_{i_1}, \ldots, k_{i_{t(\kappa)}})$ $(i_j \in I_\Sigma)$ be pairwise independent random variables indexed by (I_Σ, f_Σ), and each variable be uniformly distributed over Σ. That is, for any pair of (k_{i_0}, k_{i_j}) $(j = 1, \ldots, t\kappa)$, for any $(x, y) \in \Sigma^2$, we have $\Pr[k_{i_0} \to x \wedge k_{i_j} \to y] = 1/|\Sigma|^2$. Consider a PPT algorithm $\mathcal{A}^{F,I_\Sigma}$ that can issue oracle queries. When $\mathcal{A}$ sends $q_j \in D$ and $i_j \in I_\Sigma$ to the query, the oracle replies with $F^{\kappa,\Sigma,\mathcal{D},\mathcal{R}}_{\bar{k}_j}(q_j)$ for each $j = 0, 1, \ldots, t(\kappa)$, where $(\bar{k}_{i_0}, \bar{k}_{i_1}, , \ldots, \bar{k}_{i_{t(\kappa)}}) \in_R (k_{i_0}, k_{i_1}, \ldots, k_{i_{t(\kappa)}})$. $\mathcal{A}^{RF,I_\Sigma}$ is the same as $\mathcal{A}^{F,I_\Sigma}$ except for $F^{\kappa,\Sigma,\mathcal{D},\mathcal{R}}_{\bar{k}_0}(q_0)$ is replaced by a truly random function $\mathsf{RF}(q_0)$.

Definition 3. We say that F is secure πPRF family if for any PPT adversary $\mathcal{A}$ running in time t has at most an advantage of $\epsilon_{\pi\mathsf{PRF}}$ to distinguish the πPRF from a truly random function, i.e.

$$|\Pr[\mathcal{A}^{F,I_\Sigma}(1^\kappa, \mathcal{D}, \mathcal{R}) = 1] - \Pr[\mathcal{A}^{\mathsf{RF},I_\Sigma}(1^\kappa, \mathcal{D}, \mathcal{R}) = 1]| \le \epsilon_{\pi\mathsf{PRF}},$$

where F is a πPRF family with index $(I_\Sigma, f_\Sigma) \in \mathcal{K}$.

2.4 Bilinear Groups

In the following, we briefly recall some of the basic properties of bilinear groups. Our AKE solution mainly consist of elements from a single group $\mathbb{G}$. We therefore concentrate on symmetric bilinear map (pairing).

Definition 4 (Symmetric Bilinear groups)**.** Let two cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p. Let g be a generator of $\mathbb{G}$. The function

$$e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$$

is an (admissible) bilinear map if it holds that:

1. **Bilinear:** for all $a, b \in \mathbb{G}$ and $x, y \in \mathbb{Z}$, we have $e(a^x, b^y) = e(a, b)^{xy}$.
2. **Non-degenerate:** $e(g, g) \neq 1_{\mathbb{G}_T}$, is a generator of group $\mathbb{G}_T$.
3. **Efficiency:** e is efficiently computable for all $a, b \in \mathbb{G}$.

We call $(\mathbb{G}, g, \mathbb{G}_T, p, e)$ a symmetric bilinear groups.

2.5 The Decision Linear Diffie-Hellman Assumption

Let $\mathbb{G}$ be a group of prime order p. Let g be a generator of $\mathbb{G}$, and along with arbitrary generators g_1 and g_2 of $\mathbb{G}$. Given tuple $(g, g_1, g_2, g_1^a, g_2^b, g^c)$ for $(a, b, c) \in_R Z_p^3$ the decisional linear Diffie-Hellman assumption says that it is hard to decide whether $c = a + b \bmod p$.

Definition 5. We say that the DLIN assumption holds if

$$\left|\Pr\left[\mathcal{A}(g, g_1, g_2, g_1^a, g_2^b, g^{a+b}) = 1\right] - \Pr\left[\mathcal{A}(g, g_1, g_2, g_1^a, g_2^b, g^c) = 1\right]\right| \leq \epsilon,$$

where $(a, b, c) \in_R Z_p^3$, for all probabilistic polynomial-time adversaries $\mathcal{A}$, where $\epsilon = \epsilon(\kappa)$ is some negligible function in the security parameter.

2.6 The Bilinear Decisional Diffie-Hellman Assumption

Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order p. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map as definition 4. The Bilinear Decisional DH problem is stated as follows: given the tuple (g, g^a, g^b, g^c) for $(a, b, c) \in \mathbb{Z}_p^3$ as input to distinguish the $e(g, g)^{abc}$ from a random value.

Definition 6. We say that the (t, ϵ)-BDDH assumption holds if

$$\left|\Pr\left[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1\right] - \Pr\left[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^{\gamma}) = 1\right]\right| \leq \epsilon,$$

where $(a, b, c, \gamma) \in_R \mathbb{Z}_p^4$, for all probabilistic t-time adversaries $\mathcal{A}$, where $\epsilon = \epsilon(\kappa)$ is some negligible function in the security parameter.

3 AKE Security

In this section we present the formal security model for two party PKI-based authenticated key-exchange (AKE) protocol. While modeling the active adversaries, we provide with an 'execution environment' following an important line of research [5, 7, 10, 19] dates back to Bellare and Rogaway [3]. We will use the framework as in [19] with slight modification.

Execution Environment. Assume there exist a fixed number of parties $\{P_1, \dots, P_\ell\}$ for $\ell \in \mathbb{N}$, where each party $P_i \in \{P_1, \dots, P_\ell\}$ is a potential protocol participant and each party has a long-term key pair $(pk_i, sk_i) \in (\mathcal{PK}, \mathcal{SK})$ corresponds to its identity i, where $\{\mathcal{PK}, \mathcal{SK}\}$ are keyspaces of long-term keys. To model several sequential and parallel executions of the protocol, each party P_i is modeled by a collection of oracles $\pi_i^1, \dots, \pi_i^d$ for $d \in \mathbb{N}$. Each oracle π_i^s represents one single process that executes an instance of the protocol. All oracles $\pi_i^1, \dots, \pi_i^d$ representing party P_i have access to the same long-term key pair (pk_i, sk_i) of P_i and to all public keys $pk_1, \dots, pk_\ell$. Moreover, each oracle π_i^s maintains a separate internal state

- a variable Φ storing the identity j of an intended communication partner P_j,
- a variable $\Psi \in \{\texttt{accept}, \texttt{reject}\}$,
- a variable $K \in \mathcal{K}$ storing the session key used for symmetric encryption between π_i^s and party P_Φ, where $\mathcal{K}$ is the keyspace of the protocol.
- and some additional temporary state variable st (which may, for instance, be used to store ephemeral Diffie-Hellman exponents or other intermediate values).

The internal state of each oracle is initialized to $(\Phi, \Psi, K, st) = (\emptyset, \emptyset, \emptyset, \emptyset)$. At some point during the protocol execution each party would generate the session key according to the key exchange protocol specification when turning to state $(\Psi, K) = (\texttt{accept}, K)$ for some K, and at some point with internal state $(\Psi, K) = (\texttt{reject}, \emptyset)$ where $\emptyset$ denotes the empty string. We will always assume (for simplicity) that $K \neq \emptyset$ if an oracle has reached $\texttt{accept}$ state.

An adversary may interact with these oracles by issuing the following queries.

- $\mathsf{Send}(\pi_i^s, m)$: The adversary can use this query to send any message m of his own choice to oracle π_i^s. The oracle will respond according to the protocol specification, depending on its internal state. If the first message $m = (\top, \tilde{j})$ consists of a special symbol $\top$ and a value $\tilde{j}$ which is either $\emptyset$ or identity j, then π_i^s will set $\Phi = \tilde{j}$ and respond with the first protocol message. If $\tilde{j} = \emptyset$ then Φ will be set as identity j at some point according to protocol specification. [2]

- $\mathsf{RevealKey}(\pi_i^s)$: Oracle π_i^s responds to a $\mathsf{RevealKey}$-query with the contents of variable K.

- $\mathsf{StateReveal}(\pi_i^s)$: Oracle π_i^s responds the contents secret state stored in variable st.

- $\mathsf{EstablishParty}(pk_m, sk_m, P_m)$ This query registers an identity $m(\ell < m < \mathbb{N})$ and a static public/private key pair (pk_m, sk_m) on behalf of a party P_m, if one of the following conditions is held: (i) $sk_m = \emptyset$ and $pk_m \in \mathcal{PK}$, (ii) $sk_m \in \mathcal{SK}$ and sk_m is the correct private key for public key pk_m; otherwise a failure symbol $\perp$ is returned. Parties established by this query are called corrupted or adversary controlled.

- $\mathsf{Corrupt}(P_i)$: Oracle π_i^1 responds with the long-term secret key sk_i of party P_i. After this query, oracles π_i^s can still be asked queries using the compromised key sk_i.

- $\mathsf{Test}(\pi_i^s)$: This query may only be asked once throughout the game. Oracle π_i^s handles this query as follows: If the oracle has state $\Psi = \texttt{reject}$ or $K = \emptyset$, then it returns some failure symbol $\perp$. Otherwise it flips a fair coin b, samples a random element $K_0 \stackrel{\$}{\leftarrow} \mathcal{K}$, sets $K_1 = K$ to the 'real' session key, and returns K_b.

We note that the exact meaning of the $\mathsf{StateReveal}$ must be defined for each protocol separately, namely the content stored in the variable st during protocol execution. In $\mathsf{EstablishParty}$ query, the private key sk_m corresponds to the proof of knowledge assumptions for public key registration, which should be specified in the security proof of each protocol. If $sk_m = \emptyset$ then the plain public key or arbitrary key registration assumption is modeled, otherwise the knowledge of secret key assumption is modeled.

Secure AKE Protocols. We first define the partnering of two oracles via *matching conversations* that was first introduced by Bellare and Rogaway [3] in order to define correctness and security of an AKE protocol precisely, and refined latter in [19]. In the following let T_i^s denote the transcript of messages sent and received by oracle π_i^s. We assume that messages in a transcript T_i^s are represented as binary strings. Let $|T_i^s|$ denote the number of its messages. Assume there are two transcripts T_i^s and T_j^t, where $m := |T_i^s|$ and $n := |T_j^t|$. We say that T_i^s is a prefix of T_j^t if $0 < m \leq n$ and the first m messages in transcripts T_i^s and T_j^t are pairwise equivalent as binary strings.

[2] A protocol might be run in either pre- or post-specified peer model here [8].

Definition 7. We say that a process π_i^s has a *matching conversation* to oracle π_j^t, if

- π_i^s has sent the last message(s) and T_j^t is a prefix of T_i^s, or
- π_j^t has sent the last message(s) and T_i^s is a prefix of T_j^t.

We say that two oracles π_i^s and π_j^t have *matching conversations* if π_i^s has a *matching conversation* to process π_j^t, and vice versa.

Definition 8 (Freshness)**.** Let π_i^s be a completed oracle held by an honest party P_i with honest peer P_j, and both parties P_i and P_j are not registered by $\mathsf{EstablishParty}$ query. Let π_j^t be a completed oracle, if it exists, such that π_i^s and π_j^t have matching conversations. Then the oracle π_i^s is said to be fresh (unexposed) if none of the following conditions holds:

1. The adversary $\mathcal{A}$ either issued $\mathsf{RevealKey}(\pi_i^s)$, or $\mathsf{RevealKey}(\pi_j^t)$ (if such π_j^t exists).
2. If π_j^t exists, $\mathcal{A}$ either issued:
   (a) Both $\mathsf{Corrupt}(P_i)$ and $\mathsf{StateReveal}(\pi_i^s)$, or.
   (b) Both $\mathsf{Corrupt}(P_j)$ and $\mathsf{StateReveal}(\pi_j^t)$.
3. If π_j^t does not exist, $\mathcal{A}$ either issued:
   (a) Both $\mathsf{Corrupt}(P_i)$ and $\mathsf{StateReveal}(\pi_i^s)$, or
   (b) $\mathsf{Corrupt}(P_j)$.

Definition 9 (Security Experiment)**.** In the experiment, the following steps are performed:

1. The challenger implements the collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$. At the beginning of the experiment, the challenger generates ℓ long-term key pairs (pk_i, sk_i) for all $i \in [\ell]$, and gives the adversary $\mathcal{A}$ all public keys $pk_1, \ldots, pk_\ell$ as input.
2. $\mathcal{A}$ may issue polynomial number (in the security parameter κ) of queries as described above in the execution environment, namely $\mathcal{A}$ makes queries: Send, $\mathsf{StateReveal}$, $\mathsf{EstablishParty}$, $\mathsf{Corrupt}$ and $\mathsf{RevealKey}$.
3. At some point, $\mathcal{A}$ issues a $\mathsf{Test}(\pi_i^s)$ query on a fresh oracle π_i^s during the experiment with only once.
4. At the end of the experiment, the $\mathcal{A}$ terminates with outputting a bit b' as its guess for bit b of Test query.

Security of AKE protocols is now defined by requiring that the protocol is a secure AKE protocol, thus an adversary cannot distinguish the session key K of a fresh oracle from a random key.

Definition 10 (Secure Authenticated Key Exchange Protocol)**.** We say an AKE protocol is secure in the security experiment as Definition 9, if for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$ and for some negligible probability $\epsilon = \epsilon(\kappa)$ in the security parameter hold that:

- If two fresh oracles π_i^s and π_j^t accept with matching conversations, then both oracles hold the same session key K.

- When $\mathcal{A}$ returns b' such that
  - $\mathcal{A}$ has issued a Test query on an oracle π_i^s without failure, and
  - π_i^s has internal state $\Phi = j$, and
  - π_i^s is fresh throughout the security experiment.

  Then the probability that b' equals the bit b sampled by the **Test**-query is bounded by

$$\left|\Pr[b = b'] - 1/2\right| \leq \epsilon.$$

4 A Strong AKE Protocol Without Random Oracles

In this section we present a pairing-based strong AKE protocol without random oracles and under malicious insider setting, which is informally depicted in Figure 1.

4.1 Protocol Description

The AKE protocol takes as input the following building blocks:

- Symmetric bilinear groups $(\mathbb{G}, g, \mathbb{G}_T, p, e)$, where the generator of group $\mathbb{G}_T$ is $e(g, g)$ and along with another random generators g_1, g_2 and h of $\mathbb{G}$.
- A collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$,
- A pairwise independent pseudo-random function (πPRF) F, with index $\{I_{\mathbb{G}_T}, f_{\mathbb{G}_T}\}$ where $I_{\mathbb{G}_T} := \{(U, V, \alpha) | (U, V, \alpha) \in \mathbb{G}_T^2 \times \mathbb{Z}_p\}$ and $f_{\mathbb{G}_T} := (U, V, \alpha) \rightarrow U^{r_1 + \alpha r_2} V$ with $(r_1, r_2) \in_R \mathbb{Z}_p^2$.

Long-term Key Generation: on input the security parameter κ, the long-term keys of each party $\hat{A}$ is generated as following:

- $\hat{A}$ selects long-term private keys : $(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8) \in_R \mathbb{Z}_p^8$, and compute the long-term public keys: $(A_1, A_2, A_3, A_4, A_5, A_6, A_7) := (g_1^{a_1} g^{a_3}, g_2^{a_2} g^{a_3}, g_1^{a_4} g^{a_6}, g_2^{a_5} g^{a_6}, g_1^{a_7}, g_2^{a_8}, g^{a_7 + a_8})$.

Protocol Execution :

1. Upon activation a session $(\hat{A}, \hat{B})$[3], the initiator $\hat{A}$ performs the steps:
   (a) Choose three ephemeral private keys $x_1, x_2, x, \in_R \mathbb{Z}_p^3$.
   (b) Compute $X_1 := g^{x_1}, X_2 := g^{x_2}, X_3 := g^{x_1 + x_2}$ and $X := g^x$.
   (c) Create an active session with identifier $sid_{\hat{A}} := (\hat{A}, \hat{B}, X_1, X_2, X_3, X)$.
   (d) Send $(X_1, X_2, X_3, X, \hat{A}, \hat{B})$ to $\hat{B}$.
2. Upon receiving $(X_1, X_2, X_3, X, \hat{A}, \hat{B})$, the responder $\hat{B}$ does the following:
   (a) Verify that $(X_1, X_2, X_3, X) \in \mathbb{G}^4$.
   (b) Choose three ephemeral private keys $y_1, y_2, y \in_R Z_p^3$.

[3]We stress that the identifier $\hat{A}$ and $\hat{B}$ are required to be distinct

$\hat{A}$		$\hat{B}$
$(sk_A =< a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8 >)$		$(sk_B =< b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8 >)$
$A_1 = g_1^{a_1} g^{a_3}, A_2 = g_2^{a_2} g^{a_3}$		$B_1 = g_1^{b_1} g^{b_3}, B_2 = g_2^{b_2} g^{b_3}$
$A_3 = g_1^{a_4} g^{a_6}, A_4 = g_2^{a_5} g^{a_6}$		$B_3 = g_1^{b_4} g^{b_6}, B_4 = g_2^{b_5} g^{b_6}$
$A_5 = g_1^{a_7}, A_6 = g_2^{a_8}, A_7 = g^{a_7+a_8}$		$B_5 = g_1^{b_7}, B_6 = g_2^{b_8}, B_7 = g^{b_7+b_8}$
$x_1, x_2, x, \in_R \mathbb{Z}_p^*$		$y_1, y_2, y \in_R \mathbb{Z}_p^*$
$X_1 := g_1^{x_1}, X_2 := g_2^{x_2},$		$Y_1 := g_1^{y_1}, Y_2 := g_2^{y_2},$
$X_3 := g^{x_1+x_2}, X := g^x$		$Y_3 := g^{y_1+y_2}, Y := g^y$
	$\xrightarrow{X_1, X_2, X_3, X, \hat{A}, \hat{B}}$	
	$\xleftarrow[\hat{B}, X_1, X_2, X_3, X]{Y_1, Y_2, Y_3, Y, \hat{A},}$	
$(Y_1, Y_2, Y_3, Y)? \in \mathbb{G}^4$		$(X_1, X_2, X_3, X)? \in \mathbb{G}^4$
$sid_{\hat{A}} := (\hat{A}, \hat{B}, X_1, X_2, X_3, X, Y_1, Y_2, Y_3, Y)$		$sid_{\hat{B}} := (\hat{A}, \hat{B}, X_1, X_2, X_3, X, Y_1, Y_2, Y_3, Y)$
$\beta := H(sid_{\hat{A}})$		$\beta := H(sid_{\hat{B}})$
$\sigma_{\hat{A}} := (B_1 B_3^{\beta})^{\frac{x_1+a_7}{x}} \cdot (B_2 B_4^{\beta})^{\frac{x_2+a_8}{x}}$		$\sigma_{\hat{B}} := (A_1 A_3^{\beta})^{\frac{y_1+b_7}{y}} \cdot (A_2 A_4^{\beta})^{\frac{y_2+b_8}{y}}$
$\cdot (Y_1 B_5)^{\frac{a_1+a_4\beta}{x}} \cdot (Y_2 B_6)^{\frac{a_2+a_5\beta}{x}} \cdot (Y_3 B_7)^{\frac{a_3+a_6\beta}{x}} \cdot Y$		$\cdot (X_1 A_5)^{\frac{b_1+b_4\beta}{y}} \cdot (X_2 A_6)^{\frac{b_2+b_5\beta}{y}} \cdot (X_3 A_7)^{\frac{b_3+b_6\beta}{y}} \cdot X$
$\sigma_{\hat{A}} := e(\sigma_{\hat{A}}, h^x)$		$\sigma_{\hat{B}} := e(\sigma_{\hat{B}}, h^y)$
$k := F(\sigma_{\hat{A}}, sid_{\hat{A}})$		$k := F(\sigma_{\hat{B}}, sid_{\hat{B}})$

Figure 1: The AKE Protocol without Random Oracles.

(c) Compute $Y_1 := g^{y_1}, Y_2 := g^{y_2}, Y_3 := g^{y_1+y_2}$ and $Y := g^y$.

(d) Create an active session with identifier $sid_{\hat{B}} := (\hat{A}, \hat{B}, X_1, X_2, X_3, X, Y_1, Y_2, Y_3, Y)$ and compute $\beta := H(sid_{\hat{B}})$.

(e) Compute $\sigma_{\hat{B}} := (A_1 A_3^{\beta})^{\frac{y_1+b_7}{y}} \cdot (A_2 A_4^{\beta})^{\frac{y_2+b_8}{y}} \cdot (X_1 A_5)^{\frac{b_1+b_4\beta}{y}} \cdot (X_2 A_6)^{\frac{b_2+b_5\beta}{y}} \cdot (X_3 A_7)^{\frac{b_3+b_6\beta}{y}} \cdot X$ and $\sigma_{\hat{B}} := e(\sigma_{\hat{B}}, h^y)$

(f) Compute session key $k := F(\sigma_{\hat{B}}, sid_{\hat{B}})$ and erase all intermediate values and y_1, y_2, y.

(g) Send $(Y_1, Y_2, Y_3, Y, \hat{A}, \hat{B}, X_1, X_2, X_3, X)$ to $\hat{A}$.

3. Upon receiving $(Y_1, Y_2, Y_3, Y, \hat{A}, \hat{B}, X_1, X_2, X_3, X)$ does the following:

   (a) Verify that exist a session identified by $(\hat{A}, \hat{B}, X_1, X_2, X_3, X)$ and $(Y_1, Y_2, Y_3, Y) \in \mathbb{G}^4$.

   (b) Update session identifier $sid_{\hat{A}} := (\hat{A}, \hat{B}, X_1, X_2, X_3, X, Y_1, Y_2, Y_3, Y)$, and compute $\beta := H(sid_{\hat{A}})$.

   (c) Compute $\sigma_{\hat{A}} := (B_1 B_3^{\beta})^{\frac{x_1+a_7}{x}} \cdot (B_2 B_4^{\beta})^{\frac{x_2+a_8}{x}} \cdot (Y_1 B_5)^{\frac{a_1+a_4\beta}{x}} \cdot (Y_2 B_6)^{\frac{a_2+a_5\beta}{x}} \cdot (Y_3 B_7)^{\frac{a_3+a_6\beta}{x}} \cdot Y$ and $\sigma_{\hat{A}} := e(\sigma_{\hat{A}}, h^x)$.

   (d) Compute session key as $k := F(\sigma_{\hat{A}}, sid_{\hat{A}})$ and erase all intermediate values and y_1, y_2, y.

   We assume, only the ephemeral private keys, i.e. (x_1, x_2, x) and (y_1, y_2, y) would be stored as secret in the state variable st.[4]

[4]This can be achieved by performing the computation in steps 2e and 2f (resp. steps 3c and 2f) on a smart card,

4.2 Security Analysis

Theorem 1. *Suppose that the $(t, q, \epsilon_{\mathsf{BDDH}})$-Bilinear DDH assumption and $(t, q, \epsilon_{\mathsf{DLIN}})$-Decision linear assumption hold in bilinear groups $(\mathbb{G}, g, \mathbb{G}_T, p, e)$, the hash function H is $(t, \epsilon_{\mathsf{CR}})$-secure, and a $(t, \epsilon_{\pi\mathsf{PRF}})$-secure $\pi\mathsf{PRF}$ family with index $\{I_{\mathbb{G}_T}, f_{\mathbb{G}_T}\}$ where $I_{\mathbb{G}_T} := \{(U, V, \alpha) | (U, V, \alpha) \in \mathbb{G}^2 \times \mathbb{Z}_p\}$ and $f_{\mathbb{G}_T} := (U, V, \alpha) \to U^{r_1 + \alpha r_2} V$ with $(r_1, r_2) \in_R \mathbb{Z}_p^2$, with respect to the definitions in Section 2. Then the proposed protocol is a (t', ϵ')-eCK secure AKE in the sense of Definition 10.*

Proof of Theorem 1. We now verify that no polynomially bounded adversary can distinguish the real session key of a fresh oracle from a random key. In the security experiment, the adversary is allowed to query $\mathsf{EstablishParty}(pk_m, sk_m, P_m)$ with $sk_m = \emptyset$ while registering a public key pk_m for dishonest party P_m. Namely, we allow arbitrary public key registration.

We fist introduce the notations might be used in the proof. When describing the communication between oracle $\pi^s_{\hat{A}}$ and party $\hat{C}^{(s)} (s \in [\ell - 1])$, we use the following notations:

- $(x_1^{(s)}, x_2^{(s)}, x^{(s)})$: the ephemeral private keys of oracle $\pi^s_{\hat{A}}$.
- $(X_1^{(s)}, X_2^{(s)}, X_3^{(s)}, X^{(s)}) = (g_1^{x_1^{(s)}}, g_2^{x_2^{(s)}}, g^{x_1^{(s)} + x_2^{(s)}}, g^{x^{(s)}})$: the ephemeral public keys of oracle $\pi^s_{\hat{A}}$.
- $(C_1^{(s)}, C_2^{(s)}, C_3^{(s)}, C_4^{(s)}, C_5^{(s)}, C_6^{(s)}, C_7^{(s)}) = (g_1^{c_1^{(s)}} g^{C_3^{(s)}}, g_2^{c_2^{(s)}} g^{C_3^{(s)}}, g_1^{C_4^{(s)}} g^{c_6^{(s)}}, g_2^{c_5^{(s)}} g^{c_6^{(s)}}, g_1^{c_7^{(s)}}, g_2^{c_8^{(s)}}, g^{b_7 + b_8})$: the public keys of party $\hat{C}^{(s)}$.
- $(W_1^{(s)}, W_2^{(s)}, W_3^{(s)}, W^{(s)}) = (g_1^{w_1^{(s)}}, g_2^{w_2^{(s)}}, g^{w_1^{(s)} + w_2^{(s)}}, g^{w^{(s)}})$: the ephemeral public keys received by oracle $\pi^s_{\hat{A}}$.
- $sid^{(s)}_{\hat{A}} = (\hat{A}, \hat{C}^{(s)}, X_1^{(s)}, X_2^{(s)}, X_3^{(s)}, X^{(s)}, W_1^{(s)}, W_2^{(s)}, W_3^{(s)}, W^{(s)})$.
- $\beta^{(s)}_{\hat{A}} = H(sid^{(s)}_{\hat{A}})$.
- $$\sigma^{(s)}_{\hat{A}} = (C_1^{(s)} C_3^{(s)\beta^{(s)}_{\hat{A}}})^{x_1^{(s)} + a_7} \cdot (C_2^{(s)} C_4^{(s)\beta^{(s)}_{\hat{A}}})^{x_2^{(s)} + a_8} \cdot (W_1^{(s)} C_5^{(s)})^{a_1 + a_4 \beta^{(s)}_{\hat{A}}} \cdot (W_2^{(s)} C_6^{(s)})^{a_2 + a_5 \beta^{(s)}_{\hat{A}}} \cdot (W_3^{(s)} C_7^{(s)})^{a_3 + a_6 \beta^{(s)}_{\hat{A}}} \cdot W^{(s)x^{(s)}}.$$
- $\sigma^{(s)}_{\hat{A}} = e((\sigma^{(s)}_{\hat{A}}), h)$.
- $k^{(s)}_{\hat{A}} = F(\sigma^{(s)}_{\hat{A}}, sid^{(s)}_{\hat{A}})$.

We describe the communication between oracle $\pi^t_{\hat{B}}$ and party $\hat{D}^{(t)} (t \in [\ell - 1])$, using the following notations:

- $(y_1^{(t)}, y_2^{(t)}, y^{(t)})$: the ephemeral private keys of oracle $\pi^t_{\hat{B}}$.
- $(Y_1^{(t)}, Y_2^{(t)}, Y_3^{(t)}, Y^{(t)}) = (g_1^{y_1^{(t)}}, g_2^{y_2^{(t)}}, g^{y_1^{(t)} + y_2^{(t)}}, g^{y^{(t)}})$: the ephemeral public keys of oracle $\pi^t_{\hat{B}}$.

---

where the long-term keys are stored. In this case, the intermediate values would not be exposed due to e.g. malware attacks on the PC, which we model with $\mathsf{StateReveal}$ query.

- $(D_1^{(t)}, D_2^{(t)}, D_3^{(t)}, D_4^{(t)}, D_5^{(t)}, D_6^{(t)}, D_7^{(t)}) = (g_1^{d_1^{(t)}} g^{D_3^{(t)}}, g_2^{d_2^{(t)}} g^{D_3^{(t)}}, g_1^{D_4^{(t)}} g^{d_6^{(t)}}, g_2^{d_5^{(t)}} g^{d_6^{(t)}}, g_1^{d_7^{(t)}}, g_2^{d_8^{(t)}}, g^{b_7+b_8})$: the public keys of party $\hat{C}^{(s)}$.

- $(N_1^{(t)}, N_2^{(t)}, N_3^{(t)}, N^{(t)}) = (g_1^{n_1^{(t)}}, g_2^{n_2^{(t)}}, g^{n_1^{(t)}+n_2^{(t)}}, g^{n^{(t)}})$: the ephemeral public keys received by oracle $\pi_{\hat{B}}^{t}$.

- $sid_{\hat{A}}^{(s)} = (\hat{A}, \hat{C}^{(s)}, Y_1^{(t)}, Y_2^{(t)}, Y_3^{(t)}, Y^{(t)}, N_1^{(t)}, N_2^{(t)}, N_3^{(t)}, N^{(t)})$.

- $\beta_{\hat{A}}^{(s)} = H(sid_{\hat{A}}^{(s)})$.

- $$\sigma_{\hat{A}}^{(s)} = (D_1^{(t)} D_3^{(t)\beta_{\hat{A}}^{(s)}})^{y_1^{(t)}+a_7} \cdot (D_2^{(t)} D_4^{(t)\beta_{\hat{A}}^{(s)}})^{y_2^{(t)}+a_8} \cdot (N_1^{(t)} D_5^{(t)})^{a_1+a_4\beta_{\hat{A}}^{(s)}} \cdot (N_2^{(t)} D_6^{(t)})^{a_2+a_5\beta_{\hat{A}}^{(s)}} \cdot (N_3^{(t)} D_7^{(t)})^{a_3+a_6\beta_{\hat{A}}^{(s)}} \cdot N^{(t)y^{(t)}}.$$

- $\sigma_{\hat{B}}^{(t)} = e((\sigma_{\hat{B}}^{(t)}), h)$.

- $k_{\hat{B}}^{(t)} = F(\sigma_{\hat{B}}^{(t)}, sid_{\hat{B}}^{(t)})$.

We examine the probability of adversary in breaking the indistinguishability of session key of the test oracle, according to the following complementary events and all freshness related cases as Definition 8:

- **Event 1**: There is a oracle $\pi_{\hat{B}}^{t^*}$ held by $\hat{B}$, such that $\pi_{\hat{A}}^{s^*}$ and $\pi_{\hat{B}}^{t^*}$ have matching conversations, and we have the following disjoint cases:
  - Case 1 ($C1$): The adversary doesn't issue Corrupt($\hat{A}$) and Corrupt($\hat{B}$).
  - Case 2 ($C2$): The adversary doesn't issue StateReveal($\pi_{\hat{A}}^{s^*}$) and StateReveal($\pi_{\hat{B}}^{t^*}$).
  - Case 3 ($C3$): The adversary doesn't issue StateReveal($\pi_{\hat{A}}^{s^*}$) and Corrupt($\hat{B}$).
  - Case 4 ($C4$): The adversary doesn't issue Corrupt($\hat{A}$) and StateReveal($\pi_{\hat{B}}^{t^*}$).

- **Event 2**: There is no oracle $\pi_{\hat{B}}^{t^*}$ held by $\hat{B}$, such that $\pi_{\hat{A}}^{s^*}$ and $\pi_{\hat{B}}^{t^*}$ have matching conversations, and we have the following disjoint events:
  - Case 5 ($C5$): The adversary doesn't issue Corrupt($\hat{A}$) and Corrupt($\hat{B}$).
  - Case 6 ($C6$): The adversary doesn't issue StateReveal($\pi_{\hat{A}}^{s^*}$) and Corrupt($\hat{B}$).

In order to complete the proof of Theorem 1, we must provide the security proofs for above six cases. However, due to the Propositions 1 and 2 from [13], the security proofs for Cases $C1, C2, C3$ and $C6$ can be reduced to the security proof for Case $C5$. Therefore, we prove the advantage of the adversary is negligible in security parameter κ, for cases $C4$ and $C5$ respectively .

Proof of Case $C4$:

The proof proceeds in a sequence of games, following [4, 18]. The first game is the real security experiment, as assumed that there exists an *AKE-adversary* $\mathcal{A}_1$ that breaks the security of the proposed protocol. We then describe several intermediate games that step-wisely modify the original game. Finally we prove that (under the stated security assumptions) no adversary can distinguish any of these games (G^1_{i+1} from its predecessor G^1_i), namely the adversary has only negligible advantage in breaking the indistinguishability security property of the protocol in the security parameter κ. Let S^1_i be the event that the adversary wins the security experiment under the Game G^1_i.

Game G^1_0. This is the original eCK game with adversary in Case $C4$.

Game G^1_1. This game proceeds exactly as Game G^1_0, but the simulator aborts the game if it does not correctly guess the test oracle and its partner. Since the challenger activates d oracles for each ℓ parties. Then the probability that the challenger guesses correctly the test oracle and its partner is at least $1/(\ell^2 d^2)$. Thus we have that

$$\Pr[S^1_0] \leq \ell^2 d^2 \cdot \Pr[S^1_1]. \tag{1}$$

Game G^1_2. This game proceeds exactly like the previous game, except that we replace the value $e({Y^{(s^*)}}^{x^{(s^*)}}, h)$ of test oracle with a random one.

If there exists adversary $\mathcal{A}_1$ can distinguish Game G^1_2 from Game G^1_1, then we can use it to construct an efficient algorithm $\mathcal{B}$ to solve the BDDH problem. Given a BDDH challenge instance $(\bar{g}, u, v, w, c)$, $\mathcal{B}$ sets $g = \bar{g}$, and $h = w$, and simulates the execution environment as Game G^1_1 except for the ephemeral keys $X^{(s^*)}$ and $Y^{(s^*)}$. $\mathcal{B}$ sets $X^{(s^*)} = u$ and $Y^{(s^*)} = v$ as the ephemeral public key for the test oracle and its partner respectively. Moreover, $\mathcal{B}$ sets $g_1 = g^{\mu_1}$ and $g_2 = g^{\mu_2}$, where $\mu_1, \mu_2 \in_R \mathcal{Z}^2_p$. To compute the session key of test oracle, $\mathcal{B}$ does following

1. Compute secret exponent $\gamma = \mu_1(x_1^{(s^*)} + a_7)(b_1 + b_4\beta_{\hat{A}}^{(s^*)}) + \mu_2(x_2^{(s^*)} + a_8)(b_2 + b_5\beta_{\hat{A}}^{(s^*)}) + (x_1^{(s^*)} + x_2^{(s^*)} + a_7 + a_8)(b_3 + b_6\beta_{\hat{A}}^{(s^*)}) + \mu_1(y_1^{(s^*)} + b_7)(a_1 + a_4\beta_{\hat{A}}^{(s^*)}) + \mu_2(y_2^{(s^*)} + b_8)(a_2 + b_5\beta_{\hat{A}}^{(s^*)}) + ((y_1^{(s^*)} + y_2^{(s^*)})\beta_{\hat{A}}^{(s^*)} + b_7 + b_8)(a_3 + a_6\beta_{\hat{A}}^{(s^*)})$, such that $g^\gamma = (B_1 B_3^{\beta_{\hat{A}}^{(s^*)}})^{x_1^{(s^*)} + a_7} \cdot (B_2 B_4^{\beta_{\hat{A}}^{(s^*)}})^{x_2^{(s^*)} + a_8} \cdot (Y_1^{(s^*)} B_5)^{a_1 + a_4\beta_{\hat{A}}^{(s^*)}} \cdot (Y_2^{(s^*)} B_6)^{a_2 + a_5\beta_{\hat{A}}^{(s^*)}} \cdot (Y_3^{(s^*)} B_7)^{a_3 + a_6\beta_{\hat{A}}^{(s^*)}}$.

2. Compute secret key material as $\sigma^{(s^*)} = e(h, g)^\gamma \cdot c = e(\sigma_{\hat{A}}^{(s^*)}, h)$.[5]

3. Compute $k^{(s^*)} = F(\sigma^{(s^*)}, sid_{\hat{A}}^{(s^*)})$.

Note that if $c = e(h, X^{(s^*)})^{y^{(s^*)}}$, then the simulation is equivalent to Game G^1_1. Otherwise the simulation is equivalent to Game G^1_2. Now when the algorithm $\mathcal{A}_1$ which is able to distinguish game G^1_2 from G^1_2 outputs 1 (meaning this game proceeds like Game G^1_2), $\mathcal{B}$ outputs 1 as answer to the BDDH challenge (meaning c is chosen at random), otherwise $\mathcal{B}$ outputs 0. Therefore, we obtain that

$$|\Pr[S^1_1] - \Pr[S^1_2]| \leq \epsilon_{\mathsf{BDDH}}. \tag{2}$$

Game G^1_3. We modify Game G^1_2 to G^1_3 by changing pseudo-random function F to a truly random function RF for test oracle.

In Game G^1_3 we make use of the fact that the seed $\sigma^{(s^*)} = e((\sigma_{\hat{A}}^{(s^*)})^{x^{(s^*)}}, h)$ of F is chosen uniformly random, and independent of any message sent. If the adversary $\mathcal{A}_1$ distinguishes Game

[5]Please note that we omit the value U (e.g., set $U = 1$) when generating the seed of $\pi\mathsf{PRF}$, since plain PRF is enough here.

G_3^1 from Game G_2^1 with non-negligible probability, we can use it to construct algorithm $\mathcal{B}$ that breaks the security of PRF function F. Specifically, the $\mathcal{B}$ is given access to an PRF oracle, which access either F or a truly random function RF. $\mathcal{B}$ simulates the execution environment for $\mathcal{A}_1$ as Game G_2^1 except for the test oracle. When $\mathcal{A}_1$ issue test query, $\mathcal{B}$ send $(\sigma^{(s^*)}, sid_{\hat{A}}^{(s^*)})$ to the PRF oracle. If the oracle is F, the simulation is equivalent to that in Game G_2^1, otherwise the simulation is equivalent to Game G_3^1. Then the algorithm $\mathcal{B}$ outputs 1when $\mathcal{A}_1$ outputs 1 meaning this game proceeds like Game G_3^1. Thus, we have that

$$|\Pr[S_2^1] - \Pr[S_3^1]| \leq \epsilon_{\mathsf{PRF}}. \tag{3}$$

Collect the advantages from Game G_0^1 to Game G_3^1, we have that

$$\epsilon' \leq \ell^2 d^2 \cdot (\epsilon_{\mathsf{BDDH}} + \epsilon_{\mathsf{PRF}}). \tag{4}$$

Proof of Case $C5$:

Similarly, we proceed in Games G_i^2 with adversary $\mathcal{A}_2$ for Case $C5$ as follows. Let S_i^2 be the event that the adversary wins the security experiment in Game G_i^2 respectively.

Game G_0^2. This is the original eCK game with adversary in Case $C5$.

Game G_1^2. This game proceeds as the previous game, except that the simulator aborts if the adversary completes an oracle $\pi_{\hat{B}}^t$ such that $H(sid_{\hat{A}}^{(s^*)}) = H(s_{\hat{B}}^{(t)})$ and $\pi_{\hat{B}}^t$ has no matching conversation to test oracle. Hence we have for any $sid_{\hat{A}}^{(s^*)} \neq sid_{\hat{B}}^{(t)} (t \in [d])$. Then, if the collision event does not occur, the simulation is equivalent to Game G_0^2. When the event does occur, we can easily construct algorithm that breaks the collision-resistant hash function H by outputting $(sid_{\hat{A}}^{(s^*)}, sid_{\hat{B}}^{(t)})$. Hence we have that

$$|\Pr[S_0^2] - \Pr[S_1^2]| \leq \epsilon_{\mathsf{CR}}. \tag{5}$$

Please note that, the rejection rule is possible, since we assume that the simulator would catches all session id s for each oracle. Such rejection event introduced by this game would happen at any point after issuing the test query.

Game G_2^2. The challenger proceeds as Game G_1^2 but aborts the game if it does not correctly guess the test oracle and its peer . Then the probability that the challenger guesses correctly is at least $1/d\ell^2$.

Game G_3^2. We modify game G_2^2 to game G_3^2 by changing the value of

$$e((C_1^{(s)} C_3^{(s)^{\beta^{(s)}\hat{A}}})^{x_1^{(s)}+a_7} \cdot (C_2^{(s)} C_4^{(s)^{\beta_{\hat{A}}^{(s)}}})^{x_2^{(s)}+a_8}, h)$$

in computation of secret material $\sigma_{\hat{A}}^{(s)}$ for oracles $\pi_{\hat{A}}^s$ to

$$(e(X_1^{(s)} A_5, C_1^{(s)} C_4^{(s)^{\beta_{\hat{A}}^{(s)}}}) \cdot e(X_2^{(s)} A_6, C_2^{(s)} C_5^{(s)^{\beta_{\hat{A}}^{(s)}}}) \cdot e(X_3^{(s)} A_7, C_3^{(s)} C_6^{(s)^{\beta_{\hat{A}}^{(s)}}}))^r,$$

where r is uniform random exponent of $h = g^r$ which is chosen by simulator.

This change is purely conceptual, since our long-term private/public keys are all determined when simulator initiates the execution environment of this game. Therefore, we have that

$$\Pr[S_2^2] = \Pr[S_3^2]. \tag{6}$$

Game G_4^2. This game proceeds as the previous game, except that we change the DH tuple $(g, g_1, g_2, A_5, A_6, A_7)$ to a random tuple. First note that the probability that the chosen tuple $(g, g_1, g_2, A_5, A_6, A_7)$, such that $g \neq 1_{\mathcal{G}}, g_1 \neq 1_{\mathcal{G}}, g_2 \neq 1_{\mathcal{G}}, g_1 \neq g_2 \neq g$ and $\log_{g_1} A_5 + \log_{g_2} A_6 \neq \log_g A_7$, is at least $1 - 6/p$. Since the elements in DH tuple $(g, g_1, g_2, A_5, A_6, A_7)$ are uniformly selected at random. If there exists adversary $\mathcal{A}_2$ can distinguish game G_4^2 from game G_3^2, then we can use it to construct an efficient algorithm $\mathcal{B}$ to solve the DLIN problem. Given a DLIN challenge instance (u, u_1, u_2, v, w, c), $\mathcal{B}$ sets $g = u, g_1 = u_1$ and $g_2 = u_2$, and simulates the execution environment as Game G_3^2 except for the long-term public keys of $\hat{A}$: A_5, A_6 and A_7. $\mathcal{B}$ sets $A_5 = v$, $A_6 = w$ and $A_7 = c$. Note that if $c = g^{a_7+a_8}$, then the simulation is equivalent to Game G_3^2. Otherwise the simulation is equivalent to game G_4^2. Now when the algorithm $\mathcal{A}_2$ which is able to distinguish game G_4^2 from G_3^2 outputs 1 (meaning this game proceeds like game G_4^2), $\mathcal{B}$ outputs 1 as answer to the DLIN challenge (meaning c is chosen at random), otherwise $\mathcal{B}$ outputs 0. Therefore, we obtain that

$$|\Pr[S_3^2] - \Pr[S_4^2]| \leq \epsilon_{\mathsf{DLIN}} + 6/p. \tag{7}$$

Game G_5^2. We modify Game G_5^2 to G_6^2 by changing $\pi\mathsf{PRF}$ function F to a truly random function RF for test oracle. Due to the modifications of Game G_3^2 and G_4^2, we first show that key secret $\sigma_{\hat{A}}^{(s^*)}$ and each the key secret $\sigma_{\hat{B}}^{(t)}$ of oracle $\pi_{\hat{B}}^{t}$ are pairwise independent. The tuple $(B_1, B_2, B_3, B_4, \sigma_{\hat{A}}^{(s^*)}, \sigma_{\hat{B}}^{(t)})$ (before evaluating pairing) is denoted by the following equations:

$$\log_g B_1 \equiv \mu_1 b_1 + b_3 \tag{8}$$

$$\log_g B_2 \equiv \mu_2 b_2 + b_3 \tag{9}$$

$$\log_g B_3 \equiv \mu_1 b_4 + b_6 \tag{10}$$

$$\log_g B_4 \equiv \mu_2 b_5 + b_6 \tag{11}$$

$$\begin{aligned}\log_g \sigma_{\hat{A}}^{(s^*)} = mu_1(x_1^{(s^*)} + a_7)(b_1 + b_4\beta_{\hat{A}}^{(s^*)}) + \mu_2(x_2^{(s^*)} + a_8)(b_2 + b_5\beta_{\hat{A}}^{(s^*)}) + \\ (x_1^{(s^*)} + x_2^{(s^*)} + a_0)(b_3 + b_6\beta_{\hat{A}}^{(s^*)}) + \delta^{(s^*)}\end{aligned} \tag{12}$$

$$\begin{aligned}\log_g \sigma_{\hat{B}}^{(t)} = mu_1(n_1^{(t)} + d_7^{(t)})(b_1 + b_4\beta_{\hat{B}}^{(t)}) + \mu_2(n_2^{(t)} + d_8^{(t)})(b_2 + b_5\beta_{\hat{B}}^{(t)}) + \\ (n_1^{(t)} + n_2^{(t)} + d_0^{(t)})(b_3 + b_6\beta_{\hat{B}}^{(t)}) + \delta_{\hat{B}}^{(t)}\end{aligned} \tag{13}$$

Where $g_1 = g^{\mu_1}$, $g_2 = g^{\mu_2}$, $a_0 = a_7 + a_8 + \Delta_a$ (where $\Delta_a \neq 0$), $d_0^{(t)} = d_7^{(t)} + d_8^{(t)} + \Delta_d^{(t)}$,

$$g^{\delta^{(s^*)}} = (X_1^{(s^*)} A_5)^{b_1 + b_4\beta_{\hat{A}}^{(s^*)}} \cdot (X_2^{(s^*)} A_6)^{b_2 + b_5\beta_{\hat{A}}^{(s^*)}} \cdot (X_3^{(s^*)} A_7)^{b_3 + b_6\beta_{\hat{A}}^{(s^*)}} \cdot {Y^{(s^*)}}^{x^{(s^*)}}$$

, and

$$g^{\delta_{\hat{B}}^{(t)}} = (N_1^{(t)} D_5^{(t)})^{b_1 + b_4\beta_{\hat{B}}^{(t)}} \cdot (N_2^{(t)} D_6^{(t)})^{b_2 + b_5\beta_{\hat{B}}^{(t)}} \cdot (N_3^{(t)} D_7^{(t)})^{b_3 + b_6\beta_{\hat{B}}^{(t)}} \cdot {N^{(t)}}^{y^{(t)}}.$$

Let $z_1^{(s^*)} = x_1^{(s^*)} + a_7$, $z_2^{(s^*)} = x_2^{(s^*)} + a_8$, $z_3^{(s^*)} = x_1^{(s^*)} + x_2^{(s^*)} + a_0$, $z_1^{(t)} = n_1^{(t)} + d_7^{(t)}$, $z_2^{(t)} = n_2^{(t)} + d_8^{(t)}$, and $z_3^{(t)} = n_1^{(t)} + n_2^{(t)} + d_0^{(t)}$. We now can obtain a 6×6 matrix denoted by M from the above equations as following:

$$\begin{pmatrix} \log_g B_1 \\ \log_g B_2 \\ \log_g B_3 \\ \log_g B_4 \\ \log_g \sigma_{\hat{A}}^{(s^*)} - \delta^{(s^*)} \\ \log_g \sigma_{\hat{B}}^{(t)} - \delta_{\hat{B}}^{(t)} \end{pmatrix} = \begin{pmatrix} \mu_1 & 0 & 1 & 0 & 0 & 0 \\ 0 & \mu_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu_1 & 0 & 1 \\ 0 & 0 & 0 & 0 & \mu_2 & 1 \\ \mu_1 z_1^{(s^*)} & \mu_2 z_2^{(s^*)} & z_3^{(s^*)} & mu_1 \beta_{\hat{A}}^{(s^*)} z_1^{(s^*)} & \mu_2 \beta_{\hat{A}}^{(s^*)} z_2^{(s^*)} & \beta_{\hat{A}}^{(s^*)} z_3^{(s^*)} \\ \mu_1 z_1^{(t)} & \mu_2 z_2^{(t)} & z_3^{(t)} & \mu_1 \beta_{\hat{B}}^{(t)} z_1^{(t)} & \mu_2 \beta_{\hat{B}}^{(t)} z_2^{(t)} & \beta_{\hat{B}}^{(t)} z_3^{(t)} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{pmatrix} \pmod{\text{p}}$$

One the one hand, If $(g, g_1 g_2, g_3, g^{z_1^{(t)}}, g^{z_2^{(t)}}, g^{z_3^{(t)}})$ is a Linear tuple, then corresponding $\sigma_{\hat{B}}^{(t)}$ is independent from $\sigma_{\hat{A}}^{(s^*)}$, since

$$\log_g \sigma_{\hat{B}}^{(t)} - \delta_{\hat{B}}^{(t)} = \mu_1 z_1^{(t)} (b_1 + b_4 \beta_{\hat{B}}^{(t)}) + \mu_2 z_2^{(t)} (b_2 + b_5 \beta_{\hat{B}}^{(t)}) + z_3^{(t)} (b_3 + b_6 \beta_{\hat{B}}^{(t)})$$

is linearly dependent on $\log_g B_1, \log_g B_2, \log_g B_3$ and $\log_g B_4$, while $\sigma_{\hat{A}}^{(s^*)}$ is linearly independent from $\log_g B_1, \log_g B_2, \log_g B_3$ and $\log_g B_4$ with overwhelming probability (at least $1 - 6/p$). On the other hand, consider that $(g, g_1 g_2, g_3, g^{z_1^{(t)}}, g^{z_2^{(t)}}, g^{z_3^{(t)}})$ is not a linear tuple. We observe that

$$DetM := \mu_1^2 \mu_2^2 (\beta_{\hat{A}}^{(s^*)} - \beta_{\hat{B}}^{(t)})(z_3^{(s^*)} - z_1^{(s^*)} - z_2^{(s^*)})(z_3^{(t)} - z_1^{(t)} - z_2^{(t)}) \neq 0,$$

since $\beta_{\hat{A}}^{(s^*)} - \beta^{(t)} \neq 0$ (by the rejection rule in game G_1^2), $(z_3^{(s^*)} - z_1^{(s^*)} - z_2^{(s^*)}) = \Delta_a \neq 0$ and $z_3^{(t)} - z_1^{(t)} - z_2^{(t)} = \Delta_z^{(t)} = \Delta_n^{(t)} \beta_{\hat{B}}^{(t)} + \Delta_d^{(t)} \neq 0$. Hence, the $\sigma_{\hat{A}}^{(s^*)}$ is independent to $\sigma_{\hat{B}}^{(t)}$, as well as $e(\sigma_{\hat{A}}^{(s^*)}, \sigma_{\hat{A}}^{(s^*)})$ and $e(\sigma_{\hat{B}}^{(t)}, \sigma_{\hat{B}}^{(t)})$. Now if there exist adversary $\mathcal{A}_2$ that is able to distinguish game G_5^2 from game G_5^2 with non-negligible probability (i.e. $b = b'$). Then we can use it to construct an efficient algorithm $\mathcal{B}$ to break the πPRF function with index $\{I_{\mathbb{G}_T}, f_{\mathbb{G}_T}\}$ where $I_{\mathbb{G}_T} := \{(U, V, \alpha) | (U, V, \alpha) \in \mathbb{G}_T^2 \times \mathbb{Z}_p\}$ and $f_{\mathbb{G}_T} := (U, V, \alpha) \to U^{r_1 + \alpha r_2} V$ with $(r_1, r_2) \in_R \mathbb{Z}_p^2$. Algorithm $\mathcal{B}$ simulates the execution environment for $\mathcal{A}_2$ as the challenger in Game G_4^2. In particularly, $\mathcal{B}$ uniformly chooses $(\mu_1, \mu_2) \in_R \mathcal{Z}_p^2$, and sets

$$\eta_1 := \mu_1 b_1 + b_3, \eta_2 := \mu_2 b_2 + b_3,$$

$$\eta_3 := \mu_1 b_4 + b_6, \eta_4 := \mu_2 b_5 + b_6,$$

$$U_{\hat{A}}^{(s^*)} := e(A_7 / (A_5^{\mu_1^{-1}} A_6^{\mu_2^{-1}}), h), U_{\hat{B}}^{(t)} := e(N_3^{(t)} D_7^{(t)} / ((N_1^{(t)} A_5)^{\mu_1^{-1}} (N_2^{(t)} A_6)^{\mu_2^{-1}}), h),$$

$$V_{\hat{A}}^{(s^*)} = e((B_1 B_3^{\beta_{\hat{A}}^{(s^*)}})^{x_1^{(s^*)} + a_7} \cdot (B_2 B_4^{\beta_{\hat{A}}^{(s^*)}})^{x_2^{(s^*)} + a_8} \cdot (W_1^{(s^*)\beta_{\hat{A}}^{(s^*)}} B_5)^{a_1 + a_4 \beta_{\hat{A}}^{(s^*)}} \cdot (W_2^{(s^*)} B_6)^{a_2 + a_5 \beta_{\hat{A}}^{(s^*)}} \cdot$$
$$(W_3^{(s^*)} B_7)^{a_3 + a_6 \beta_{\hat{A}}^{(s^*)}} \cdot (g^{x_1} g^{a_7})^{\eta_1 + \beta_{\hat{A}}^{(s^*)} \eta_3} \cdot (g^{x_2} g^{a_8})^{\eta_2 + \beta_{\hat{A}}^{(s^*)} \eta_4} \cdot W^{(s^*)^{x^{(s^*)}}}, h),$$

$$V^{(t)} := e((D_1^{(t)} D_3^{(t)\beta_{\hat{B}}^{(t)}})^{y_1 + b_7} \cdot (D_2^{(t)} D_4^{(t)\beta_{\hat{B}}^{(t)}})^{y_2 + b_8} \cdot (N_1^{(t)} g^{a_7})^{\eta_1 + \beta_{\hat{B}}^{(t)} \eta_3} \cdot$$
$$(N_2^{(t)} g^{a_7})^{\eta_2 + \beta_{\hat{B}}^{(t)} \eta_4} \cdot N^{(t)^{y^{(t)}}}, h),$$

and $(r_1, r_2) := (b_3, b_6)$. Then the indexes $\sigma_{(U^{(s^*)}, V^{(s^*)}, \beta_{\hat{A}}^{(s^*)})}^{(s^*)} = e(\sigma_{\hat{A}}^{(s^*)}, h)$, and $\sigma_{(U^{(t)}, V^{(t)}, \beta_{\hat{B}}^{(t)})}^{(t)} = e(\sigma_{\hat{B}}^{(t)}, h)$ for $t \in [d]$, where the $\sigma_{\hat{A}}^{(s^*)}$ and $\sigma_{\hat{B}}^{(t)} (t \in [d])$ are the secret key materials of corresponding oracles as in game G_5^2. Thereafter, $\mathcal{B}$ simulates game G_5^2 with adversary $\mathcal{A}_2$ except the computation of $k^{(s^*)}$ and $k^{(t)}(t \in [d])$, where $\mathcal{B}$ gives index $(U^{(s^*)}, V^{(s^*)}, \beta_{\hat{A}}^{(s^*)})$ and $(U^{(t)}, V^{(t)}, \beta_{\hat{B}}^{(t)})(t \in [d])$ to the oracle $(F, I_{\mathcal{G}_T})$ in the experiment of the $\pi\mathsf{PRF}$ security Definition 3 and sets the values returned from the oracle as session keys $k_e^{(s^*)}$ and $k_e^{(t)}(t \in [d])$ respectively. When $\mathcal{A}_2$ output 1 (meaning the simulated game is G_5^2), then $\mathcal{B}$ outputs 1 (meaning the oracle is $(F, I_{\mathcal{G}_T})$). Otherwise $\mathcal{B}$ outputs 0. Since if the oracle is $(F, I_{\mathcal{G}_T})$, the simulated game is equivalent to Game G_4^2, otherwise the simulated game is equivalent to Game G_5^2. Therefore, we obtain that

$$|\Pr[S_4^2] - \Pr[S_5^2]| \leq \epsilon_{\pi\mathsf{PRF}}. \tag{14}$$

Note that, in game G_5^2 the bit b is never use in test query, so that the $\Pr[S_5^2] = 1/2$. Collect the advantages from Game G_0^2 to Game G_5^2, we have that

$$\epsilon' \leq \epsilon_{\mathsf{CR}} + \ell^2 d \cdot (\epsilon_{\mathsf{DLIN}} + \epsilon_{\pi\mathsf{PRF}} + 6/p) \tag{15}$$

5 Conclusions

We have presented an efficient eCK-secure key exchange protocols without random oracles (and without NAXOS trick), that the security against chosen public key attacks based on the plain public key assumption (i.e. without KOSK assumption). An open question here is how to construct an eCK secure protocol without $\pi\mathsf{PRF}$, we leave out this for future work.

Acknowledgements. We would like to thank the anonymous reviewers for their helpful comments.

References

[1] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.

[2] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 390–399. ACM, 2006.

[3] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1993.

[4] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Vaudenay [21], pages 409–426.

[5] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In *IMA Int. Conf.*, pages 30–45, 1997.

[6] Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (sts) protocol. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 154–170. Springer, 1999.

[7] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.

[8] Ran Canetti and Hugo Krawczyk. Security analysis of ike's signature-based key-exchange protocol. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 143–161. Springer, 2002.

[9] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[10] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2007.

[11] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Vaudenay [21], pages 465–485.

[12] Alfred Menezes and Berkant Ustaoglu. Comparing the pre- and post-specified peer models for key agreement. *IJACT*, 1(3):236–250, 2009.

[13] Daisuke Moriyama and Tatsuaki Okamoto. An eck-secure authenticated key exchange protocol without random oracles. *TIIS*, 5(3):607–625, 2011.

[14] Tatsuaki Okamoto. Authenticated key exchange and key encapsulation in the standard model. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 474–484. Springer, 2007.

[15] Inc. RSA Data Security. Certification request syntax standard. RSA Data Security, Inc., 2000.

[16] Augustin P. Sarr, Philippe Elbaz-Vincent, and Jean-Claude Bajard. A new security model for authenticated key agreement. In *SCN*, pages 219–234, 2010.

[17] Augustin P. Sarr, Philippe Elbaz-Vincent, and Jean-Claude Bajard. A secure and efficient authenticated diffie-hellman protocol. In *Proceedings of the 6th European conference on Public key infrastructures, services and applications*, EuroPKI'09, pages 83–98, Berlin, Heidelberg, 2010. Springer-Verlag.

[18] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

[19] Sven Schaege Tibor Jager, Florian Kohlar and Joerg Schwenk. A standard-model security analysis of tls-dhe. Cryptology ePrint Archive, Report 2011/219, 2011. `http://eprint.iacr.org/`.

[20] Berkant Ustaoglu. Comparing *sessionstatereveal* and *ephemeralkeyreveal* for diffie-hellman protocols. In Josef Pieprzyk and Fangguo Zhang, editors, *ProvSec*, volume 5848 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2009.

[21] Serge Vaudenay, editor. *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*. Springer, 2006.

[22] Kazuki Yoneyama and Yunlei Zhao. Taxonomical security consideration of authenticated key exchange resilient to intermediate computation leakage. In Xavier Boyen and Xiaofeng Chen, editors, *ProvSec*, volume 6980 of *Lecture Notes in Computer Science*, pages 348–365. Springer, 2011.