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

April 19, 2013

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 [15].

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 [11] 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. [15, 20, 13, 14]) 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 [11], 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 [5], 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 [12]), 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 [16]) 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 [14] 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.

1.1 Contributions

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 function family, bilinear decision Diffie-Hellman (BDDH) and decision linear Diffie-Hellman (DLIN) assumptions, and pseudo-random function family with pairwise independent random source $\pi\mathsf{PRF}$ [15]. Surprisingly, although our protocol is constructed based on bilinear groups, we avoid any pairing operations during protocol execution. Moreover, our construction doesn't rely on KOSK assumption for public key registration, instead an alternative proof of public key and corresponding check procedure are given for key registration.

1.2 Related Work

In the first eCK security model introduced by LaMacchia, Lauter and Mityagin [11], 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 $\mathsf{EstablishParty}$ query in later literatures.

Since then many eCK secure protocols, e.g. [11, 15, 13, 18, 17], 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 [15] 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 [14, 20] are motivated to propose eCK-secure key exchange protocols without the NAXOS tricks.

Sarr et al. [18], 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. [17] strengthened the eCK model by allowing the adversary 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 [17] 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.

3 Collision-Resistant Hash Functions

Let $\mathsf{CRHF} : \mathcal{K}_{\mathsf{CRHF}} \times \mathcal{M}_{\mathsf{CRHF}} \to \mathcal{Y}_{\mathsf{CRHF}}$ be a family of keyed-hash functions where $\mathcal{K}_{\mathsf{CRHF}}$ is the key space, $\mathcal{M}_{\mathsf{CRHF}}$ is the message space and $\mathcal{Y}_{\mathsf{CRHF}}$ is the hash value space. The public key $hk_{\mathsf{CRHF}} \in \mathcal{K}_{\mathsf{CRHF}}$ defines a hash function, denoted by $\mathsf{CRHF}(hk_{\mathsf{CRHF}}, \cdot)$, which is generated by a PPT algorithm $\mathsf{CRHF.KG}(1^\kappa)$ on input security parameter κ. On input a message $m \in \mathcal{M}_{\mathsf{TCRHF}}$, this function $\mathsf{CRHF}(hk_{\mathsf{CRHF}}, m)$ generates a hash value $y \in \mathcal{Y}_{\mathsf{CRHF}}$.

Definition 1. CRHF is called $(t_{\mathsf{CRHF}}, \epsilon_{\mathsf{CRHF}})$-secure if all $t_{\mathsf{CRHF}} = t_{\mathsf{CRHF}}|(\kappa)$-time adversaries $\mathcal{A}$ have negligible advantage $\epsilon_{\mathsf{CRHF}} = \epsilon_{\mathsf{CRHF}}(\kappa)$ with

$$\Pr\begin{bmatrix} hk_{\mathsf{CRHF}} \leftarrow \mathsf{CRHF.KG}(1^\kappa), (m, m') \leftarrow \mathcal{A}(1^\kappa, hk_{\mathsf{CRHF}}); \\ m \neq m', (m, m') \in \mathcal{M}_{\mathsf{CRHF}}, \mathsf{CRHF}(hk_{\mathsf{CRHF}}, m) = \mathsf{CRHF}(hk_{\mathsf{CRHF}}, m') \end{bmatrix} \leq \epsilon_{\mathsf{CRHF}}$$

where the probability is over the random coins of the adversary and $\mathsf{CRHF.KG}$.

If the hash key hk_{CRHF} is obvious from the context, we write $\mathsf{CRHF}(m)$ for $\mathsf{CRHF}(hk_{\mathsf{CRHF}}, m)$.

3.1 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.

3.2 Pseudo-Random Functions with Pairwise Independent Random Sources ($\pi\mathsf{PRF}$)

This is a specific class of PRF introduced by Okamoto [15]. The $\pi\mathsf{PRF}$ 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.

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

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

3.3 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.

3.4 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.

3.5 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.

4 Security Model

In this section we present the formal security model for two party PKI-based authenticated key-exchange (AKE) protocol. In this model, while emulating the real-world capabilities of an active adversary, we provide an 'execution environment' for adversaries following an important line of research [4, 7, 11] which is initiated by Bellare and Rogaway [3]. In the sequel, we will use the similar framework as [19]. Let $\mathcal{K}_{\mathsf{KE}}$ be the key space of session key, and $\{\mathcal{PK}, \mathcal{SK}\}$ be key spaces for long-term public/private key respectively. Let $\mathcal{IDS}_{\mathsf{KE}}$ be an identity space. Those spaces are associated with security parameter κ of considered protocol.

Execution Environment. In the execution environment, we fix a set of honest parties $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_\ell\}$ for $\ell \in \mathbb{N}$, where ID is identity of a party which is chosen uniquely from space $\mathcal{IDS}_{\mathsf{KE}}$. Each identity is associated with a long-term key pair $(sk_i, pk_i) \in (\mathcal{SK}, \mathcal{PK})$ for entity authentication. Note that those identities are also lexicographically indexed via variable $i \in [\ell]$. For public key registration, each party ID_i might be required to provide extra information (denoted by pf) to prove either the knowledge of the secret key or correctness of registered public key (via e.g. non-interactive proof of knowledge schemes). In practice, the concrete implementation of pf is up to the CA [6] and may be either interactive or non-interactive. Examples can be found in RFC 4210 [6] and PKCS#10. In this model we foucuse on non-interactive proof. Each honest party ID_i can sequentially and concurrently execute the protocol multiple times with different indented partners, this is characterized by a collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$ for $d \in \mathbb{N}$. Oracle π_i^s behaves as party ID_i carrying out a process to execute the s-th protocol instance, which has access to the long-term key pair (sk_i, pk_i) and to all other public keys. Moreover, we assume each oracle π_i^s maintains a list of independent internal state variables with semantics listed in Table 1.

Variable	Decryption
Ψ_i^s	recording the identity ID_j of intended communication partner
Φ_i^s	denoting the decision $\Phi_i^s \in \{\mathtt{accept}, \mathtt{reject}\}$
K_i^s	recording the session key $K_i^s \in \mathcal{K}_{\mathsf{KE}}$ used for symmetric encryption
st_i^s	storing the maximum secret states that allows to be revealed by adversary
sT_i^s	recording the transcript of messages sent by oracle π_i^s
rT_j^t	recording the transcript of messages sent by oracle π_i^s

Table 1: Internal States of Oracles

All those variables of each oracle are initialized with empty string which is denoted by the symbol $\emptyset$ in the following. At some point, each oracle π_i^s may complete the execution always with a decision state $\Phi_i^s \in \{\mathtt{accept}, \mathtt{reject}\}$. Furthermore, we assume that the session key is assigned to the variable K_i^s (such that $K_i^s \neq \emptyset$) iff oracle π_i^s has reached an internal state $\Phi_i^s = \mathtt{accept}$.

Adversarial Model. An adversary $\mathcal{A}$ in our model is a PPT Turing Machine taking as input the security parameter 1^κ and the public information (e.g. generic description of above environment), which 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 the next message m^* (if any) to be sent according to the protocol specification and its internal states. Oracle π_i^s would be initiated as *initiator* via sending the oracle the first message $m = (\top, \widetilde{\mathsf{ID}}_j)$ consisting of a special initialization symbol $\top$ and a value $\widetilde{\mathsf{ID}}_j$. The $\widetilde{\mathsf{ID}}_j$ is either the identity ID_j of intended partner or empty string $\emptyset$. After answering a Send query, the variables $(\Psi_i^s, \Phi_i^s, K_i^s, st_i^s, T_i^s)$ will be updated depending on the specific protocol.[1]

- $\mathsf{RevealKey}(\pi_i^s)$: Oracle π_i^s responds with the contents of variable K_i^s.
- $\mathsf{StateReveal}(\pi_i^s)$: Oracle π_i^s responds with the secret state stored in variable st_i^s.
- $\mathsf{Corrupt}(\mathsf{ID}_i)$: Oracle π_i^1 responds with the long-term secret key sk_i of party ID_i. After this query, oracles $\pi_i^s (s > 1)$ can still answer other queries.
- $\mathsf{EstablishParty}(\mathsf{ID}_m, pk_m, \mathsf{pf}_m)$: This query allows the adversary to register an identity $\mathsf{ID}_m(\ell < m < \mathbb{N})$ and a static public key pk_m on behalf of a party P_m, if either the adversary is ensured to knows the secret key sk_m or the pk_m is ensured to be correctly formed corresponding to the public key pk_m by evaluating pf_m based on pk_m. Parties established by this query are called corrupted or dishonest.[2]
- $\mathsf{Test}(\pi_i^s)$: This query may only be asked once throughout the experiment. Oracle π_i^s handles this query as follows: If the oracle has state $\Omega = \mathtt{reject}$ or $K_i^s = \emptyset$, then it returns some failure symbol $\perp$. Otherwise it flips a fair coin b, samples a random element K_0 from key space $\mathcal{K}_{\mathsf{KE}}$, sets $K_1 = K_i^s$ to the real session key, and returns K_b.

We stress that the exact meaning of the StateReveal must be defined by each protocol separately, and each protocol should be proven secure to resist with such kind of state leakage as its claimed. Namely a protocol should specify the content stored in the variable st during protocol execution.[3] The EstablishParty query is used to model the chosen public key attacks. In this query, the detail form of pf should be specified by each protocol.Please note that one could specify $\mathsf{pf} = \emptyset$ to model arbitrary public key registration without checking anything.

Secure AKE Protocols. To formalize the notion that two oracles are engaged in an on-line communication, we define the partnership via *matching sessions* which was first formulated by Krawczyk [10].

Definition 7. We say that an oracle π_i^s has a *matching session* to oracle π_j^t, if both oracles accept and

- $\Psi_i^s = \mathsf{ID}_j$ and $\Psi_j^t = \mathsf{ID}_i$ and

[1] For example, the variable Ψ_i^s will be set as identity j at some point when the oracle receives a message containing the identity of its partner; the messages (m, m^*) will be orderly appended to transcript T_i^s. A protocol here might be either run in pre- or post-specified peer setting [8].

[2] We only require that the proof is non-interactive to simplify the model (if a common reference string is required we may assume that it is held by the execution environment and made publicly available). In practice, the concrete implementation of these proofs of possession is up to the CA and may also be interactive. We only require that it is secure under concurrent executions.

[3] To make our model simple, we only use a unified StateReveal query other than using different kinds of such queries with various 'aliases'. Since no matter how many different kinds of 'StateReveal' are modelled, each protocol should specify which 'StateReveal' query it can resist with. We here just assume all states stored on untrusted host machine (e.g. the personal computer) are susceptible to StateReveal query.

- $sT_i^s = rT_j^t$ and $rT_i^s = rT_j^s$

Security Game. This game is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, where the following steps are performed:

1. At the beginning of the game, the challenger $\mathcal{C}$ implements the collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$, and generates ℓ long-term key pairs (pk_i, sk_i) and corresponding proof pf_i (if any) for all honest parties ID_i for $i \in [\ell]$ where the identity $\mathsf{ID}_i \in \mathcal{IDS}_{\mathsf{KE}}$ of each party is chosen uniquely. $\mathcal{C}$ gives adversary $\mathcal{A}$ all identities, public keys and corresponding proofs $\{(\mathsf{ID}_1, pk_1, \mathsf{pf}_1), \ldots, (\mathsf{ID}_\ell, pk_\ell, \mathsf{pf}_\ell)\}$ as input.

2. $\mathcal{A}$ may issue polynomial number of queries as aforementioned, namely $\mathcal{A}$ makes queries: Send, StateReveal, Corrupt, EstablishParty and RevealKey.

3. At some point, $\mathcal{A}$ may issue a $\mathsf{Test}(\pi_i^s)$ query on an oracle π_i^s during the game with only once.

4. At the end of the game, the $\mathcal{A}$ terminates with outputting a bit b' as its guess for b of Test query.

For the security definition, we need the notion of *freshness* of oracle.

Definition 8 (Freshness)**.** Let π_i^s be an accepted oracle held by an party ID_i with intended partner ID_j. Meanwhile, let π_j^t be an oracle (if it exists) held by a party ID_j with intended partner ID_i, such that π_i^s has a matching conversation to π_j^t. Then the oracle π_i^s is said to be *fresh* if none of the following conditions holds:

1. The party ID_j is established by adversary $\mathcal{A}$ via EstablishParty query.

2. $\mathcal{A}$ either makes query $\mathsf{RevealKey}(\pi_i^s)$, or $\mathsf{RevealKey}(\pi_j^t)$ (if such π_j^t exists).

3. If π_j^t exists, $\mathcal{A}$ either makes queries:

   (a) Both $\mathsf{Corrupt}(\mathsf{ID}_i)$ and $\mathsf{StateReveal}(\pi_i^s)$, or

   (b) Both $\mathsf{Corrupt}(\mathsf{ID}_j)$ and $\mathsf{StateReveal}(\pi_j^t)$.

4. If π_j^t does not exist, $\mathcal{A}$ either makes queries:

   (a) Both $\mathsf{Corrupt}(\mathsf{ID}_i)$ and $\mathsf{StateReveal}(\pi_i^s)$, or

   (b) $\mathsf{Corrupt}(\mathsf{ID}_j)$.

Definition 9 (Session Key Security)**.** A key exchange protocol Σ is called (t, ϵ)-session-key-secure if for all adversaries $\mathcal{A}$ running within time t in the above security game and for some negligible probability $\epsilon = \epsilon(\kappa)$ in the security parameter κ, it holds that:

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

- When $\mathcal{A}$ returns b' such that

  – $\mathcal{A}$ has issued a Test query on an oracle π_i^s without failure,

  – 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

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

5 A One-round Two Party AKE Protocol from $\pi\mathsf{PRF}$

In this section we present a pairing-based strong AKE protocol without random oracles and NAXOS trick based on pseudo-Random function family with pairwise independent random sources ($\pi\mathsf{PRF}$) as key derivation function.

5.1 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 $\mathsf{CRHF}(hk, \cdot) : \mathcal{K}_{\mathsf{CRHF}} \times \{0,1\}^* \to \mathbb{Z}_p$ where hk is chosen uniformly at random from space $\mathcal{K}_{\mathsf{CRHF}}$.
- A pairwise independent pseudo-random function family ($\pi\mathsf{PRF}$) $F(\cdot,\cdot) : \mathbb{G}_T \times \{0,1\}^* \to \mathcal{K}_{\mathsf{KE}}$, 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 $(r_1, r_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$.

The variable pms stores the public system parameters
$pms := (\mathcal{PG}, e(g, h), e(g_1, h), e(g_2, h), hk)$ where each element is initialized by corresponding building block as above.

$\hat{A}$:

$$sk_A = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)$$
$$pk_A = (A_1, A_2, A_3, A_4, A_5, A_6, A_7) := (e(g_1^{a_1} g^{a_3}, h), e(g_2^{a_2} g^{a_3}, h), e(g_1^{a_4} g^{a_6}, h), e(g_2^{a_5} g^{a_6}, h), e(g_1^{a_7}, h), e(g_2^{a_8}, h), e(g^{a_7+a_8}, h))$$

$$x_1, x_2, x, \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$$
$$X_1 := e(g_1, h)^{x_1}, X_2 := e(g_2, h)^{x_2}, X_3 := e(g, h)^{x_1+x_2}, X := (g, h)^x$$

$\hat{B}$:

$$sk_B = (b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8)$$
$$pk_B = (B_1, B_2, B_3, B_4, B_5, B_6, B_7) := (e(g_1^{b_1} g^{b_3}, h), e(g_2^{b_2} g^{b_3}, h), e(g_1^{b_4} g^{b_6}, h), e(g_2^{b_5} g^{b_6}, h), e(g_1^{b_7}, h), e(g_2^{b_8}, h), e(g^{b_7+b_8}, h))$$

$$y_1, y_2, y \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$$
$$Y_1 := e(g_1, h)^{y_1}, Y_2 := e(g_2, h)^{y_2}, Y_3 := e(g, h)^{y_1+y_2}, Y := e(g, h)^y$$

$\hat{A} \to \hat{B}$: $\xrightarrow{X_1, X_2, X_3, X, \hat{A}}$

$\hat{B} \to \hat{A}$: $\xleftarrow{Y_1, Y_2, Y_3, Y, \hat{B}}$

$\hat{A}$:

$$(Y_1, Y_2, Y_3, Y)? \in \mathbb{G}_T^4$$
$$\mathsf{sid} := \hat{A}||\hat{B}||X_1||X_2||X_3||X||Y_1||Y_2||Y_3||Y$$
$$\beta := \mathsf{CRHF}(sid)$$
$$\sigma_{\hat{A}} := (B_1 B_3^{\beta})^{x_1+a_7} \cdot (B_2 B_4^{\beta})^{x_2+a_8} \cdot (Y_1 B_5)^{a_1+a_4\beta} \cdot (Y_2 B_6)^{a_2+a_5\beta} \cdot (Y_3 B_7)^{a_3+a_6\beta} \cdot Y^x$$
$$k := F(\sigma_{\hat{A}}, \beta)$$

$\hat{B}$:

$$(X_1, X_2, X_3, X)? \in \mathbb{G}_T^4$$
$$\mathsf{sid} := \hat{A}||\hat{B}||X_1||X_2||X_3||X||Y_1||Y_2||Y_3||Y$$
$$\beta := \mathsf{CRHF}(sid)$$
$$\sigma_{\hat{B}} := (A_1 A_3^{\beta})^{y_1+b_7} \cdot (A_2 A_4^{\beta})^{y_2+b_8} \cdot (X_1 A_5)^{b_1+b_4\beta} \cdot (X_2 A_6)^{b_2+b_5\beta} \cdot (X_3 A_7)^{b_3+b_6\beta} \cdot X^y$$
$$k := F(\sigma_{\hat{B}}, \beta)$$

Figure 1: AKE Protocol from $\pi\mathsf{PRF}$

Long-term key generation and Registration: On input the system parameter pms, the long-term keys of each party $\hat{A}$ is generated as following:

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

In order to register the public key pk_A, we require each party $\hat{A}$ to provide the $\mathsf{pf}_{\hat{A}} := (g_1^{a_1}, g_2^{a_2}, g_1^{a_4}, g_2^{a_5}, g^{a_1}, g^{a_2}, g^{a_3}, g^{a_4}, g^{a_5}, g^{a_6})$. Upon receiving a request on registering public key $pk_{\hat{A}}$ with $\mathsf{pf}_{\hat{A}}$, the CA does the following steps:

1. choose random values $\theta_1, \theta_2, \theta_3, \theta_4 \xleftarrow{\$} \mathbb{Z}_p$.

2. reject the registration if

$$e((g_1^{a_1})^{\theta_1}(g_2^{a_2})^{\theta_2}(g_1^{a_4})^{\theta_3}(g_2^{a_5})^{\theta_4}, g) \neq (e(g^{a_1}, g_1)^{\theta_1}(e(g^{a_2}, g_2))^{\theta_2}$$
$$(e(g^{a_4}, g_1))^{\theta_3}(e(g^{a_5}, g_2))^{\theta_4}.$$

3. reject the registration if

$$A_1^{\theta_1} A_2^{\theta_2} A_3^{\theta_3} A_4^{\theta_4} \neq e((g_1^{a_1} g^{a_3})^{\theta_1}(g_2^{a_2} g^{a_3})^{\theta_2}(g_1^{a_4} g^{a_6})^{\theta_3}(g_2^{a_5} g^{a_6})^{\theta_4}, h).$$

4. accept the registration.

Those checks ensure that the public key $pk_{\hat{A}}$ is consistent with the protocol specification (i.e. it is correctly formed). In particular the $\mathsf{pf}_{\hat{A}}$ is required while $\mathcal{A}$ queries the $\mathsf{EstablishParty}(\hat{A}, pk_{\hat{A}}, \mathsf{pf}_{\hat{A}})$ in which the above checks are performed. We will show in the proof that the $\mathsf{pf}_{\hat{A}}$ corresponding to registered public key $pk_{\hat{A}}$ would make security proof go through.
Protocol Execution : On input pms, the protocol between two parties $\hat{A}$ and $\hat{B}$ is proceed as follows, which is also briefly illustrated in the Figure 1.

1. Upon activation a session at a party $\hat{A}$, it performs the steps: (a) choose three ephemeral private keys $x_1, x_2, x, \xleftarrow{\$} \mathbb{Z}_p^3$;

   (b) compute $X_1 := e(g_1, h)^{x_1}, X_2 := e(g_2, h)^{x_2}, X_3 := e(g, h)^{x_1+x_2}$ and $X := (g, h)^x$; (c) send $(X_1, X_2, X_3, X, \hat{A})$ to $\hat{B}$.

2. Upon activation a session at a party $\hat{B}$, it performs the steps: (a) choose three ephemeral private keys $y_1, y_2, y, \xleftarrow{\$} \mathbb{Z}_p^3$;

   (b) compute $Y_1 := e(g_1, h)^{y_1}, Y_2 := e(g_2, h)^{y_2}, Y_3 := e(g, h)^{y_1+y_2}$ and $Y := e(g, h)^y$; (c) send $(Y_1, Y_2, Y_3, Y, \hat{B})$ to $\hat{A}$.

3. Upon receiving $(Y_1, Y_2, Y_3, Y, \hat{B})$ from $\hat{B}$, $\hat{A}$ does the following: (a) verify that $(Y_1, Y_2, Y_3, Y) \in \mathbb{G}_T^4$; (b) set session identifier
$\mathsf{sid} := \hat{A}||\hat{B}||X_1||X_2||X_3||X||Y_1||Y_2||Y_3||Y$, and compute $\beta := \mathsf{CRHF}(\mathsf{sid})$; (c) compute $\sigma_{\hat{A}} := (B_1 B_3^{\beta})^{x_1+a_7} \cdot (B_2 B_4^{\beta})^{x_2+a_8} \cdot (Y_1 B_5)^{a_1+a_4\beta} \cdot (Y_2 B_6)^{a_2+a_5\beta} \cdot (Y_3 B_7)^{a_3+a_6\beta} \cdot Y^x$; (d) compute session key as $k := F(\sigma_{\hat{A}}, \beta)$ and erase all intermediate values and y_1, y_2, y.

4. Upon receiving $(X_1, X_2, X_3, X, \hat{A})$ from $\hat{A}$, $\hat{B}$ does the following: (a) verify that $(X_1, X_2, X_3, X) \in \mathbb{G}_T^4$; (b) set session identifier
$\mathsf{sid} := \hat{A}||\hat{B}||X_1||X_2||X_3||X||Y_1||Y_2||Y_3||Y$, and compute $\beta := \mathsf{CRHF}(\mathsf{sid})$; (c) compute $\sigma_{\hat{B}} := (A_1 A_3^{\beta})^{y_1+b_7} \cdot (A_2 A_4^{\beta})^{y_2+b_8} \cdot (X_1 A_5)^{b_1+b_4\beta} \cdot (X_2 A_6)^{b_2+b_5\beta} \cdot (X_3 A_7)^{b_3+b_6\beta} \cdot X^y$; (d) compute session key $k := F(\sigma_{\hat{B}}, \beta)$ and erase all intermediate values and y_1, y_2, y.

   Session States: We thus assume, only the ephemeral private keys, i.e. (x_1, x_2, x) would be stored as secret in the state variable st. This can be achieved by performing the computation in steps 2.(c) and 2.(d) on a smart card, 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.

5.2 Security Analysis

We show the security of proposed protocol in our strong security model. Assume each ephemeral key chosen during key exchange has bit-size $\lambda \in \mathbb{N}$.

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 $(t, \epsilon_{\mathsf{CRHF}})$-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 $(r_1, r_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$. Then the proposed protocol is a (t', ϵ)-eCK secure AKE in the sense of Definition 9 with $t' \approx t$ and $\epsilon \leq \frac{d^2\ell^2}{2^\lambda} + \epsilon_{\mathsf{CRHF}} + 3 \cdot (d^2\ell^2 \cdot \epsilon_{\mathsf{BDDH}} + d\ell \cdot (\epsilon_{\mathsf{DLIN}} + 6/2^\lambda)) + \epsilon_{\pi\mathsf{PRF}}$.*

We now verify that no polynomially bounded adversary can distinguish the real session key of a fresh oracle from a random key. We fist introduce the notations might be used in the proof. When describing the communication between oracle $\pi_{\hat{A}}^s$ 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 $\pi_{\hat{A}}^s$.
- $(X_1^{(s)}, X_2^{(s)}, X_3^{(s)}, X^{(s)}) = (e(g_1, h)^{x_1^{(s)}}, e(g_2, h)^{x_2^{(s)}}, e(g, h)^{x_1^{(s)}+x_2^{(s)}}, e(g, h)^{x^{(s)}})$: the ephemeral public keys of oracle $\pi_{\hat{A}}^s$.
- $(C_1^{(s)}, C_2^{(s)}, C_3^{(s)}, C_4^{(s)}, C_5^{(s)}, C_6^{(s)}, C_7^{(s)}) = (e(g_1^{c_1^{(s)}} g^{c_3^{(s)}}, h), e(g_2^{c_2^{(s)}} g^{c_3^{(s)}}, h),$
$e(g_1^{c_4^{(s)}} g^{c_6^{(s)}}, h), e(g_2^{c_5^{(s)}} g^{c_6^{(s)}}, h), e(g_1, h)^{c_7^{(s)}}, e(g_2, h)^{c_8^{(s)}}, e(g, h)^{c_7^{(s)}+c_8^{(s)}})$: the public keys of party $\hat{C}^{(s)}$.
- $\mathsf{pf}_{\hat{C}^{(s)}} := (g_1^{c_1^{(s)}}, g_2^{c_2^{(s)}}, g_1^{c_4^{(s)}}, g_2^{c_5^{(s)}}, g^{c_1^{(s)}}, g^{c_2^{(s)}}, g^{c_3^{(s)}}, g^{c_4^{(s)}}, g^{c_5^{(s)}}, g^{c_6^{(s)}})$: the valid consistent proof of $pk_{\hat{C}^{(s)}}$.
- $(W_1^{(s)}, W_2^{(s)}, W_3^{(s)}, W^{(s)}) = (e(g_1, h)^{w_1^{(s)}}, e(g_2, h)^{w_2^{(s)}}, e(g, h)^{w_1^{(s)}+w_2^{(s)}}, e(g, h)^{w^{(s)}})$: the ephemeral public keys received by oracle $\pi_{\hat{A}}^s$.
- $\mathsf{sid}_{\hat{A}}^{(s)} = (\hat{A}, \hat{C}^{(s)}, X_1^{(s)}, X_2^{(s)}, X_3^{(s)}, X^{(s)}, W_1^{(s)}, W_2^{(s)}, W_3^{(s)}, W^{(s)})$.
- $\beta_{\hat{A}}^{(s)} = \mathsf{CRHF}(\mathsf{sid}_{\hat{A}}^{(s)})$.

- $$\sigma_{\hat{A}}^{(s)} = (C_1^{(s)} {C_3^{(s)}}^{\beta_{\hat{A}}^{(s)}})^{x_1^{(s)}+a_7} \cdot (C_2^{(s)} {C_4^{(s)}}^{\beta_{\hat{A}}^{(s)}})^{x_2^{(s)}+a_8} \cdot (W_1^{(s)} C_5^{(s)})^{a_1+a_4\beta_{\hat{A}}^{(s)}} \cdot (W_2^{(s)} C_6^{(s)})^{a_2+a_5\beta_{\hat{A}}^{(s)}} \cdot (W_3^{(s)} C_7^{(s)})^{a_3+a_6\beta_{\hat{A}}^{(s)}} \cdot {W^{(s)}}^{x^{(s)}}.$$
- $k_{\hat{A}}^{(s)} = F(\sigma_{\hat{A}}^{(s)}, \beta_{\hat{A}}^{(s)})$.

We describe the communication between oracle $\pi_{\hat{B}}^{t}$ 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 $\pi_{\hat{B}}^{t}$.
- $(Y_1^{(t)}, Y_2^{(t)}, Y_3^{(t)}, Y^{(t)}) = (e(g_1, h)^{y_1^{(t)}}, e(g_2, h)^{y_2^{(t)}}, e(g, h)^{y_1^{(t)}+y_2^{(t)}}, e(g, h)^{y^{(t)}})$: the ephemeral public keys of oracle $\pi_{\hat{B}}^{t}$.
- $(D_1^{(t)}, D_2^{(t)}, D_3^{(t)}, D_4^{(t)}, D_5^{(t)}, D_6^{(t)}, D_7^{(t)}) = (e(g_1^{d_1^{(t)}} g^{d_3^{(t)}}, h),$ $e(g_2^{d_2^{(t)}} g^{d_3^{(t)}}, h), e(g_1^{d_4^{(t)}} g^{d_6^{(t)}}, h), e(g_2^{d_5^{(t)}} g^{d_6^{(t)}}, h), e(g_1, h)^{d_7^{(t)}}, e(g_2, h)^{d_8^{(t)}}, e(g, h)^{b_7+b_8})$: the public keys of party $\hat{D}^{(t)}$.
- $\mathsf{pf}_{\hat{D}^{(t)}} := (g_1^{d_1^{(t)}}, g_2^{d_2^{(t)}}, g_1^{d_4^{(t)}}, g_2^{d_5^{(t)}}, g^{d_1^{(t)}}, g^{d_2^{(t)}}, g^{d_3^{(t)}}, g^{d_4^{(t)}}, g^{d_5^{(t)}}, g^{d_6^{(t)}})$: the valid consistent proof of public key $pk_{\hat{D}^{(t)}}$.
- $(N_1^{(t)}, N_2^{(t)}, N_3^{(t)}, N^{(t)}) = (e(g_1, h)^{n_1^{(t)}}, e(g_2, h)^{n_2^{(t)}}, e(g, h)^{n_1^{(t)}+n_2^{(t)}}, e(g, h)^{n^{(t)}})$: the ephemeral public keys received by oracle $\pi_{\hat{B}}^{t}$.
- $\mathsf{sid}_{\hat{A}}^{(s)} = (\hat{A}, \hat{C}^{(s)}, Y_1^{(t)}, Y_2^{(t)}, Y_3^{(t)}, Y^{(t)}, N_1^{(t)}, N_2^{(t)}, N_3^{(t)}, N^{(t)})$.
- $\beta_{\hat{A}}^{(s)} = \mathsf{CRHF}(\mathsf{sid}_{\hat{A}}^{(s)})$.
- $$\sigma_{\hat{A}}^{(s)} = (D_1^{(t)} {D_3^{(t)}}^{\beta_{\hat{A}}^{(s)}})^{y_1^{(t)}+a_7} \cdot (D_2^{(t)} {D_4^{(t)}}^{\beta_{\hat{A}}^{(s)}})^{y_2^{(t)}+a_8} \cdot (N_1^{(t)} D_5^{(t)})^{a_1+a_4\beta_{\hat{A}}^{(s)}} \cdot (N_2^{(t)} D_6^{(t)})^{a_2+a_5\beta_{\hat{A}}^{(s)}} \cdot (N_3^{(t)} D_7^{(t)})^{a_3+a_6\beta_{\hat{A}}^{(s)}} \cdot {N^{(t)}}^{y^{(t)}}.$$
- $k_{\hat{B}}^{(t)} = F(\sigma_{\hat{B}}^{(t)}, \beta_{\hat{B}}^{(t)})$.

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 [14], the security proofs for Cases $C1, C3, C4$ 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 $C2$, $C5$ and $C6$ respectively.

Let S_δ be the event that the adversary wins the security experiment under the Game δ and freshness cases in the set $\{C2, C5, C6\}$. Let $\mathsf{Adv}_\delta := \Pr[\mathsf{S}_\delta] - 1/2$ denote the advantage of $\mathcal{A}$ in Game δ.

Game 0. This is the original eCK game with adversary $\mathcal{A}$ under freshness cases C_2, C_5 and C_6. Meanwhile, the challenger chooses an uniform random value $r \xleftarrow{\$} \mathbb{Z}_p$, and sets $h := g^r$ as public parameter. Thus we have that

$$\Pr[\mathsf{S}_0] = 1/2 + \epsilon = \mathsf{Adv}_0 + 1/2.$$

Game 1. In this game, the challenger proceeds exactly like previous game, except that we add an abort rule. The challenger raises event $\mathsf{abort_{eph}}$ and aborts, if during the simulation an ephemeral key X or Y replied by an oracle π_i^s but it has been sample by another oracle π_j^t or sent by adversary before. Since there are $d\ell$ such ephemeral keys would be sampled uniform randomly from $\mathbb{Z}_p$. Thus, the event $\mathsf{abort_{eph}}$ occurs with probability $\Pr[\mathsf{abort_{eph}}] \leq \frac{d^2\ell^2}{2^\lambda}$. We therefore have that

$$|\mathsf{Adv}_0 - \mathsf{Adv}_1| \leq \frac{d^2\ell^2}{2^\lambda}.$$

Note that the ephemeral key chosen by each oracle is unique in this game, so that the adversary can't replay any ephemeral key to result in two oracles generating the same session key but without matching sessions.

Game 2. This game proceeds as the previous game, except that the simulator aborts if the adversary completes two oracles $\pi_{\hat{A}}^s$ and $\pi_{\hat{B}}^t$ such that $\mathsf{CRHF}(\mathsf{sid}_{\hat{A}}^{(s)}) = \mathsf{CRHF}(\mathsf{sid}_{\hat{B}}^{(t)})$ but those two oracles have no matching sessions. Hence we have for any $\mathsf{sid}_{\hat{A}}^{(s)} \neq \mathsf{sid}_{\hat{B}}^{(t)} (s, t \in [d]^2)$. Then, if the collision event does not occur, the simulation is equivalent to Game 1. When the event does occur, we can easily construct algorithm that breaks the collision-resistant hash function CRHF by outputting $(\mathsf{sid}_{\hat{A}}^{(s)}, \mathsf{sid}_{\hat{B}}^{(t)})$. Hence we have that

$$|\mathsf{Adv}_1 - \mathsf{Adv}_2| \leq \epsilon_{\mathsf{CRHF}}.$$

Game 3. In this game proceeds as the previous one, but we modify the game according to guessed freshness cases as: (i) case $C2$: replace the key $k_{\hat{A}}^{(s^*)}$ of test oracle $\pi_{\hat{A}}^{s^*}$ and its partner oracle $\pi_{\hat{B}}^{t^*}$ with random value $\widetilde{k_{\hat{A}}^{(s^*)}}$; (ii) case $C5$: replace the DH tuple $(g, g_1, g_2, A_5, A_6, A_7)$ with a random tuple; and (iii) case $C6$: replace the DH tuple $(g, g_1, g_2, X_1^{(s)}, X_2^{(s)}, X_3^{(s)})$ with a random tuple. First note that the probability that the chosen tuple for instance $(g, g_1, g_2, A_5, A_6, A_7)$, such that $g_1 \neq g_2 \neq g \neq 1_{\mathcal{G}}$ and $\log_{g_1} A_5 + \log_{g_2} A_6 \neq \log_g A_7$, is at least $1 - 6/2^{\lambda}$. 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 an adversary $\mathcal{A}$ can distinguish between the Game 3 from Game 2 then we can use it to construct a distinguisher $\mathcal{D}$ to solve either the BDDH problem in case $C2$ or DLIN in cases $C5, C6$. Basically, $\mathcal{D}$ first guesses: (i) the fresh cases in set $\{C2, C5, C6\}$ with probability at least $1/3$. (ii) the test oracle and its partner oracle for the case $C2$ with probability at least$1/(d^2\ell^2)$; (iii) the test oracle for the cases $C5$ and $C6$ with probability at least $1/(d\ell)$. Since the challenger activates d oracles for each ℓ parties. Meanwhile, if $\mathcal{D}$ guesses incorrectly in either case, then it aborts the game. Next $\mathcal{D}$ simulates game for $\mathcal{A}$ as the simulator in Game 2 except for the following modifications:

1. **Case** $C2$. Given a BDDH challenge instance $(\bar{g}, u, v, w, \Gamma)$, $\mathcal{B}$ does modifications:
   - set $g = \bar{g}$, $h = w$ and $X^{(s^*)} = e(u, h)$ and $Y^{(s^*)} = e(v, h)$;
   - set $g_1 = g^{\mu_1}$ and $g_2 = g^{\mu_2}$, where $\mu_1, \mu_2 \xleftarrow{\$} \mathcal{Z}_p^2$;
   - compute the session key of test oracle as:
     - generate 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 $e(g, h)^{\gamma} = (B_1 B_3^{\beta_{\hat{A}}^{(s^*)}})^{x_1^{(s^*)} + a_7} \cdot (B_2 B_4^{\beta_{\hat{A}}^{(s^*)}})^{x_2^{(s^*)} + a_8}$ $\cdot (Y_1^{(s^*)} B_5)^{a_1 + a_4\beta_{\hat{A}}^{(s^*)}} \cdot (Y_2^{(s^*)} B_6)^{a_2 + a_5\beta_{\hat{A}}^{(s^*)}} \cdot (Y_3^{(s^*)} B_7)^{a_3 + a_6\beta_{\hat{A}}^{(s^*)}}$.
     - compute secret key material $\sigma_{\hat{A}}^{(s^*)} = e(g, h)^{\gamma} \cdot \Gamma$.
     - compute $k^{(s^*)} = F(\sigma^{(s^*)}, \beta_{\hat{A}}^{(s^*)})$.

2. **Case** $C5$. Given a DLIN challenge instance $(u, u_1, u_2, v, w, \Gamma)$, $\mathcal{B}$ does modifications:
   - set $g = u, g_1 = u_1$ and $g_2 = u_2$, and simulates all public/private keys as previous game except for the long-term public keys of $\hat{A}$: A_5, A_6 and A_7;
   - set $A_5 = e(v, h)$, $A_6 = e(w, h)$ and $A_7 = e(\Gamma, h)$, and the values (v, w, Γ) would be included in the proof $\mathsf{pf}_{\hat{A}}$ of $\hat{A}'s$ public key.
   - change the computation of $\sigma_{\hat{A}}^{(s)}$ for any oracle $\pi_{\hat{A}}^{s}$ (including the test oracle) to

   $$\begin{aligned}(e(g^{x_1^{(s)}} v, g^{c_1^{(s)}} g^{c_4^{(s)}\beta_{\hat{A}}^{(s)}} \cdot e(g^{x_2^{(s)}} w, g^{c_3^{(s)}} g^{c_5^{(s)}\beta_{\hat{A}}^{(s)}}) \cdot e(g^{x_1^{(s)} + x_2^{(s)}} \Gamma, g^{c_3^{(s)}} g^{c_6^{(s)}\beta_{\hat{A}}^{(s)}}))^{r} \\ \cdot (W_1^{(s)} C_5^{(s)})^{a_1 + a_4\beta_{\hat{A}}^{(s)}} \cdot (W_2^{(s)} C_6^{(s)})^{a_2 + a_5\beta_{\hat{A}}^{(s)}} \\ \cdot (W_3^{(s)} C_7^{(s)})^{a_3 + a_6\beta_{\hat{A}}^{(s)}} \cdot (W^{(s)})^{x^{(s)}},\end{aligned}$$

   where the values $(g^{c_1^{(s)}}, g^{c_2^{(s)}}, g^{c_3^{(s)}}, g^{c_4^{(s)}}, g^{c_5^{(s)}}, g^{c_6^{(s)}})$ are included in $\mathsf{pf}_{\hat{C}^{(s)}}$.

3. **Case** $C6$. Given a DLIN challenge instance (u, u_1, u_2, v, w, c), $\mathcal{B}$ does modifications:

   - set $g = u, g_1 = u_1$ and $g_2 = u_2$, and simulates all public/private keys as previous game except for the long-term public keys of $\hat{A}$: A_5, A_6 and A_7;
   - set $X_1^{(s^*)} = e(v, h)$, $X_2^{(s^*)} = e(w, h)$ and $X_3^{(s^*)} = e(c, h)$.
   - change the computation of $\sigma_{\hat{A}}^{(s^*)}$ for test oracle $\pi_{\hat{A}}^{(s^*)}$ to

$$
\begin{aligned}
(e(vA_5, g^{c_1^{(s^*)}} g^{c_4^{(s^*)}\beta_{\hat{A}}^{(s^*)}} \cdot e(wA_6, g^{c_3^{(s^*)}} g^{c_5^{(s^*)}\beta_{\hat{A}}^{(s^*)}}) \cdot e(\Gamma A_7, g^{c_3^{(s^*)}} g^{c_6^{(s^*)}\beta_{\hat{A}}^{(s^*)}}))^r \\
\cdot (C_1^{(s^*)} C_3^{(s^*)\beta_{\hat{A}}^{(s^*)}})^{a_7} \cdot (C_2^{(s^*)} C_4^{(s^*)\beta_{\hat{A}}^{(s^*)}})^{a_8} \cdot (W_1^{(s)} C_5^{(s^*)})^{a_1 + a_4\beta_{\hat{A}}^{(s^*)}} \\
\cdot (W_2^{(s^*)} C_6^{(s^*)})^{a_2 + a_5\beta_{\hat{A}}^{(s^*)}} \cdot (W_3^{(s^*)} C_7^{(s^*)})^{a_3 + a_6\beta_{\hat{A}}^{(s^*)}} \cdot (W^{(s^*)})^{x^{(s^*)}}.
\end{aligned}
$$

Note that those change are purely conceptual. If Γ is true value in each case, then the simulation is equivalent to Game 2, otherwise the simulation is equivalent to Game 3. Now when the algorithm $\mathcal{A}$ which is able to distinguish Game 3 from Game 2 outputs 1 (meaning this game proceeds as this game), $\mathcal{B}$ outputs 1 as answer to either the DLIN challenge instance or BDDH challenge instance (meaning Γ is chosen at random), otherwise $\mathcal{B}$ outputs 0. Therefore, we obtain that

$$|\mathsf{Adv}_2 - \mathsf{Adv}_3| \leq 3 \cdot (d^2\ell^2 \cdot \epsilon_{\mathsf{BDDH}} + d\ell \cdot (\epsilon_{\mathsf{DLIN}} + 6/2^\lambda)).$$

Game 4. We modify Game 3 to Game 4 by changing πPRF F to a truly random function $\mathsf{RF}(\cdot)$ for test oracle. If the guessed case is $C2$ then we make use of the fact that the seed $\sigma^{(s^*)}$ of F is chosen uniformly at random. If the adversary $\mathcal{A}$ distinguishes Game 4 from Game 3 with non-negligible probability, we can use it to construct algorithm $\mathcal{B}$ that breaks the security of πPRF F, since such seed is chosen to be independent to other seeds.

As for case $C5$ and $C6$, 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 in this game. We take the simulation for case $C5$ as example (the simulation for case $C6$ is quite similar), the tuple $(B_1, B_2, B_3, B_4, \sigma_{\hat{A}}^{(s^*)}, \sigma_{\hat{B}}^{(t)})$ is denoted by the following equations:

$$
\begin{aligned}
\log_{e(g,h)} B_1 &\equiv \mu_1 b_1 + b_3 \\
\log_{e(g,h)} B_2 &\equiv \mu_2 b_2 + b_3 \\
\log_{e(g,h)} B_3 &\equiv \mu_1 b_4 + b_6 \\
\log_{e(g,h)} B_4 &\equiv \mu_2 b_5 + b_6
\end{aligned}
$$

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

$$
\begin{aligned}
\log_{e(g,h)} \sigma_{\hat{B}}^{(t)} = \mu_1(n_1^{(t)} + d_7^{(t)})(b_1 + b_4\beta_{\hat{B}}^{(t)}) + \mu_2(n_2^{(t)} + d_8^{(t)})(b_2 + b_5\beta_{\hat{B}}^{(t)}) + \\
(n_1^{(t)} + n_2^{(t)} + d_0^{(t)})(b_3 + b_6\beta_{\hat{B}}^{(t)}) + \delta_{\hat{B}}^{(t)}
\end{aligned}
$$

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

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

, 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:

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

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 to $\sigma_{\hat{A}}^{(s^*)}$, since

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

is linearly dependent on $\log_{e(g,h)} B_1, \log_{e(g,h)} B_2, \log_{e(g,h)} B_3$ and $\log_{e(g,h)} B_4$, while $\sigma_{\hat{A}}^{(s^*)}$ is linearly independent to $\log_{e(g,h)} B_1, \log_{e(g,h)} B_2, \log_{e(g,h)} B_3$ and $\log_{e(g,h)} B_4$ with overwhelming probability (at least $1 - 6/2^{\lambda}$). 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 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 $\sigma_{\hat{B}}^{(t)}$. Now if there exist adversary $\mathcal{A}$ that is able to distinguish this game from previous game with non-negligible probability (i.e. $b = b'$). Then we can use it to construct an efficient algorithm $\mathcal{B}$ to break the security of 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 $f_{\mathbb{G}_T} := (U, V, \alpha) \to U^{r_1 + \alpha r_2} V$ with $(r_1, r_2) \xleftarrow{\$} \mathbb{Z}_p^2$. Algorithm $\mathcal{B}$ simulates the game for $\mathcal{A}$ as the challenger in previous game. In particularly, $\mathcal{B}$ uniformly chooses $(\mu_1, \mu_2) \xleftarrow{\$} \mathcal{Z}_p^2$ at random, and sets

$$\eta_1 := \mu_1 b_1 + b_3, \eta_2 := \mu_2 b_2 + b_3,$$
$$\eta_3 := \mu_1 b_4 + b_6, \eta_4 := \mu_2 b_5 + b_6,$$

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

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

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

and $(r_1, r_2) := (b_3, b_6)$. Thereafter, $\mathcal{B}$ simulates this game for adversary $\mathcal{A}$ as previous game except for the computations of $k^{(s^*)}$ and $k^{(t)}(t \in [d])$, where $\mathcal{B}$ gives indices $(U^{(s^*)}, V^{(s^*)}, \beta_{\hat{A}}^{(s^*)})$ 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 $\pi\mathsf{PRF}$ as Definition 3 and sets the values returned from the oracle as session keys $k^{(s^*)}$ and $k^{(t)}(t \in [d])$ respectively. When $\mathcal{A}$ output 1 (meaning the simulation is as the same as the Game 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 3, otherwise the simulated game is equivalent to Game 4. We therefore obtain that

$$|\mathsf{Adv}_3 - \mathsf{Adv}_4| \leq \epsilon_{\pi\mathsf{PRF}}.$$

Note that, in this game the bit b is never used in test query, so that we have that $\mathsf{Adv}_4 = 0$. Collect the advantages from Game 0 to Game 4, we have that

$$\epsilon \leq \frac{d^2\ell^2}{2^\lambda} + \epsilon_{\mathsf{CRHF}} + 3 \cdot (d^2\ell^2 \cdot \epsilon_{\mathsf{BDDH}} + d\ell \cdot (\epsilon_{\mathsf{DLIN}} + 6/2^\lambda)) + \epsilon_{\pi\mathsf{PRF}}.$$

6 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 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 of PROVSEC'12 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] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In *IMA Int. Conf.*, pages 30–45, 1997.

[5] 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.

[6] T. Kause C. Adams, S. Farrell and T. Monen. *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)*. Internet Engineering Task Force RFC 4210,, 2005.

[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] Hugo Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, 2005.

[11] 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.

[12] 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.

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

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

[15] 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.

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

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

[18] 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.

[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.