# Flaw in the Security Analysis of Leakage-resilient Authenticated Key Exchange Protocol from CT-RSA 2016 and Restoring the Security Proof

Suvradip Chakraborty[1], Goutam Paul[2] and C. Pandu Rangan[1]

[1] Department of Computer Science and Engineering,
Indian Institute of Technology Madras, India
{suvradip1111, prangan55}@gmail.com
[2] Cryptology and Security Research Unit (CSRU),
R. C. Bose Centre for Cryptology and Security,
Indian Statistical Institute, Kolkata, India
goutam.paul@isical.ac.in

**Abstract.** In this paper, we revisit the security result of an authenticated key exchange (AKE) protocol recently proposed in CT-RSA 2016 by Chen, Mu, Yang, Susilo and Guo (we refer to this scheme as the CMYSG scheme). The security of the CMYSG scheme is shown in a new (stronger) challenge-dependent leakage-resilient eCK (CLR-eCK) model that captures (bounded) leakage from both the long term secret key of the parties as well the (per-session) randomness of the parties involved in an AKE protocol even after the challenge/test session. In this model, they proposed a generic framework for constructing one-round AKE protocols. The main tool employed in their construction is a (extended) 2-smooth projective hash proof system. The security of their protocol is reduced to the security of the underling hash-proof system, the existence of pseudo-random functions (PRF) and $\pi$-PRFs, collision-resistant hash functions and the Decisional Diffie-Hellman (DDH) hardness assumption. However, we disprove their security result and show that the security of the CMYSG protocol is incorrectly reduced to that of the DDH assumption. We then re-prove the security of the CMYSG scheme in the CLR-eCK model under the Gap Diffie-Hellman (GDH) hardness assumption in the random oracle model. Our security analysis continues the troubled past of the make-and-break efforts of constructing leakage-resilient AKE protocols and also leaves open the construction of CLR-eCK secure AKE protocol in the standard model.

**Keywords:** Authenticated Key Exchange, DDH, GDH, CLR-eCK, leakage-resilient, cryptanalysis, random oracle

## 1 Introduction

Until the 1970s, cryptography was generally viewed as a "dark art" practiced by secret government agencies and a few eccentric amateurs. In developing a strong foundation for modern cryptography, a central task is to decide on the "right" definitions for security of various types of protocols. Provable security is the paradigm in modern cryptography that bridged this gap by transforming cryptography from being a dark art to a science. The "provable security" paradigm roughly states that: A cryptographic scheme $A$ is secure in the security model $B$ provided a set of assumptions $C$ hold. These assumptions are generally based on hard mathematical problems that admit no "efficient"

algorithms for solving them till date or can be based on the security of some underlying primitives such as an encryption scheme or pseudo-random permutations etc. This not only gives us a way to prove security of the proposed cryptographic schemes or primitives from a theoretical perspective but also gives confidence on the soundness of the constructions from a practical standpoint. Given that this is fascinating enough, this approach also has its own limitations. In particular, there has been instances of provably-secure cryptosystems which were later shown to be insecure in practice and/or in theory for various reasons (see [CBH05, PYG08, NLP+11, YPGH11]). In this paper we add one more scheme in this questionable list, namely the work of [CMY+16]. This gives rise to a natural question: Do the cryptographic schemes with security proofs provide security guarantees as claimed?

*Authenticated key exchange* (AKE) protocols allow two parties to establish a common shared secret key with each other over an insecure network. Besides it also allows both the parties to mutually authenticate (implicit or explicit) each other with the assurance that the shared secret key is known only to them. AKE protocols have been widely deployed in many real-world applications for securing communication channels and they form a central component in many network standards, such as IPSec, SSL/TLS, SSH. Bellare and Rogaway [BR94] gave the first formal indistinguishability based security definition for AKE (referred to as BR model). Over the past two decades few variants of the BR model were proposed capturing powerful classes of attacks and stronger security guarantees like the Canetti-Krawzyck (CK) model [CK01] and the extended CK (eCK) models [LLM07]. The eCK model is now the *de-facto* standard for AKE security analysis since it captures more realistic and powerful attack scenarios like Key Compromise Impersonation (KCI), ephemeral-KCI attacks, weak perfect forward secrecy (wPFS), maximal exposure attacks (MEX) etc. We refer the reader to Cremers *et al.* [Cre11] for a detailed comparative analysis of these security models.

However, in all these models the adversary has well-defined (restricted) interface with which it interacts with this primitive (treated as mathematical objects). The security of the AKE protocols hold only in this idealistic setting as long as the adversarial access is limited to that defined in the security model. However in the real-world implementation much more information may leak un-intentionally due to side-channel attacks which includes timing measurements, power analysis, fault injection attacks, electromagnetic measurements, microwave attacks, memory attacks and many more [KJJ99, Koc96, HSH+09]. Leakage-resilient cryptography was introduced to deal with this problem from a theoretical standpoint. It guarantees the security of the cryptosystems even in the face of such side-channel attacks. In the last few years the area of leakage-resilient AKE has been studied each under different assumptions and under different leakage models. Recently, Chen, Mu, Yang, Susilo and Guo [CMY+16] gave a general framework for designing leakage-resilient AKE protocols under the challenge-dependent eCK (CLR-eCK) model which is considered to be more powerful than the sate-of-the-art security models for analyzing leakage-resilient AKE protocols. More precisely, their model captures leakage not only from the static secret keys of the honest parties involved in the AKE protocol, but also captures partial leakage from the (session-specific) ephemeral randomness of the parties. Besides, it also captures *after-the-fact* or *challenge-dependent* leakage where the adversary still has access to the leakage oracle even after the challenge/test session. We notice that the CLR-eCK model is particularly modified from the extended Canetti-Krawczyk (eCK) model [LLM07]. At a high level, they use pseudo-random functions, 2-smooth projective hash functions and strong randomness extractors in their protocol for computing the ephemeral public key and session key to obtain security in the presence of key leakage. The security of the CMYSG scheme is done by reducing it to the security of the underlying cryptographic building

blocks (namely smooth projective hash functions, PRFs and strong randomness extractors) and the Decisional Diffie-Hellman (DDH) hard problem without random oracles. However, in this work we refute their claims and show that the security of the CMYSG scheme is incorrectly reduced to the DDH problem. In fact our analysis shows the inconsistency of the security proof even in the eCK model which also automatically implies the flaw in the proof of the AKE protocol in (stronger) CLR-eCK model also. We fix the flaw and re-prove the eCK security of the CMYSG in the random oracle (RO) model based on the Gap Diffie-Hellman (GDH) assumption. Our result may not directly translate into a direct real world attack against the protocol in [CMY+16]. However, our result points out the inconsistency in their proof and fixes it by rigorously re-proving its security unlike the original paper. It is straightforward to extend our modified security proof of the CMYSG scheme in the CLR-eCK model. Our analysis re-iterates the difficulty of constructing leakage-resilient AKE protocols in stronger leakage models. We hope that further constructions of future leakage-resilient AKE protocols avoids these subtle flaws.

## 2 Related Works

The first construction of leakage-resilient AKE was given by Alwen, Dodis and Wichs [ADW09] from an entropically-unforgeable digital signature scheme secure under chosen-message attacks in the RO model. They considered a leakage-resilient security model (BRM-CK) by extending the CK model to the setting of *Bounded-Retrieval model* (BRM). However, it required 3 passes and also does not capture challenge-dependent leakage.

Moriyama and Okamoto [MO11] introduced a notion of $\lambda$-leakage resilient eCK (LR-eCK) security which is an extension of the eCK security model with the notion of $\lambda$-leakage resilience introduced in [AGV09]. They constructed a 2-pass AKE protocol in this model without random oracles. However, their model did not capture challenge-dependent leakage as in [ADW09].

Alawatugoda *et al.* [ASB14] first modeled after-the-fact leakage or challenge-dependent leakage (it is named as ASB model) for AKE in both the bounded and continuous leakage setting and gave a somewhat generic construction of a two-pass AKE in the model. However, they could not instantiate their generic construction in the ASB continuous leakage model due to non-availability of the underlying continuous leakage-resilient cryptographic primitive in the literature.

In an attempt to solve the mentioned open problem, Alawatugoda *et al.* introduced the Continuous After-the-Fact Leakage (CAFL) model [ABS14] which is a weaker variant of the ASB continuous leakage model in terms of the freshness conditions of the model. They proposed a generic construction of a 2-pass AKE protocol and proved its security formally in the CAFL model. However, both these protocols were later shown to be insecure in their respective models and the security claims were also invalidated in subsequent works. The protocol of [ASB14] was shown to be insecure in the ASB model and the security proof was also shown to be flawed [YL16]. The protocol of [ABS14] was also shown to be completely insecure in the CAFL model by Toorani [Too15] who gave an ephemeral-KCI attack on their protocol. This invalidates the formal security proofs in [ABS14]. In fact this was shown to be insecure in the eCK model also.

## 3 Preliminaries

In this section we provide some basic notations, definitions and tools needed.

### 3.1 Notations

Throughout this work, we denote the security parameter by $\lambda$. We assume that all the algorithms take as input (implicitly) the security parameter represented in unary, i.e., $1^\lambda$. For an integer $n$, we use the notation $[n]$ to denote the set $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$. For a randomized function $f$, we write $f(x; r)$ to denote the unique output of $f$ on input $x$ with random coins $r$. We write $f(x)$ to denote a random variable for the output of $f(x; r)$, over the random coins $r$. For a set $S$, we let $U_S$ denote the uniform distribution over $S$. For an integer $r \in \mathbb{N}$, let $U_r$ denote the uniform distribution over $\{0,1\}^r$, the bit strings of length $r$. For a distribution or random variable $X$, we denote $x \leftarrow X$ the action of sampling an element $x$ according to $X$. For a set $S$, we write $s \stackrel{\$}{\leftarrow} S$ to denote sampling $s$ uniformly at random from the $S$. We use $\text{negl}(\lambda)$ to denote a function that vanishes faster than the inverse of any polynomial, i.e., it denotes the set of negligible functions $\mu(\lambda) = \lambda^{-\omega(1)}$. We denote an ensemble $\mathcal{X}$ as a collection of distributions $\{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$. We sometimes drop the subscript $\lambda$ when clear from context and write $x \leftarrow \mathcal{X}$ instead of $x \leftarrow \mathcal{X}_\lambda$ to denote sampling an element $x$ from $\mathcal{X}_\lambda$.

**Definition 1. (Statistical Distance).** *The statistical distance between two random variables $X$ and $Y$ over a finite domain $\Omega$ denoted by $\Delta(X, Y)$ is defined as:*

$$\Delta(X, Y) = \frac{1}{2} \Sigma_{w \in \Omega} \mid \mathsf{Pr}[X = w] - \mathsf{Pr}[Y = w] \mid .$$

*We say that two variables are $\epsilon$-close, and write $X \approx_\epsilon Y$, if their statistical distance is at most $\epsilon$, i.e., $\Delta(X, Y) \leq \epsilon$.*

We write $X \equiv Y$ to mean that $X$ and $Y$ are identically distributed.

**Definition 2. (Computational Indistinguishability).** *Let $X_n$ and $Y_n$ be two probability distributions over the finite set $\{0,1\}^n$. We say that $X$ and $Y$ are computationally indistinguishable if for all PPT algorithm $\mathcal{A}$, there exists a negligible function $\varepsilon(n)$ such that for every $n \in \mathbb{N}$, we have:*

$$\mid \mathsf{Pr}[t \leftarrow X_n : \mathcal{A}(t) = 1] - \mathsf{Pr}[t \leftarrow Y_n : \mathcal{A}(t) = 1] \mid \leq \varepsilon(n).$$

*We denote $X_n \approx_c Y_n$ to mean $X_n$ and $Y_n$ are computationally indistinguishable.*

### 3.2 Entropy and Randomness Extraction

We begin with some definitions and then state an useful result.

**Definition 3. (Min-Entropy).** The *min-entropy* of a random variable $X$, *denoted as* $\mathrm{H}_\infty(X)$ *is defined as* $\mathrm{H}_\infty(X) \stackrel{\text{def}}{=} \text{-}\log(\max_x \Pr[X = x])$. This is a standard notion of entropy used in cryptography, since it measures the worst-case predictability of $X$.

**Definition 4. (Average Conditional Min-Entropy).** The *average-conditional min-entropy* of a random variable $X$ conditioned on a (possibly) correlated variable $Z$, denoted as $\widetilde{H}_\infty(X|Z)$ is defined as

$$\widetilde{H}_\infty(X|Z) = \text{-}\log\left(\mathbb{E}_{z \leftarrow Z}\left[\max_x \Pr[X = x|Z = z]\right)\right) = \text{-}\log\left(\mathbb{E}_{z \leftarrow Z}\left[2^{\mathrm{H}_\infty(X|Z=z)}\right]\right).$$

This measures the worst-case predictability of $X$ by an adversary that may observe a correlated variable $Z$.

The following bound on average min-entropy was proved in [DORS08].

**Lemma 1.** [DORS08] *For any random variable $X$, $Y$ and $Z$, if $Y$ takes on values in $\{0,1\}^l$, then*

$$\widetilde{\mathrm{H}}_\infty(X|Y,Z) \geq \widetilde{\mathrm{H}}_\infty(X|Z) - l \quad \text{and} \quad \widetilde{\mathrm{H}}_\infty(X|Y) \geq \widetilde{\mathrm{H}}_\infty(X) - l$$

**Definition 5.** (Randomness Extractor). We say that an efficient randomized function $\mathsf{Ext}\colon \mathcal{X}\times\mathcal{S} \to \mathcal{Y}$ is an $(\upsilon,\varepsilon)$-extractor if for all (correlated) random variables $X$, $Z$ such that the support of $X$ is $\mathcal{X}$ and $\widetilde{\mathrm{H}}_\infty(X|Z) \geq \upsilon$, we get $(Z,S,\mathsf{Ext}(X;S)) \approx_\varepsilon (Z,S,U_\mathcal{Y})$, where $S$ is uniform over $\mathcal{S}$, and $U_\mathcal{Y}$ denotes the uniform distribution over the range of the extractor $\mathcal{Y}$.

### 3.3 Pseudo-Random Functions

**Definition 6.** (**Pseudo-random functions**). A function family $\mathcal{F}$ associated with seed space $\{\mathsf{Seed}_k\}_{k\in\mathbb{N}}$, domain $\{\mathsf{Dom}_k\}_{k\in\mathbb{N}}$, and range $\{\mathsf{Rng}_k\}_{k\in\mathbb{N}}$. Formally, for any $\Sigma \xleftarrow{\$} \mathsf{Seed}_k$, $\sigma \xleftarrow{\$} \Sigma$, $\mathcal{D} \xleftarrow{\$} \mathsf{Dom}_k$, $\mathcal{R} \xleftarrow{\$} \mathsf{Rng}_k$, $\mathcal{F}_\sigma^{\lambda,\Sigma,\mathcal{D},\mathcal{R}}$ defines a function which maps an element of $\mathcal{D}$ to an element of $\mathcal{R}$. That is, $\mathcal{F}_\sigma^{\lambda,\Sigma,\mathcal{D},\mathcal{R}}(\rho) \in \mathcal{R}$, for any $\rho \in \mathcal{D}$. We say $\mathcal{F}$ is a pseudo-random function (PRF) family if

$$\{\mathcal{F}_\sigma^{\lambda,\Sigma,\mathcal{D},\mathcal{R}}(\rho_i)\} \approx_c \{\mathsf{RF}(\rho_i)\}.$$

for any ($\rho_i \in \mathcal{D}$) adaptively chosen by any polynomial time distinguisher, where $\mathsf{RF}$ is a truly random function. That is, for any $\rho \in \mathcal{D}$, $\mathsf{RF}(\rho) \xleftarrow{\$} \mathcal{R}$.

**Definition 7. ($\pi$-PRF)**. Roughly speaking, a pseudo-random function with *pairwise-independent* random sources ($\pi$-PRF) refers to a pseudo-random function family that if a specific key $\sigma$ is pairwise-independent from other keys, then the output of function with key $\sigma$ is computationally indistinguishable from a random element.
Formally, let $Z_\Sigma$ be a set of random variables over $\Sigma$, and $I_\Sigma$ be a set of indices regarding $\Sigma$ such that there exits a deterministic polynomial-time algorithm, $f_\Sigma : I_\Sigma \to Z_\Sigma$ which on input the index $i \in I_\Sigma$ outputs $\sigma_i \in Z_\Sigma$. Consider the random variables $\{\sigma_{i_j}\}_{j=0,\cdots,q(\lambda)} = \{f_\Sigma(i_j)\}_{j=0,\cdots,q(\lambda)}$, where $i_j \in I_\Sigma$ and $q(\lambda)$ a polynomial function of $\lambda$. We say that $\sigma_{i_0}$ is *pairwise independent* from other variables $\sigma_{i_1},\cdots,\sigma_{i_{q(\lambda)}}$ if for any pair of $(\sigma_{i_0},\sigma_{i_j})$ $(j=1,\cdots,q(\lambda))$, for any $(x,y) \in \Sigma^2$, we have $\mathsf{Pr}[\sigma_{i_0} \to x \ \wedge \ \sigma_{i_j} \to y] = \frac{1}{\Sigma^2}$. Define $\widetilde{\mathcal{F}} = \mathcal{F}_{\sigma_{i_j}}^{\lambda,\Sigma,\mathcal{D},\mathcal{R}}(\rho_j)$, for $i_j \in I_\Sigma$, $\rho_j \in \mathcal{D}$. We say that $F$ is a $\pi$-PRF family if

$$\{\widetilde{\mathcal{F}}(\rho_j)\} \approx_c \{\widetilde{\mathsf{RF}}(\rho_j)\}.$$

for any $\{i_j \in I_\Sigma, \rho_j \in \mathcal{D}\}$ $(j=1,\cdots,q(\lambda))$ adaptively chosen by any polynomial time distinguisher such that $\sigma_{i_0}$ is pairwisely independent from $\sigma_{i_j}(j>0)$ where $\widetilde{\mathsf{RF}}$ is the same as $\widetilde{\mathcal{F}}$ except that $\widetilde{\mathsf{RF}}(\rho_0)$ is replaced by a truly random value in $\mathcal{R}$.

### 3.4 Smooth Projective Hash Functions

The paradigm of smooth projective hash functions (SPHF) was originally introduced by Cramer Shoup [CS02] for constructing a practical CCA-secure public key encryption. In this section we

describe an extended version of smooth projective hashing as will be required for the construction later on.

A SPHF requires the existence of a well-defined domain $\mathcal{X}$ and an NP language $\mathcal{L}$ underlying it, where the elements of $\mathcal{L}$ form a proper subset of the domain $\mathcal{X}$, i.e., $\mathcal{L} \subset \mathcal{X}$. Informally, a projective hash family is a family of *keyed* hash functions associated with two types of keys: the primary hashing key $\mathsf{hk}$ and a projective key $\mathsf{hp}$, which can be seen analogous to private and public keys respectively. The hashing key $\mathsf{hk}$ can be used to evaluate the hash function on every point of its domain; whereas the projection key $\mathsf{hp}$ can be used to compute the hash function on a "special subset" of the domain, namely, the valid words that belong to the language $\mathcal{L}$. The correctness of such a hash proof system guarantees that: when computed on a word $W \in \mathcal{L}$, both functions should lead to the same result: $\mathsf{Hash}(\mathsf{hk}, params, \mathcal{L}, W, aux)$ with the hashing key, public parameters $params$, some auxiliary information $aux$ and $\mathsf{ProjHash}(\mathsf{hp}, params, \mathcal{L}, W, w, aux)$ with the projection key and a witness $w$ for the fact that $W \in \mathcal{L}$ (since this is a valid word), along with $params$ and $aux$. Apart from these, we also need another word generation algorithm $\mathsf{WordG}(param, \mathcal{L}, w)$, that generates a word $W \in \mathcal{L}$ using the witness $w$. Of course, for an invalid word $W \in \mathcal{X} \setminus \mathcal{L}$ there does not exist such a witness, and the *smoothness* property states that the hash value $\mathsf{Hash}(\mathsf{hk}, param, \mathcal{L}, W)$ for such an invalid word is independent of $\mathsf{hp}$. To summarize, a SPHF has the property that the projection key uniquely determines the hash value of any word in the language $\mathcal{L}$ but gives almost no information about the hash value of any point in $\mathcal{X} \setminus \mathcal{L}$. An important property that is used in all the applications of such families is that it is hard to distinguish members of the special subset $\mathcal{L}$ from non-members of the set $\mathcal{X} \setminus \mathcal{L}$. This is called the *hard subset membership* property.

**Syntax.** A SPHF over a language $\mathcal{L} \subset \mathcal{X}$, onto a set $\mathcal{Y}$, is defined by six polynomial-time algorithms:

- $\mathsf{SPHFSetup}(1^\lambda)$: The setup algorithm takes as input the security parameter $\lambda$ (in unary) and outputs the public parameters $params$ and the description of the underlying NP language $\mathcal{L}$.
- $\mathsf{HashKG}(params, \mathcal{L})$: Generates the hashing key $\mathsf{hk}$ for the language $\mathcal{L}$.
- $\mathsf{ProjKG}(\mathsf{hk}, params, \mathcal{L}, )$: Derives the projection key $\mathsf{hp}$ from the hashing key $\mathsf{hk}$.
- $\mathsf{WordG}(param, \mathcal{L}, w)$: Takes as input a witness $w$, and generates a word $W \in \mathcal{L}$.
- $\mathsf{Hash}(\mathsf{hk}, params, \mathcal{L}, W, aux)$: Takes as input the hashing key $\mathsf{hk}$ and a word $W$, along with some auxiliary information $aux$, and outputs the hash value $hv \in \mathcal{Y}$.
- $\mathsf{ProjHash}(\mathsf{hp}, params, \mathcal{L}, W, w, aux)$: Takes as input the the projection key $\mathsf{hp}$ and a word $W \in \mathcal{L}$, along with the witness $w$, auxiliary information $aux$, and outputs the hash value $hv' \in \mathcal{Y}$.

A (extended) SPHF = ($\mathsf{SPHFSetup}$, $\mathsf{HashKG}$, $\mathsf{ProjKG}$, $\mathsf{WordG}$, $\mathsf{Hash}$, $\mathsf{ProjHash}$) should satisfy the following properties:

1. **Correctness:** Let $W = \mathsf{WordG}(param, \mathcal{L}, w)$, then $\forall$ $\mathsf{hk}$ and $\mathsf{hp}$, we have:

$$\mathsf{Hash}(\mathsf{hk}, params, \mathcal{L}, W, aux) = \mathsf{ProjHash}(\mathsf{hp}, params, \mathcal{L}, W, w, aux).$$

2. **Smoothness:** For any $W \in \mathcal{X} \setminus \mathcal{L}$, the following two distributions are perfectly indistinguishable:

$$\mu_1 = \{(\mathcal{L}, params, W, \mathsf{hp}, aux, hv) \,|\, hv = \mathsf{Hash}(\mathsf{hk}, params, \mathcal{L}, W, aux)\},$$
$$\mu_2 = \{(\mathcal{L}, params, W, \mathsf{hp}, aux, hv) \,|\, hv \xleftarrow{\$} \mathcal{Y}\}.$$

**Definition 8.** (2-smooth SPHF). For any $W_1, W_2 \in \mathcal{X} \setminus \mathcal{L}$ and auxiliary informations $aux_1$ and $aux_2$, such that $(W_1, aux_1) \neq (W_2, aux_2)$, we say an SPHF is 2-smooth if the following two distributions are perfectly indistinguishable:

$$\mu_1 = \{(\mathcal{L}, params, W_1, W_2, \mathsf{hp}, aux_1, aux_2, hv_1, hv_2) \,|\, hv_2 = \mathsf{Hash}(\mathsf{hk}, params, \mathcal{L}, W_2, aux_2)\},$$
$$\mu_2 = \{(\mathcal{L}, params, W_1, W_2, \mathsf{hp}, aux_1, aux_2, hv_1, hv_2) \,|\, hv_2 \xleftarrow{\$} \mathcal{Y}\},$$

where $hv_1 = \mathsf{Hash}(\mathsf{hk}, params, \mathcal{L}, W_1, aux_1)$.

**Definition 9.** (**Hard Subset Membership problem**). As a computational problem we require that the subset membership problem is *hard*, which means that for random valid word $W_0 \in \mathcal{L}$, and a random invalid word $W_1 \in \mathcal{X} \setminus \mathcal{L}$, the two words $W_0$ and $W_1$ are computationally indistinguishable. This is formally captured by defining the advantage function $\mathsf{Adv}^{\mathsf{SM}}_{\mathsf{SPHF},\mathcal{A}}(\lambda)$ of an adversary $\mathcal{A}$ as:

$$\mathsf{Adv}^{\mathsf{SM}}_{\mathsf{SPHF},\mathcal{A}}(\lambda) = \mid \mathsf{Pr}_{W_0 \xleftarrow{\$} \mathcal{L}}[\mathcal{A}(params, \mathcal{X}, \mathcal{L}, W_0) = 1] - \mathsf{Pr}_{W_1 \xleftarrow{\$} \mathcal{X} \setminus \mathcal{L}}[\mathcal{A}(params, \mathcal{X}, \mathcal{L}, W_1) = 1] \mid,$$

where $params$, $\mathcal{X}$, and $\mathcal{L}$ are generated using $\mathsf{SPHFSetup}(1^\lambda)$. We define the subset membership to be $(t_{\mathsf{SM}}, \epsilon_{\mathsf{SM}})$ hard if for all $t_{\mathsf{SM}}$-time adversaries the above advantage $\mathsf{Adv}^{\mathsf{SM}}_{\mathsf{SPHF},\mathcal{A}}(\lambda)$ is at most $\epsilon_{\mathsf{SM}}$.

### 3.5 Complexity Assumptions

In this section, we present a brief overview of the hard problem assumptions.

**Definition 10. Computation Diffie-Hellman Problem (CDH) -** Given $(g, g^a, g^b) \xleftarrow{\$} \mathbb{G}^3$ for unknown $a, b \in \mathbb{Z}_p^*$, where $\mathbb{G}$ is a cyclic prime order multiplicative group with $g$ as a generator and $p$ the order of the group, the CDH problem in $\mathbb{G}$ is to compute $g^{ab}$.

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the CDH problem in $\mathbb{G}$ is defined as

$$\mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{A}} = \mathsf{Pr}\left[\mathcal{A}(g, g^a, g^b) = g^{ab} \mid a, b \in \mathbb{Z}_p^*\right].$$

The *CDH Assumption* is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{A}}$ is negligibly small.

**Definition 11. Decisional Diffie-Hellman Problem (DDH) -** Given $(g, g^a, g^b, h) \xleftarrow{\$} \mathbb{G}^4$ for unknown $a, b \in \mathbb{Z}_p^*$, where $\mathbb{G}$ is a cyclic prime order multiplicative group with $g$ as a generator and $p$ the order of the group, the DDH problem in $\mathbb{G}$ is to determine whether $h \stackrel{?}{=} g^{ab}$ or a random group element.

The advantage of any probabilistic polynomial time algorithm $\mathcal{A}$ in solving the DDH problem in $\mathbb{G}$ is defined as

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}} = |\mathsf{Pr}\left[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1\right] - \mathsf{Pr}\left[\mathcal{A}(g, g^a, g^b, h) = 1\right] \mid a, b \in \mathbb{Z}_q^*|.$$

The *DDH Assumption* is that, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A}}$ is negligibly small.

**Definition 12. Gap Diffie Hellman Assumption (GDH).** Given $(g, g^a, g^b) \xleftarrow{\$} \mathbb{G}^3$ and access to a Decision Diffie Hellman (DDH) oracle $\mathcal{DDH}(\cdot,\cdot,\cdot)$ which on input $g^a$, $g^b$ and $g^c$ outputs `True` if and only if $c = ab$, the Gap Diffie Hellman problem is to compute $g^{ab} \in \mathbb{G}$.
The advantage of an adversary $\mathcal{A}$ in solving the Gap Diffie Hellman problem is defined as

$$\mathsf{Adv}^{\mathsf{GDH}}_{\mathcal{A}} = \mathsf{Pr}\left[\mathcal{A}^{\mathcal{DDH}(\cdot,\cdot,\cdot)}(g, g^a, g^b) = g^{ab}\right].$$

The Gap Diffie Hellman assumption holds in $\mathbb{G}$ if for all polynomial time adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{GDH}}_{\mathcal{A}}$ is negligible.

## 4 eCK security model for AKE

In this section we describe the extended Canetti-Krawzyck (eCK) security model for AKE proposed by Lauter, LaMacchia and Mityagin [LLM07]. The original Challenge-Dependent Leakage-Resilient eCK (CLR-eCK) model introduced in [CMY+16] builds on top of the eCK model and considers bounded leakage from both the static secret keys of parties and session-specific randomness. However, for our analysis we do not need the CLR-eCK model. The eCK model is enough to show our result, which in turn automatically implies the relevance of our results in the CLR-eCK model as well. Hence, for simplicity, we specify the eCK model here. While emulating the real world capabilites of an active adversary, we provide an execution environment for adversaries, similar to the approach initiated by Bellare and Rogaway [BR94] and later further investigated in [LLM07, JKSS12, YS12].

### 4.1 Execution environment

In the execution environment, we fix a set of $\ell$ honest parties $\{\mathsf{ID}_1, \cdots, \mathsf{ID}_\ell\}_{\ell \in \mathbb{N}}$, where $\mathsf{ID}_i$ ($i \in [\ell]$) denotes the identity of a party chosen uniquely from some identity space $\mathcal{IDS}$ (note that we are not in the identity-based setting). Each identity is associated with a long-term key pair $(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_i}) \in (\mathcal{SK} \times \mathcal{PK})$ for authentication. We also assume that the certification authority (CA) issues a certificate that binds the long term public key to the party. Note that these identities are also lexicographically indexed via variable $i \in [\ell]$. Each honest party $\mathsf{ID}_i$ can sequentially and also concurrently execute the protocol multiple times with different intended partners. This is characterized by a collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$, for $d \in \mathbb{N}$. Oracle $\pi_i^s$ behaves as party $\mathsf{ID}_i$ carrying out a process to execute the $s$-th protocol instance (session), which has access to the long-term key pair $(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_i})$ and to all other public keys. Moreover, we assume each oracle $\pi_i^s$ maintains a list of independent internal state variables with semantics listed in Table 1.

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

### 4.2 Adversarial Model

An adversary $\mathcal{A}$ is a PPT Turing Machine taking as input the security parameter $1^\lambda$ and the public information (e.g. generic description of above environment), which may interact with these oracles by issuing the following queries.

**Table 1.** Internal states of oracles.

| Variable | Description |
|---|---|
| $\psi_i^s$ | storing the identity and public key of its intended communication partner e.g., $(\mathsf{ID}_j, pk_{\mathsf{ID}_j})$) |
| $\Phi_i^s$ | denoting the decision $\Phi_i^s \in$ {accept; reject} |
| $\rho_i^s$ | denoting the role identifier $\rho_i^s \in \{\text{Initiator}(\mathcal{I}), \text{Responder}(\mathcal{R})\}$ |
| $K_i^s$ | recording the session key of the session $\pi_i^s$ |
| $st_i^s$ | storing the ephemeral keys that allows to be revealed, e.g. the randomness used to generate ephemeral public key |
| $sT_i^s$ | recording the transcript of messages sent by oracle $\pi_i^s$ |
| $rT_i^s$ | recording the transcript of messages received by oracle $\pi_i^s$ |

1. $\mathsf{Send}(\pi_i^s, m)$: The adversary can use this query to send any message $m$ of his own choice to $\pi_i^s$. The oracle will respond the next message $m^*$ (if any) to be sent according to the protocol specification and its internal states. Oracle $\pi_i^s$ 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 value $\widetilde{\mathsf{ID}_j}$ is either the identity $\mathsf{ID}_j$ of intended partner or the empty string $\emptyset$. After answering a $\mathsf{Send}$ query, the variables $(\psi_i^s, \Phi_i^s, \rho_i^s, K_i^s, st_i^s, rT_j^s)$ will be updated depending on the specific protocol.
2. $\mathsf{RevealKey}(\pi_i^s)$: Oracle $\pi_i^s$ responds with the contents of variable $K_i^s$.
3. $\mathsf{EphemeralKeyReveal}(\pi_i^s)$: Oracle $\pi_i^s$ responds with its ephemeral secret keys (i.e. per-session randomness of the oracle).
4. $\mathsf{Corrupt}(\mathsf{ID}_i)$: The long-term secret key $sk_{\mathsf{ID}_i}$ of party $\mathsf{ID}_i$ is returned if $i \in [l]$; otherwise a failure symbol $\perp$ is returned.
5. $\mathsf{EstablishParty}(\mathsf{ID}_\tau)$: This query allows an adversary to register a public key on behalf of party $(\mathsf{ID}_\tau)$ $(l < \tau)$ to the system. Parties established by this query are said to be *dishonest*.
6. $\mathsf{Test}(\pi_i^s)$: If the oracle has state $\Phi_i^s = \mathsf{reject}$ or $K_i^s = \emptyset$, then the oracle $\pi_i^s$ returns some failure symbol $\perp$. Otherwise, it flips a fair coin $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, it samples a random element $K_0$ from key space $\mathcal{K}$; if $b = 1$, it sets $K_1 = K_i^s$. Finally, the key $K_b$ is returned.

### 4.3 Secure AKE protocols

To formalize the notion that two oracles are engaged in an online communication, we define the partnership via *matching* sessions.

**Definition 13. (Matching Session)**. We say that an oracle $\pi_i^s$ has a matching session to oracle $\pi_j^t$, if $\pi_i^s$ has sent all protocol messages and all the following conditions hold:

- $\psi_i^s = \mathsf{ID}_j$ and $\psi_j^t = \mathsf{ID}_i$.
- $rT_j^t = sT_i^s$ and $rT_i^s = sT_j^t$.
- $\rho_i^s \neq \rho_j^t$.

If an oracle $\pi_i^s$ has a matching session to oracle $\pi_j^t$, then $\pi_i^s$ is said to be a *partner* oracle of $\pi_j^t$.

**Correctness:** We say an AKE protocol $\Pi$ is correct, if two oracles $\pi_i^s$ and $\pi_j^t$ accept with matching sessions, then both oracles hold the same session key, i.e. $K_i^s = K_j^t$.

**Definition 14. (Session/Oracle Freshness).** Let $\Pi$ be a protocol, and $\mathsf{ID}_i$ and $\mathsf{ID}_j$ be two honest parties, $\pi_i^s$ be an accepting oracle with intended partner $\mathsf{ID}_j$. Meanwhile let $\pi_j^t$ be an oracle (if it exists) with intended partner $\mathsf{ID}_i$, such that $\pi_i^s$ has a matching session to $\pi_j^t$. Then the oracle $\pi_i^s$ is said to be $fresh$ if none of the following holds:

1. $\mathcal{A}$ issued either a $\mathsf{RevealKey}(\pi_i^s)$ or $\mathsf{RevealKey}(\pi_j^t)$ query (if it exists).
2. $\mathcal{A}$ issued both $\mathsf{Corrupt}(\mathsf{ID}_i)$ and $\mathsf{EphemeralKeyReveal}(\pi_i^s)$ queries.
3. If $\pi_j^t$ exists, $\mathcal{A}$ issued both $\mathsf{Corrupt}(\mathsf{ID}_j)$ and $\mathsf{EphemeralKeyReveal}(\pi_j^t)$ queries.
4. If $\pi_j^t$ does not exist, $\mathcal{A}$ issued $\mathsf{Corrupt}(\mathsf{ID}_j)$.

**Security Experiment** $\mathsf{EXP}_{\Pi,\mathcal{A}}^{\mathsf{eCK}}(1^\lambda)$: On input security parameter $1^\lambda$, the security experiment proceeds as a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ based on an AKE protocol $\Pi$, 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 $\ell$ long-term key pairs $(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_i})$ for all honest parties $\mathsf{ID}_i$ for $i \in [\ell]$ where the identity $\mathsf{ID}_i \in \mathcal{IDS}$ of each party is chosen uniquely. $\mathcal{C}$ gives adversary $\mathcal{A}$ all identities and public keys $\{(\mathsf{ID}_1, pk_{\mathsf{ID}_1}), \cdots, (\mathsf{ID}_\ell, pk_{\mathsf{ID}_\ell})\}$ as input.
2. $\mathcal{A}$ may issue polynomial number of aforementioned queries adaptively, namely $\mathcal{A}$ makes queries: $\mathsf{Send}$, $\mathsf{EphemeralKeyReveal}$, $\mathsf{EstablishParty}$, $\mathsf{Corrupt}$ and $\mathsf{RevealKey}$.
3. At some point, $\mathcal{A}$ may issue a $\mathsf{Test}(\pi_i^s)$ query on an oracle $\pi_i^s$ during the game only once.
4. $\mathcal{A}$ may continue to issue $\mathsf{Send}$, $\mathsf{EphemeralKeyReveal}$, $\mathsf{EstablishParty}$, $\mathsf{Corrupt}$ and $\mathsf{RevealKey}$ queries adaptively, provided the $\mathsf{Test}$ oracle remains *fresh* throughout the game.
5. At the end of the game, the $\mathcal{A}$ may terminate with returning a bit $b'$ as its guess for $b$ of $\mathsf{Test}$ query. Return 1, if $b' = b$, otherwise return 0.

**Definition 15.** (**eCK security**). We say that an adversary $\mathcal{A}$ $(t, \epsilon)$-breaks the security of a correct AKE protocol $\Pi$, if $\mathcal{A}$ runs the AKE security game within time $t$, and the probability bound $|\mathsf{Pr}[\mathsf{EXP}_{\Pi,\mathcal{A}}^{\mathsf{eCK}}(1^\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ holds for all adversaries $\mathcal{A}$ running within time $t$ in the above security experiment and for some negligible probability $\epsilon = \epsilon(\lambda)$ in the security parameter $\lambda$. We refer $|\mathsf{Pr}[\mathsf{EXP}_{\Pi,\mathcal{A}}^{\mathsf{eCK}}(1^\lambda) = 1] - \frac{1}{2}|$ as the advantage $\mathsf{Adv}_{\Pi}^{\mathsf{eCK}}(\mathcal{A})$ of the adversary $\mathcal{A}$ in the AKE protocol $\Pi$ in the eCK model.

## 5 The CMYSG CLR-eCK AKE protocol [CMY+16]

In this section we review the AKE protocol by Chen, Mu, Yang, Susilo and Guo [CMY+16] (i.e., the CMYSG scheme) (please see **Table 2**).
The CMYSG scheme makes use of the following cryptographic building blocks:

- Group $\mathbb{G}$ of prime order $p$ and a random generator $g$.

**Table 2.** CMYSG protocol $\Pi$ [CMY+16]

| $\mathsf{ID}_A$ | | $\mathsf{ID}_B$ |
|---|---|---|
| **Long-Term Key Generation** | | |
| $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(params, \mathcal{L})$ | | $\mathsf{hk}' \xleftarrow{\$} \mathsf{HashKG}(params, \mathcal{L})$ |
| $\mathsf{hp} \xleftarrow{\$} \mathsf{ProjKG}(\mathsf{hk}, params, \mathcal{L})$ | | $\mathsf{hp}' \xleftarrow{\$} \mathsf{ProjKG}(\mathsf{hk}', params, \mathcal{L})$ |
| $r_{A_1} \xleftarrow{\$} \{0,1\}^{t_1(\lambda)}$, $r_{A_2} \xleftarrow{\$} \{0,1\}^{t_2(\lambda)}$ | | $r_{B_1} \xleftarrow{\$} \{0,1\}^{t_1(\lambda)}$, $r_{B_2} \xleftarrow{\$} \{0,1\}^{t_2(\lambda)}$ |
| $lsk_A = \mathsf{hk}$, $lpk_A = (\mathsf{hp}, r_{A_1}, r_{A_2})$ | | $lsk_B = \mathsf{hk}'$, $lpk_B = (\mathsf{hp}', r_{B_1}, r_{B_2})$ |
| **Session Execution** | | |
| $esk_A \xleftarrow{\$} \{0,1\}^{u(\lambda)}$, $t_A \xleftarrow{\$} \{0,1\}^{t_3(\lambda)}$ | | $esk_B \xleftarrow{\$} \{0,1\}^{u(\lambda)}$, $t_B \xleftarrow{\$} \{0,1\}^{t_3(\lambda)}$ |
| $\widehat{lsk}_A = \mathsf{Ext}_1(lsk_A, r_{A_1})$ | | $\widehat{lsk}_B = \mathsf{Ext}_1(lsk_B, r_{B_1})$ |
| $\widehat{esk}_A = \mathsf{Ext}_2(esk_A, r_{A_2})$ | | $\widehat{esk}_B = \mathsf{Ext}_2(esk_B, r_{B_2})$ |
| $(w_A, x) = \hat{\mathcal{F}}_{\widehat{lsk}_A}(esk_A) + \bar{\mathcal{F}}_{\widehat{esk}_A}(r_{A_1})$ | | $(w_B, y) = \hat{\mathcal{F}}_{\widehat{lsk}_B}(esk_B) + \bar{\mathcal{F}}_{\widehat{esk}_B}(r_{B_1})$ |
| $W_A = \mathsf{WordG}(param, \mathcal{L}, w_A), X = g^x$ | | $W_B = \mathsf{WordG}(param, \mathcal{L}, w_B), Y = g^y$ |
| Erase all state except $(esk_A, t_A, W_A, X)$ | | Erase all state except $(esk_B, t_B, W_B, Y)$ |
| | $\xrightarrow{(B, A, W_A, X, t_A)}$ | |
| | $\xleftarrow{(A, B, W_B, Y, t_B)}$ | |
| **Session Key Generation** | | |
| Set $\mathsf{sid} = (A, B, W_A, X, t_A, W_B, Y, t_B)$ | | Set $\mathsf{sid} = (A, B, W_A, X, t_A, W_B, Y, t_B)$ |
| $aux = H_1(\mathsf{sid})$, $K_{A_1} = Y^x$ | | $aux = H_1(\mathsf{sid})$, $K_{B_1} = X^y$ |
| $K_{A_2} = \mathsf{ProjHash}(lpk_B, params, \mathcal{L}, W_A, w_A, aux)$ | | $K_{B_2} = \mathsf{Hash}(lsk_B, params, \mathcal{L}, W_A, aux)$ |
| $K_{A_3} = \mathsf{Hash}(lsk_A, params, \mathcal{L}, W_B, aux)$ | | $K_{B_3} = \mathsf{ProjHash}(lpk_A, params, \mathcal{L}, W_B, w_B, aux)$ |
| $s_A = \mathsf{Ext}_3(H_2(K_{A_1}) \oplus K_{A_2} \oplus K_{A_3}, t_A \oplus t_B)$ | | $s_B = \mathsf{Ext}_3(H_2(K_{B_1}) \oplus K_{B_2} \oplus K_{B_3}, t_A \oplus t_B)$ |
| $SK_A = \widetilde{\mathcal{F}}_{s_A}(\mathsf{sid})$ | | $SK_B = \widetilde{\mathcal{F}}_{s_B}(\mathsf{sid})$ |

- 2-smooth SPHF $\mathcal{SPHF}$ over $\mathcal{L} \subset \mathcal{X}$ onto the set $\mathcal{Y}$ with hashing key space $\mathcal{HK}$, projection key space $\mathcal{HP}$, witness space $\mathcal{W}$ and auxiliary input space $\mathcal{AUX}$, such that the subset membership problem between $\mathcal{L}$ and $\mathcal{X}$ is hard.
- Collision Resistant hash functions $H_1 : \{0,1\}^* \to \mathcal{AUX}$ and $H_2 : \mathbb{G} \to \mathcal{Y}$.
- Strong extractors $\mathsf{Ext}_1$, $\mathsf{Ext}_2$ and $\mathsf{Ext}_3$, where:
  - $\mathsf{Ext}_1 : \mathcal{HK} \times \{0,1\}^{t_1(\lambda)} \to \{0,1\}^{l_1(\lambda)}$ is an average-case $(|\mathcal{HK}| - \lambda_1, \epsilon_1)$-strong extractor, where $\lambda_1$ is the leakage bound on the long term secret key.
  - $\mathsf{Ext}_2 : \{0,1\}^{u(\lambda)} \times \{0,1\}^{t_2(\lambda)} \to \{0,1\}^{l_2(\lambda)}$ is an average-case $(\lambda - \lambda_2, \epsilon_2)$-strong extractor, where $\lambda_2$ is the leakage bound on the ephemeral secret key.
  - $\mathsf{Ext}_3 : \mathcal{Y} \times \{0,1\}^{t_3(\lambda)} \to \{0,1\}^{l_3(\lambda)}$ is an average-case $(|\mathcal{Y}| - \lambda_1, \epsilon_3)$-strong extractor.
- PRF families $\hat{\mathcal{F}}$ and $\bar{\mathcal{F}}$ and $\pi$-PRF family $\widetilde{\mathcal{F}}$ where:
  - $\hat{\mathcal{F}}^{\lambda, \Sigma_{\hat{\mathcal{F}}}, \mathcal{D}_{\hat{\mathcal{F}}}, \mathcal{R}_{\hat{\mathcal{F}}}} : \Sigma_{\hat{\mathcal{F}}} \times \mathcal{D}_{\hat{\mathcal{F}}} \to \mathcal{R}_{\hat{\mathcal{F}}}$, where $\mathcal{K}_{\hat{\mathcal{F}}} = \{0,1\}^{l_1(\lambda)}$, $\mathcal{D}_{\hat{\mathcal{F}}} = \{0,1\}^{u(\lambda)}$ and $\mathcal{R}_{\hat{\mathcal{F}}} = \mathcal{W} \times \mathbb{Z}_p$.
  - $\bar{\mathcal{F}}^{\lambda, \Sigma_{\bar{\mathcal{F}}}, \mathcal{D}_{\bar{\mathcal{F}}}, \mathcal{R}_{\bar{\mathcal{F}}}} : \Sigma_{\bar{\mathcal{F}}} \times \mathcal{D}_{\bar{\mathcal{F}}} \to \mathcal{R}_{\bar{\mathcal{F}}}$, where $\mathcal{K}_{\bar{\mathcal{F}}} = \{0,1\}^{l_2(\lambda)}$, $\mathcal{D}_{\bar{\mathcal{F}}} = \{0,1\}^{t_1(\lambda)}$ and $\mathcal{R}_{\bar{\mathcal{F}}} = \mathcal{W} \times \mathbb{Z}_p$.

– $\widetilde{\mathcal{F}}^{\lambda,\Sigma_{\widetilde{\mathcal{F}}},\mathcal{D}_{\widetilde{\mathcal{F}}},\mathcal{R}_{\widetilde{\mathcal{F}}}} : \Sigma_{\widetilde{\mathcal{F}}} \times \mathcal{D}_{\widetilde{\mathcal{F}}} \to \mathcal{R}_{\widetilde{\mathcal{F}}}$, where $\mathcal{K}_{\widetilde{\mathcal{F}}} = \{0,1\}^{l_3(\lambda)}$, $\mathcal{D}_{\widetilde{\mathcal{F}}} = \mathcal{PK}^2 \times \mathcal{L}^2 \times \mathbb{G}^2 \times \{0,1\}^{2t_3(\lambda)}$ and $\mathcal{R}_{\widetilde{\mathcal{F}}} = \{0,1\}^{l_4(\lambda)}$, where we assume $\mathcal{PK}$ is the space of certified public keys.

**Correctness Analysis.** The correctness of the AKE scheme follows from the correctness of Diffie Hellman key exchange and the correctness property of the smooth projective hash function. In particular $K_{A_1} = K_{B_1}$, as $K_{A_1} = Y^x = X^y = K_{B_1} = g^{xy}$. Due to the correctness property of SPHF, we have $K_{A_2} = \mathsf{ProjHash}(lpk_B, params, \mathcal{L}, W_A, w_A, aux) = \mathsf{Hash}(lsk_B, params, \mathcal{L}, W_A, aux) = K_{B_2}$, and $K_{A_3} = \mathsf{Hash}(lsk_A, params, \mathcal{L}, W_B, aux) = \mathsf{ProjHash}(lpk_A, params, \mathcal{L}, W_B, w_B, aux) = K_{B_3}$.

# 6 Existing Security Analysis [CMY+16] of CMYSG Protocol and Its Flaw

The CMYSG scheme is claimed to be secure in the CLR-eCK model [CMY+16, Theorem 1], based on Decisional Diffie-Hellman (DDH) assumption and the security of underlying cryptographic primitives. However, we disprove their claim by showing that the CMYSG scheme is incorrectly reduced to the DDH problem in the eCK model. Note that the security of an AKE protocol in the CLR-eCK model already implies its security in the eCK model. Hence our result also falsifies their claim in the CLR-eCK model. In the following sections we give a sketch of the security proof as given in the CMYSG scheme (Section 6.1) and demonstrate the flaw in the security reduction (Section 6.2). Finally we rigorously re-prove the security of the CMYSG scheme in the eCK model (Section 7).

## 6.1 Proof Sketch

In this section we give the proof sketch of the CMYSG protocol. This will help the reader to connect with our result in the next section. The security of the AKE protocol can be analyzed in the following two disjoint cases.

**Case I.** *The test session corresponding to oracle $\pi_i^s$ has a matching session to oracle $\pi_j^t$.* Based on the freshness of the eCK model, we know that the adversary cannot get both the long term secret key of the party involved in test session and the ephemeral secret key of the test session. Without loss of generality, let us assume the adversary gets the long term secret key $lsk_A^*$ of party $\mathsf{ID}_A$. Now, the witness, exponent pair $(w_A^*, x^*)$ of the ephemeral public keys is generated using a kind of twisted PRF trick [FSXY15], namely, $(w_A^*, x^*) = \hat{\mathcal{F}}_{\widehat{lsk}_A^*}(esk_A) + \bar{\mathcal{F}}_{\widehat{esk}_A^*}(r_{A_1})$. In this case, since the adversary does not know $esk_A^*$, the output of the PRF $\bar{\mathcal{F}}_{\widehat{esk}_A^*}(r_{A_1})$ is random; and hence the entire output $(w_A^*, x^*)$ appears random to the adversary. Similarly, when the adversary knows the ephemeral secret key $esk_A^*$, the output of the PRF $\hat{\mathcal{F}}_{\widehat{lsk}_A^*}(esk_A^*)$ is hidden from the view of the adversary, and hence the output $(w_A, x)$ appears random. So instead of computing $(w_A^*, x^*)$ as before, we choose $(w_A^*, x^*) \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$. A similar argument holds for the party $\mathsf{ID}_B$, and hence instead of computing $(w_B^*, y^*)$ using the twisted PRF trick, we choose $(w_B^*, y^*) \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$. Hence, regardless of the type of reveal queries, $(x^*, y^*)$ are uniform random elements in $\mathbb{Z}_p^2$ from the view of the adversary $\mathcal{A}$. Therefore the value $K_{A_1}^* = K_{B_1}^* = g^{x^* y^*}$ is computationally indistinguishable from a random element in $\mathbb{G}$ according to the DDH assumption and hence $H_2(K_{A_1}^*)$ is a uniform random string from the view of $\mathcal{A}$ who is given $X^* = g^{x^*}$, $Y^* = g^{y^*}$. Thus the seed $s_A^*$ and $s_B^*$ for the $\pi$-PRF function is uniformly distributed and unknown to the adversary and thus the derived session key $SK_A^*$ is computationally indistinguishable from a random string.

**Case II.** *The test session corresponding to oracle $\pi_i^s$ has no matching session.* Note that, in his case, the adversary cannot ask $\mathsf{Corrupt}(\mathsf{ID}_B)$ query to obtain the long term secret key $lsk_B$ of party $\mathsf{ID}_B$ by the eCK freshness condition (Definition 14). In the simulation we proceed via a sequence of games. Instead of computing $K_{A_2} = \mathsf{ProjHash}(lpk_B, params, \mathcal{L}, W_A, w_A, aux)$, we compute $K_{A_2} = \mathsf{Hash}(lsk_B, params, \mathcal{L}, W_A, aux)$, and then in the next game choose $W_A^* \in \mathcal{X} \setminus \mathcal{L}$ instead of deriving it from $\mathcal{L}$ by the algorithm $\mathsf{WordG}$. This new game is indistinguishable from the previous game by the correctness of the underlying SPHF and the hard subset membership problem. The adversary may also activate a non-matching session with $\mathsf{ID}_B$, by simply choosing $W \in \mathcal{X} \setminus \mathcal{L}$, send $W$ to $\mathsf{ID}_B$ and issue $\mathsf{RevealKey}$ query to learn the session key. According to the property of the underlying 2-smooth projective hash function, we have that $K_{A_2}^*$ is pairwise independent from any other such key (denoted by $\tilde{K}$ ) and all public information. Hence, $\widetilde{H}_\infty(K_{A_2}^* | \tilde{K}, params, \mathcal{L}, lpk_B, W_A^*, aux^*) = |\mathcal{Y}|$. By the property of the strong extractor the value $s_A^*$ is statistically close to uniform and thus the derived session key $SK_A^*$ is computationally indistinguishable from a truly random element from $\mathcal{A}'$s view due to the application of $\pi$-PRF.

Another difficulty arises for the simulation of non-test session involving $\mathsf{ID}_A$ or $\mathsf{ID}_B$. When the $\mathsf{Corrupt}(\mathsf{ID}_A)$ or $\mathsf{Corrupt}(\mathsf{ID}_B)$ queries are not issued by the adversary the session execution simulated should be identical to the session run by $\mathsf{ID}_A$ or $\mathsf{ID}_B$ from the view of $\mathcal{A}$. In this case, the simulator does not know the long term secret key of neither $\mathsf{ID}_A$ nor $\mathsf{ID}_B$. In this case, the simulator chooses $(r_1, r_2) \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$, and then compute $W_A = \mathsf{WordG}(param, \mathcal{L}, r_1)$, $X = g^{r_2}$. In this way the simulation of the session owned by $\mathsf{ID}_A$ can be made identical to the real session from the view of $\mathcal{A}$ due to pseudo-randomness of the PRF, namely the output $\hat{\mathcal{F}}_{\widehat{lsk}_A}(esk_A)$.

### 6.2 Incorrect security reduction to the DDH assumption in the eCK model

In this section, we show that the above security reduction is flawed due to incorrect reduction to the DDH problem based on the eCK model. The details of the problem is illustrated in **Algorithm 1**. Suppose that there exists a distinguisher $\mathcal{D}$, which tries solve the DDH problem using the eCK adversary $\mathcal{M}$, i.e., given a DDH challenge instance $(g, g^a, g^b, Z)$, the goal of $\mathcal{D}$ is to distinguish whether $Z = g^{ab}$ or a random group element. The distinguisher must properly set up the simulation environment to the AKE adversary $\mathcal{M}$ in order to take advantage of $\mathcal{M}$ in solving the DDH challenge. We use the superscript '*' to denote the test session/oracle. Assume that $\mathcal{M}$ selects $\pi_B^{t^*}$ as the test oracle with the intended communication partner $\mathsf{ID}_A$. Note that in the simulation, $\mathcal{D}$ needs to set the ephemeral public key of $\pi_B^{t^*}$ (namely the value $Y$ in **Algorithm 1**) to be one of the challenge values in $(g^a, g^b)$ and the ephemeral public key of an oracle $\pi_A^s$ of party $\mathsf{ID}_A$ (namely the value $X$ in **Algorithm 1**) to be another challenge value (which is the one received by test oracle). Meanwhile the adversary $\mathcal{M}$ corrupts party $\mathsf{ID}_B$ (note that this is allowed according to the eCK freshness definition without asking $\mathsf{EphemeralKeyReveal}(\pi_B^{t^*})$ and $\mathsf{Corrupt}(\mathsf{ID}_A)$ query). Then the adversary generates a valid protocol message of its own choice and sends such a message to $\pi_A^s$ (see Algorithm 1). Note that the adversary can replay any message from party $\mathsf{ID}_A$ to the oracle $\pi_B^{t^*}$. Recall that the ephemeral public key of $\pi_A^s$ is set to be DDH challenge value $g^a$ in order to do the reduction. In this case, $\mathcal{D}$ cannot generate the session key for $\pi_A^s$ without knowing both the exponent of received ephemeral public value chosen by adversary $\mathcal{M}$ and the exponent of the ephemeral public value of $\pi_A^s$. More precisely, $\mathcal{D}$ cannot compute the value $K_{\mathcal{M}_1}$ without knowing either of the exponents of $\widetilde{Y}$ or $X$. Note that the adversary $\mathcal{M}$ can compute the value $K_{\mathcal{M}_1}$ using the exponent $c$ chosen by him/her. The adversary can compute the values $K_{\mathcal{M}_2} = K_{A_2}$

and $K_{\mathcal{M}_3} = K_{A_3}$ using the private key $lsk_B$ of $\mathsf{ID}_B$ (recall that $\mathcal{M}$ issued $\mathsf{Corrupt}(\mathsf{ID}_B)$ query and has $lsk_B$) and the witness $w_M$ chosen by $\mathcal{M}$ respectively. Note that if the adversary can derive the values $K_{\mathcal{M}_1}$, $K_{\mathcal{M}_2}$ and $K_{\mathcal{M}_3}$ he/she can also compute the session key. In order to fix the problem above we model the hash function $H_2$ to be a random oracle and instead use the Gap Diffie-Hellman (GDH) assumption in place of DDH assumption to recover the security of the CMYSG protocol in the eCK model. The modified security proof is shown in the next section.

DDH Challenge Instance $(g, g^a, g^b, Z)$

| $\mathsf{ID}_A$ | $\mathcal{M}$(Adversary) | $\mathsf{ID}_B$ |
|---|---|---|
| **Long-Term Key Generation** | | |
| $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(params, \mathcal{L})$ | | $\mathsf{hk}' \xleftarrow{\$} \mathsf{HashKG}(params, \mathcal{L})$ |
| $\mathsf{hp} \xleftarrow{\$} \mathsf{ProjKG}(\mathsf{hk}, params, \mathcal{L})$ | | $\mathsf{hp}' \xleftarrow{\$} \mathsf{ProjKG}(\mathsf{hk}', params, \mathcal{L})$ |
| $r_{A_1} \xleftarrow{\$} \{0,1\}^{t_1(\lambda)}$, $r_{A_2} \xleftarrow{\$} \{0,1\}^{t_2(\lambda)}$ | | $r_{B_1} \xleftarrow{\$} \{0,1\}^{t_1(\lambda)}$, $r_{B_2} \xleftarrow{\$} \{0,1\}^{t_2(\lambda)}$ |
| $lsk_A = \mathsf{hk}$, $lpk_A = (\mathsf{hp}, r_{A_1}, r_{A_2})$ | | $lsk_B = \mathsf{hk}'$, $lpk_B = (\mathsf{hp}', r_{B_1}, r_{B_2})$ |
| **Protocol Simulation** | | |
| $\pi_A^s$ | | $\pi_B^{t^*}$ |
| $W_A = \mathsf{WordG}(param, \mathcal{L}, w_A)$ | | $W_B = \mathsf{WordG}(param, \mathcal{L}, w_B)$ |
| $\boxed{X \leftarrow g^a}$ | $\boxed{\widetilde{Y} \leftarrow g^c}$ | $\boxed{Y \leftarrow g^b}$ |
| | $W_M = \mathsf{WordG}(params, \mathcal{L}, w_m)$, where, $(w_m, c) \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ | |
| Erase all state except $(esk_A, W_A, X, t_A)$ | $\boxed{\text{Corrupt } \mathsf{ID}_B}$ | Erase all state except $(esk_B, W_B, Y, t_B)$ |
| $\xrightarrow{X, t_A, B, A, W_A}$ | | $\xrightarrow{X, t_A, B, A, W_A}$ |
| $\xleftarrow{\widetilde{Y}, t_B, A, B, W_M}$ | | $\xleftarrow{Y, t_B, A, B, W_B}$ |
| **Session Key Generation** | | |
| | $aux = H_1(\mathsf{sid})$, $\boxed{K_{\mathcal{M}_1} = X^c}$, $K_{\mathcal{M}_2} = \mathsf{Hash}(lsk_B, params, \mathcal{L}, W_A, aux)$, $K_{\mathcal{M}_3} = \mathsf{ProjHash}(lpk_A, params, \mathcal{L}, W_M, w_m, aux)$, $s_M = \mathsf{Ext}_3(H_2(K_{M_1}) \oplus K_{M_2} \oplus K_{M_3}, t_A \oplus t_B)$, $SK_{\mathcal{M}} = \widetilde{\mathcal{F}}_{s_{\mathcal{M}}}(\mathsf{sid})$ | |
| $aux = H_1(\mathsf{sid})$ | | $aux = H_1(\mathsf{sid})$ |
| $\boxed{\text{Cannot compute } K_{\mathcal{M}_1}}$ | | Set $\boxed{K_{\mathcal{B}_1} \leftarrow Z}$ |
| $K_{A_2} = \mathsf{ProjHash}(lpk_B, params, \mathcal{L}, W_A, w_A, aux)$ | | $K_{B_2} = \mathsf{Hash}(lsk_B, params, \mathcal{L}, W_A, aux)$ |
| $K_{A_3} = \mathsf{Hash}(lsk_A, params, \mathcal{L}, W_M, aux)$ | | $K_{B_3} = \mathsf{ProjHash}(lpk_A, params, \mathcal{L}, W_B, w_B, aux)$ |
| | | $s_B = \mathsf{Ext}_3(H_2(K_{B_1}) \oplus K_{B_2} \oplus K_{B_3}, t_A \oplus t_B)$ |
| | | $SK_B = \widetilde{\mathcal{F}}_{s_B}(\mathsf{sid})$ |

**Algorithm 1:** Proof Reduction problem of CMGSY to DDH assumption

## 7 Our Modified Security Analysis of CMYSG Protocol

In this section we show our modified security analysis of the CMYSG protocol. Our theorem follows:

**Theorem 1.** *The AKE protocol in Section 5 is eCK-secure if the underlying smooth projective hash function is 2-smooth, the GDH assumption holds in $\mathbb{G}$, $H_1$ is a collision-resistant hash function, $H_2$ is modeled as a random oracle and and $\hat{\mathcal{F}}$, $\bar{\mathcal{F}}$ are PRF families and $\tilde{\mathcal{F}}$ is a $\pi$-PRF family. Let $\mathcal{A}$ be a PPT adversary against the protocol $\Pi$ in Section 5. Then the advantage $\mathsf{Adv}_{\Pi}^{eCK}(\mathcal{A})$ of $\mathcal{A}$ against eCK-security of protocol $\Pi$ is:*

$$\mathsf{Adv}_{\Pi}^{eCK}(\mathcal{A}) \leq \max\{(\mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}) + \epsilon_{\mathsf{SM}} + \epsilon_3 + \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}})), \\ (\ell^2 S^2(2[\epsilon_1 + \epsilon_2 + \epsilon_3 + \mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}) + \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}})] + \mathsf{Adv}_{\mathcal{S}}^{\mathsf{GDH}})\}.$$

*where $\mathcal{B}$, $\tilde{\mathcal{B}}$, $\mathcal{S}$ are efficient algorithms constructed using the adversary $\mathcal{A}$ against the underlying PRF, $\pi$-PRF, and GDH problem respectively. $\epsilon_1$, $\epsilon_2$ and $\epsilon_3$ are the parameters of the three strong randomness extractors, and $\epsilon_{\mathsf{SM}}$ represents the hardness of the underlying subset membership problem of the 2-smooth projective hash function.*

The proof of this theorem will proceed via the game hopping technique [Sho04]: define a sequence of games and relate the adversary's advantage of distinguishing each game from the previous game to the advantage of breaking one of the underlying cryptographic primitive.

We have to prove that the adversary is unable to distinguish a random value from the session key space from the session key of any fresh oracle. The proof is split into two main cases: when the partner to the test session exists, and when it does not. We use the superscript '*' to high-light corresponding values processed in test oracle ${\pi_A^s}^*$. Let $\pi_A^s$ be an accepted oracle with intended partner $\mathsf{ID}_B$. Let $\pi_B^t$ be an oracle (if it exists) with intended partner $\mathsf{ID}_A$, such that $\pi_A^s$ has a matching session to $\pi_B^t$. Throughout the proof we use the notation $Succ_{Game_\delta}(\mathcal{A})$ to denote the event that the adversary $\mathcal{A}$ wins Game $\delta$ and the notation $\mathsf{Adv}_{Game_\delta}(\mathcal{A})$ to denote the advantage of the adversary $\mathcal{A}$ of winning Game $\delta$. In all the cases Game 0 denotes the original game. According to the freshness condition of the eCK model we consider the cases and sub-cases.

### 7.1 Case I. Partner to the test session exists

In this case we consider that the partner to the test session/oracle *exists*, i.e., ${\pi_A^s}^*$ has a matching session to $\pi_B^t$. We consider the following sub-cases under this case and we follow a game-based proof technique.

1. The adversary $\mathcal{A}$ issued $\mathsf{Corrupt}(\mathsf{ID}_A)$ and $\mathsf{Corrupt}(\mathsf{ID}_B)$ query.
2. $\mathcal{A}$ issued $\mathsf{EphemeralKeyReveal}(\pi_A^s)$ and $\mathsf{Corrupt}(\mathsf{ID}_B)$ query.
3. $\mathcal{A}$ issued $\mathsf{Corrupt}(\mathsf{ID}_A)$ and $\mathsf{EphemeralKeyReveal}(\pi_B^t)$ query.
4. $\mathcal{A}$ issued $\mathsf{EphemeralKeyReveal}(\pi_A^s)$ and $\mathsf{EphemeralKeyReveal}(\pi_B^t)$ query.

#### I.1 $\mathcal{A}$ issued $\mathsf{Corrupt}(\mathsf{ID}_A)$ and $\mathsf{Corrupt}(\mathsf{ID}_B)$ query

**Game 0.** This is the original security game with adversary $\mathcal{A}$. When the $\mathsf{Test}$ query is asked, the Game 0 challenger chooses a random bit $b \xleftarrow{\$} \{0,1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$, otherwise a random value chosen from the same session key space is given. So, we have:

$$\mathsf{Adv}_{Game_0}(\mathcal{A}) = \mathsf{Adv}^{eCK}_{\Pi}(\mathcal{A}). \tag{1}$$

**Game 1.** Same as Game 0 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $\mathsf{ID}_A$, $\mathsf{ID}_B \xleftarrow{\$} \{ID_1, ID_2, \cdots, ID_\ell\}$ are chosen and two random numbers $s^*, t^* \xleftarrow{\$} \{1, 2 \cdots, S\}$ are chosen, where $\ell$ is the number of protocol principals or parties and $S$ is the maximum number of sessions that can be executed by a party. The oracle $\pi^{s^*}_A$ is chosen as the target/test session and the oracle $\pi^{t^*}_B$ is chosen as the partner to the target session. If the test session is not the oracle $\pi^{s^*}_A$ or partner to the oracle is not $\pi^{t^*}_B$, the Game 1 challenger aborts the game. The probability of Game 1 to be halted due to incorrect choice of the test session is $(1 - \frac{1}{\ell(\ell-1)S^2})$. Unless the incorrect choice happens, Game 1 is identical to Game 0. Hence,

$$\mathsf{Adv}_{Game_1}(\mathcal{A}) \geq \frac{1}{\ell^2 S^2} \mathsf{Adv}_{Game_0}(\mathcal{A}). \tag{2}$$

**Game 2.** Same as Game 1 with the following exception: Instead of computing $\widehat{esk}^*_A = \mathsf{Ext}_2(esk^*_A, r_{A_2})$ and $\widehat{esk}^*_B = \mathsf{Ext}_2(esk^*_B, r_{B_2})$, the challenger chooses $\widehat{esk}^*_A \xleftarrow{\$} \{0,1\}^{l_2(\lambda)}$ and $\widehat{esk}^*_B \xleftarrow{\$} \{0,1\}^{l_2(\lambda)}$ respectively. Note that Game 1 and Game 2 are indistinguishable by the property of the strong average case randomness extractor. To be more precise, since $\mathsf{Ext}_2$ is an average-case $(k - \lambda_2, \epsilon_2)$-strong extractor, $(esk^*_A, r_{A_2}, \mathsf{Ext}_2(esk^*_A; r_{A_2})) \approx_{\epsilon_2} (esk^*_A, r_{A_2}, U_{l_2(\lambda)})$, where $U_{l_2(\lambda)}$ denotes the uniform distribution over $\{0,1\}^{l_2(\lambda)}$. Similar case holds for the other party $\mathsf{ID}_B$. So,

$$|\mathsf{Adv}_{Game_2}(\mathcal{A}) - \mathsf{Adv}_{Game_1}(\mathcal{A})| \leq 2\,\epsilon_2. \tag{3}$$

**Game 3.** Same as Game 2 with the following exception: Instead of computing $(w^*_A, x^*) = \hat{\mathcal{F}}_{\widehat{lsk}_A}(esk^*_A) + \bar{\mathcal{F}}^*_{\widehat{esk}_A}(r_{A_1})$, the challenge now chooses $(w'_A, x') \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $(w^*_A, x^*) = \hat{\mathcal{F}}_{\widehat{lsk}_A}(esk^*_A) + (w'_A, x')$. Similarly for party $\mathsf{ID}_B$ the challenger chooses $(w'_B, y') \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $(w^*_B, y^*) = \hat{\mathcal{F}}_{\widehat{lsk}_B}(esk^*_B) + (w'_B, y')$.

If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session key value $SK$ is computed using the real PRF with a hidden key, or using a random function. The adversary $\mathcal{A}$ is given a $SK$, such that a part of the key is computed using the PRF or randomly chosen from the session key space. Note that for simulating non-test session involving $\mathsf{ID}_A$ or $\mathsf{ID}_B$, the challenger knows the long term key $lsk_A$ and $lsk_B$ (since the adversary queried $\mathsf{Corrupt}$ query on both) and hence can choose any ephemeral secret key and compute the ephemeral public key correctly by using the long-term secret key and long-term public key. Hence the session execution simulated is identical to the session run by $\mathsf{ID}_A$ or $\mathsf{ID}_B$ from the view of $\mathcal{A}$. So we have,

$$|\mathsf{Adv}_{Game_3}(\mathcal{A}) - \mathsf{Adv}_{Game_2}(\mathcal{A})| \leq 2\ \mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}). \tag{4}$$

**Game 4.** Same as Game 3 with the following exception: Instead of computing $K_{A_1} = Y^{*^{x^*}} = X^{*^{y^*}} = K_{B_1}$, the challenger chooses a random key $K \xleftarrow{\$} \mathcal{Y}$ and adds an abort event. Namely, the challenger aborts if the adversary queried a $H_2$ random oracle query with input secret $Y^{*^{x^*}} = X^{*^{y^*}}$ which is a CDH solution for the target keys $X^*$ and $Y^*$. It is straightforward to see that if the abort

event happens with non-negligible advantage, then we could make use of the adversary to build a CDH solver $\mathcal{S}$.
$\mathcal{S}$ receives a CDH challenge instance $(g, g^a, g^b)$, and tries to compute $g^{ab}$ by running $\mathcal{A}$ as subroutine. $\mathcal{S}$ simulates the game for $\mathcal{A}$, and sets the target Diffie-Hellman keys (i.e., $X$ and $Y$) to be $g^a$ and $g^b$ respectively. For each unique $H_2$ random oracle query, $\mathcal{S}$ returns a uniform random value if the input to the oracle $X^{*^r}$ or $Y^{*^{r'}}$ is a correct CDH value for corresponding ephemeral public keys $X^*$ and $Y^*$ (this check could be done via its $\mathcal{DDH}$ oracle query). As for two $H_2$ oracle queries with the same input, $\mathcal{S}$ returns the identical value. As for those $\mathsf{Send}$, $\mathsf{EphemeralKeyReveal}$, $\mathsf{Corrupt}$ queries, $\mathcal{S}$ would answer them honestly based on the secrets chosen by herself following the protocol specification. With respect to some $\mathsf{RevealKey}$ query and $\mathsf{Test}$ query involving target Diffie-Hellman (DH) keys $X^*$ and $Y^*$, $\mathcal{S}$ just returns a random key which should be identical to the session key of its partner session. If the adversary asks a $H_2$ oracle query for the test session with valid key material $Y^{*^{x^*}} = X^{*^{y^*}}$, then $\mathcal{S}$ aborts the game in success. Due to the security of GDH assumption, we have that:

$$|\mathsf{Adv}_{Game_4}(\mathcal{A}) - \mathsf{Adv}_{Game_3}(\mathcal{A})| \leq \mathsf{Adv}^{\mathsf{GDH}}_{\mathcal{S}}. \tag{5}$$

**Game 5.** Same as Game 4 with the following exception: Instead of computing $s^*_A = \mathsf{Ext}_3(H_2(K^*_{A_1}) \oplus K^*_{A_2} \oplus K^*_{A_3}, t^*_A \oplus t^*_B)$ and $s^*_B = \mathsf{Ext}_3(H_2(K^*_{B_1}) \oplus K^*_{B_2} \oplus K^*_{B_3}, t^*_A \oplus t^*_B)$, the challenge chooses $s^*_A, s^*_B \xleftarrow{\$} \{0,1\}^{l_3(\lambda)}$. Since $\mathsf{Ext}_3$ is an average-case $(|\mathcal{Y}| - \lambda_1, \epsilon_3)$-strong extractor, $(H_2(K^*_{A_1}) \oplus K^*_{A_2} \oplus K^*_{A_3}, t^*_A \oplus t^*_B, s^*_A)) \approx_{\epsilon_3} (H_2(K^*_{A_1}) \oplus K^*_{A_2} \oplus K^*_{A_3}, t^*_A \oplus t^*_B, U_{l_3(\lambda)})$, where $U_{l_3(\lambda)}$ denotes the uniform distribution over $\{0,1\}^{l_3(\lambda)}$. Similar case holds for party $\mathsf{ID}_B$. So,

$$|\mathsf{Adv}_{Game_5}(\mathcal{A}) - \mathsf{Adv}_{Game_4}(\mathcal{A})| \leq 2\,\epsilon_3. \tag{6}$$

**Game 6.** Same as Game 5 with the following exception: Instead of computing $SK_A = \widetilde{\mathcal{F}}_{s^*_A}(\mathsf{sid}^*)$, the challenger now chooses $SK_A \xleftarrow{\$} \{0,1\}^{l_4(\lambda)}$. If $\mathcal{A}$ can distinguish the difference between Game 5 and Game 6, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\tilde{\mathcal{B}}$, which is used to distinguish whether the session key value $SK$ is computed using the real $\pi$-PRF with a hidden key, or using a random function. So,

$$|\mathsf{Adv}_{Game_6}(\mathcal{A}) - \mathsf{Adv}_{Game_5}(\mathcal{A})| \leq 2\ \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}}). \tag{7}$$

**Semantic Security of the session key in Game 6.** Since the session key $SK_A = SK_B$ of $\pi^s_A$ and its matching session $\pi^t_B$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in Game 6. Hence,

$$\mathsf{Adv}_{Game_6}(\mathcal{A}) = 0. \tag{8}$$

Using equations (1) -(8), we get,

$$\mathsf{Adv}^{eCK}_{\Pi}(\mathcal{A}) \leq \ell^2 S^2 (2(\epsilon_2 + \mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}) + \epsilon_3 + \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}})) + \mathsf{Adv}^{\mathsf{GDH}}_{\mathcal{S}}). \tag{9}$$

### I.2 $\mathcal{A}$ issued $\mathsf{EphemeralKeyReveal}(\pi^s_A)$ and $\mathsf{Corrupt}(\mathsf{ID}_B)$ query

**Game 0.** Same as Game 0 of **Case I.1**.

**Game 1.** Same as Game 1 of **Case I.1**.

**Game 2.** Same as Game 1 with the following exception: Instead of computing $\widehat{lsk}_A = \mathsf{Ext}_1(lsk_A, r_{A_1})$ and $\widehat{esk}_B^* = \mathsf{Ext}_2(esk_B^*, r_{B_2})$, the challenger chooses $\widehat{lsk}_A \xleftarrow{\$} \{0,1\}^{l_1(\lambda)}$ and $\widehat{esk}_B^* \xleftarrow{\$} \{0,1\}^{l_2(\lambda)}$ respectively. Note that Game 1 and Game 2 are indistinguishable by the property of the strong average case randomness extractors. To be more precise, since $\mathsf{Ext}_1$ is an average-case $(|\mathcal{HK}|-\lambda_1, \epsilon_1)$-strong extractor, $(lsk_A, r_{A_1}, \mathsf{Ext}_1(lsk_A; r_{A_1})) \approx_{\epsilon_1} (lsk_A, r_{A_1}, U_{l_1(\lambda)})$, where $U_{l_1(\lambda)}$ denotes the uniform distribution over $\{0,1\}^{l_1(\lambda)}$. Similarly, since $\mathsf{Ext}_2$ is an average-case $(k-\lambda_2, \epsilon_2)$-strong extractor, $(esk_B^*, r_{B_2}, \mathsf{Ext}_2(esk_B^*; r_{B_2})) \approx_{\epsilon_2}$
$(esk_B^*, r_{B_2}, U_{l_2(\lambda)})$, where $U_{l_2(\lambda)}$ denotes the uniform distribution over $\{0,1\}^{l_2(\lambda)}$. So,

$$|\mathsf{Adv}_{Game_2}(\mathcal{A}) - \mathsf{Adv}_{Game_1}(\mathcal{A})| \leq \epsilon_1 + \epsilon_2. \tag{10}$$

**Game 3.** Same as Game 2 with the following exception: Instead of computing $(w_A^*, x^*) = \hat{\mathcal{F}}_{\widehat{lsk}_A}(esk_A^*) + \bar{\mathcal{F}}_{\widehat{esk}_A^*}(r_{A_1})$, the challenge now chooses $(w'_A, x') \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $(w_A^*, x^*) = (w'_A, x') + \hat{\mathcal{F}}_{\widehat{esk}_A^*}(r_{A_1})$. For party $\mathsf{ID}_B$ the challenger chooses $(w'_B, y') \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $(w_B^*, y^*) = \hat{\mathcal{F}}_{\widehat{lsk}_B}(esk_B^*) + (w'_B, y')$.
If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session key value $SK$ is computed using the real PRF with a hidden key, or using a random function. The adversary $\mathcal{A}$ is given a $SK$, such that a part of the key is computed using the PRF or randomly chosen from the session key space. Note that for simulating non-test session involving $\mathsf{ID}_A$ or $\mathsf{ID}_B$, the challenger knows the long term key $lsk_B$ of party $\mathsf{ID}_B$ (since the adversary queried $\mathsf{Corrupt}(\mathsf{ID}_B)$ query) and hence can choose any ephemeral secret key $esk_B$ and $t_B$ and compute the ephemeral public key correctly by using the long-term secret key and long-term public key. However for party $\mathsf{ID}_A$, the adversary does not know its long term secret key $lsk_A$ and hence cannot simulate the session as mentioned above. In this case, the challenger chooses $(r_1, r_2) \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and compute $W_A = \mathsf{WordG}(params, \mathcal{L}, r_1)$ and $X = g^{r_2}$. The session simulated in this way is identical to the real session from $\mathcal{A}$'s view due to the pseudo-randomness of the PRF. Hence the session execution simulated is identical to the session run by $\mathsf{ID}_A$ or $\mathsf{ID}_B$ from the view of $\mathcal{A}$. So we have,

$$|\mathsf{Adv}_{Game_3}(\mathcal{A}) - \mathsf{Adv}_{Game_2}(\mathcal{A})| \leq 2\ \mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}). \tag{11}$$

**Game 4.** Same as Game 4 of **Case I.1**.

**Game 5.** Same as Game 5 of **Case I.1**.

**Game 6.** Same as Game 6 of **Case I.1**.

Putting these together we get:

$$\mathsf{Adv}_{\Pi}^{eCK}(\mathcal{A}) \leq \ell^2 S^2 (2(\mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}) + \epsilon_3 + \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}})) + \epsilon_1 + \epsilon_2 + \mathsf{Adv}_{\mathcal{S}}^{\mathsf{GDH}}). \tag{12}$$

### I.3 $\mathcal{A}$ issued $\mathsf{Corrupt}(\mathsf{ID}_A)$ and $\mathsf{EphemeralKeyReveal}(\pi_B^t)$ query

**Game 0.** Same as **Case I.2**.

**Game 1.** Same as **Case I.2**.

**Game 2.** Same as **Case I.2**. except the following: Instead of computing $\widehat{esk}_A^* = \mathsf{Ext}_2(esk_A^*, r_{A_2})$ and $\widehat{lsk}_B = \mathsf{Ext}_1(lsk_B, r_{B_1})$, the challenger chooses $\widehat{esk}_A^* \xleftarrow{\$} \{0,1\}^{l_2(\lambda)}$ and $\widehat{lsk}_B \xleftarrow{\$} \{0,1\}^{l_1(\lambda)}$ respectively. Note that Game 1 and Game 2 are indistinguishable by the property of the strong average case randomness extractors. By a similar argument as in Game 2 in **Case I.2** we have:

$$|\mathsf{Adv}_{Game_2}(\mathcal{A}) - \mathsf{Adv}_{Game_1}(\mathcal{A})| \leq \epsilon_1 + \epsilon_2. \tag{13}$$

**Game 3.** Same as Game 2 with the following exception: Instead of computing $(w_A^*, x^*) = \hat{\mathcal{F}}_{\widehat{lsk}_A}(esk_A) + \bar{\mathcal{F}}_{\widehat{esk}_A^*}(r_{A_1})$, the challenge now chooses $(w'_A, x') \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $(w_A^*, x^*) = \hat{\mathcal{F}}_{\widehat{lsk}_A}(esk_A^*) + (w'_A, x')$. For party $\mathsf{ID}_B$ the challenger chooses $(w'_B, y') \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $(w_B^*, y^*) = (w'_B, y') + \hat{\mathcal{F}}_{\widehat{esk}_B^*}(r_{B_1})$.

If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session key value $SK$ is computed using the real PRF with a hidden key, or using a random function. The simulation for non-test session involving $\mathsf{ID}_A$ or $\mathsf{ID}_B$ is done similarly to game 3 in **Case I.2** except that the challenger now chooses $(r'_1, r'_2) \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $W_B = \mathsf{WordG}(params, \mathcal{L}, r'_1)$ and $Y = g^{r'_2}$. For party $\mathsf{ID}_A$, the challenger knows the long term secret key $lsk_A$ (since the adversary queried $\mathsf{Corrupt}(\mathsf{ID}_A)$ query) and hence can choose any ephemeral secret key $esk_A$ and $t_A$ and compute the ephemeral public key correctly by using the long-term secret key and long-term public key. Hence the session execution simulated is identical to the session run by $\mathsf{ID}_A$ or $\mathsf{ID}_B$ from the view of $\mathcal{A}$. So we have,

$$|\mathsf{Adv}_{Game_3}(\mathcal{A}) - \mathsf{Adv}_{Game_2}(\mathcal{A})| \leq 2\ \mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}). \tag{14}$$

**Game 4.** This is similar to the **Case I.2**.

**Game 5.** This is similar to the **Case I.2**.

**Game 6.** This is similar to the **Case I.2**.

Putting these together we get:

$$\mathsf{Adv}_{\Pi}^{eCK}(\mathcal{A}) \leq \ell^2 S^2 (2(\mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}) + \epsilon_3 + \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}})) + \epsilon_1 + \epsilon_2 + \mathsf{Adv}_{\mathcal{S}}^{\mathsf{GDH}}). \tag{15}$$

### I.4 $\mathcal{A}$ issued $\mathsf{EphemeralKeyReveal}(\pi_A^s)$ and $\mathsf{EphemeralKeyReveal}(\pi_B^t)$ query

**Game 0.** Same as **Case I.2**.

**Game 1.** Same as **Case I.2**.

**Game 2.** Same as Game 1 with the following exception: Instead of computing $\widehat{lsk}_A = \mathsf{Ext}_1(lsk_A, r_{A_1})$ and $\widehat{lsk}_B = \mathsf{Ext}_1(lsk_B, r_{B_1})$, the challenger chooses $\widehat{lsk}_A \xleftarrow{\$} \{0,1\}^{l_1(\lambda)}$ and $\widehat{lsk}_B \xleftarrow{\$} \{0,1\}^{l_1(\lambda)}$ respectively. Note that Game 2 and Game 1 are indistinguishable by the property of the strong average case randomness extractor. To be more precise, since $\mathsf{Ext}_1$ is an average-case $(|\mathcal{HK}| - \lambda_1, \epsilon_1)$-strong extractor, $(lsk_A, r_{A_1}, \mathsf{Ext}_1(lsk_A; r_{A_1})) \approx_{\epsilon_1} (\widehat{lsk}_A, r_{A_1}, U_{l_1(\lambda)})$ and $(lsk_B, r_{B_1}, \mathsf{Ext}_1(lsk_B; r_{B_1})) \approx_{\epsilon_1} (\widehat{lsk}_B, r_{B_1}, U_{l_1(\lambda)})$, where $U_{l_1(\lambda)}$ denotes the uniform distribution over $\{0,1\}^{l_1(\lambda)}$. So,

$$|\mathsf{Adv}_{Game_2}(\mathcal{A}) - \mathsf{Adv}_{Game_1}(\mathcal{A})| \leq 2\epsilon_1. \tag{16}$$

**Game 3.** Same as Game 2 with the following exception: Instead of computing $(w_A^*, x^*) = \hat{\mathcal{F}}_{\widehat{lsk}_A}(esk_A^*) + \bar{\mathcal{F}}_{\widehat{esk}_A^*}(r_{A_1})$, the challenge now chooses $(w'_A, x') \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $(w_A^*, x^*) = (w'_A, x') + \hat{\mathcal{F}}_{\widehat{esk}_A^*}(r_{A_1})$. For party $\mathsf{ID}_B$ the challenger chooses $(w'_B, y') \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $(w_B, y) = (w'_B, y') + \hat{\mathcal{F}}_{\widehat{esk}_B^*}(r_{B_1})$.

If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session key value $SK$ is computed using the real PRF with a hidden key, or using a random function. Now, in this case for the simulation of non-test session involving $\mathsf{ID}_A$ or $\mathsf{ID}_B$ the challenger does not know either the long term key $lsk_A$ of party $\mathsf{ID}_A$ or the long term key $lsk_B$ of party $\mathsf{ID}_B$. In this case, the challenger chooses $(r_1, r_2) \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $W_A = \mathsf{WordG}(params, \mathcal{L}, r_1)$ and $X = g^{r_2}$. For party $\mathsf{ID}_B$, the challenger again chooses $(r'_1, r'_2) \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $W_B = \mathsf{WordG}(params, \mathcal{L}, r'_1)$ and $Y = g^{r'_2}$. The session simulated in this way is identical to the real session from $\mathcal{A}$'s view due to the pseudo-randomness of the PRF. Hence the session execution simulated is identical to the session run by $\mathsf{ID}_A$ or $\mathsf{ID}_B$ from the view of $\mathcal{A}$. So we have,

$$|\mathsf{Adv}_{Game_3}(\mathcal{A}) - \mathsf{Adv}_{Game_2}(\mathcal{A})| \leq 2\ \mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}). \tag{17}$$

**Game 4.** This is similar to the **Case I.2**.

**Game 5.** This is similar to the **Case I.2**.

**Game 6.** This is similar to the **Case I.2**.

Putting these together we get:

$$\mathsf{Adv}_{\Pi}^{eCK}(\mathcal{A}) \leq \ell^2 S^2 (2(\epsilon_1 + \mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}) + \epsilon_3 + \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}})) + \mathsf{Adv}_{\mathcal{S}}^{\mathsf{GDH}}). \tag{18}$$

Combining the four cases (**Case I.1**) - (**Case I.4**), from equations (9), (12), (15) and (18), we get:

$$\mathsf{Adv}_{\Pi}^{eCK}(\mathcal{A}) \leq \ell^2 S^2 (2(\epsilon_1 + \epsilon_2 + \epsilon_3 + \mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}) + \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}})) + \mathsf{Adv}_{\mathcal{S}}^{\mathsf{GDH}}). \tag{19}$$

### 7.2 Case II. Partner to the test session does not exist

Here we consider the case when there is *no partner* to the test session/oracle ,i.e., $\pi_A^{s^*}$ does not have a matching session. When the partner session does not exist, the owner of the test session shares the session key with the active adversary. In this situation where there is no matching session corresponding to oracle $\pi_A^{s^*}$, the adversary is not allowed to corrupt the intended partner principal to the test session by the freshness condition (Definition 14). We have the following sub-cases:

1. The adversary $\mathcal{A}$ issued $\mathsf{EphemeralKeyReveal}(\pi_A^{s^*})$ query.
2. $\mathcal{A}$ issued $\mathsf{Corrupt}(\mathsf{ID}_A)$ query.

#### II.1 $\mathcal{A}$ issued $\mathsf{EphemeralKeyReveal}(\pi_A^{s^*})$ query

**Game 0.** This is the original security game with adversary $\mathcal{A}$. When the $\mathsf{Test}$ query is asked, the Game 0 challenger chooses a random bit $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$, otherwise a random value chosen from the same session key space is given. Hence,

$$\mathsf{Adv}_{Game_0}(\mathcal{A}) = \mathsf{Adv}^{eCK}_{\Pi}(\mathcal{A}). \tag{20}$$

**Game 1.** Same as Game 0 with the following exception: before $\mathcal{A}$ begins, the Game 1 challenger guesses the identity $\mathsf{ID}_B$ of the partner principal to the test session and if the guess in incorrect it aborts the game. The probability of Game 1 to be aborted due to incorrect guess of the partner principal to the test session is $(1 - \frac{1}{\ell})$. Unless the incorrect guess happens, Game 1 is identical to Game 0. Hence,

$$\mathsf{Adv}_{Game_1}(\mathcal{A}) = \frac{1}{\ell}\ \mathsf{Adv}_{Game_0}(\mathcal{A}). \tag{21}$$

**Game 2.** Same as Game 1 with the following exception: Instead of computing $\widehat{lsk}_A = \mathsf{Ext}_1(lsk_A, r_{A_1})$, the challenger chooses $\widehat{lsk}_A \xleftarrow{\$} \{0,1\}^{l_1(\lambda)}$. Note that Game 1 and Game 2 are indistinguishable by the property of the strong average case randomness extractor. To be more precise, since $\mathsf{Ext}_1$ is an average-case $(|\mathcal{HK}| - \lambda_1, \epsilon_1)$-strong extractor, $(lsk_A, r_{A_1}, \mathsf{Ext}_1(lsk_A; r_{A_1})) \approx_{\epsilon_1} (lsk_A, r_{A_1}, U_{l_1(\lambda)})$, where $U_{l_1(\lambda)}$ denotes the uniform distribution over $\{0,1\}^{l_1(\lambda)}$. So,

$$|\mathsf{Adv}_{Game_2}(\mathcal{A}) - \mathsf{Adv}_{Game_1}(\mathcal{A})| \leq \epsilon_1. \tag{22}$$

**Game 3.** Same as Game 2 with the following exception: Instead of computing $(w^*_A, x^*) = \hat{\mathcal{F}}_{\widehat{lsk}_A}(esk^*_A) + \bar{\mathcal{F}}_{\widehat{esk}_A}(r_{A_1})$, the challenge now chooses $(w'_A, x') \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $(w^*_A, x^*) = (w'_A, x') + \hat{\mathcal{F}}_{\widehat{esk}^*_A}(r_{A_1})$.

If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session key value $SK$ is computed using the real PRF with a hidden key, or using a random function. So we have,

$$|\mathsf{Adv}_{Game_3}(\mathcal{A}) - \mathsf{Adv}_{Game_2}(\mathcal{A})| \leq\ \mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}). \tag{23}$$

**Game 4.** Same as Game 3 with the following exception: Replace $K^*_{A_2} = \mathsf{ProjHash}(lpk_B, params, \mathcal{L}, W^*_A, w^*_A, aux^*)$ by $K^*_{A_2} = \mathsf{Hash}(lsk_B, params, \mathcal{L}, W^*_A, aux^*)$. Note that Game 1 and Game 2 are identical by the correctness of the underlying 2-smooth projective hash function $\mathcal{SPHF}$. So,

$$\mathsf{Adv}_{Game_4}(\mathcal{A}) = \mathsf{Adv}_{Game_3}(\mathcal{A}). \tag{24}$$

**Game 5.** Same as Game 4 with the following exception: Choose $W^*_A \in \mathcal{X} \setminus \mathcal{L}$ instead of deriving it through the algorithm $\mathsf{WordG}$. Game 4 and Game 5 are computationally indistinguishable due to the $(t_{\mathsf{SM}}, \epsilon_{\mathsf{SM}})$-hard subset membership problem of the underlying 2-smooth projective hash function $\mathcal{SPHF}$. So,

$$|\mathsf{Adv}_{Game_5}(\mathcal{A}) - \mathsf{Adv}_{Game_4}(\mathcal{A})| \leq \epsilon_{\mathsf{SM}}. \tag{25}$$

**Game 6.** Same as Game 5 with the following exception: Instead of computing $K^*_{A_2} = \mathsf{Hash}(lsk_B, params, \mathcal{L}, W^*_A, aux^*)$, the challenger now chooses $K^*_{A_2} \xleftarrow{\$} \mathcal{Y}$. Game 5 and Game 6 are perfectly indistinguishable to the view of the adversary $\mathcal{A}$ given the public parameters ($params$, $\mathcal{L}$, $aux^*$), i.e., $\tilde{H}_\infty(K^*_{A_2} | \ params, \mathcal{L}, aux^*) = |\mathcal{Y}|$. This follows from the *smoothness*

property of the underlying 2-smooth projective hash function. Note that adversary $\mathcal{A}$ may activate a session $\pi_j^t$ with $\mathsf{ID}_B$ which is not matching to the test session corresponding to the oracle $\pi_A^{s^*}$. More precisely, $\mathcal{A}$ can choose $W \in \mathcal{X} \setminus \mathcal{L}$, send $W$ to $\mathsf{ID}_B$ and issues $\mathsf{RevealKey}(\pi_j^t)$ query to learn the shared key. According to the property of 2-smooth of the underlying smooth projective hash function, we have that $K_{A_2}^*$ is pairwise independent from any other such key (denoted by $\widetilde{K}$) and all public information ($params$, $\mathcal{L}$, $W_A^*$, $aux^*$), and hence $\widetilde{H}_\infty(K_{A_2}^* | \widetilde{K}, params, \mathcal{L}, aux^*) = |\mathcal{Y}|$. So we have,

$$\mathsf{Adv}_{Game_6}(\mathcal{A}) = \mathsf{Adv}_{Game_5}(\mathcal{A}). \tag{26}$$

**Game 7.** Same as Game 6 with the following exception: Instead of computing $s_A^* = \mathsf{Ext}_3(H_2(K_{A_1}^*) \oplus K_{A_2}^* \oplus K_{A_3}^*, t_A^* \oplus t_B^*)$, the challenge chooses $s_A^* \xleftarrow{\$} \{0,1\}^{l_3(\lambda)}$. Since $\mathsf{Ext}_3$ is an average-case $(|\mathcal{Y}| - \lambda_1, \epsilon_3)$-strong extractor, $(H_2(K_{A_1}^*) \oplus K_{A_2}^* \oplus K_{A_3}^*, t_A^* \oplus t_B^*, s_A^*)) \approx_{\epsilon_3} (H_2(K_{A_1}^*) \oplus K_{A_2}^* \oplus K_{A_3}^*, t_A^* \oplus t_B^*, U_{l_3(\lambda)})$, where $U_{l_3(\lambda)}$ denotes the uniform distribution over $\{0,1\}^{l_3(\lambda)}$. So,

$$|\mathsf{Adv}_{Game_7}(\mathcal{A}) - \mathsf{Adv}_{Game_6}(\mathcal{A})| \leq \epsilon_3. \tag{27}$$

**Game 8.** Same as Game 7 with the following exception: Instead of computing $SK_A = \widetilde{\mathcal{F}}_{s_A^*}(\mathsf{sid}^*)$, the challenger now chooses $SK_A \xleftarrow{\$} \{0,1\}^{l_4(\lambda)}$. If $\mathcal{A}$ can distinguish the difference between Game 7 and Game 8, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\tilde{\mathcal{B}}$, which is used to distinguish whether the session key value $SK$ is computed using the real $\pi$-PRF with a hidden key, or using a random function. So,

$$|\mathsf{Adv}_{Game_8}(\mathcal{A}) - \mathsf{Adv}_{Game_7}(\mathcal{A})| \leq \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}}). \tag{28}$$

**Semantic Security of the session key in Game 8.** Since the session key $SK_A$ of $\pi_A^{s^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in Game 6. Hence,

$$\mathsf{Adv}_{Game_8}(\mathcal{A}) = 0. \tag{29}$$

Using equations (20) - (29), we get,

$$\mathsf{Adv}_{\Pi}^{eCK}(\mathcal{A}) \leq \ell \, (\mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}) + \epsilon_{\mathsf{SM}} + \epsilon_3 + \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}})). \tag{30}$$

**II.2** $\mathcal{A}$ issued $\mathsf{Corrupt}(\mathsf{ID}_A)$ query

**Game 0.** Same as Game 0 as in **Case II.1**.

**Game 1.** Same as Game 1 as in **Case II.1**.

**Game 2.** Same as Game 1 as in **Case II.1** except the following: Instead of computing $\widehat{esk}_A^* = \mathsf{Ext}_2(esk_A^*, r_{A_2})$, the challenger chooses $\widehat{esk}_A^* \xleftarrow{\$} \{0,1\}^{l_2(\lambda)}$. Game 1 and Game 2 are indistinguishable by the property of the strong average case randomness extractor $\mathsf{Ext}_2$. By a similar argument as in **Case I.1**, we get:

$$|\mathsf{Adv}_{Game_2}(\mathcal{A}) - \mathsf{Adv}_{Game_1}(\mathcal{A})| \leq \epsilon_2. \tag{31}$$

**Game 3.** Same as Game 2 as in **Case II.1** except the following: Instead of computing $(w_A^*, x^*) = \hat{\mathcal{F}}_{\widehat{lsk}_A}(esk_A^*) + \bar{\mathcal{F}}_{\widehat{esk}_A^*}(r_{A_1})$, the challenge now chooses $(w'_A, x') \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and computes $(w_A^*, x^*) = \hat{\mathcal{F}}_{\widehat{lsk}_A}(esk_A^*) + (w'_A, x')$.
If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session key value $SK$ is computed using the real PRF with a hidden key, or using a random function. So we have,

$$|\mathsf{Adv}_{Game_3}(\mathcal{A}) - \mathsf{Adv}_{Game_2}(\mathcal{A})| \leq \mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}). \tag{32}$$

**Game 4.** Same as Game 4 as in **Case II.1**.

**Game 5.** Same as Game 5 as in **Case II.1**.

**Game 6.** Same as Game 6 as in **Case II.1**.

**Game 4.** Same as Game 4 as in **Case II.1**.

So, we have:

$$\mathsf{Adv}_{\Pi}^{eCK}(\mathcal{A}) \leq \ell \, (\mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}) + \epsilon_{\mathsf{SM}} + \epsilon_3 + \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}})). \tag{33}$$

Finally, combining **Case I** and **Case II**, we get the following:

$$\begin{aligned}\mathsf{Adv}_{\Pi}^{eCK}(\mathcal{A}) \leq \max\{&(\mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}) + \epsilon_{\mathsf{SM}} + \epsilon_3 + \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}})), \\ &(\ell^2 S^2 (2[\epsilon_1 + \epsilon_2 + \epsilon_3 + \mathsf{Adv}_{\mathsf{PRF}}(\mathcal{B}) + \mathsf{Adv}_{\pi\text{-}\mathsf{PRF}}(\tilde{\mathcal{B}})] + \mathsf{Adv}_{\mathcal{S}}^{\mathsf{GDH}})\}.\end{aligned}$$

This completes the proof of **Theorem 1**. □

## 8 Conclusion

In CT-RSA 2016, Chen et al. proposed a new authenticated key exchange protocol and showed it to be secure under challenge-dependent leakage-resilient eCK (CLR-eCK) model. The security of their protocol is reduced to the existence of hash-proof systems, pseudo-random functions (PRF) and $\pi$-PRFs, collision-resistant hash functions and the hardness of the Decisional Diffie-Hellman (DDH) problem. In the paper, we refute their security claim by showing that the reduction to DDH is incorrect. Further, we restore the security proof under the Gap Diffie-Hellman (GDH) hardness assumption in the random oracle model. Restoring the security under the DDH assumption with possibly a different reduction seems unlikely and our restoration under the GDH assumption is also quite non-trivial. Moreover, due to our work, the existence of CLR-eCK secure AKE protocol in the standard model becomes open once again.

## References

[ABS14] Janaka Alawatugoda, Colin Boyd, and Douglas Stebila. Continuous after-the-fact leakage-resilient key exchange. In *ACISP*, pages 258–273. Springer, 2014.

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *Advances in Cryptology-CRYPTO 2009*, pages 36–54. Springer, 2009.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptography Conference*, pages 474–495. Springer, 2009.

[ASB14] Janaka Alawatugoda, Douglas Stebila, and Colin Boyd. Modelling after-the-fact leakage for key exchange. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 207–216. ACM, 2014.

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – CRYPTO93*, pages 232–249. Springer, 1994.

[CBH05] Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. Errors in computational complexity proofs for protocols. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 624–643. Springer, 2005.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.

[CMY+16] Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo, and Fuchun Guo. Strongly leakage-resilient authenticated key exchange. In *Cryptographers Track at the RSA Conference*, pages 19–36. Springer, 2016.

[Cre11] Cas Cremers. Examining indistinguishability-based security models for key exchange protocols: the case of ck, ck-hmqv, and eck. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 80–91. ACM, 2011.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 45–64. Springer, 2002.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008.

[FSXY15] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. *Designs, Codes and Cryptography*, 76(3):469–504, 2015.

[HSH+09] J Alex Halderman, Seth D Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A Calandrino, Ariel J Feldman, Jacob Appelbaum, and Edward W Felten. Lest we remember: cold-boot attacks on encryption keys. *Communications of the ACM*, 52(5):91–98, 2009.

[JKSS12] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of tls-dhe in the standard model. In *Advances in Cryptology–CRYPTO 2012*, pages 273–293. Springer, 2012.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in CryptologyCRYPTO99*, pages 388–397. Springer, 1999.

[Koc96] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in CryptologyCRYPTO96*, pages 104–113. Springer, 1996.

[LLM07] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In *Provable Security*, pages 1–16. Springer, 2007.

[MO11] Daisuke Moriyama and Tatsuaki Okamoto. Leakage resilient eck-secure key exchange protocol without random oracles. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 441–447. ACM, 2011.

[NLP+11] Junghyun Nam, Kwangwoo Lee, Juryon Paik, Woojin Paik, and Dongho Won. Security improvement on a group key exchange protocol for mobile networks. In *International Conference on Computational Science and Its Applications*, pages 123–132. Springer, 2011.

[PYG08] Raphael C-W Phan, Wei-Chuen Yau, and Bok-Min Goi. Cryptanalysis of simple three-party key exchange protocol (s-3pake). *Information sciences*, 178(13):2849–2856, 2008.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[Too15] Mohsen Toorani. On continuous after-the-fact leakage-resilient key exchange. In *Proceedings of the Second Workshop on Cryptography and Security in Computing Systems*, page 31. ACM, 2015.

[YL16] Zheng Yang and Shuangqing Li. On security analysis of an after-the-fact leakage resilient key exchange protocol. *Information Processing Letters*, 116(1):33–40, 2016.

[YPGH11] Wei-Chuen Yau, Raphael C-W Phan, Bok-Min Goi, and Swee-Huay Heng. Cryptanalysis of a provably secure cross-realm client-to-client password-authenticated key agreement protocol of cans09. In *International Conference on Cryptology and Network Security*, pages 172–184. Springer, 2011.

[YS12] Zheng Yang and Jörg Schwenk. Strongly authenticated key exchange protocol from bilinear groups without random oracles. In *International Conference on Provable Security*, pages 264–275. Springer, 2012.