Strongly Leakage-Resilient Authenticated Key Exchange*

Rongmao Chen[1,2], Yi Mu[1], Guomin Yang[1], Willy Susilo[1], and Fuchun Guo[1]

[1]Centre for Computer and Information Security Research
School of Computing and Information Technology
University of Wollongong, Australia
{rc517,ymu,gyang,wsusilo,fuchun}@uow.edu.au
[2]College of Computer
National University of Defense Technology, China

Abstract. Authenticated Key Exchange (AKE) protocols have been widely deployed in many real-world applications for securing communication channels. In this paper, we make the following contributions. First, we revisit the security modelling of leakage-resilient AKE protocols, and show that the existing models either impose some unnatural restrictions or do not sufficiently capture leakage attacks in reality. We then introduce a new strong yet meaningful security model, named challenge-dependent leakage-resilient eCK (CLR-eCK) model, to capture challenge-dependent leakage attacks on both long-term secret key and ephemeral secret key (i.e., randomness). Second, we propose a general framework for constructing one-round CLR-eCK-secure AKE protocols based on smooth projective hash functions (SPHFs). This framework ensures the session key is private and authentic even if the adversary learns a large fraction of both long-term secret key and ephemeral secret key, and hence provides stronger security guarantee than existing AKE protocols which become insecure if the adversary can perform leakage attacks during the execution of a session. Finally, we also present a practical instantiation of the general framework based on the Decisional Diffie-Hellman assumption without random oracle. Our result shows that the instantiation is efficient in terms of the communication and computation overhead and captures more general leakage attacks.

Keywords: Authenticated key exchange, challenge-dependent leakage, strong randomness extractor, smooth projective hash function.

1 Introduction

Leakage-resilient cryptography, particularly leakage-resilient cryptographic primitives such as encryption, signature, and pseudo-random function, has been extensively studied in recent years. However, there are only very few works that have been done on the modelling and construction of leakage-resilient authenticated key exchange (AKE) protocols. This is somewhat surprising since AKE protocols are among the most widely used cryptographic primitives. In particular, they form a central component in many network standards, such as IPSec, SSL/TLS, SSH. In practice, the communication channel over a public network can be easily attacked by a malicious attacker and hence is insecure by default for message transmission. An AKE protocol enables a secure channel to be established among a set of communicating parties by first allowing them to agree on a cryptographically strong secret key, and then applying efficient symmetric key tools to ensure the data confidentiality and authenticity.

Many practical AKE protocols such as the ISO protocol (a.k.a. SIG-DH) [1, 12] and the Internet Key Exchange protocol (a.k.a. SIGMA) [27] have been proposed and deployed in the aforementioned network standards. In such an AKE protocol, each party holds a *long-term public key* and the corresponding *long-term secret key*, which are static in the establishment of different session keys for multiple communication sessions. In order to establish a unique

* An extended abstract of this paper is published in the proceedings of CT-RSA2016 . This is the full version. The final publication is available at http://link.springer.com/book/10.1007%2F978-3-319-29485-8.

session key for an individual session, each party also generates their own *ephemeral secret key* and exchanges the corresponding *ephemeral public key*. Both parties can derive a common session key based on their own secret keys and the public keys of the peer entity. We should note that in practice, an AKE protocol proven secure in the traditional model could be completely insecure in the presence of leakage attacks. For example, an attacker can launch a memory attack [22, 3] to learn partial information about the static long-term secret key, and also obtain partial information about the ephemeral secret key (i.e., randomness) of an AKE session (e.g., via poorly implemented PRNGs [29, 34, 38]).

1.1 Motivations of This Work

The general theme in formulating leakage resilience of cryptographic primitives is that in addition to the normal black-box interaction with an honest party, the adversary can also learn some partial information of a user secret via an abstract leakage function f. More precisely, the adversary is provided with access to a leakage oracle: the adversary can query the oracle with a polynomial-time computable function f, and then receive $f(sk)$, where sk is the user secret key. This approach was applied to model leakage resilience of many cryptographic schemes, such as pseudorandom generators [36], signature schemes [11] and encryption schemes [32, 14]. One of the major problems of leakage resilient cryptography is to define a meaningful leakage function family $\mathcal{F}$ for a cryptographic primitive such that the leakage functions in $\mathcal{F}$ can cover as many leakage attacks as possible while at the same time it is still feasible to construct a scheme that can be proven secure. That is, in order to allow the software-level solution to solve the leakage problem in one go, the leakage function set $\mathcal{F}$ should be *as large as possible* and *adaptively* chosen by the adversary under minimal restrictions.

Limitations in Existing Leakage-Resilient AKE Models. The above modelling approach has been applied to define leakage-resilient AKE protocols in [6, 17, 31, 5]. This was done by allowing the adversary to access the leakage oracle in addition to other oracles defined in a traditional AKE security model. However, we find that the existing leakage-resilient AKE models fail to fully capture general leakage attacks due to the following reasons.

UNNATURAL RESTRICTIONS. The *de facto* security definition of AKE requires that the real challenge session key should be indistinguishable from a randomly chosen key even when the adversary has obtained some information (e.g., by passively eavesdropping the ephemeral public keys, or injecting an ephemeral public key in an active attack) of the challenge session. However, such a definition will bring a problem when it comes to the leakage setting. During the execution of the challenge session, the adversary can access to the leakage oracle by encoding the available information about the challenge session into the leakage function and obtain partial information about the real session key. The previous security definitions for leakage-resilient AKE, e.g., [6, 17, 31, 35], bypassed the definitional difficulty outlined above by only considering *challenge-independent leakage*. Namely, the adversary *cannot make a leakage query which involves a leakage function f that is related to the challenge session*. Specifically, in those models, the adversary is disallowed to make any leakage query during the challenge session. This approach indeed bypasses the technical problem, but it also puts some unnatural restrictions on the adversary by assuming leakage would not happen during the challenge AKE session. Such a definitional difficulty was also recognized in the prior work on leakage-resilient encryption schemes. For example, Naor and Segev wrote in [32] that "it will be very interesting to

find an appropriate framework that allows a certain form of challenge-dependent leakage." We should note that there are some recent works on challenge-dependent leakage-resilient encryption schemes [24, 37], which addressed the problem by weakening the security notions.

INSUFFICIENT LEAKAGE CAPTURING. Although the notions proposed in [6, 17, 31, 35, 5] have already captured some leakage attacks, they only focused on partial leakage of the long-term secret key. We should note that *the partial leakage is independent from the (long-term/ephemeral) secret key reveal queries* in CK/eCK models. In reality, an attacker may completely reveal one (long-term/ephemeral) secret key and learn partial information about the other (ephemeral/long-term) secret key. Such an adversarial capability has never been considered in the previous models. In practice, as mentioned before, potential weakness of the randomness can be caused due to different reasons such as the poor implementation of pseudo-random number generators (PRNGs) [29, 34, 38]. Moreover, real leakage attacks (e.g., timing or power consumption analysis) can also be closely related to the randomness. The problem has been recognized in prior work on leakage-resilient encryption and signature schemes. For example, Halevi and Lin mentioned in [24] that "Another interesting question is to handle leakage from the encryption randomness, not just the secret key", which was later answered by the works in [10, 37]. In terms of the signature schemes, the notion of fully leakage-resilient signatures was also proposed by Katz and Vaikuntanathan [25]. In a fully leakage-resilient setting, the adversary is allowed to obtain leakage of the state information, including the secret keys and internal random coins. However, to date there is no formal treatment on the randomness leakage in AKE protocols. This is surprising as randomness plays a crucial role in AKE protocols and determines the value of a session key.

On After-the-Fact Leakage. It is worth noting that inspired by the work in [24], Alawatugoda et al. [5] modelled after-the-fact leakage for AKE protocols. Their proposed model, named bounded after-the-fact leakage eCK model (BAFL-eCK), captures the leakage of long-term secret keys during the challenge session. However, the BAFL-eCK model has implicitly assumed that the long-term secret has split-state since otherwise their definition is unachievable in the eCK-model. Moreover, the central idea of their AKE construction is to utilize a split-state encryption scheme with a special property (i.e., pair generation indistinguishability), which is a strong assumption. We also note that the split-state approach seems not natural for dealing with ephemeral secret leakage. The work in [4] also introduced a continuous after-the-fact leakage eCK model which is a weaker variant of the one in [5] and hence also suffers from the aforementioned limitations.

Goal of This Work. In this work, we are interested in designing a more general and powerful leakage-resilient AKE model without the aforementioned limitations. Particularly, we ask two questions: *how to generally define a challenge-dependent leakage-resilient AKE security model capturing both long-term and ephemeral secret leakage*, and *how to construct an efficient AKE protocol proven secure under the proposed security model*. The motivation of this work is to solve these two outstanding problems which are of both practical and theoretical importance.

1.2 Related Work

Traditional AKE Security Notions. The Bellare-Rogaway (BR) model [8] gives the first formal security notion for AKE based on an indistinguishability game, where an adversary is required to differentiate between the real session key from a randomly chosen session key. Its

variants are nowadays the *de facto* standard for AKE security analysis. In particular, the Canetti-Krawczyk (CK) model [12], which can be considered as the extension and combination of the BR model and the Bellare-Canetti-Krawczyk (BCK) model [7], has been used to prove the security of many widely used AKE protocols such as SIG-DH and SIGMA. Noting that the CK model does not capture several attacks such as the Key Compromise Impersonation (KCI) attacks, LaMacchia et al. [28] introduced an extension of the CK model, named eCK model, to consider stronger adversaries (in some aspects) who is allowed to access either the long-term secret key or the ephemeral secret key in the target session chosen by the adversary. We refer the readers to Choo et al. [13] for a detailed summary of the differences among the aforementioned AKE models, and to Cremers et al. [16] for a full analysis of these models.

Modelling Leakage Resilience. The method of protecting against leakage attacks by treating them in an abstract way was first proposed by Micali and Reyzin [30] based on the assumption that *only computation leaks information*. Inspired by the cold boot attack presented by Halderman et al. [22], Akavia et al. [3] formalized a general framework, namely, *Relative Leakage Model*, which implicitly assumes that, a leakage attack can reveal a fraction of the secret key, no matter what the secret key size is. The *Bounded-Retrieval Model* (BRM) [6] is a generalization of the relative leakage model. In BRM, the leakage-parameter forms an independent parameter of the system. The secret key-size is then chosen flexibly depending on the leakage parameter. Another relatively stronger leakage model is the *Auxiliary Input Model* [18] where the leakage is not necessarily bounded in length, but it is assumed to be computationally hard to recover the secret-key from the leakage.

Leakage-Resilient AKE. Alwen, Dodis and Wichs [6] presented an efficient leakage-resilient AKE protocol in the random oracle model. They considered a leakage-resilient security model (BRM-CK) by extending the CK model to the BRM leakage setting. They then showed that a leakage-resilient AKE protocol can be constructed from an entropically-unforgeable digital signature scheme secure under chose-message attacks. Such a leakage-resilient signature-based AKE protocol, namely eSIG-DH, however, is at least 3-round and does not capture ephemeral secret key leakage. Also, the security model considered in [6] does not capture challenge-dependent leakage since the adversary is not allowed to make leakage queries during the execution of the challenge session. In [17], Dodis et al. proposed new constructions of AKE protocols that are leakage-resilient in the CK security model (LR-CK). Their first construction follows the result of [6], i.e., authenticating Diffie-Hellman (DH) key exchange using a leakage-resilient signature scheme. The second construction, i.e., Enc-DH, is based on a leakage-resilient CCA-secure PKE scheme: both parties authenticate each other by requiring the peer entity to correctly decrypt the DH ephemeral public key encrypted under the long-term public key. Similar to Alwen at al. [6], the security model given by Dodis et al. [17] is not challenge-dependent, and both constructions have at least 3-round and didn't consider randomness leakage. Another leakage-resilient model for AKE protocols is introduced by Moriyama and Okamoto [31]. Their notion, named λ-leakage resilient eCK (LR-eCK) security, is an extension of the eCK security model with the notion of λ-leakage resilience introduced in [3]. They also presented a 2-round AKE protocol that is λ-leakage resilient eCK secure without random oracles. One limitation of their model is that they just considered the long-term secret key leakage (when the ephemeral secret key is revealed) but not the ephemeral secret key leakage (when the long-term secret key is revealed). Also, their model is not challenge-dependent. Yang et al. [35] initiated the study on leakage resilient AKE in the auxiliary input model. They showed that in the random oracle

model, an AKE protocol secure under auxiliary input attacks can be built based on a digital signature scheme that is random message unforgeable under random message and auxiliary input attacks (RU-RMAA). However, their model is based on the CK model and only captures the challenge-independent leakage of lone-term secret.

1.3 Our Results and Techniques

In this work, we address the aforementioned open problems by designing a strong yet meaningful AKE security model, namely challenge-dependent leakage-resilient eCK (CLR-eCK) model, to capture the challenge-dependent leakage attacks on both the long-term secret key and the ephemeral secret key; we then present a general framework for the construction of CLR-eCK-secure one-round AKE protocol as well as an efficient instantiation based on the DDH assumption. Below we give an overview of our results.

Overview of Our Model. As shown in Table 1, our model is the first *split-state-free* model that captures challenge-dependent leakage on both the long-term secret key and the ephemeral secret key (or randomness), which could occur in practice due to side-channel attacks and weak randomness implementations. In our proposed model, we consider the partial *Relative-Leakage* [3]. We should note that the partial leakage here is independent from the secret key reveal queries in CK/eCK models. In our CLR-eCK model, the adversary can make both leakage and key reveal queries for the long-term and ephemeral secret keys. To be more precise, our model allows one (long-term/ephemeral) secret key to be completely revealed and the other (ephemeral/long-term) secret key to be partially leaked. Such an adversarial capability has never been considered in the previous models.

Our CLR-eCK security model addresses the limitations of the previous leakage-resilient models by allowing both long-term and ephemeral key leakage queries before, during and after the test (i.e., challenge) session. Nevertheless, we should prevent an adversary $\mathcal{M}$ from submitting a leakage function which encodes the session key derivation function of the test session since otherwise the adversary can trivially distinguish the real session key from a random key. To address this technical problem, instead of asking adversary $\mathcal{M}$ to specify the leakage functions before the system setup (i.e., non-adaptive leakage), we require $\mathcal{M}$ to commit a set of leakage functions before it obtains (via key reveal queries) all the inputs, except the to-be-leaked one, of the session key derivation function for the test session. Once $\mathcal{M}$ obtains all the other inputs, it can only use the leakage functions specified in the committed set to learn the partial information of the last unknown secret. To be more precise, in the CLR-eCK model, after $\mathcal{M}$ reveals the ephemeral secret key of the test session, it can only use any function $f_1 \in \mathcal{F}_1$ as the long-term secret key leakage function where $\mathcal{F}_1$ is the set of leakage functions committed by $\mathcal{M}$ before it reveals the ephemeral secret key. A similar treatment is done for the ephemeral secret key leakage function f_2. Under such a restriction, neither f_1 nor f_2 can be embedded with the session key derivation function of the test session and $\mathcal{M}$ cannot launch a trivial attack against the AKE protocol. Therefore, the adversary can still make leakage queries during and after the test session, and if the long-term/ephemeral key is not revealed, then the adversary even doesn't need to commit the ephemeral/long-term key leakage functions $\mathcal{F}_1$ or $\mathcal{F}_2$. We can see that our approach still allows the adversary to adaptively choose leakage functions and meanwhile can capture challenge-dependent leakage under the minimum restriction.

Table 1. Comparison with Existing Leakage-Resilient AKE Security Models

AKE Models	Partial Leakage Setting				Basic Models
	Challenge-Dependent	Long-Term Key	Ephemeral Key	Leakage Model	
BRM-CK [6]	No	$\surd$	$\times$	*Bounded-Retrieval*	CK
LR-CK [17]	No	$\surd$	$\times$	*Relative Leakage*	CK
AI-CK [35]	No	$\surd$	$\times$	*Auxiliary Input*	CK
LR-eCK [31]	No	$\surd$	$\times$	*Relative Leakage*	eCK
BAFL-eCK [5]	Yes (w/ split-state)	$\surd$	$\times$	*Relative Leakage*	eCK
CLR-eCK	Yes (w/o split-state)	$\surd$	$\surd$	*Relative Leakage*	eCK

Generic AKE Construction. To illustrate the practicality of the model, we present a general framework for the construction of AKE protocol secure in our newly proposed challenge-dependent leakage-resilient eCK model. The framework can be regarded as a variant of the AKE protocols proposed by Okamoto et al. [33, 31]. Roughly speaking, we apply both pseudo-random functions (PRFs) and strong randomness extractors in the computation of ephemeral public key and session key to obtain the security in the presence of key leakage. Specifically, we employ an (extended) smooth projective hash function (SPHF) which is defined based on a domain $\mathcal{X}$ and an $\mathcal{NP}$ language $\mathcal{L} \subset \mathcal{X}$. For any word $W \in \mathcal{L}$, the hash value of W can be computed using either a secret hashing key or a public projection key with the knowledge of the witness for W. The key property of SPHF is that the projection key uniquely determines the hash value of any word in the language $\mathcal{L}$ (*projective*) but gives almost no information about the hash value of any point in $\mathcal{X} \setminus \mathcal{L}$ (*smooth*). During the session execution, both parties generate their ephemeral secret key and apply a strong extractor to extract a fresh seed for a PRF in order to derive a word in $\mathcal{L}$. They then exchange their words with the corresponding witness kept secret locally. Additionally, they also run an ephemeral Diffie-Hellman protocol using the exponent which is also output by the PRF. At the end of session, they derive the session key by computing the hash value of both words along with the Diffie-Hellman shared key. The correctness of the framework can be easily obtained due to the property of SPHF and Diffie-Hellman protocol while the security is guaranteed by the strong extractors, pseudo-random functions, along with the underlying (2-)smooth SPHF bulit on an $\mathcal{NP}$ language where the subgroup decision problem is hard.

An Efficient Instantiation. We show that the building blocks in our framework can be instantiated efficiently based on the DDH assumption. Precisely, we first introduce the Diffie-Hellman language $\mathcal{L}_{\mathsf{DH}} = \{(u_1, u_2) | \exists r \in \mathbb{Z}_p, s.t., u_1 = g_1^r, u_2 = g_2^r\}$ where $\mathbb{G}$ is a group of primer order p and $g_1, g_2 \in \mathbb{G}$ are generators. We then show that the subset membership problem over $\mathcal{X} = \mathbb{G}^2$ and $\mathcal{L}_{\mathsf{DH}}$ is hard and use it to construct a 2-smooth SPHF, denoted by $\mathcal{SPHF}_{\mathsf{DH}}$. A concrete protocol based on $\mathcal{SPHF}_{\mathsf{DH}}$ is then presented and proved to be CLR-eCK-secure. A comparison between our protocol and the previous ones is given in Table 2. We should note that the communication cost in eSIG-DH [6] and Enc-DH [17] is higher than our protocol due to the reason that they require their underlying primitive, i.e., signature or encryption scheme, to be leakage-resilient. For example, according to the result (**Theorem 5.2**) of [17], to obtain $(1-\varepsilon)$-leakage resilience, the ciphertexts CT transferred in the Enc-DH protocol has the size of $O(1/\varepsilon)|\mathbb{G}|$. Due to the same reason, the computation overhead of those protocols is also higher than that of our protocol.

Table 2. Comparison with Existing Leakage-Resilient AKE Protocols

Protocols	Round	Communication[1]	Computation[1]	Relative Leakage[2]		Security	AKE Models
				lsk	esk		
eSIG-DH [6]	3	3·|Cer|+2·|$\mathbb{G}$|+2·|Sig|	4·Exp+2·Sgn+2·Ver	$(1-\varepsilon)$	0	w/ RO	BRM-CK [6]
Enc-DH [17]	3	4·|Cer|+|$\mathbb{G}$|+2·|CT|	4·Exp+2·Enc+2·Dec	$(1-\varepsilon)$	0	w/o RO	LR-CK [17]
MO [31]	2	4·|Cer|+9·|$\mathbb{G}$|+3·|Exk|	20·Exp	$(1/4-\varepsilon)$	0	w/o RO	LR-eCK [31]
π [5]	2	4·|Cer|+2·|$\mathbb{G}$|+2·|Sig|	24·Exp	$(1/n-\varepsilon)$	0	w/o RO	BAFL-eCK [5]
Our Protocol	1	4·|Cer|+6·|$\mathbb{G}$|+2·|Exk|	16 ·Exp	$(1/4-\varepsilon)$	$(1-\varepsilon)$	w/o RO	CLR-eCK

[1] For the communication cost, we use Cer to denote the certificate of a long-term public key, $\mathbb{G}$ a group of primer order p, CT a ciphertext, Sig a signature and Exk the key of a randomness extractor. For the computation cost, we use Exp to denote exponentiation, Sgn the signing operation, Ver the verification operation, Enc the encryption operation and Dec the decryption operation.

[2] The "Relative Leakage" column indicates the leakage ratio of a secret key. We use lsk to denote the long-term secret key and esk the ephemeral secret key. In [5], the secret key is split into n parts.

2 Preliminaries

2.1 Notation

For a finite set Ω, $\omega \xleftarrow{\$} \Omega$ denotes that ω is selected uniformly at random from Ω.

Statistical Indistinguishability. Let X and Y be two random variables over a finite domain Ω, the *statistical distance* between X and Y is defined as $\mathsf{SD}(X,Y) = 1/2\sum_{\omega\in\Omega}|\Pr[X=\omega]-\Pr[Y=\omega]|$. We say that X and Y are ϵ-*statistically indistinguishable* if $\mathsf{SD}(X,Y)\leq\epsilon$ and for simplicity we denote it by $X \overset{s}{\equiv}_{\epsilon} Y$. If $\epsilon = 0$, we say that X and Y are *perfectly indistinguishable*.

Computational Indistinguishability. Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be two probability distribution over a finite set Ω where $|\Omega| \geq 2^k$ and k is a security parameter. We then define a distinguisher $\widetilde{\mathcal{D}}$ as follows. In the game, $\widetilde{\mathcal{D}}$ takes as input $\mathcal{V}_1$ and $\mathcal{V}_2$, the challenger flips a coin $\gamma \xleftarrow{\$} \{0,1\}$. $\widetilde{\mathcal{D}}$ is then given an element $v_1 \xleftarrow{\$} \mathcal{V}_1$ if $\gamma = 1$, otherwise an element $v_2 \xleftarrow{\$} \mathcal{V}_2$. Finally, $\widetilde{\mathcal{D}}$ outputs a bit $\gamma' \in \{0,1\}$ as its guess on γ. We define the advantage of $\widetilde{\mathcal{D}}$ in this game as $\mathsf{Adv}^{\mathcal{V}_1,\mathcal{V}_2}_{\widetilde{\mathcal{D}}}(k) = \Pr[\gamma'=\gamma]-1/2$. We say that $\mathcal{V}_1$ and $\mathcal{V}_2$ are *computationally indistinguishable* if for any polynomial-time distinguisher $\mathcal{D}$, $\mathsf{Adv}^{\mathcal{V}_1,\mathcal{V}_2}_{\widetilde{\mathcal{D}}}(k)$ is negligible, and we denote it by $\mathcal{V}_1 \overset{c}{\equiv} \mathcal{V}_2$.

2.2 Randomness Extractor

A central part of our work in this paper is a strong randomness extractor. Here we recall the notion of average-case strong extractor described in [19]. We start with the introduction of average-case min-entropy.

Average-Case Min-Entropy. The *min-entropy* of a random variable X is $\mathrm{H}_\infty(X) = -\log(\max_x \Pr[X=x])$. Dodis et al. [19] formalized the notion of average min-entropy that captures the unpredictability of a random variable X given the value of a random variable Y, formally defined as $\widetilde{\mathrm{H}}_\infty(X|Y) = -\log(\mathrm{E}_{y\leftarrow Y}[2^{-\mathrm{H}_\infty(X|Y=y)}])$. They also showed the following result on average min-entropy in [19].

Lemma 1)([19]). *If Y has 2^λ possible values, then $\widetilde{H}_\infty(X|Y) \geq \widetilde{H}_\infty(X) - \lambda$.*

Definition 1 (Average-Case Strong Extractor)[19]. *Let $k \in \mathbb{N}$ be a security parameter. A function* $\mathsf{Ext} : \{0,1\}^{n(k)} \times \{0,1\}^{t(k)} \leftarrow \{0,1\}^{l(k)}$ *is said to be an average-case (m,ϵ)-strong*

extractor if for all pairs of random variables (X, I) *such that* $X \in \{0,1\}^{n(k)}$ *and* $\widetilde{H}_\infty(X|I) \geq m$*, it holds that*

$$\mathsf{SD}((\mathsf{Ext}(X,S),S,I),(U,S,I)) \leq \epsilon,$$

as long as $l(k) \leq m - 2\log(1/\epsilon)$*, where* $S \xleftarrow{\$} \{0,1\}^{t(k)}$ *is the extraction key and* $U \xleftarrow{\$} \{0,1\}^{l(k)}$.

2.3 Pseudo-Random Function

Here we describe the notion of pseudo-random function (PRF) defined in [21] and its specific class, namely pseudo-random function with pairwise-independent random sources (πPRF), which was proposed by Okamoto in [33].

PRF. Let $k \in \mathbb{N}$ be a security parameter. A function family F is associated with $\{\mathsf{Seed}_k\}_{k\in\mathbb{N}}$, $\{\mathsf{Dom}_k\}_{k\in\mathbb{N}}$ and $\{\mathsf{Rng}_k\}_{k\in\mathbb{N}}$. Formally, for any $\sum \xleftarrow{\$} \mathsf{Seed}_k$, $\sigma \xleftarrow{\$} \sum$, $\mathcal{D} \xleftarrow{\$} \mathsf{Dom}_k$ and $\mathcal{R} \xleftarrow{\$} \mathsf{Rng}_k$, $\mathsf{F}_\sigma^{k,\sum,\mathcal{D},\mathcal{R}}$ defines a function which maps an element of $\mathcal{D}$ to an element of $\mathcal{R}$. That is, $\mathsf{F}_\sigma^{k,\sum,\mathcal{D},\mathcal{R}}(\rho) \in \mathcal{R}$ for any $\rho \in \mathcal{D}$.

Definition 2 (PRF). *We say that* F *is a pseudo-random function (PRF) family if*

$$\{\mathsf{F}_\sigma^{k,\sum,\mathcal{D},\mathcal{R}}(\rho_i)\} \stackrel{c}{\equiv} \{RF(\rho_i)\}$$

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

πPRF. Roughly speaking, πPRF refers to a pseudo-random function family that if a specific key σ is pairwise-independent from other keys, then the output of function with key σ is computationally indistinguishable from a random element.

Formally, let $Z_\sum$ be a set of random variables over $\sum$, and $I_\sum$ be a set of indices regarding $\sum$ such that there exits a deterministic polynomial-time algorithm, $f_\sum : I_\sum \to Z_\sum$, which on input the index $i \in I_\sum$, output $\sigma_i \in Z_\sum$. Consider the random variables $\{\sigma_{i_j}\}_{j=0,...,q(k)} = \{f_\sum(i_j)\}_{j=0,...,q(k)}$ where $i_j \in I_\sum$ and $q(k)$ a polynomial function of k. We say that σ_{i_0} is *pairwisely independent* from other variables $\sigma_{i_1}, ..., \sigma_{i_{q(k)}}$ if for any pair of $(\sigma_{i_0}, \sigma_{i_j})(j = 1, ..., q(k))$, for any $(x, y) \in \sum^2$, we have $\Pr[\sigma_{i_0} \to x \wedge \sigma_{i_j} \to y] = 1/|\sum|^2$.

Definition 3 (πPRF). Define $\widetilde{\mathsf{F}}(\rho_j) = \mathsf{F}_{\sigma_{i_j}}^{k,\sum,\mathcal{D},\mathcal{R}}(\rho_j)$ for $i_j \in I_\sum$, $\rho_j \in \mathcal{D}$. We say that F is a πPRF family if

$$\{\widetilde{\mathsf{F}}(\rho_j)\} \stackrel{c}{\equiv} \{\widetilde{\mathsf{RF}}(\rho_j)\}$$

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

2.4 Smooth Projective Hash Function

Smooth projective hash function(SPHF) is originally introduced by Cramer and Shoup [15] and extended for constructions of many cryptographic primitives [20, 23, 26, 2, 9]. We start with the original definition.

Syntax. Roughly speaking, the definition of an SPHF requires the existence of a domain $\mathcal{X}$ and an underlying $\mathcal{NP}$ language $\mathcal{L}$, where elements of $\mathcal{L}$ form a subset $\mathcal{X}$, i.e., $\mathcal{L} \subset \mathcal{X}$. A key

property of SPHF is that, for any point W in the language $\mathcal{L}$ ($W \in \mathcal{L}$), the hash value of W can be computed by using either a secret hashing key which also works for the computation of any point in the set $\mathcal{X} \setminus \mathcal{L}$, or a public projection key which only works for any point $W \in \mathcal{L}$ and requires the knowledge of the witness w for the fact that $W \in \mathcal{L}$. Formally, an SPHF over a language $\mathcal{L} \subset \mathcal{X}$, onto a set $\mathcal{Y}$, is defined by the following algorithms

- $\mathsf{SPHFSetup}(1^k)$: generates the global parameters param and the description of an $\mathcal{NP}$ language $\mathcal{L}$ from the security parameter k;
- $\mathsf{HashKG}(\mathcal{L}, \mathsf{param})$: generates a hashing key hk for the language $\mathcal{L}$;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}))$: derives the projection key hp from the hashing key hk;
- $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$: outputs the hash value $\mathsf{hv} \in \mathcal{Y}$ on the word W from the hashing key hk;
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, w)$: outputs the hash value $\mathsf{hv}' \in \mathcal{Y}$, on the word W from the projection key hp, and the witness w for the fact that $W \in \mathcal{L}$.

Extension. In order to make the SPHF notion well applied for our work, similar to [15], we also need an extension of the SPHF in this paper. Precisely, we introduce the WordG algorithm and slightly modify the $\mathsf{Hash}, \mathsf{ProjHash}$ algorithms for SPHF as follows.[1]

- $\mathsf{WordG}(\mathcal{L}, \mathsf{param}, w)$: generates a word $W \in \mathcal{L}$ with w the witness ;
- $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W, aux)$: outputs the hash value $\mathsf{hv} \in \mathcal{Y}$ on the word W from the hashing key hk and the auxiliary input aux;
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, w, aux)$: outputs the hash value $\mathsf{hv}' \in \mathcal{Y}$, on the word W from the projection key hp, the witness w for the fact that $W \in \mathcal{L}$ and the auxiliary input aux.

Property. A smooth projective hash function $\mathcal{SPHF}$=($\mathsf{SPHFSetup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{WordG}, \mathsf{Hash}, \mathsf{ProjHash}$) should satisfy the following properties,

- *Correctness*. Let $W = \mathsf{WordG}(\mathcal{L}, \mathsf{param}, w)$, then for all hashing key hk and projection key hp , we have

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

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

$$\mathcal{V}_1 = \{(\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, aux, \mathsf{hv}) | \mathsf{hv} = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W, aux)\},$$

$$\mathcal{V}_2 = \{(\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, aux, \mathsf{hv}) | \mathsf{hv} \xleftarrow{\$} \mathcal{Y}\}.$$

To summary, a smooth projective hash function 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 $\mathcal{X} \setminus \mathcal{L}$.

Definition 4 (2-smooth SPHF). *For any $W_1, W_2 \in \mathcal{X} \setminus \mathcal{L}$, let aux_1, aux_2 be the auxiliary inputs such that $(W_1, aux_1) \neq (W_2, aux_2)$, we say an SPHF is 2-$\mathsf{smooth}$ if the following two distributions are perfectly indistinguishable :*

$$\mathcal{V}_1 = \{(\mathcal{L}, \mathsf{param}, W_1, W_2, \mathsf{hp}, aux_1, aux_2, \mathsf{hv}_1, \mathsf{hv}_2) | \mathsf{hv}_2 = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W_2, aux_2)\},$$

[1] In the rest of paper, all the SPHFs are referred to as the extended SPHF and defined by algorithms ($\mathsf{SPHFSetup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{WordG}, \mathsf{Hash}, \mathsf{ProjHash}$).

$$\mathcal{V}_2 = \{(\mathcal{L}, \mathsf{param}, W_1, W_2, \mathsf{hp}, aux_1, aux_2, \mathsf{hv}_1, \mathsf{hv}_2) | \mathsf{hv}_2 \xleftarrow{\$} \mathcal{Y}\}.$$

where $\mathsf{hv}_1 = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W_1, aux_1)$.

Definition 5 (Hard Subset Membership Problem). *For a finite set $\mathcal{X}$ and an NP language $\mathcal{L} \subset \mathcal{X}$, we say the subset membership problem is hard if for any word $W \xleftarrow{\$} \mathcal{L}$, W is computationally indistinguishable from any random element chosen from $\mathcal{X} \backslash \mathcal{L}$.*

3 A New Strong Leakage-Resilient AKE Security Model

We are now ready to introduce our proposed challenge-dependent leakage-resilient eCK (CLR-eCK) security model.

3.1 AKE Protocol

An AKE protocol is run among parties $(\mathcal{A}, \mathcal{B}, \mathcal{C}, ...)$ which are modelled as probabilistic polynomial-time Turing Machines. Each party has a *long-term secret key* (lsk) together with a certificate that binds the corresponding *long-term public key* (lpk) to the party. Here we denote $\widehat{A}$ $(\widehat{B})$ as the long-term public key of party $\mathcal{A}$ $(\mathcal{B})$ with the certificate issued by a trusted certificate authority $\mathcal{CA}$.

Any two parties, say $\mathcal{A}$ and $\mathcal{B}$, can be activated to run an instance of the AKE protocol, which is referred to as a *session*, and obtain a shared session key. In this paper, we only focus on one-round (i.e., two-pass) AKE protocols. Specifically, during the execution of a session, party $\mathcal{A}$ generates an *ephemeral public/secret key* pair $(epk_{\mathcal{A}}, esk_{\mathcal{A}})$ and sends $(\widehat{B}, \widehat{A}, epk_{\mathcal{A}})$ to the peer $\mathcal{B}$, and vice versa. At the end of the session execution, each party derives the shared session key by taking as input his/her own long-term secret key and ephemeral secret key, along with the long-term public key and ephemeral public key received from the other party.

A session of party $\mathcal{A}$ with peer $\mathcal{B}$ is identified by the session identifier $(\widehat{A}, \widehat{B}, epk_{\mathcal{A}}, epk_{\mathcal{B}})$, and the session $(\widehat{B}, \widehat{A}, epk_{\mathcal{B}}, epk_{\mathcal{A}})$ of party $\mathcal{B}$ is referred to as the *matching session* of $(\widehat{A}, \widehat{B}, epk_{\mathcal{A}}, epk_{\mathcal{B}})$. If the party outputs a session key at the end of the session, we call the session is completed successfully.

3.2 eCK Security Model

The extended Canetti-Krawczyk (eCK) model was proposed by LaMacchia, Lauter and Mityagin [28] based on the CK model which was formulated by Canetti and Krawczyk [12] for the AKE protocols.

Roughly speaking, in the eCK definition, the adversary $\mathcal{M}$ is modelled as a probabilistic polynomial time Turing machine that controls all communications between the honest parties. Note that $\mathcal{M}$ cannot interfere with communication between a single party and the $\mathcal{CA}$ but is able to register fictitious parties. The adversary plays a central role in the model and is responsible for activating all other parties. That is, $\mathcal{M}$ schedules all activations of parties and message delivery. Initially and upon the completion of each activation, $\mathcal{M}$ decides which party to activate next. The adversary $\mathcal{M}$ also decides which incoming message or external request the activated party is to receive.

To be more precise, in the eCK model, adversary $\mathcal{M}$ is given the (certified) public keys of a set of honest users, and is allowed to issue the following oracle queries.

- $\mathsf{Send}(\mathcal{A}, \mathcal{B}, message)$. Send $message$ to party $\mathcal{A}$ on behalf of party $\mathcal{B}$, and obtain $\mathcal{A}$'s response for this message.
- $\mathsf{EstablishParty}(\mathsf{pid})$. This query allows the adversary to register a long-term public key on behalf of party pid, which is said to be *dishonest*.
- $\mathsf{LongTermKeyReveal}(\mathsf{pid})$. This query allows the adversary to learn the long-term secret key of honest party pid.
- $\mathsf{SessionKeyReveal}(\mathsf{sid})$. This query allows the adversary to obtain the session key of the completed session sid.
- $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$. This query allows the adversary to obtain the ephemeral secret key of session sid.

Eventually, in the challenge phase, adversary $\mathcal{M}$ selects a completed session sid^* as the *test session* and makes a query $\mathsf{Test}(\mathsf{sid}^*)$ as follows.

- $\mathsf{Test}(\mathsf{sid}^*)$. To answer this query, the challenger pick $b \xleftarrow{\$} \{0,1\}$. If $b = 1$, the challenger returns $SK^* \leftarrow \mathsf{SessionKeyReveal}(\mathsf{sid}^*)$. Otherwise, the challenger sends $\mathcal{M}$ a random key $R^* \xleftarrow{\$} \{0,1\}^{|SK^*|}$.

Note that the Test query can be issued only once but at any time during the game, and the game terminates as soon as $\mathcal{M}$ outputs its guess b' on b. Here, we require the *test session* to be a *fresh session* which is defined as follows.

Definition 6 (**Fresh Session in eCK Model**). *Let* sid *be the completed session owned by an honest party $\mathcal{A}$ with peer $\mathcal{B}$, who is also honest. If there exists the matching session to session* sid*, we denote the matching session as* $\overline{\mathsf{sid}}$*. Session* sid *is said to be fresh if none of the following conditions hold:*

- *$\mathcal{M}$ issues a* $\mathsf{SessionKeyReveal}(\mathsf{sid})$ *query or a* $\mathsf{SessionKeyReveal}(\overline{\mathsf{sid}})$ *query (If* $\overline{\mathsf{sid}}$ *exists).*
- $\overline{\mathsf{sid}}$ *exists and $\mathcal{M}$ issues either*
  - $\mathsf{LongTermKeyReveal}(\mathcal{A}) \wedge \mathsf{EphemeralKeyReveal}(\mathsf{sid})$*, or*
  - $\mathsf{LongTermKeyReveal}(\mathcal{B}) \wedge \mathsf{EphemeralKeyReveal}(\overline{\mathsf{sid}})$.
- $\overline{\mathsf{sid}}$ *does not exist and $\mathcal{M}$ issues either*
  - $\mathsf{LongTermKeyReveal}(\mathcal{A}) \wedge \mathsf{EphemeralKeyReveal}(\mathsf{sid})$*, or*
  - $\mathsf{LongTermKeyReveal}(\mathcal{B})$.

We remark that the freshness of the test session can be identified only after the game is completed as $\mathcal{M}$ can continue the other queries after the Test query. That is, $\mathcal{M}$ wins the game if he correctly guesses the challenge for the test session which remains fresh until the end of the game. Formally, we have the following notion for eCK security.

Definition 7 (**eCK Security**). *Let the test session* sid^* *be fresh where adversary $\mathcal{M}$ issues* $\mathsf{Test}(sid^*)$ *query. We define the advantage of $\mathcal{M}$ in the eCK game by*

$$\mathsf{Adv}^{\mathsf{eCK}}_{\mathcal{M}}(k) = \Pr[b' = b] - 1/2,$$

where k is the security parameter of the AKE protocol. We say the AKE protocol is eCK-secure if the matching session computes the same session key and for any probabilistic polynomial-time adversary $\mathcal{M}$, $\mathsf{Adv}^{\mathsf{eCK}}_{\mathcal{M}}(k)$ *is negligible.*

3.3 Challenge-Dependent Leakage-Resilient eCK Model

We introduce a new eCK-based security notion to capture various *side-channel attacks* against AKE protocols. Our notion, named *Challenge-Dependent Leakage-Resilient eCK* (CLR-eCK) model is the first split-state-free security model that captures both long-term and ephemeral key leakage *and* allows the adversary to issue leakage queries even *after the activation of the test session*. Formally, adversary $\mathcal{M}$ is allowed to issue the following queries.

- $\mathsf{Send}(\mathcal{A}, \mathcal{B}, message)$. Send $message$ to party $\mathcal{A}$ on behalf of party $\mathcal{B}$, and obtain $\mathcal{A}$'s response for this message.
- $\mathsf{EstablishParty}(\mathsf{pid})$. Register a long-term public key on behalf of party pid, which is said to be *dishonest*.
- $\mathsf{LongTermKeyReveal}(\mathsf{pid})$. Query the long-term secret key of honest party pid.
- $\mathsf{SessionKeyReveal}(\mathsf{sid})$. Query the session key of the completed session sid.
- $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$. Query the ephemeral secret key of session sid.
- $\mathsf{LongTermKeyLeakage}(f_1, \mathsf{pid})$. This query allows $\mathcal{M}$ to learn $f_1(lsk)$ where f_1 denotes the leakage function and lsk denotes the long-term secret key of party pid.
- $\mathsf{EphemeralKeyLeakage}(f_2, \mathsf{sid})$. This query allows $\mathcal{M}$ to learn $f_2(esk)$ where f_2 denotes the leakage function and esk denotes the ephemeral secret key used by an honest user in the session sid.
- $\mathsf{Test}(\mathsf{sid}^*)$. To answer this query, the challenger pick $b \xleftarrow{\$} \{0,1\}$. If $b = 1$, the challenger returns $SK^* \leftarrow \mathsf{SessionKeyReveal}(\mathsf{sid}^*)$. Otherwise, the challenger sends the adversary a random key $R^* \xleftarrow{\$} \{0,1\}^{|SK^*|}$.

Note that the Test query can be issued only once but at any time during the game, and the game terminates as soon as $\mathcal{M}$ outputs its guess b' on b.

Restrictions on the Leakage Function. In our CLR-eCK security model, we consider several restrictions on the leakage function to prevent the adversary $\mathcal{M}$ from trivially breaking the AKE protocol.

The first restriction is that the output size of the leakage function f_1 and f_2 must be less than $|lsk|$ and $|esk|$, respectively. Specifically, following some previous work on leakage resilient cryptography [32], we require the output size of a leakage function f is at most λ bits, which means the entropy loss of sk is at most λ bits upon observing $f(sk)$. Formally, we define the bounded leakage function family $\mathcal{F}_{\mathsf{bbd\text{-}I}}$ for the long-term secret key and $\mathcal{F}_{\mathsf{bbd\text{-}II}}$ for the ephemeral secret key as follows. $\mathcal{F}_{\mathsf{bbd\text{-}I}}(k)$ is defined as the class of all polynomial-time computable functions: $f : \{0,1\}^{|lsk|} \rightarrow \{0,1\}^{\leq \lambda_1(k)}$, where $\lambda_1(k) < |lsk|$. $\mathcal{F}_{\mathsf{bbd\text{-}II}}(k)$ is defined as the class of all polynomial-time computable functions: $f : \{0,1\}^{|esk|} \rightarrow \{0,1\}^{\leq \lambda_2(k)}$, where $\lambda_2(k) < |esk|$. We then require that the leakage function submitted by the adversary should satisfy that $f_1 \in \mathcal{F}_{\mathsf{bbd\text{-}I}}$ and $f_2 \in \mathcal{F}_{\mathsf{bbd\text{-}II}}$.

Another restriction that must be enforced is related to the challenge-dependent leakage security of AKE protocols. Consider a test session sid^* which is owned by party $\mathcal{A}$ with peer $\mathcal{B}$. Note that for a 2-pass AKE protocol, the session key of sid^* is determined by $(\widehat{A}, \widehat{B}, lsk_{\mathcal{A}}, esk^*_{\mathcal{A}}, lpk_{\mathcal{B}}, epk^*_{\mathcal{B}})$ which contains only two secret keys (i.e., $lsk_{\mathcal{A}}, esk^*_{\mathcal{A}}$). Since $\mathcal{M}$ is allowed to reveal $esk^*_{\mathcal{A}}$ ($lsk_{\mathcal{A}}$) in the eCK model, $\mathcal{M}$ can launch a trivial attack by encoding the session key derivation function into the leakage function of $lsk_{\mathcal{A}}$ ($esk^*_{\mathcal{A}}$) and hence wins the security game. Therefore, adversary $\mathcal{M}$ should not be allowed to adaptively issue leakage query after it obtains all the

other (secret) information for session key computation, otherwise the security of AKE protocol is unachievable. More precisely, we describe the restrictions on $\mathsf{LongTermKeyLeakage}(f_1, \mathcal{A})$ and $\mathsf{EphemeralKeyLeakage}(f_2, \mathsf{sid}^*)$ as follows.

- $\mathcal{M}$ is allowed to ask for arbitrary leakage function $f_1 \in \mathcal{F}_{\mathsf{bbd\text{-}I}}$ before it obtains the ephemeral secret key $esk^*_{\mathcal{A}}$, i.e., by issuing $\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*)$ query; however, after obtaining $esk^*_{\mathcal{A}}$, $\mathcal{M}$ can only use the leakage functions $f_1 \in \mathcal{F}_1 \subset \mathcal{F}_{\mathsf{bbd\text{-}I}}$ where $\mathcal{F}_1$ is a set of leakage functions chosen and submitted by $\mathcal{M}$ before it issues $\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*)$.
- $\mathcal{M}$ is allowed to ask for arbitrary leakage function $f_2 \in \mathcal{F}_{\mathsf{bbd\text{-}II}}$ before it obtains the long-term secret key $lsk_{\mathcal{A}}$, i.e., by issuing $\mathsf{LongTermKeyReveal}(\mathcal{A})$ query; however, after obtaining $lsk_{\mathcal{A}}$, $\mathcal{M}$ can only use the leakage functions $f_2 \in \mathcal{F}_2 \subset \mathcal{F}_{\mathsf{bbd\text{-}II}}$ where $\mathcal{F}_2$ is a set of leakage functions chosen and submitted by $\mathcal{M}$ before it issues $\mathsf{LongTermKeyReveal}(\mathcal{A})$.

We should note that if $\overline{\mathsf{sid}^*}$ exists, the above restriction must also be enforced for the leakage query $\mathsf{LongTermKeyLeakage}(f_1, \mathcal{B})$ and $\mathsf{EphemeralKeyLeakage}(f_2, \overline{\mathsf{sid}^*})$, since the session key of sid^* is also determined by $(\widehat{A}, \widehat{B}, lpk_{\mathcal{A}}, epk^*_{\mathcal{A}}, lsk_{\mathcal{B}}, esk^*_{\mathcal{B}})$.

Adaptive Leakage. One can see that our proposed model enables adversary $\mathcal{M}$ to choose $\mathcal{F}_1, \mathcal{F}_2$ adaptively and $\mathcal{M}$ can submit $\mathcal{F}_1, \mathcal{F}_2$ even after the challenge phase as long as the restriction holds. That is, $\mathcal{M}$ can specify function set $\mathcal{F}_1, \mathcal{F}_2$ after seeing $epk^*_{\mathcal{A}}$ and $epk^*_{\mathcal{B}}$. Also, if there is no long-term (ephemeral, respectively) key reveal query, then $\mathcal{F}_1$ ($\mathcal{F}_2$, respectively) is the same as $\mathcal{F}_{\mathsf{bbd\text{-}I}}$ ($\mathcal{F}_{\mathsf{bbd\text{-}II}}$, respectively). Implicitly, $\mathcal{M}$ is allowed to obtain $f_1(lsk_{\mathcal{A}}), f'_1(lsk_{\mathcal{B}}), f_2(esk^*_{\mathcal{A}}), f'_2(esk^*_{\mathcal{B}})$ where $f_1, f'_1 \in \mathcal{F}_{\mathsf{bbd\text{-}I}}, f_2, f'_2 \in \mathcal{F}_{\mathsf{bbd\text{-}II}}$ can be dependent on $(lpk_{\mathcal{A}}, lpk_{\mathcal{B}}, epk^*_{\mathcal{A}}, epk^*_{\mathcal{B}})$, or to obtain $f_1(lsk_{\mathcal{A}}), f_2(esk^*_{\mathcal{B}})$ where $f_1 \in \mathcal{F}_1, f_2 \in \mathcal{F}_2$ can be dependent on $(lpk_{\mathcal{A}}, lpk_{\mathcal{B}}, lsk_{\mathcal{B}}, epk^*_{\mathcal{A}}, epk^*_{\mathcal{B}})$ and $(lpk_{\mathcal{A}}, lpk_{\mathcal{B}}, epk^*_{\mathcal{A}}, esk^*_{\mathcal{A}}, epk^*_{\mathcal{B}})$, respectively. Since the leakage can happen during or after the challenge session and can be related to the challenge session, our proposed security model captures the challenge-dependent leakage security for AKE protocols.

We define the notion of a *fresh session* in the CLR-eCK model as follows.

Definition 8 (**(λ_1, λ_2)-Leakage Fresh Session in the CLR-eCK Model**). *Let* sid *be a completed session owned by an honest party $\mathcal{A}$ with peer $\mathcal{B}$, who is also honest. Let* $\overline{\mathsf{sid}}$ *denote the matching session of* sid*, if it exists. Session* sid *is said to be fresh in the* CLR-eCK *model if the following conditions hold:*

- sid *is a fresh session in the sense of eCK model.*
- *$\mathcal{M}$ only issues the queries* $\mathsf{LongTermKeyLeakage}(f_1, \mathcal{A})$, $\mathsf{LongTermKeyLeakage}(f'_1, \mathcal{B})$, $\mathsf{EphemeralKeyLeakage}(f_2, \mathsf{sid})$, $\mathsf{EphemeralKeyLeakage}(f'_2, \overline{\mathsf{sid}})$ *(if* $\overline{\mathsf{sid}}$ *exists), such that f_1, f'_1, f_2, f'_2 satisfy the restriction given above.*
- *The total output length of all the* LongTermKeyLeakage *queries to $\mathcal{A}$ ($\mathcal{B}$, respectively) is at most λ_1.*
- *The total output length of all the* EphemeralKeyLeakage *query to* sid *(*$\overline{\mathsf{sid}}$*, respectively, if it exists) is at most λ_2.*

We now describe the notion of CLR-eCK security.

Definition 9 (CLR-eCK **Security**). *Let the test session* sid^* *be (λ_1, λ_2)-leakage fresh where adversary $\mathcal{M}$ issues* $\mathsf{Test}(sid^*)$ *query. We define the advantage of $\mathcal{M}$ in the* CLR-eCK *game*

by $\mathsf{Adv}_{\mathcal{M}}^{\mathsf{CLR\text{-}eCK}}(k) = \Pr[b' = b] - 1/2$*, where* k *is the security parameter of the AKE protocol. We say the AKE protocol is* (λ_1, λ_2)*-challenge-dependent leakage-resilient eCK-secure* ((λ_1, λ_2)-CLR-eCK-*secure*) *if the matching session computes the same session key and for any probabilistic polynomial-time adversary* $\mathcal{M}$*,* $\mathsf{Adv}_{\mathcal{M}}^{\mathsf{CLR\text{-}eCK}}(k)$ *is negligible.*

Remark. Here we give a further discussion on the relationship between the reveal oracle, e.g., LongTermKeyReveal and the leakage oracle, e.g., LongTermKeyLeakage. We can see that it is meaningless for $\mathcal{M}$ to issue the leakage query on the long-term secret key (ephemeral secret key) if it has already obtained the whole key through querying the reveal oracle. Indeed, adversary $\mathcal{M}$ can compute by itself the leakage function $f_1(lsk_{\mathcal{A}})$ if $lsk_{\mathcal{A}}$ is known to him.

Therefore, we can observe that the meaningful queries that adversary $\mathcal{M}$ will ask in CLR-eCK model are as follows. Suppose session sid^* is the test session owned by $\mathcal{A}$ with the peer $\mathcal{B}$. If $\overline{\mathsf{sid}^*}$ exists, $\mathcal{M}$ will only make queries that form a subset of any one of the following cases:

- $\{\mathsf{LongTermKeyReveal}(\mathcal{A}), \mathsf{LongTermKeyReveal}(\mathcal{B}), \mathsf{EphemeralKeyLeakage}(\mathsf{sid}^*), \mathsf{EphemeralKeyLeakage}(\overline{\mathsf{sid}^*})\}$,[2]
- $\{\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*), \mathsf{EphemeralKeyReveal}(\overline{\mathsf{sid}^*}), \mathsf{LongTermKeyLeakage}(\mathcal{A}), \mathsf{LongTermKeyLeakage}(\mathcal{B})\}$,
- $\{\mathsf{LongTermKeyReveal}(\mathcal{A}), \mathsf{EphemeralKeyReveal}(\overline{\mathsf{sid}^*}), \mathsf{EphemeralKeyLeakage}(\mathsf{sid}^*), \mathsf{LongTermKeyLeakage}(\mathcal{B})\}$,
- $\{\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*), \mathsf{LongTermKeyReveal}(\mathcal{B}), \mathsf{LongTermKeyLeakage}(\mathcal{A}), \mathsf{EphemeralKeyLeakage}(\overline{\mathsf{sid}^*})\}$.

If $\overline{\mathsf{sid}^*}$ does not exist, we have the following cases:

- $\{\mathsf{LongTermKeyReveal}(\mathcal{A}), \mathsf{EphemeralKeyLeakage}(\mathsf{sid}^*), \mathsf{LongTermKeyLeakage}(\mathcal{B})\}$,
- $\{\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*), \mathsf{LongTermKeyLeakage}(\mathcal{A}), \mathsf{LongTermKeyLeakage}(\mathcal{B})\}$.

4 One-Round CLR-eCK-Secure AKE

In this section, we present a generic construction of one-round CLR-eCK-secure AKE protocol.

4.1 General Framework

Fig. 1 describes a generic construction of the CLR-eCK secure AKE protocol. Suppose that k is the system security parameter. Let $\mathbb{G}$ be a group with prime order p and g is a random generator of $\mathbb{G}$. Let $\mathcal{SPHF}$ denote a 2-smooth SPHF over $\mathcal{L} \subset \mathcal{X}$ and onto the set $\mathcal{Y}$ such that the subset membership problem between $\mathcal{L}$ and $\mathcal{X}$ is hard. Denote the hashing key space by $\mathcal{HK}$, the projection key space by $\mathcal{HP}$, the auxiliary input space by $\mathcal{AUX}$ and the witness space by $\mathcal{W}$. Pick two collision-resistant hash functions $H_1 : \{0,1\}^* \to \mathcal{AUX}, H_2 : \mathbb{G} \to \mathcal{Y}$.

Let $\lambda_1 = \lambda_1(k)$ be the bound on the amount of long-term secret key leakage and $\lambda_2 = \lambda_2(k)$ be that of the ephemeral secret key leakage. Let $\mathsf{Ext}_1, \mathsf{Ext}_2, \mathsf{Ext}_3$ be strong extractors as follows. $\mathsf{Ext}_1 : \mathcal{HK} \times \{0,1\}^{t_1(k)} \to \{0,1\}^{l_1(k)}$ is an average-case $(|\mathcal{HK}| - \lambda_1, \epsilon_1)$-strong extractor. $\mathsf{Ext}_2 : \{0,1\}^{u(k)} \times \{0,1\}^{t_2(k)} \to \{0,1\}^{l_2(k)}$ is an average-case $(k - \lambda_2, \epsilon_2)$-strong extractor. $\mathsf{Ext}_3 : \mathcal{Y} \times \{0,1\}^{t_3(k)} \to \{0,1\}^{l_3(k)}$ is an average-case $(|\mathcal{Y}| - \lambda_1, \epsilon_3)$-strong extractor. Here $\epsilon_1 = \epsilon_1(k), \epsilon_2 = \epsilon_2(k), \epsilon_3 = \epsilon_3(k)$ are negligible.

[2] For simplicity, we will omit the leakage function in the input of the leakage query in the rest of the paper.

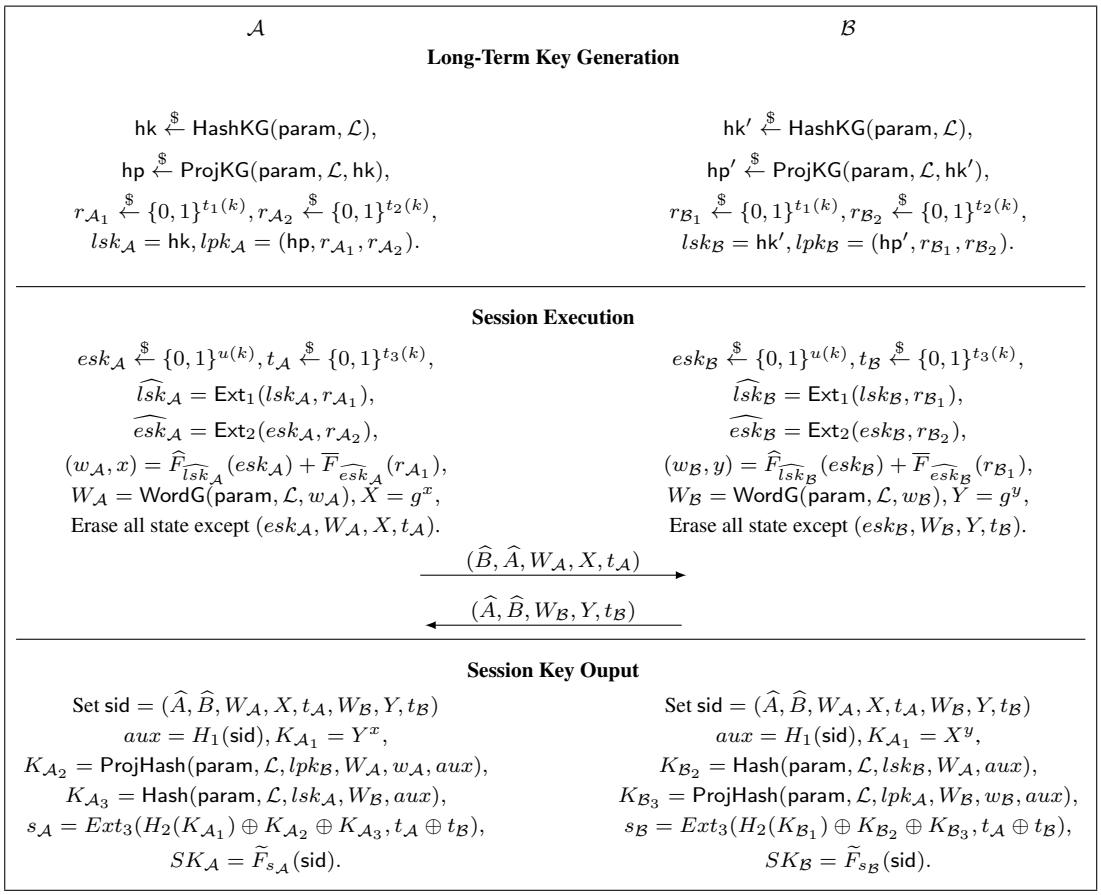

Fig. 1. Framework for CLR-eCK secure AKE

Let $\widehat{\mathsf{F}}$ and $\overline{\mathsf{F}}$ be PRF families and $\widetilde{\mathsf{F}}$ be a πPRF family as follows.
$\widehat{\mathsf{F}}^{k,\sum_{\widehat{\mathsf{F}}},\mathcal{D}_{\widehat{\mathsf{F}}},\mathcal{R}_{\widehat{\mathsf{F}}}} : \sum_{\widehat{\mathsf{F}}} = \{0,1\}^{l_1(k)}, \mathcal{D}_{\widehat{\mathsf{F}}} = \{0,1\}^{u(k)}, \mathcal{R}_{\widehat{\mathsf{F}}} = \mathcal{W} \times \mathbb{Z}_p$,
$\overline{\mathsf{F}}^{k,\sum_{\overline{\mathsf{F}}},\mathcal{D}_{\overline{\mathsf{F}}},\mathcal{R}_{\overline{\mathsf{F}}}} : \sum_{\overline{\mathsf{F}}} = \{0,1\}^{l_2(k)}, \mathcal{D}_{\overline{\mathsf{F}}} = \{0,1\}^{t_1(k)}, \mathcal{R}_{\overline{\mathsf{F}}} = \mathcal{W} \times \mathbb{Z}_p$,
$\widetilde{\mathsf{F}}^{k,\sum_{\widetilde{\mathsf{F}}},\mathcal{D}_{\widetilde{\mathsf{F}}},\mathcal{R}_{\widetilde{\mathsf{F}}}} : \sum_{\widetilde{\mathsf{F}}} = \{0,1\}^{l_3(k)}, \mathcal{D}_{\widetilde{\mathsf{F}}} = (\Lambda_k)^2 \times \mathcal{L}^2 \times \mathbb{G}^2 \times \{0,1\}^{2t_3(k)}, \mathcal{R}_{\widetilde{\mathsf{F}}} = \{0,1\}^{l_4(k)}$.[3]
Let $\widehat{F} \leftarrow \widehat{\mathsf{F}}^{k,\sum_{\widehat{\mathsf{F}}},\mathcal{D}_{\widehat{\mathsf{F}}},\mathcal{R}_{\widehat{\mathsf{F}}}}$, $\overline{F} \leftarrow \overline{\mathsf{F}}^{k,\sum_{\overline{\mathsf{F}}},\mathcal{D}_{\overline{\mathsf{F}}},\mathcal{R}_{\overline{\mathsf{F}}}}$ and $\widetilde{F} \leftarrow \widetilde{\mathsf{F}}^{k,\sum_{\widetilde{\mathsf{F}}},\mathcal{D}_{\widetilde{\mathsf{F}}},\mathcal{R}_{\widetilde{\mathsf{F}}}}$.

The system parameter is $(\mathsf{param}, \mathbb{G}, p, g, H_1, H_2, \mathsf{Ext}_1, \mathsf{Ext}_2, \mathsf{Ext}_3, \widehat{F}, \overline{F}, \widetilde{F})$ where $\mathsf{param} \leftarrow \mathsf{SPHFSetup}(1^k)$.

Long-Term Key Generation. At the long-term key generation stage, $\mathcal{A}$ runs the algorithm HashKG to obtain a hashing key hk and then the algorithm ProjKG to obtain the projection key hp, picks $r_{\mathcal{A}_1} \xleftarrow{\$} \{0,1\}^{t_1(k)}, r_{\mathcal{A}_2} \xleftarrow{\$} \{0,1\}^{t_2(k)}$, then sets its long-term key pair as $lsk_{\mathcal{A}} = \mathsf{hk}, lpk_{\mathcal{A}} = (\mathsf{hp}, r_{\mathcal{A}_1}, r_{\mathcal{A}_2})$. Similarly, $\mathcal{B}$ generates its long-term key pair as $lsk_{\mathcal{B}} = \mathsf{hk}', lpk_{\mathcal{B}} = (\mathsf{hp}', r_{\mathcal{B}_1}, r_{\mathcal{B}_2})$.

Session Execution $(\mathcal{A} \rightleftharpoons \mathcal{B})$**.** The key exchange protocol between $\mathcal{A}$ and $\mathcal{B}$ executes as follows.

– $(\mathcal{A} \rightharpoonup \mathcal{B})$. $\mathcal{A}$ performs the following steps.

1. Selects the ephemeral secret key $esk_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^{u(k)}$ and picks $t_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^{t_3(k)}$.

[3] In this paper, we denote the space of a certified long-term public key (such as $\widehat{\mathcal{A}}$) by Λ_k.

2. Sets $\widehat{lsk}_{\mathcal{A}} = \mathsf{Ext}_1(lsk_{\mathcal{A}}, r_{\mathcal{A}_1}), \widehat{esk}_{\mathcal{A}} = \mathsf{Ext}_2(esk_{\mathcal{A}}, r_{\mathcal{A}_2})$.
3. Computes $(w_{\mathcal{A}}, x) = \widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(esk_{\mathcal{A}}) + \overline{F}_{\widehat{esk}_{\mathcal{A}}}(r_{\mathcal{A}_1})$.
4. Runs the algorithm $\mathsf{WordG}(\mathsf{param}, \mathcal{L}, w_{\mathcal{A}})$ to obtain a word $W_{\mathcal{A}}$ and computes $X = g^x$.
5. Erase all state except $(esk_{\mathcal{A}}, W_{\mathcal{A}}, X, t_{\mathcal{A}})$, sets $(W_{\mathcal{A}}, X, t_{\mathcal{A}})$ as the ephemeral public key and sends $(\widehat{B}, \widehat{A}, W_{\mathcal{A}}, X, t_{\mathcal{A}})$ to $\mathcal{B}$.

– ($\mathcal{B} \rightharpoonup \mathcal{A}$). Similarly, $\mathcal{B}$ executes the following steps.

1. Selects the ephemeral secret key $esk_{\mathcal{B}} \xleftarrow{\$} \{0,1\}^{u(k)}$ and picks $t_{\mathcal{B}} \xleftarrow{\$} \{0,1\}^{t_3(k)}$.
2. Sets $\widehat{lsk}_{\mathcal{B}} = \mathsf{Ext}_1(lsk_{\mathcal{B}}, r_{\mathcal{B}_1}), \widehat{esk}_{\mathcal{B}} = \mathsf{Ext}_2(esk_{\mathcal{B}}, r_{\mathcal{B}_2})$.
3. Computes $(w_{\mathcal{B}}, y) = \widehat{F}_{\widehat{lsk}_{\mathcal{B}}}(esk_{\mathcal{B}}) + \overline{F}_{\widehat{esk}_{\mathcal{B}}}(r_{\mathcal{B}_1})$.
4. Runs the algorithm $\mathsf{WordG}(\mathsf{param}, \mathcal{L}, w_{\mathcal{B}})$ to obtain a word $W_{\mathcal{B}}$ and computes $Y = g^y$.
5. Erase all state except $(esk_{\mathcal{B}}, W_{\mathcal{B}}, Y, t_{\mathcal{B}})$, sets $(W_{\mathcal{B}}, Y, t_{\mathcal{B}})$ as the ephemeral public key and sends $(\widehat{A}, \widehat{B}, W_{\mathcal{B}}, Y, t_{\mathcal{B}})$ to $\mathcal{A}$.

Session Key Output. When $\mathcal{A}$ receives $(\widehat{A}, \widehat{B}, W_{\mathcal{B}}, Y, t_{\mathcal{B}})$, $\mathcal{A}$ sets $\mathsf{sid} = (\widehat{A}, \widehat{B}, W_{\mathcal{A}}, X, t_{\mathcal{A}}, W_{\mathcal{B}}, Y, t_{\mathcal{B}})$ and computes the session key as follows.

1. Reconstructs $(w_{\mathcal{A}}, x)$ from $(lsk_{\mathcal{A}}, lpk_{\mathcal{A}}, esk_{\mathcal{A}})$, and computes $aux = H_1(\mathsf{sid})$.
2. Computes $K_{\mathcal{A}_1} = Y^x$, $K_{\mathcal{A}_2} = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, lpk_{\mathcal{B}}, W_{\mathcal{A}}, w_{\mathcal{A}}, aux)$, $K_{\mathcal{A}_3} = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, lsk_{\mathcal{A}}, W_{\mathcal{B}}, aux)$.
3. Sets $s_{\mathcal{A}} = Ext_3(H_2(K_{\mathcal{A}_1}) \oplus K_{\mathcal{A}_2} \oplus K_{\mathcal{A}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}})$.
4. Computes $SK_{\mathcal{A}} = \widetilde{F}_{s_{\mathcal{A}}}(\mathsf{sid})$.

Similarly, party $\mathcal{B}$ sets $\mathsf{sid} = (\widehat{A}, \widehat{B}, W_{\mathcal{A}}, X, t_{\mathcal{A}}, W_{\mathcal{B}}, Y, t_{\mathcal{B}})$ and then computes the session key as follows.

1. Reconstructs $(w_{\mathcal{B}}, y)$ from $(lsk_{\mathcal{B}}, lpk_{\mathcal{B}}, esk_{\mathcal{B}})$ and computes $aux = H_1(\mathsf{sid})$.
2. Computes $K_{\mathcal{B}_1} = X^y$, $K_{\mathcal{B}_2} = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, lsk_{\mathcal{B}}, W_{\mathcal{A}}, aux)$, $K_{\mathcal{B}_3} = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, lpk_{\mathcal{A}}, W_{\mathcal{B}}, w_{\mathcal{B}}, aux)$.
3. Sets $s_{\mathcal{B}} = Ext_3(H_2(K_{\mathcal{B}_1}) \oplus K_{\mathcal{B}_2} \oplus K_{\mathcal{B}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}})$.
4. Computes $SK_{\mathcal{B}} = \widetilde{F}_{s_{\mathcal{B}}}(\mathsf{sid})$.

Correctness Analysis. One can note that $K_{\mathcal{A}_1} = K_{\mathcal{B}_1}$ as $K_{\mathcal{A}_1} = Y^x = X^y = K_{\mathcal{B}_1} = g^{xy}$. Due to the property of SPHF, we have $K_{\mathcal{A}_2} = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, lpk_{\mathcal{B}}, W_{\mathcal{A}}, w_{\mathcal{A}}, aux) = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, lsk_{\mathcal{B}}, W_{\mathcal{A}}, aux) = K_{\mathcal{B}_2}$, $K_{\mathcal{A}_3} = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, lsk_{\mathcal{A}}, W_{\mathcal{B}}, aux) = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, lpk_{\mathcal{A}}, W_{\mathcal{B}}, w_{\mathcal{B}}, aux) = K_{\mathcal{B}_3}$. Therefore, we can obtain that $s_{\mathcal{A}} = Ext_3(H_2(K_{\mathcal{A}_1}) \oplus K_{\mathcal{A}_2} \oplus K_{\mathcal{A}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}}) = s_{\mathcal{B}} = Ext_3(H_2(K_{\mathcal{B}_1}) \oplus K_{\mathcal{B}_2} \oplus K_{\mathcal{B}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}})$, which guarantees that $SK_{\mathcal{A}} = SK_{\mathcal{B}}$.

4.2 Security Analysis

Theorem 1. *The AKE protocol following the general framework is* (λ_1, λ_2)-CLR-eCK-*secure if the underlying smooth projective hash function is 2-*smooth*, the DDH assumption holds in* $\mathbb{G}$, H_1, H_2 *are collision-resistant hash functions,* $\widehat{\mathsf{F}}$ *and* $\overline{\mathsf{F}}$ *are PRF families and* $\widetilde{\mathsf{F}}$ *is a* π*PRF family. Here* $\lambda_1 \leq \min\{|\mathcal{HK}| - 2\log(1/\epsilon_1) - l_1(k), |\mathcal{Y}| - 2\log(1/\epsilon_3) - l_3(k)\}, \lambda_2 \leq u(k) - 2\log(1/\epsilon_2) - l_2(k)$.

Proof. Let session $\mathsf{sid}^* = (\widehat{A}, \widehat{B}, W^*_{\mathcal{A}}, X^*, t^*_{\mathcal{A}}, W^*_{\mathcal{B}}, Y^*, t^*_{\mathcal{B}})$ be the target session chosen by adversary $\mathcal{M}$. $\mathcal{A}$ is the owner of the session sid^* and $\mathcal{B}$ is the peer. We then analyze the security of the AKE protocol in the following two disjoint cases.

Case I. *There exists a matching session,* $\overline{\mathsf{sid}^*}$*, of the target session* sid^*.

we analyse the security based on the type of the reveal query and leakage query that the adversary issues to the target session, the matching session and the corresponding parties.

- $\mathsf{LongTermKeyReveal}(\mathcal{A}), \mathsf{LongTermKeyReveal}(\mathcal{B}), \mathsf{EphemeralKeyLeakage}(\mathsf{sid}^*)$, $\mathsf{EphemeralKeyLeakage}(\overline{\mathsf{sid}^*})$. In this sub-case, suppose that the adversary obtains at most λ_2-bits of the ephemeral secret key of target session sid^*, we have that

$$\widehat{esk}^*_{\mathcal{A}} = \mathsf{Ext}_2(esk^*_{\mathcal{A}}, r_{\mathcal{A}_2}) \overset{s}{\equiv}_{\epsilon_2} \widehat{esk}'_{\mathcal{A}} \overset{\$}{\leftarrow} \{0,1\}^{l_2(k)}, \tag{1}$$

  Therefore, $(w^*_{\mathcal{A}}, x^*) = \widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(esk^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r_{\mathcal{A}_1}) \overset{c}{\equiv} (w'_{\mathcal{A}}, x') \overset{\$}{\leftarrow} \mathcal{W} \times \mathbb{Z}_p$. Similarly, suppose that the adversary obtains at most λ_2-bits of the ephemeral secret key of matching session $\overline{\mathsf{sid}^*}$, we have that

$$\widehat{esk}^*_{\mathcal{B}} = \mathsf{Ext}_2(esk^*_{\mathcal{B}}, r_{\mathcal{B}_2}) \overset{s}{\equiv}_{\epsilon_2} \widehat{esk}'_{\mathcal{B}} \overset{\$}{\leftarrow} \{0,1\}^{l_2(k)}, \tag{2}$$

  and thus $(w^*_{\mathcal{B}}, y^*) = \widehat{F}_{\widehat{lsk}_{\mathcal{B}}}(esk^*_{\mathcal{B}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{B}}}(r_{\mathcal{B}_1}) \overset{c}{\equiv} (w'_{\mathcal{B}}, y') \overset{\$}{\leftarrow} \mathcal{W} \times \mathbb{Z}_p$.
- $\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*), \mathsf{EphemeralKeyReveal}(\overline{\mathsf{sid}^*}), \mathsf{LongTermKeyLeakage}(\mathcal{A})$, $\mathsf{LongTermKeyLeakage}(\mathcal{B})$. In this sub-case, suppose that the adversary obtains at most λ_1-bits of the long-term secret key of party $\mathcal{A}$, we have that

$$\widehat{lsk}^*_{\mathcal{A}} = \mathsf{Ext}_1(lsk_{\mathcal{A}}, r_{\mathcal{A}_1}) \overset{s}{\equiv}_{\epsilon_1} \widehat{lsk}'_{\mathcal{A}} \overset{\$}{\leftarrow} \{0,1\}^{l_1(k)}, \tag{3}$$

  hence $(w^*_{\mathcal{A}}, x^*) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{A}}}(esk^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r_{\mathcal{A}}) \overset{c}{\equiv} (w'_{\mathcal{A}}, x') \overset{\$}{\leftarrow} \mathcal{W} \times \mathbb{Z}_p$. Similarly, suppose that the adversary obtains at most λ_1-bits of the long-term secret key of party $\mathcal{B}$, we have that

$$\widehat{lsk}^*_{\mathcal{B}} = \mathsf{Ext}_1(lsk_{\mathcal{B}}, r_{\mathcal{B}_1}) \overset{s}{\equiv}_{\epsilon_1} \widehat{lsk}'_{\mathcal{B}} \overset{\$}{\leftarrow} \{0,1\}^{l_1(k)}, \tag{4}$$

  and therefore $(w^*_{\mathcal{B}}, y^*) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{B}}}(esk^*_{\mathcal{B}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{B}}}(r_{\mathcal{B}_1}) \overset{c}{\equiv} (w'_{\mathcal{B}}, y') \overset{\$}{\leftarrow} \mathcal{W} \times \mathbb{Z}_p$.
- $\mathsf{LongTermKeyReveal}(\mathcal{A}), \mathsf{EphemeralKeyReveal}(\overline{\mathsf{sid}^*}), \mathsf{EphemeralKeyLeakage}(\mathsf{sid}^*)$, $\mathsf{LongTermKeyLeakage}(\mathcal{B})$. In this sub-case, suppose that the adversary obtains at most λ_2-bits of the ephemeral secret key of target session sid^*, at most λ_1-bits of the long-term secret key of party $\mathcal{B}$, then based on the Equation (1),(4), we have that $(w^*_{\mathcal{A}}, x^*) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{A}}}(esk^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r_{\mathcal{A}_1}) \overset{c}{\equiv} (w'_{\mathcal{A}}, x') \overset{\$}{\leftarrow} \mathcal{W} \times \mathbb{Z}_p$ and $(w^*_{\mathcal{B}}, y^*) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{B}}}(esk^*_{\mathcal{B}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{B}}}(r_{\mathcal{B}_1}) \overset{c}{\equiv} (w'_{\mathcal{B}}, y') \overset{\$}{\leftarrow} \mathcal{W} \times \mathbb{Z}_p$.
- $\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*), \mathsf{LongTermKeyReveal}(\mathcal{B}), \mathsf{LongTermKeyLeakage}(\mathcal{A})$, $\mathsf{EphemeralKeyLeakage}(\overline{\mathsf{sid}^*})$. In this sub-case, suppose that the adversary obtains at most λ_1-bits of the long-term secret key of party $\mathcal{A}$, at most λ_2-bits of the ephemeral secret key of matching session $\overline{\mathsf{sid}^*}$, then based on Equation (2),(3), we have that $(w^*_{\mathcal{A}}, x^*) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{A}}}(esk^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r_{\mathcal{A}_1}) \overset{c}{\equiv} (w'_{\mathcal{A}}, x') \overset{\$}{\leftarrow} \mathcal{W} \times \mathbb{Z}_p$ and $(w^*_{\mathcal{B}}, y^*) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{B}}}(esk^*_{\mathcal{B}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{B}}}(r_{\mathcal{B}_1}) \overset{c}{\equiv} (w'_{\mathcal{B}}, y') \overset{\$}{\leftarrow} \mathcal{W} \times \mathbb{Z}_p$.

Therefore, regardless of the type of the reveal query and leakage query, (x^*, y^*) are uniformly random elements in $\mathbb{Z}_p^2$ from the view of adversary $\mathcal{M}$. Therefore, $K^*_{\mathcal{A}_1} = K^*_{\mathcal{B}_1} = g^{x^*y^*}$ is computationally indistinguishable from a random element in $\mathbb{G}$ according to the DDH assumption and hence $H_2(K^*_{\mathcal{A}_1})$ is a uniform random string from the view of $\mathcal{M}$ who is given $X^* = g^{x^*}, Y^* = g^{y^*}$. We then have that the seed $s^*_{\mathcal{A}}$ for the πPRF function is uniformly distributed and unknown to the adversary and thus the derived session key $SK^*_{\mathcal{A}}$ is computationally indistinguishable from a random string. It is worth noting that in this case we only require $\tilde{F}$ to be a normal PRF.

Case II. *There exists no matching session of the test session* sid^*.

In this case, the adversary cannot issues $\mathsf{LongTermKeyReveal}$ query to reveal the long-term secret key of $\mathcal{B}$ but may issues the leakage query $\mathsf{LongTermKeyLeakage}$ to learn some bit-information of $lsk_{\mathcal{B}}$. We prove the security of the AKE protocol as follows.

In the simulation, we modify the security game via the following steps to obtain two new games.

- Game 1: Replace $K^*_{\mathcal{A}_2} = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, lpk_{\mathcal{B}}, W^*_{\mathcal{A}}, w^*_{\mathcal{A}}, aux^*)$ by $K^*_{\mathcal{A}_2} = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, lsk_{\mathcal{B}}, W^*_{\mathcal{A}}, aux^*)$.
- Game 2: Choose $W^*_{\mathcal{A}} \in \mathcal{X} \setminus \mathcal{L}$ instead of deriving it from $\mathcal{L}$ through the algorithm WordG.

We can see that Game 1 is identical to the original game from the view of adversary $\mathcal{M}$ due to the fact that $\mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, lpk_{\mathcal{B}}, W^*_{\mathcal{A}}, w^*_{\mathcal{A}}) = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, lsk_{\mathcal{B}}, W^*_{\mathcal{A}})$, and Game 2 is indistinguishable from Game 1 (and hence also the original game) due to the difficulty of the subset membership problem which ensures that the distribution of $\mathcal{X} \setminus \mathcal{L}$ is indistinguishable from $\mathcal{L}$.

Note that adversary $\mathcal{M}$ may actives a session sid, which is not matching to session sid^*, with $\mathcal{B}$. Precisely, $\mathcal{M}$ can choose $W \in \mathcal{X} \setminus \mathcal{L}$ (e.g., by replaying $W^*_{\mathcal{A}}$), send W to $\mathcal{B}$ and issues $\mathsf{SessionKeyReveal}(\mathsf{sid})$ query to learn the shared key. According to the property of 2-smooth of the underlying smooth projective hash function, we have that $K^*_{\mathcal{A}_2}$ is pairwisely independent from any other such key (denoted by $\widetilde{K}$) and all public information (i.e., $\mathsf{param}, \mathcal{L}, lpk_{\mathcal{B}}, W^*_{\mathcal{A}}, aux^*$) and hence

$$\widetilde{\mathrm{H}}_\infty(K^*_{\mathcal{A}_2} | \widetilde{K}, \mathsf{param}, \mathcal{L}, lpk_{\mathcal{B}}, W^*_{\mathcal{A}}, aux^*) = |\mathcal{Y}|.$$

Suppose that the leakage of $lsk_{\mathcal{B}}$ is at most λ_1-bits (denoted by $\widetilde{lsk_{\mathcal{B}}}$), and therefore (see *Lemma 1*)

$$\begin{aligned}\widetilde{\mathrm{H}}_\infty(K^*_{\mathcal{A}_2} | \widetilde{K}, \mathsf{param}, \mathcal{L}, lpk_{\mathcal{B}}, W^*_{\mathcal{A}}, aux^*, \widetilde{lsk_{\mathcal{B}}}) &\geq \widetilde{\mathrm{H}}_\infty(K^*_{\mathcal{A}_2} | \widetilde{K}, \mathsf{param}, \mathcal{L}, lpk_{\mathcal{B}}, W^*_{\mathcal{A}}, aux^*) - \lambda_1 \\ &= |\mathcal{Y}| - \lambda_1.\end{aligned}$$

Therefore, by using the strong extractor Ext_3, it holds that

$$s^*_{\mathcal{A}} = \mathsf{Ext}_3(H_2(K_{\mathcal{A}_1})^* \oplus K^*_{\mathcal{A}_2} \oplus K^*_{\mathcal{A}_3}, t^*_{\mathcal{A}} \oplus t^*_{\mathcal{B}}) \overset{s}{\equiv}_{\epsilon_3} s'_{\mathcal{A}} \overset{\$}{\leftarrow} \{0,1\}^{l_3(k)}.$$

One can see that $\mathcal{A}$ obtains a variable $s^*_{\mathcal{A}}$ which is pairwisely independent from any other such variables and thus the derived session key $SK^*_{\mathcal{A}}$ is computationally indistinguishable from a truly random element from $\mathcal{M}$'s view due to the application of πPRF, which completes the proof.

Simulation for Non-test Session. Note that for the two cases above, we have to simulate the non-test session correctly with the adversary. Specifically, when adversary $\mathcal{M}$ activates a non-test session with $\mathcal{A}$ or $\mathcal{B}$, the session execution simulated should be identical to the session run by $\mathcal{A}$ or $\mathcal{B}$ from the view of $\mathcal{M}$. One can note that this can be easily guaranteed when the query $\mathsf{LongTermKeyReveal}(\mathcal{A})$ or $\mathsf{LongTermKeyReveal}(\mathcal{B})$ is issued in the game. Since we know the long-term secret key of $\mathcal{A}$ or $\mathcal{B}$, we can just select an ephemeral secret key and compute the ephemeral public key correctly by using the long-term secret key and long-term public key. Nevertheless, if the query $\mathsf{LongTermKeyReveal}(\mathcal{A})$ or $\mathsf{LongTermKeyReveal}(\mathcal{B})$ is not issued, that is, without the long-term secret key of $\mathcal{A}$ or $\mathcal{B}$, the simulation of the non-test session owned by $\mathcal{A}$ or $\mathcal{B}$ can no longer be simulated as shown above. In this case, we simulate the session as follows. Suppose that we are to simulate the session owned by $\mathcal{A}$ without knowing $lsk_{\mathcal{A}}$, we pick $(r_1, r_2) \xleftarrow{\$} \mathcal{W} \times \mathbb{Z}_p$ and then compute $W_{\mathcal{A}} = \mathsf{WordG}(\mathsf{param}, \mathcal{L}, r_1), X = g^{r_2}$. We say that the session simulated in this way can be identical to the real session from $\mathcal{M}$'s view due to the pseudo-randomness of the PRF. To be more precise, even when $\mathcal{M}$ obtains at most λ_1-bits of $lsk_{\mathcal{A}}$ through $\mathsf{LongTermKeyLeakage}(\mathcal{A})$, the variable $\widehat{lsk}_{\mathcal{A}}$, which comes from $\mathsf{Ext}_1(lsk_{\mathcal{A}}, r_{\mathcal{A}})$ and inputs to the pseudo-random function $\widehat{F}$, still remains unknown to adversary $\mathcal{M}$. Therefore, the value of $\widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(esk_{\mathcal{A}})$ is computationally indistinguishable from a random element.

5 An Instantiation from DDH Assumption

In this section, we first introduce an SPHF based on the DDH assumption and then show how to construct a CLR-eCK-secure AKE protocol based on this function.

5.1 DDH-based SPHF

In the following, we present the language we use in the instantiation of our generic CLR-eCK-secure AKE protocol. Specifically, we introduce the Diffie Hellman language $\mathcal{L}_{\mathsf{DH}}$ and show how to construct a 2-smooth SPHF on $\mathcal{L}_{\mathsf{DH}}$.

Diffie-Hellman Language. Let $\mathbb{G}$ be a group of primer order p and $g_1, g_2 \in \mathbb{G}$. The Diffie-Hellman Language is as follows.

$$\mathcal{L}_{\mathsf{DH}} = \{(u_1, u_2) | \exists r \in \mathbb{Z}_p, s.t., u_1 = g_1^r, u_2 = g_2^r\}$$

One can see that the witness space of $\mathcal{L}_{\mathsf{DH}}$ is $\mathcal{W} = \mathbb{Z}_p$ and $\mathcal{L}_{\mathsf{DH}} \subset \mathcal{X} = \mathbb{G}^2$. We have the following theorems.

Theorem 2. *The subset membership problem over $\mathcal{X} = \mathbb{G}^2$ and $\mathcal{L}_{\mathsf{DH}}$ is hard.*

Proof. One can easily obtain the theorem above from the DDH assumption and hence we omit the proof here. Actually, if an adversary can distinguish a word randomly picked from $\mathcal{L}_{\mathsf{DH}}$ from a random element chosen from $\mathcal{X} \backslash \mathcal{L}_{\mathsf{DH}}$, we can build a distinguisher for the DDH problem by using the adversary as a subroutine.

SPHF on $\mathcal{L}_{\mathsf{DH}}$. Here we show how to construct a 2-smooth SPHF (denoted by $\mathcal{SPHF}_{\mathsf{DH}}$) over the language $\mathcal{L}_{\mathsf{DH}} \subset \mathcal{X} = \mathbb{G}^2$ onto the group $\mathcal{Y} = \mathbb{G}$. Let $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ denote a collision-resistant hash function. The concrete construction is as follows.

- $\mathsf{SPHFSetup}(1^\lambda)$: $\mathsf{param} = (\mathbb{G}, p, g_1, g_2)$;
- $\mathsf{HashKG}(\mathcal{L}_{\mathsf{DH}}, \mathsf{param})$: $\mathsf{hk} = (\alpha_1, \alpha_2, \beta_1, \beta_2) \xleftarrow{\$} \mathbb{Z}_p^4$;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}_{\mathsf{DH}}, \mathsf{param}))$: $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2) = (g_1^{\alpha_1} g_2^{\alpha_2}, g_1^{\beta_1} g_2^{\beta_2}) \in \mathbb{G}_p^2$;
- $\mathsf{WordG}(\mathsf{hk}, (\mathcal{L}_{\mathsf{DH}}, \mathsf{param}), w = r)$: $W = (g_1^r, g_2^r)$;
- $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}_{\mathsf{DH}}, \mathsf{param}), W = (u_1, u_2) = (g_1^r, g_2^r), aux = d = H_1(W, aux'))$: $\mathsf{hv} = u_1^{\alpha_1 + d\beta_1} u_2^{\alpha_2 + d\beta_2}$;
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}_{\mathsf{DH}}, \mathsf{param}), W = (u_1, u_2) = (g_1^r, g_2^r), w = r, aux = d = H_1(W, aux'))$: $\mathsf{hv}' = \mathsf{hp}_1^r \mathsf{hp}_2^{dr}$.

Note that $\mathcal{Y} = \mathbb{G}, \mathcal{HK} = \mathbb{Z}_p^4, \mathcal{HP} = \mathbb{G}_p^2, \mathcal{AUX} = \mathbb{Z}_p, \mathcal{W} = \mathbb{Z}_p$. Then we have the following theorem.

Theorem 3. *$\mathcal{SPHF}_{\mathsf{DH}}$ is a 2-*smooth *SPHF.*

Proof. We show that $\mathcal{SPHF}_{\mathsf{DH}}$ is projective and smooth (2-smooth).

- *Correctness*. With the above notations, for a word $W = (u_1, u_2) = (g_1^r, g_2^r)$ we have

  $$\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}_{\mathsf{DH}}, \mathsf{param}), W, d) = u_1^{\alpha_1 + d\beta_1} u_2^{\alpha_2 + d\beta_2} = \mathsf{hp}_1^r \mathsf{hp}_2^{dr} = \mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}_{\mathsf{DH}}, \mathsf{param}), W, r, d).$$

- *Smoothness* (2-smooth). Suppose $g_2 = g_1^\theta$. Note that $\mathsf{hp}_1 = g_1^{\alpha_1} g_2^{\alpha}, \mathsf{hp}_2 = g_1^{\beta_1} g_2^{\beta_2}$ which constraints $(\alpha_1, \alpha_2, \beta_1, \beta_2)$ to satisfy

  $$\log_{g_1} \mathsf{hp}_1 = \alpha_1 + \theta\alpha_2. \tag{5}$$

  $$\log_{g_1} \mathsf{hp}_2 = \beta_1 + \theta\beta_2. \tag{6}$$

  Let $W_1 = (g_1^{r_1}, g_2^{r_2}), W_2 = (g_1^{r_1'}, g_2^{r_2'}) \in \mathcal{X} \backslash \mathcal{L}_{\mathsf{DH}}$ where $r_1 \neq r_2, r_1' \neq r_2'$, suppose $aux_1 = d_1 = H_1(W_1, aux_1'), aux_2 = d_2 = H_1(W_2, aux_2')$, then the hash value hv_1 of W_1, hv_2 of W_2 are as follows,

  $$\mathsf{hv}_1 = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}_{\mathsf{DH}}, \mathsf{param}), W_1, aux_1) = g_1^{r_1(\alpha_1 + d_1\beta_1)} g_2^{r_2(\alpha_2 + d_1\beta_2)},$$

  $$\mathsf{hv}_2 = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}_{\mathsf{DH}}, \mathsf{param}), W_2, aux_2) = g_1^{r_1'(\alpha_1 + d_2\beta_1)} g_2^{r_2'(\alpha_2 + d_2\beta_2)},$$

  which also constraint $(\alpha_1, \alpha_2, \beta_1, \beta_2)$ to satisfy

  $$\log_{g_1} \mathsf{hv}_1 = r_1\alpha_1 + r_2\theta\alpha_2 + r_1 d_1 \beta_1 + r_2 d_1 \theta\beta_2. \tag{7}$$

  $$\log_{g_1} \mathsf{hv}_2 = r_1'\alpha_1 + r_2'\theta\alpha_2 + r_1' d_2 \beta_1 + r_2' d_2 \theta\beta_2. \tag{8}$$

  From the above equations, we have

  $$(\alpha_1, \alpha_2, \beta_1, \beta_2) \cdot \mathbf{A} = (\log_{g_1} \mathsf{hp}_1, \log_{g_1} \mathsf{hp}_2, \log_{g_1} \mathsf{hv}_1, \log_{g_1} \mathsf{hv}_2),$$

  where $\mathbf{A}$ is a matrix defined as

  $$\mathbf{A} = \begin{bmatrix} 1 & \theta & 0 & 0 \\ 0 & 0 & 1 & \theta \\ r_1 & \theta r_2 & r_1 d_1 & \theta r_2 d_1 \\ r_1' & \theta r_2' & r_1' d_2 & \theta r_2' d_2 \end{bmatrix}.$$

  Since $(W_1, aux_1) \neq (W_2, aux_2)$ where $aux_1 = d_1 = H_1(W_1, aux_1'), aux_2 = d_2 = H_1(W_2, aux_2')$, we have that $d_1 \neq d_2$. Furthermore, as $\theta \neq 0$, $r_1 \neq r_2$ and $r_1' \neq r_2'$, we can obtain that the determinant of $\mathbf{A}$ is $\theta^2 \cdot (r_2 - r_1) \cdot (r_2' - r_1') \cdot (d_2 - d_1) \neq 0$ and hence the equation (8) is independent of the equation (7). Therefore, we have that hv_2 is perfectly indistinguishable from any element randomly chosen from $\mathbb{G}$.

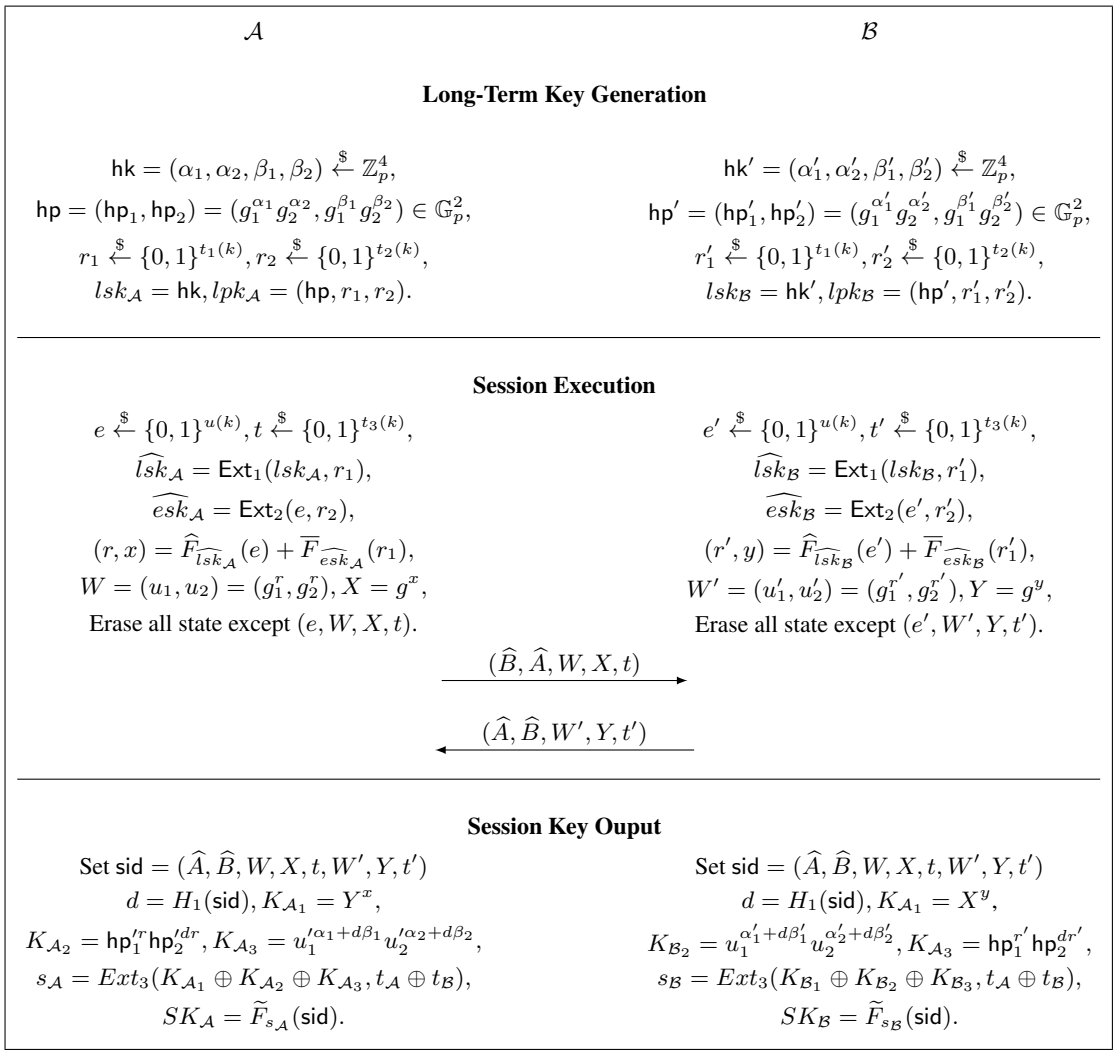

Fig. 2. CLR-eCK secure AKE Protocol

5.2 Concrete AKE Protocol

We then show a concrete AKE protocol based on $\mathcal{SPHF}_{\mathsf{DH}}$ in Fig. 2.

Protocol Description. In the system setup phase, let $\mathbb{G}$ be a group of primer order p and $g_1, g_2 \in \mathbb{G}$. For the $\mathcal{SPHF}_{\mathsf{DH}}$, we have that $\mathcal{Y} = \mathbb{G}, \mathcal{HK} = \mathbb{Z}_p^4, \mathcal{HP} = \mathbb{G}_p^2, \mathcal{AUX} = \mathbb{Z}_p, \mathcal{W} = \mathbb{Z}_p$. We then choose a collision-resistant hash functions $H_1 : \{0,1\}^* \to \mathbb{G}$. [4] We pick strong extractors as follows. Let $\mathsf{Ext}_1 : \mathbb{Z}_p^4 \times \{0,1\}^{t_1(k)} \to \{0,1\}^{l_1(k)}$ be average-case $(4 \cdot \log p - \lambda_1, \epsilon_1)$-strong extractor, $\mathsf{Ext}_2 : \{0,1\}^{u(k)} \times \{0,1\}^{t_2(k)} \to \{0,1\}^{l_2(k)}$ be average-case $(u(k) - \lambda_2, \epsilon_2)$-strong extractor and $\mathsf{Ext}_3 : \mathbb{G} \times \{0,1\}^{t_3(k)} \to \{0,1\}^{l_3(k)}$ be average-case $(\log p - \lambda_1, \epsilon_3)$-strong extractor. Choose $\widehat{F} \leftarrow \widehat{\mathsf{F}}^{k, \sum_{\widehat{\mathsf{F}}}, \mathcal{D}_{\widehat{\mathsf{F}}}, \mathcal{R}_{\widehat{\mathsf{F}}}}$, $\overline{F} \leftarrow \overline{\mathsf{F}}^{k, \sum_{\overline{\mathsf{F}}}, \mathcal{D}_{\overline{\mathsf{F}}}, \mathcal{R}_{\overline{\mathsf{F}}}}$ and $\widetilde{F} \leftarrow \widetilde{\mathsf{F}}^{k, \sum_{\widetilde{\mathsf{F}}}, \mathcal{D}_{\widetilde{\mathsf{F}}}, \mathcal{R}_{\widetilde{\mathsf{F}}}}$. The system parameter is $(\mathbb{G}, p, g_1, g_2, g, H_1, \mathsf{Ext}_1, \mathsf{Ext}_2, \mathsf{Ext}_3, \widehat{F}, \overline{F}, \widetilde{F})$.

For the long-term key generation, $\mathcal{A}$ chooses $(\alpha_1, \alpha_2, \beta_1, \beta_2) \xleftarrow{\$} \mathbb{Z}_p^4$ as its long-term secret key, computes $(\mathsf{hp}_1, \mathsf{hp}_2) = (g_1^{\alpha_1} g_2^{\alpha_2}, g_1^{\beta_1} g_2^{\beta_2})$, picks $r_1 \xleftarrow{\$} \{0,1\}^{t_1(k)}, r_2 \xleftarrow{\$} \{0,1\}^{t_2(k)}$ and sets

[4] Note that in the concrete construction, H_2 is not needed as the hash value space $\mathcal{Y} = \mathbb{G}$.

its long-term public key as $(\mathsf{hp}_1, \mathsf{hp}_2, r_1, r_2)$. Similarly, $\mathcal{B}$ sets its long-term secret/public key pair as $((\alpha'_1, \alpha'_2, \beta'_1, \beta'_2), (\mathsf{hp}'_1, \mathsf{hp}'_2, r'_1, r'_2))$.

After a session is activated, $\mathcal{A}$ picks an ephemeral secret key e and the extraction key $t \xleftarrow{\$} \{0,1\}^{t_3(k)}$, derives (r, x) using the secret keys and sends $(\widehat{B}, \widehat{A}, W = (u_1, u_2) = (g_1^r, g_2^r), X = g^x, t)$ to $\mathcal{B}$. Simultaneously, $\mathcal{B}$ executes the same procedure and returns $(\widehat{A}, \widehat{B}, W' = (u'_1, u'_2) = (g_1^{r'}, g_2^{r'}), Y = g^y, t')$ to $\mathcal{A}$.

To compute the shared session key, $\mathcal{A}$ runs the $\mathsf{ProjHash}$ algorithm to compute the hash value of W using the witness r and the long-term public key of $\mathcal{B}$, runs the Hash algorithm to compute the hash value of W' using its long-term secret key. $\mathcal{B}$ runs the Hash algorithm to compute the hash value of W using its long-term secret key, runs the Hash algorithm to compute the hash value of W' using the witness r' and the long-term public key of $\mathcal{A}$. Note that the auxiliary input to all the hash value computation is $d = H_1(\widehat{A}, \widehat{B}, W, X, t, W', Y, t')$. Both $\mathcal{A}$ and $\mathcal{B}$ also compute the value of g^{xy}. They then finally apply the πPRF function $\widetilde{F}$ to derive the session key.

Correctness. The correctness of the protocol can be easily obtained from the correctness of $\mathcal{SPHF}_{\mathsf{DH}}$. Precisely, $u_1^{\alpha'_1+d\beta'_1} u_2^{\alpha'_2+d\beta'_2} = \mathsf{hp}_1'^{r} \mathsf{hp}_2'^{dr}$, $u_1'^{\alpha_1+d\beta_1} u_2'^{\alpha_2+d\beta_2} = \mathsf{hp}_1^{r'} \mathsf{hp}_2^{dr'}$, $X^y = Y^x = g^{xy}$.

Based on **Theorem 1**, **Theorem 2** and **Theorem 3**, we have the following result for the concrete AKE protocol.

Theorem 4. *The concrete AKE protocol is* (λ_1, λ_2)-CLR-eCK-*secure, where* $\lambda_1 \leq \min\{4\log p - 2\log(1/\epsilon_1) - l_1(k), \log p - 2\log(1/\epsilon_3) - l_3(k)\}$, $\lambda_2 \leq u(k) - 2\log(1/\epsilon_2) - l_2(k)$.

6 Conclusion

In this paper, we introduced a new leakage-resilient security model for AKE protocols to overcome the limitations in the previous models. Our model is the first to allow the adversary to obtain challenge-dependent leakage on both long-term and ephemeral secret keys, and hence are strong yet meaningful compared with the previous models. We also presented a generic framework to construct efficient one-round AKE protocol that is secure under the proposed security model, as well as an efficient instantiation of the general framework under the DDH assumption. Our framework ensures the session key are private and authentic even if the adversary learns a large fraction of both the long-term secret key and ephemeral secret key and provides qualitatively stronger privacy guarantees than existing AKE protocols constructed in prior and concurrent works, since such protocols necessarily become insecure if the adversary can perform leakage attacks during the execution of session.

Acknowledgements. We would like to thank Janaka Alawatugoda and the anonymous reviewers for their invaluable comments on a previous version of this paper. The work of Yi Mu is supported by the National Natural Science Foundation of China (Grant No. 61170298). The work of Guomin Yang is supported by the Australian Research Council Discovery Early Career Researcher Award (Grant No. DE150101116) and the National Natural Science Foundation of China (Grant No. 61472308).

References

1. Entity authentication mechanisms-part3: Entity authentication using asymmetric techniques. ISO/IEC IS 9789-3, 1993.
2. Abdalla, M., Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D.: Sphf-friendly non-interactive commitments. In: ASIACRYPT. pp. 214–234 (2013)
3. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: TCC. pp. 474–495 (2009)
4. Alawatugoda, J., Boyd, C., Stebila, D.: Continuous after-the-fact leakage-resilient key exchange. In: ACISP. pp. 258–273 (2014)
5. Alawatugoda, J., Stebila, D., Boyd, C.: Modelling after-the-fact leakage for key exchange. In: ASIACCS. pp. 207–216 (2014)
6. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: CRYPTO. pp. 36–54 (2009)
7. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In: ACM Symposium on the Theory of Computing. pp. 419–428 (1998)
8. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: CRYPTO. pp. 232–249 (1993)
9. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for sphfs and efficient one-round PAKE protocols. In: CRYPTO. pp. 449–475 (2013)
10. Bitansky, N., Canetti, R., Halevi, S.: Leakage-tolerant interactive protocols. In: TCC. pp. 266–284 (2012)
11. Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. J. Cryptology 26(3), 513–558 (2013)
12. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: EUROCRYPT. pp. 453–474 (2001)
13. Choo, K.R., Boyd, C., Hitchcock, Y.: Examining indistinguishability-based proof models for key establishment protocols. In: ASIACRYPT. pp. 585–604 (2005)
14. Chow, S.S.M., Dodis, Y., Rouselakis, Y., Waters, B.: Practical leakage-resilient identity-based encryption from simple assumptions. In: CCS. pp. 152–161 (2010)
15. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: EUROCRYPT. pp. 45–64 (2002)
16. Cremers, C.: Examining indistinguishability-based security models for key exchange protocols: the case of ck, ck-hmqv, and eck. In: ASIACCS, 2011. pp. 80–91 (2011)
17. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: ASIACRYPT. pp. 613–631 (2010)
18. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: STOC. pp. 621–630 (2009)
19. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. 38(1), 97–139 (2008)
20. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: EUROCRYPT. pp. 524–543 (2003)
21. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986)
22. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: USENIX Security Symposium. pp. 45–60 (2008)
23. Halevi, S., Kalai, Y.T.: Smooth projective hashing and two-message oblivious transfer. J. Cryptology 25(1), 158–193 (2012)
24. Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: TCC. pp. 107–124 (2011)
25. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: ASIACRYPT. pp. 703–720 (2009)
26. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: TCC. pp. 293–310 (2011)
27. Krawczyk, H.: SIGMA: the 'sign-and-mac' approach to authenticated diffie-hellman and its use in the ike-protocols. In: CRYPTO. pp. 400–425 (2003)
28. LaMacchia, B.A., Lauter, K.E., Mityagin, A.: Stronger security of authenticated key exchange. In: Provable Security. pp. 1–16 (2007)
29. Marvin, R.: Google admits an android crypto prng flaw led to bitcoin heist (august 2013). `http://sdt.bz/64008`.
30. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: TCC. pp. 278–296 (2004)
31. Moriyama, D., Okamoto, T.: Leakage resilient eck-secure key exchange protocol without random oracles. In: ASIACCS
32. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: CRYPTO. pp. 18–35 (2009)
33. Okamoto, T.: Authenticated key exchange and key encapsulation in the standard model. In: ASIACRYPT. pp. 474–484 (2007)
34. Shumow, D., Ferguson, N.: On the possibility of a back door in the nist sp800-90 dual ec prng. `http://rump2007.cr.yp.to/15-shumow.pdf`.
35. Yang, G., Mu, Y., Susilo, W., Wong, D.S.: Leakage resilient authenticated key exchange secure in the auxiliary input model. In: ISPEC. pp. 204–217 (2013)

36. Yu, Y., Standaert, F., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: CCS. pp. 141–151 (2010)
37. Yuen, T.H., Zhang, Y., Yiu, S., Liu, J.K.: Identity-based encryption with post-challenge auxiliary inputs for secure cloud applications and sensor networks. In: ESORICS. pp. 130–147 (2014)
38. Zetter, K.: How a crypto 'backdoor' pitted the tech world against the nsa. http://www.wired.com/threatlevel/2013/09/nsa-backdoor/all/.