# Authenticated Key Exchange Resilient to Key Exposure and Weak Randomness [⋆]

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

[1]College of Computer
National University of Defense Technology, China
chromao@nudt.edu.cn
[2]Institute of Cryptology and Cybersecurity
School of Computing and Information Technology
University of Wollongong, Australia
{gyang,ymu,wsusilo,fuchun}@uow.edu.au

**Abstract.** Authenticated Key Exchange (AKE) protocols allow two parties to establish a secure communication channel over a public network. They form a central component in many network security standards such as IPSec, TLS/SSL, and SSH. In order to defend against side-channel (or in general, key leakage) attacks, leakage resilient (LR-) AKE protocols have been proposed in the literature. However, all the existing LR-AKE protocols only focused on the resistance to long-term key leakage while in reality, leakage of ephemeral secret key (or randomness) can also occur due to various reasons such as side-channel attacks, the use of poor randomness sources or insecure pseudo-random number generators (PRNGs). In this paper, we present a general framework for building secure AKE protocols that are resilient to both key leakage/exposure and weak randomness. We also show that our new framework can be efficiently instantiated under various standard assumptions with low computation and communication overhead.

**Keywords:** Authenticated key exchange, key exposure, key leakage, weak randomness.

## 1 Introduction

Key distribution is one of the fundamental security problems in network security. In order to protect the communications between two (or multiple) network entities using cryptographic methods, a shared secret key needs to be established among those parties. Since Diffie and Hellman [17] introduced their seminal key exchange protocol, which is also the first public key cryptosystem, many authenticated key exchange (AKE) protocols have been proposed and deployed in various network security standards such as IPSec, SSL/TLS, SSH, etc.

In the design of a typical AKE protocol, such as the SIG-DH protocol (i.e., ISO/IEC IS 9789-3) [1, 12] and the Internet Key Exchange (a.k.a. SIGMA) protocol [30], it is usually considered that a user holds a *long-term secret key* and generates a fresh *ephemeral secret key* in each AKE session. The long-term key is for authentication purpose while the ephemeral key ensures the freshness and uniqueness of each session key. In the design of many traditional AKE protocols, it is usually assumed that the long-term secret key is unreachable by active adversaries while the ephemeral secret key used in a session is hidden from passive adversaries that however may obtain the long-term secret key (i.e., forward secrecy). However, the advances of various side-channel [9, 22, 29, 37] and memory attacks [24, 3] voided the aforementioned assumptions underlying the traditional AKE designs. In such attacks, the adversary is able to learn some imperfect information of the user secrets when they are stored in memory and/or used during computation. We should note that the secret leakage we considered in this paper is

[⋆] A preliminary version of this paper appears in the proceedings of CT-RSA 2016 [13].

different from the exposure of an entire long-term or ephemeral secret key that has been considered in some existing AKE security models (e.g., CK [12] and eCK [31]). In particular, we are interested in the situation that one of those keys is completely revealed to the adversary while the other key is partially leaked.

## 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 executing a cryptographic algorithm, 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 leakage function $f$, and then receives $f(sk)$ where $sk$ is the user secret key. It is obvious that some restrictions on the leakage function $f$ is necessary (e.g., $f$ cannot be the identity function) in order to make the security notion meaningful and achievable. Hence, one of major challenges is to define the leakage function with necessary but minimal restrictions in order to capture as many key leakage attacks as possible.

**Limitations in Existing Leakage-Resilient AKE.** There have been a few works on the modeling and construction of leakage-resilient AKE [5, 18, 34, 4]. However, we found that there are some limitations in the existing works.

STRONG RESTRICTIONS ON LEAKAGE QUERIES. The *de facto* security definition of AKE requires that the real session key of the challenge AKE session (i.e., target session chosen by the adversary) is indistinguishable from a randomly chosen key even when the adversary has eavesdropped or intervened in the execution of the challenge session. However, such a security definition brings a problem under the leakage setting. During the execution of the challenge session, the adversary can make use of the leakage oracle by encoding the available information of the challenge session into a leakage function and obtain partial information of the real session key. This would allow the adversary to trivially win in the adversarial game defined in the security model. The previous security definitions for leakage-resilient AKE, e.g., [5, 18, 34, 39], bypassed such a problem by considering only *challenge-independent leakage*. Namely, the adversary *cannot make any leakage query that involves a leakage function $f$ related to the challenge session*. Specifically, in those models, the adversary is disallowed to make any leakage query during (or after in [34]) the execution of the challenge session. This approach indeed bypasses the problem, but it also puts some strong restrictions on the adversary's capability in making leakage queries.

It is worth noting that inspired by the work in [26], Alawatugoda et al. [4] modeled after-the-fact leakage for AKE protocols. Their proposed model, named bounded after-the-fact leakage eCK model (BAFL-eCK), captures leakage of the long-term secret key during and after the challenge session. However, the BAFL-eCK model has implicitly assumed that the long-term secret has split-state since otherwise their security notion is unachievable.

INCOMPLETE LEAKAGE COVERAGE. The leakage-resilient security notions for AKE proposed in the literature only focused on leakage of the long-term secret key. However, the leakage of the ephemeral secret key (or randomness) could also happen in practice due to various reasons. First of all, the key leakage attacks (e.g., various kinds of side-channel attacks) against the long-term secret key can also be performed against the ephemeral secret key. Moreover, the ephemeral key leakage could also happen due to other reasons such as the use of a poor randomness source or an insecure pseudo-random number generator (PRNG) that produces weak or predictable random coins [32, 38, 41].

We should note that the problem of randomness leakage has been considered in prior works on leakage-resilient encryption and signature schemes. For example, Halevi and Lin mentioned in [26] that "another interesting question is to handle leakage from the encryption randomness, not just the secret key", which was later addressed in [10, 40]. In terms of signature schemes, the notion of fully leakage-resilient signature was proposed by Katz and Vaikuntanathan [27] where the adversary is allowed to obtain leakage of the state information of the signing algorithm, including the secret key and internal random coins. However, to date there is no formal treatment for the randomness leakage in AKE protocols.

**Our Goals.** In this work, we are interested in addressing the following two questions: *(1) how to define a strong yet meaningful leakage-resilient AKE security model capturing both long-term and ephemeral secret key leakage*, and *(2) how to construct efficient AKE protocols that can achieve the proposed security notion.*

## 1.2 Related Work

**Traditional AKE Security Notions.** Bellare and Rogaway (BR) [7] introduced the first formal security notion for AKE which captured the requirements that a secure session key must be indistinguishable from a randomly chosen key and session keys established in multiple AKE sessions must be independent. The BR model and its extensions are nowadays the *de facto* security notion for AKE. In particular, the Canetti-Krawczyk (CK) model [12], which can be regarded as the extension and combination of the BR model and the Bellare-Canetti-Krawczyk (BCK) model [6], has been used to prove the security of some widely used AKE protocols such as SIG-DH and IKE. Subsequently, an extension of the CK model, named eCK model, was introduced by LaMacchia et al. [31] to consider a stronger (in certain aspects) adversary that is allowed to entirely reveal either the long-term secret key or the ephemeral secret key of the challenge session. We refer the readers to [14, 16] for some detailed analysis and comparisons among these models.

**Modelling Leakage Resilience.** The modeling of leakage attacks in an abstract way was first proposed by Micali and Reyzin [33]. Inspired by the cold boot attack presented by Halderman et al. [24], Akavia et al. [3] formalized a general framework, named *Relative Leakage Model*, which considered the situation that the adversary is able to obtain a fraction of a user secret key. The *Bounded-Retrieval Model* (BRM) proposed by Alwen et al. [5] is a generalization of the relative leakage model, in which the leakage-parameter forms an independent parameter of the system. Another relatively stronger leakage model is the *Auxiliary Input Model* proposed by Dodis et al. [19] where the leakage function is essentially any one-way function (i.e., there is no bound on the leakage size).

**Leakage-Resilient AKE.** Alwen, Dodis and Wichs [5] presented an efficient leakage-resilient AKE protocol in the random oracle model. They considered a leakage-resilient extension of the CK model under the BRM setting and showed that a leakage-resilient AKE protocol can be constructed from an entropically-unforgeable digital signature scheme secure under chose-message attacks. In [18], Dodis et al. also proposed leakage-resilient AKE constructions based on different primitives. Their first construction is similar as [5], i.e., authenticating Diffie-Hellman (DH) key exchange using a leakage-resilient signature scheme, whereas the second construction is based on a leakage-resilient CCA-secure PKE scheme. Based on the eCK model, Moriyama and Okamoto [34] proposed another leakage-resilient model for AKE. Their proposed notion, named $\lambda$-leakage resilient eCK where $\lambda$ denotes the leakage size, is an extension of the eCK model by incorporating the relative leakage introduced in [3]. They also presented a 2-round AKE protocol that is secure under their proposed notion. Yang et al. [39] studied leakage resilient AKE in the auxiliary input model. They showed that in the random oracle model, an AKE protocol secure under the auxiliary input leakage can be built based on a digital signature scheme that is random message unforgeable under random message and auxiliary input attacks (RU-RMAA) [21]. We should note that all the aforementioned works on leakage resilient AKE have the limitations we outlined in Section 1.1.

### 1.3 Our Results and Techniques

In this paper, we address the limitations in the previous works by first introducing a new AKE security model, named challenge-dependent leakage-resilient eCK (CLR-eCK), to capture the challenge-dependent leakage of both long-term and ephemeral secret keys; we then present a general framework for constructing CLR-eCK-secure AKE that can be efficiently instantiated under different assumptions.

**Overview of Our Model.** As shown in Table 1, our proposed model is the first model that captures challenge-dependent leakage without requiring a secret to have split-state; it is also the first model to consider both long-term and ephemeral secret key leakage. Our model is an extension of the eCK model [31] under the relative leakage setting [3]. In our proposed CLR-eCK model, the adversary can make both leakage and key reveal queries for the long-term and ephemeral secret keys of the challenge session. 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 model is stronger than all the previous AKE models considering secret key exposure/leakage.

Specifically, we present a new approach to address the problem that the adversary can trivially win the session key security game by encoding the information of the challenge session into a leakage function and subsequently obtaining information of the session key in a leakage query. In our model, we allow the adversary to entirely reveal one of the (long-term and ephemeral) secret keys used in the challenge session and obtain a fraction of the other unrevealed secret key, and the adversary is allowed to do this with only one condition: before the adversary fully reveals one of the secret keys, the adversary has to commit the set of leakage functions that will be subsequently used on the other secret key. Compared with the previous approach that disallows leakage queries during or after the challenge session, our approach gives the adversary more capacity and freedom in making leakage queries. First, before the adversary completely reveals one of the secrets, except the leakage size, there is no restriction on the leakage queries. Secondly, even when one of the secrets is revealed, the adversary is still allowed to query leakage functions in the committed set at any time. We should note that if the adversary reveals a secret during or after the challenge session, then the committed leakage functions against the other secret can be related to the challenge session. We can see that our model imposes the minimum restriction to prevent the adversary from trivially winning the security game and meanwhile captures the *challenge-dependent leakage of both long-term and ephemeral secret keys*.

**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 [5] | No | $\surd$ | $\times$ | *Bounded-Retrieval* | CK |
| LR-CK [18] | No | $\surd$ | $\times$ | *Relative Leakage* | CK |
| AI-CK [39] | No | $\surd$ | $\times$ | *Auxiliary Input* | CK |
| LR-eCK [34] | No | $\surd$ | $\times$ | *Relative Leakage* | eCK |
| BAFL-eCK [4] | Yes (with split-state) | $\surd$ | $\times$ | *Relative Leakage* | eCK |
| CLR-eCK | Yes (without split-state) | $\surd$ | $\surd$ | *Relative Leakage* | eCK |

**A Generic AKE Framework.** We present a general framework for the construction of AKE protocols secure in our proposed CLR-eCK model. In general, we apply both pseudo-random functions (PRFs) and strong randomness extractors to mix the long-term and ephemeral secret keys in the generation of the ephemeral public key in order to obtain leakage resilience for both secret keys.

Specifically, we first provide a generic framework based on an (extended) smooth projective hash function (SPHF) to construct CLR-eCK AKE under public-key setting[1]. An SPHF is associated with an input 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 and the witness of $W$. The key property of SPHF is that the projection key uniquely determines the hash value for every $W \in \mathcal{L}$ (*projective*) but gives essentially no information about the hash value for any $W' \in \mathcal{X} \setminus \mathcal{L}$ (*smooth*). During the protocol execution, both parties derive a word in $\mathcal{L}$ and its witness by mixing the long-term and ephemeral secret keys using PRFs and strong randomness extractors. They then exchange their words and compute the hash values using either their secret hashing key (for the word generated by the peer party) or the witness (for the word generated by themselves). In order to achieve forward secrecy, the two parties also run an additional key exchange by using a Key Encapsulation Mechanism (KEM) where the encryption key is freshly generated by the initiator of the protocol and the ciphertext is freshly generated by the responder of the protocol. The random coins used by both parties in the additional KEM-based key exchange are also derived using the same method as in the SPHF-based exchange. We prove that the proposed framework can achieve our desired security goal if the underlying primitives are secure.

We also show that our generic construction can be easily adapted to the symmetric-key setting (i.e., two parties share a long-term symmetric key) and can be efficiently instantiated under various standard assumptions.

## 1.4 Differences with [13]

This work is an extension of the preliminary results published in the proceedings of CT-RSA'16 [13]. In the conference version, we presented a semi-generic construction of CLR-eCK AKE under the public-key setting. Specifically, the construction given in [13] explicitly requires the Decisional Diffie-Hellman (DDH) assumption while in this work, the construction is more generic and does not rely on any specific assumption. As a result, we are able to provide instantiations of the generic construction under various assumptions. Moreover, in this work, we also ex-

[1] In this paper, we say an AKE protocol is in public/symmetric-key setting according to its long-term key.

tend our generic construction to the symmetric-key setting and present efficient post-quantum instantiations under this setting.

## 2 Preliminaries

### 2.1 Notation

For a finite set $\Omega$, $\omega \xleftarrow{\$} \Omega$ denotes that $\omega$ is selected uniformly at random from $\Omega$. For a probabilistic algorithm $A$, we use $A(x)$ to denote the invoking of $A$ with input $x$ and uniformly chosen random coins and $A(x;r)$ the invoking of $A$ with random coins $r$.

**Statistical Indistinguishability.** Let $X$ and $Y$ be two random variables over a finite domain $\Omega$, 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 $\epsilon$*-statistically indistinguishable* if $\mathsf{SD}(X,Y) \le \epsilon$ 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 $\mathcal{V}_0$ and $\mathcal{V}_1$ be two probability distributions over a finite set $\Omega$ where $|\Omega| \ge 2^k$ and $k$ is a security parameter. We define the computational indistinguishability between $\mathcal{V}_0$ and $\mathcal{V}_1$ against a distinguisher $\widetilde{\mathcal{D}}$ as follows. $\widetilde{\mathcal{D}}$ takes as input the descriptions of $\mathcal{V}_0$ and $\mathcal{V}_1$ and an element $v \xleftarrow{\$} \mathcal{V}_\gamma$ where $\gamma \xleftarrow{\$} \{0,1\}$. Finally, $\widetilde{\mathcal{D}}$ outputs a bit $\gamma' \in \{0,1\}$ as its guess on $\gamma$. We define the advantage of $\widetilde{\mathcal{D}}$ in this game as $\mathsf{Adv}_{\widetilde{\mathcal{D}}}^{\mathcal{V}_1,\mathcal{V}_2}(k) = \Pr[\gamma' = \gamma] - 1/2$. We say that $\mathcal{V}_0$ and $\mathcal{V}_1$ are *computationally indistinguishable* if for any polynomial-time distinguisher $\mathcal{D}$, $\mathsf{Adv}_{\widetilde{\mathcal{D}}}^{\mathcal{V}_0,\mathcal{V}_1}(k)$ is negligible, and we denote it by $\mathcal{V}_0 \overset{c}{\equiv} \mathcal{V}_1$.

### 2.2 Randomness Extractor

We recall the notion of an average-case strong extractor defined in [20].

**Average-Case Min-Entropy.** The *min-entropy* of a random variable $X$ is $\mathrm{H}_\infty(X) = -\log(\max_x \Pr[X = x])$. Following Dodis et al. [20], define $\widetilde{\mathrm{H}}_\infty(X|Y) = -\log(\mathrm{E}_{y\leftarrow Y}[2^{-\mathrm{H}_\infty(X|Y=y)}])$. The following result was proved by Dodis et al. in [20].

*Lemma 1*([20]). *If $Y$ has $2^\lambda$ possible values, then $\widetilde{H}_\infty(X|Y) \ge \widetilde{H}_\infty(X) - \lambda$.*

**Definition 1 (Average-Case Strong Extractor)**[20]. *Let $k \in \mathbb{N}$ be a security parameter. A function* $\mathsf{Ext} : \{0,1\}^{n(k)} \times \{0,1\}^{t(k)} \to \{0,1\}^{l(k)}$ *is said to be an average-case $(m,\epsilon)$-strong extractor if for all pairs of random variables $(X,I)$ such that $X \in \{0,1\}^{n(k)}$ and $\widetilde{H}_\infty(X|I) \ge m$, it holds that*

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

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

### 2.3 Pseudo-Random Function

Below we review the notion of a pseudo-random function (PRF) and its variant named pseudo-random function with pairwise-independent random sources ($\pi$PRF) introduced by Okamoto [35].

**PRF.** Let $k \in \mathbb{N}$ be a security parameter and $\mathsf{F}$ a function family associated with $\{\mathsf{Seed}_k\}_{k\in\mathbb{N}}$, $\{\mathsf{Dom}_k\}_{k\in\mathbb{N}}$ and $\{\mathsf{Rng}_k\}_{k\in\mathbb{N}}$. 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* $\mathsf{F}$ *is a pseudo-random function (PRF) family if*

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

*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* $\rho \in \mathcal{D}$, $RF(\rho) \xleftarrow{\$} \mathcal{R}$.

**$\pi$PRF.** 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 $i \in I_{\sum}$, outputs $\sigma_i \in Z_{\sum}$. Consider the random variables $\{\sigma_{i_j}\}_{j=0,\ldots,q(k)} = \{f_{\sum}(i_j)\}_{j=0,\ldots,q(k)}$ where $i_j \in I_{\sum}$ and $q(k)$ a polynomial function of $k$. We say that $\sigma_{i_0}$ is *pairwisely independent* from other variables $\sigma_{i_1}, \ldots, \sigma_{i_{q(k)}}$ if for any pair of $(\sigma_{i_0}, \sigma_{i_j})(j = 1, \ldots, q(k))$ and 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 ($\pi$PRF).** 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 $\mathsf{F}$ is a $\pi$PRF family if

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

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

## 2.4 Smooth Projective Hash Function

Smooth projective hash function(SPHF) was introduced by Cramer and Shoup in [15] and later extended for constructing various cryptographic primitives [23, 25, 28, 2, 8]. We start with the original definition.

**Syntax.** An SPHF is defined over a domain $\mathcal{X}$ and a subset $\mathcal{L} \subset \mathcal{X}$ where $\mathcal{L}$ is defined by an $\mathcal{NP}$ language. A key property of SPHF is that, for any $W \in \mathcal{L}$, its corresponding hash value can be computed by using either a secret hashing key, or a public projection key with the witness $w$ of $W$. Formally, an SPHF with domain $\mathcal{X}$ and range $\mathcal{Y}$ is defined by the following algorithms

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

**Extension.** Similar as [15], we define an extension of SPHF in this paper: we introduce a WordG algorithm and slightly modify the Hash, ProjHash algorithms as follows.

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

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

- *Correctness*. Let $W = \mathsf{WordG}(\mathcal{L}, \mathsf{param}, w)$, then for any hashing key hk and the corresponding 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 $W \in \mathcal{X} \backslash \mathcal{L}$, the following two distributions are statistically indistinguishable:

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

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

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

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

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

*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 its subset $\mathcal{L} \subset \mathcal{X}$, we say the subset membership problem is hard if the distribution $W \overset{\$}{\leftarrow} \mathcal{L}$ is computationally indistinguishable from the distribution $W' \overset{\$}{\leftarrow} \mathcal{X} \backslash \mathcal{L}$.*

## 2.5 Key Encapsulation Mechanism

A Key Encapsulation Mechanism (KEM) consists of the following algorithms

- $\mathsf{KeyGen}(1^k)$: a probabilistic algorithm that generates a public and private key pair $(PK, SK)$;
- $\mathsf{Enc}(PK)$: a probabilistic algorithm that generates a ciphertext $CT$ and a session key $K$ encapsulated in $CT$;
- $\mathsf{Dec}(SK, CT)$: a deterministic algorithm that outputs a key $K$ or a special symbol $\perp$ (decryption failure).

**Definition 6 (IND-CPA Security).** We say a KEM is IND-CPA secure if for any probabilistic polynomial time adversary

$$\{PK, CT, K\} \overset{c}{\equiv} \{PK, CT, R\}$$

where $PK \leftarrow \mathsf{KeyGen}(1^k)$, $(CT, K) \leftarrow \mathsf{Enc}(PK)$ and $R$ is randomly chosen from the session key space defined by $PK$.

## 3 A New Strong Leakage-Resilient AKE Security Model

### 3.1 AKE Protocol

A two-party AKE protocol is run between two parties that are modeled as probabilistic polynomial-time Turing Machines. In the symmetric-key setting, the two parties share a long-term symmetric key $lsk$ while in the public-key setting, each party has a secret key $lsk$ together with a certificate that binds the corresponding *long-term public key* ($lpk$) to the identity of the party. Hereafter we denote $\widehat{A}$ ($\widehat{B}$) as the (certified) long-term public key of party $\mathcal{A}$ ($\mathcal{B}$).

$\mathcal{A}$ and $\mathcal{B}$ can be activated to execute an instance of the AKE protocol, which is referred to as a *session*, and obtain a shared session key. Specifically, during the execution of a session, each party generates an *ephemeral public/secret key* pair $(epk, esk)$ and sends $epk$ as part of the message to the peer party. At the end of the session execution, each party derives the shared session key by taking as input their own long-term and ephemeral secret keys, along with the long-term and ephemeral public keys of the peer party.

We note that there can be parallel and concurrent sessions between $\mathcal{A}$ and $\mathcal{B}$. A session of party $\mathcal{A}$ with peer party $\mathcal{B}$ is identified by a session identifier, which is defined as $(\mathcal{A}, \mathcal{B}, epk_{\mathcal{A}}, epk_{\mathcal{B}})$ in this paper, and the session $(\mathcal{B}, \mathcal{A}, epk_{\mathcal{B}}, epk_{\mathcal{A}})$ of $\mathcal{B}$ is referred to as the *matching session*.

### 3.2 eCK Security Model

The extended Canetti-Krawczyk (eCK) model was proposed by LaMacchia, Lauter and Mityagin [31] based on the CK model by Canetti and Krawczyk [12]. The eCK model is designed for public-key based AKE and we will discuss how to adapt the model to the symmetric-key setting later.

In the eCK model, the adversary $\mathcal{M}$ is modelled as a probabilistic polynomial time Turing machine that controls all the communications among the parties. The adversary is also responsible for activating all the protocol sessions. Specifically, in the eCK model, adversary $\mathcal{M}$ is given (under the public-setting) the public keys of all the honest parties, 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.
- $\mathsf{EstablishParty}(\mathsf{pid}, \mathsf{PK})$. Under the public-key setting, this query allows the adversary to register a long-term public key $\mathsf{PK}$ on behalf of a party $\mathsf{pid}$ that is considered *dishonest* and controlled by $\mathcal{M}$. We should note that the adversary does not need to prove its knowledge of the secret key corresponding to $\mathsf{PK}$.
- $\mathsf{LongTermKeyReveal}(\mathsf{pid})$. This query allows the adversary to learn the long-term secret key of an honest party $\mathsf{pid}$.

- $\mathsf{SessionKeyReveal}(\mathsf{sid})$. This query allows the adversary to obtain the session key of a completed session sid.
- $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$. This query allows the adversary to obtain the ephemeral secret key used by an honest party in session sid.

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

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

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

- *$\mathcal{M}$ 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*
  - $\mathsf{LongTermKeyReveal}(\mathcal{B})$.

**Definition 8 (eCK Security).** *Let the test session* $\mathsf{sid}^*$ *be fresh according to the above definition. 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. We say an AKE protocol is eCK-secure if the matching session computes the same session key (if the matching session exists) 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 key leakage attacks against AKE protocols. Our notion, named *Challenge-Dependent Leakage-Resilient eCK* (CLR-eCK) model is the first security model that captures leakage of both long-term and ephemeral keys while allowing the adversary to issue leakage queries at any time during the game. Formally, adversary $\mathcal{M}$ is allowed to issue the following *leakage* queries on top of those defined in the eCK model.

- $\mathsf{LongTermKeyLeakage}(f_1, \mathsf{pid})$. 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 an honest 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 $\mathsf{sid}$.

**Restrictions on the Leakage Function.** In our CLR-eCK security model, there are some *necessary* restrictions on the leakage functions to prevent the adversary $\mathcal{M}$ from trivially winning the game.

Since we focus on the bounded leakage setting in this paper, the first restriction is that the total output length of the leakage functions $\{f_1\}$ and $\{f_2\}$ must be less than $|lsk|$ and $|esk|$, respectively. Specifically, 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 necessary restriction is related to the challenge-dependent leakage security of AKE protocols. W.l.o.g., we elaborate the problem under the public-key setting. Consider a test session $\mathsf{sid}^*$ which is owned by party $\mathcal{A}$ with peer $\mathcal{B}$. Note that for a 2-pass AKE protocol, the session key of $\mathsf{sid}^*$ is determined by $(\mathcal{A}, \mathcal{B}, lsk_{\mathcal{A}}, esk^*_{\mathcal{A}}, lpk_{\mathcal{B}}, epk^*_{\mathcal{B}})$ which contains only two secret keys (i.e., $lsk_{\mathcal{A}}, esk^*_{\mathcal{A}}$). Since $\mathcal{M}$ is allowed to reveal $esk^*_{\mathcal{A}}$ (or $lsk_{\mathcal{A}}$) in the eCK model, $\mathcal{M}$ can launch a trivial attack by encoding the session key derivation function into the leakage function of $lsk_{\mathcal{A}}$ (or $esk^*_{\mathcal{A}}$) and trivially win the security game. Therefore, in order to define an achievable security notion, it is impossible to let the adversary $\mathcal{M}$ have absolute freedom in making leakage queries if one of the two (i.e., long-term and ephemeral) secret keys regarding the test session has been revealed by the adversary. In order to give the adversary as much freedom as possible while ensuring achievable security, we impose the restrictions on $\mathsf{LongTermKeyLeakage}(f_1, \mathcal{A})$ and $\mathsf{EphemeralKeyLeakage}(f_2, \mathsf{sid}^*)$ as follows.

– $\mathcal{M}$ is allowed to query an arbitrary leakage function $f_1 \in \mathcal{F}_{\mathsf{bbd\text{-}I}}$ before it obtains the ephemeral secret key $esk^*_{\mathcal{A}}$ via an $\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*)$ query; however, after obtaining $esk^*_{\mathcal{A}}$, $\mathcal{M}$ can only query 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 query an arbitrary leakage function $f_2 \in \mathcal{F}_{\mathsf{bbd\text{-}II}}$ before it obtains the long-term secret key $lsk_{\mathcal{A}}$ via a $\mathsf{LongTermKeyReveal}(\mathcal{A})$ query; however, after obtaining $lsk_{\mathcal{A}}$, $\mathcal{M}$ can only query 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 $\mathsf{LongTermKeyReveal}(\mathcal{A})$.
– If the matching session $\overline{\mathsf{sid}^*}$ of the test session exists, the above restrictions also apply to the leakage queries $\mathsf{LongTermKeyLeakage}(f_1, \mathcal{B})$ and $\mathsf{EphemeralKeyLeakage}(f_2, \overline{\mathsf{sid}^*})$, since the session key of $\mathsf{sid}^*$ is also determined by $(\widehat{A}, \widehat{B}, lpk_{\mathcal{A}}, epk^*_{\mathcal{A}}, lsk_{\mathcal{B}}, esk^*_{\mathcal{B}})$.

**Remarks.** 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 $\mathcal{F}_1, \mathcal{F}_2$ after seeing the challenge session as long as the restriction holds. That is, $\mathcal{M}$ can specify leakage function sets $\mathcal{F}_1, \mathcal{F}_2$ after seeing $epk^*_{\mathcal{A}}$ and $epk^*_{\mathcal{B}}$. Also, if there is no long-term (ephemeral, respectively) key reveal query, then the adversary can choose any leakage function in $\mathcal{F}_{\mathsf{bbd\text{-}II}}$ ($\mathcal{F}_{\mathsf{bbd\text{-}I}}$, respectively), i.e., $\mathcal{M}$ 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 $(lpk_{\mathcal{A}}, lpk_{\mathcal{B}},$

$epk^*_{\mathcal{A}}, epk^*_{\mathcal{B}})$. Our model essentially gives $\mathcal{M}$ the most freedom in make key reveal and key leakage queries.

We define the notion of a *fresh session* in the $\mathsf{CLR\text{-}eCK}$ model as follows.

**Definition 9 (Session Freshness in the $(\lambda_1, \lambda_2)$-Leakage $\mathsf{CLR\text{-}eCK}$ Model).** *Let* $\mathsf{sid}$ *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* $\mathsf{sid}$*, if it exists. Session* $\mathsf{sid}$ *is said to be fresh in the* $\mathsf{CLR\text{-}eCK}$ *model if the following conditions hold:*

- $\mathsf{sid}$ *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* $\mathsf{LongTermKeyLeakage}$ *queries to* $\mathcal{A}$ *(*$\mathcal{B}$*, respectively) is at most* $\lambda_1$.
- *The total output length of all the* $\mathsf{EphemeralKeyLeakage}$ *query to* $\mathsf{sid}$ *(*$\overline{\mathsf{sid}}$*, respectively, if it exists) is at most* $\lambda_2$.

We now present the notion of $\mathsf{CLR\text{-}eCK}$ security.

**Definition 10 ($\mathsf{CLR\text{-}eCK}$ Security).** *Let the test session* $\mathsf{sid}^*$ *be fresh according to the above definition. We define the advantage of* $\mathcal{M}$ *in the* $\mathsf{CLR\text{-}eCK}$ *game by*

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

*where* $k$ *is the security parameter. We say an AKE protocol is* $(\lambda_1, \lambda_2)$-$\mathsf{CLR\text{-}eCK}$*-secure if the matching session computes the same session key (if the matching session exists) and for any probabilistic polynomial-time adversary* $\mathcal{M}$*,* $\mathsf{Adv}^{\mathsf{CLR\text{-}eCK}}_{\mathcal{M}}(k)$ *is negligible.*

***Remark.*** Here we give some further discussions on the key reveal queries, e.g., $\mathsf{LongTermKeyReveal}$, and the key leakage queries, e.g., $\mathsf{LongTermKeyLeakage}$. We can see that it is meaningless for $\mathcal{M}$ to issue a leakage query on a secret key if it has already obtained the whole key through a key reveal query. Therefore, we can conclude that the meaningful queries that adversary $\mathcal{M}$ will ask are as follows.

Let session $\mathsf{sid}^*$ be the test session owned by $\mathcal{A}$ with the peer $\mathcal{B}$. If $\overline{\mathsf{sid}^*}$ exists, $\mathcal{M}$ will only make key reveal and leakage queries in 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:

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

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

## 4 CLR-eCK-Secure AKE

In this section, we present a generic construction of CLR-eCK-secure AKE under the public-key setting.

### 4.1 General Framework

Fig. 1 describes our generic construction of a CLR-eCK-secure AKE protocol. Suppose that $k$ is the security parameter. Let $\mathcal{KEM}$ denote an IND-CPA secure key encapsulation mechanism with public key space $\mathcal{PK}$ and ciphertext space $\mathcal{C}$. Let $\mathcal{SPHF}$ denote a 2-smooth SPHF over $\mathcal{L} \subset \mathcal{X}$ and onto $\mathcal{Y}$ 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 $\mathcal{W}$. Let $H_1 : \{0,1\}^* \to \mathcal{AUX}$ denote a collision-resistant hash function.

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.

Let $\widehat{\mathsf{F}}$ and $\overline{\mathsf{F}}$ be PRF families and $\widetilde{\mathsf{F}}$ be a $\pi$PRF family as follows[3].

$\widehat{\mathsf{F}}^{k,\Sigma_{\widehat{\mathsf{F}}},\mathcal{D}_{\widehat{\mathsf{F}}},\mathcal{R}_{\widehat{\mathsf{F}}}} : \Sigma_{\widehat{\mathsf{F}}} = \{0,1\}^{l_1(k)}, \mathcal{D}_{\widehat{\mathsf{F}}} = \{0,1\}^{u(k)}, \mathcal{R}_{\widehat{\mathsf{F}}} = \mathcal{W} \times \{0,1\}^{\varsigma(k)}$,

$\overline{\mathsf{F}}^{k,\Sigma_{\overline{\mathsf{F}}},\mathcal{D}_{\overline{\mathsf{F}}},\mathcal{R}_{\overline{\mathsf{F}}}} : \Sigma_{\overline{\mathsf{F}}} = \{0,1\}^{l_2(k)}, \mathcal{D}_{\overline{\mathsf{F}}} = \{0,1\}^{t_1(k)}, \mathcal{R}_{\overline{\mathsf{F}}} = \mathcal{W} \times \{0,1\}^{\varsigma(k)}$,

$\widetilde{\mathsf{F}}^{k,\Sigma_{\widetilde{\mathsf{F}}},\mathcal{D}_{\widetilde{\mathsf{F}}},\mathcal{R}_{\widetilde{\mathsf{F}}}} : \Sigma_{\widetilde{\mathsf{F}}} = \{0,1\}^{l_3(k)}, \mathcal{D}_{\widetilde{\mathsf{F}}} = \Lambda(k)^2 \times \mathcal{L}^2 \times \mathcal{PK} \times \mathcal{C} \times \{0,1\}^{2t_3(k)}, \mathcal{R}_{\widetilde{\mathsf{F}}} = \{0,1\}^{l_4(k)}$.

Let $\widehat{F} \leftarrow \widehat{\mathsf{F}}^{k,\Sigma_{\widehat{\mathsf{F}}},\mathcal{D}_{\widehat{\mathsf{F}}},\mathcal{R}_{\widehat{\mathsf{F}}}}$, $\overline{F} \leftarrow \overline{\mathsf{F}}^{k,\Sigma_{\overline{\mathsf{F}}},\mathcal{D}_{\overline{\mathsf{F}}},\mathcal{R}_{\overline{\mathsf{F}}}}$ and $\widetilde{F} \leftarrow \widetilde{\mathsf{F}}^{k,\Sigma_{\widetilde{\mathsf{F}}},\mathcal{D}_{\widetilde{\mathsf{F}}},\mathcal{R}_{\widetilde{\mathsf{F}}}}$.

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

**Long-Term Key Generation.** At the long-term key generation stage, $\mathcal{A}$ runs the algorithm $\mathsf{HashKG}$ to obtain a hashing key $\mathsf{hk}$ and then the algorithm $\mathsf{ProjKG}$ to obtain the projection key $\mathsf{hp}$, picks $r_{\mathcal{A}_1} \xleftarrow{\$} \{0,1\}^{t_1(k)}, r_{\mathcal{A}_2} \xleftarrow{\$} \{0,1\}^{t_2(k)}$ and $t_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^{t_3(k)}$, then sets its long-term key pair as $lsk_{\mathcal{A}} = \mathsf{hk}, lpk_{\mathcal{A}} = (\mathsf{hp}, r_{\mathcal{A}_1}, r_{\mathcal{A}_2}, t_{\mathcal{A}})$. Similarly, $\mathcal{B}$ generates its long-term key pair as $lsk_{\mathcal{B}} = \mathsf{hk}', lpk_{\mathcal{B}} = (\mathsf{hp}', r_{\mathcal{B}_1}, r_{\mathcal{B}_2}, t_{\mathcal{B}})$.

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

- $(\mathcal{A} \rightharpoonup \mathcal{B})$. $\mathcal{A}$ performs the following steps.
  1. Select the ephemeral secret key $esk_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^{u(k)}$.
  2. Set $\widehat{lsk}_{\mathcal{A}} = \mathsf{Ext}_1(lsk_{\mathcal{A}}, r_{\mathcal{A}_1}), \widehat{esk}_{\mathcal{A}} = \mathsf{Ext}_2(esk_{\mathcal{A}}, r_{\mathcal{A}_2})$.
  3. Compute $(w_{\mathcal{A}}, \lambda_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(esk_{\mathcal{A}}) + \overline{F}_{\widehat{esk}_{\mathcal{A}}}(r_{\mathcal{A}_1})$.
  4. Run $\mathsf{WordG}(\mathsf{param}, \mathcal{L}, w_{\mathcal{A}})$ to obtain a word $W_{\mathcal{A}}$ and $\mathcal{KEM}.\mathsf{KeyGen}(1^k; \lambda_{\mathcal{A}})$ to obtain $(PK_{\mathcal{A}}, SK_{\mathcal{A}})$.
  5. Set $(W_{\mathcal{A}}, PK_{\mathcal{A}})$ as the ephemeral public key and send $(\widehat{B}, \widehat{A}, W_{\mathcal{A}}, PK_{\mathcal{A}})$ to $\mathcal{B}$.
- $(\mathcal{B} \rightharpoonup \mathcal{A})$. Upon receiving the message from $\mathcal{A}$, $\mathcal{B}$ executes the following steps.

[3] In this paper, we denote the space of a certified long-term public key (such as $\widehat{A}$) by $\Lambda(k)$ and the maximum size of the random coins required by $\mathcal{KEM}.\mathsf{KeyGen}$ and $\mathcal{KEM}.\mathsf{Enc}$ by $\varsigma(k)$.

| $\mathcal{A}$ | | $\mathcal{B}$ |
|---|---|---|
| | **Long-Term Key Generation** | |
| $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{param}, \mathcal{L})$, | | $\mathsf{hk}' \xleftarrow{\$} \mathsf{HashKG}(\mathsf{param}, \mathcal{L})$, |
| $\mathsf{hp} \xleftarrow{\$} \mathsf{ProjKG}(\mathsf{param}, \mathcal{L}, \mathsf{hk})$, | | $\mathsf{hp}' \xleftarrow{\$} \mathsf{ProjKG}(\mathsf{param}, \mathcal{L}, \mathsf{hk}')$, |
| $r_{\mathcal{A}_1} \xleftarrow{\$} \{0,1\}^{t_1(k)}, r_{\mathcal{A}_2} \xleftarrow{\$} \{0,1\}^{t_2(k)}, t_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^{t_3(k)}$ | | $r_{\mathcal{B}_1} \xleftarrow{\$} \{0,1\}^{t_1(k)}, r_{\mathcal{B}_2} \xleftarrow{\$} \{0,1\}^{t_2(k)}, t_{\mathcal{B}} \xleftarrow{\$} \{0,1\}^{t_3(k)}$ |
| $lsk_{\mathcal{A}} = \mathsf{hk}, lpk_{\mathcal{A}} = (\mathsf{hp}, r_{\mathcal{A}_1}, r_{\mathcal{A}_2}, t_{\mathcal{A}})$. | | $lsk_{\mathcal{B}} = \mathsf{hk}', lpk_{\mathcal{B}} = (\mathsf{hp}', r_{\mathcal{B}_1}, r_{\mathcal{B}_2}, t_{\mathcal{B}})$. |
| | **Session Execution** | |
| $esk_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^{u(k)}$ | | $esk_{\mathcal{B}} \xleftarrow{\$} \{0,1\}^{u(k)}$, |
| $\widehat{lsk}_{\mathcal{A}} = \mathsf{Ext}_1(lsk_{\mathcal{A}}, r_{\mathcal{A}_1})$, | | $\widehat{lsk}_{\mathcal{B}} = \mathsf{Ext}_1(lsk_{\mathcal{B}}, r_{\mathcal{B}_1})$, |
| $\widehat{esk}_{\mathcal{A}} = \mathsf{Ext}_2(esk_{\mathcal{A}}, r_{\mathcal{A}_2})$, | | $\widehat{esk}_{\mathcal{B}} = \mathsf{Ext}_2(esk_{\mathcal{B}}, r_{\mathcal{B}_2})$, |
| $(w_{\mathcal{A}}, \lambda_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(esk_{\mathcal{A}}) + \overline{F}_{\widehat{esk}_{\mathcal{A}}}(r_{\mathcal{A}_1})$, | | $(w_{\mathcal{B}}, \lambda_{\mathcal{B}}) = \widehat{F}_{\widehat{lsk}_{\mathcal{B}}}(esk_{\mathcal{B}}) + \overline{F}_{\widehat{esk}_{\mathcal{B}}}(r_{\mathcal{B}_1})$, |
| $W_{\mathcal{A}} = \mathsf{WordG}(\mathsf{param}, \mathcal{L}, w_{\mathcal{A}})$, | | $W_{\mathcal{B}} = \mathsf{WordG}(\mathsf{param}, \mathcal{L}, w_{\mathcal{B}})$, |
| $(PK_{\mathcal{A}}, SK_{\mathcal{A}}) = \mathcal{KEM}.\mathsf{KeyGen}(1^k; \lambda_{\mathcal{A}})$ | | |
| | $\xrightarrow{(\widehat{B}, \widehat{A}, W_{\mathcal{A}}, PK_{\mathcal{A}})}$ | |
| | | $(CT_{\mathcal{B}}, K_{\mathcal{B}_1}) = \mathcal{KEM}.\mathsf{Enc}(PK_{\mathcal{A}}; \lambda_{\mathcal{B}})$ |
| | $\xleftarrow{(\widehat{A}, \widehat{B}, W_{\mathcal{B}}, CT_{\mathcal{B}})}$ | |
| | **Session Key Ouput** | |
| Set $\mathsf{sid} = (\widehat{A}, \widehat{B}, W_{\mathcal{A}}, PK_{\mathcal{A}}, W_{\mathcal{B}}, CT_{\mathcal{B}})$ | | Set $\mathsf{sid} = (\widehat{A}, \widehat{B}, W_{\mathcal{A}}, , PK_{\mathcal{A}}, W_{\mathcal{B}}, CT_{\mathcal{B}})$ |
| $aux = H_1(\mathsf{sid}), K_{\mathcal{A}_1} = \mathcal{KEM}.\mathsf{Dec}(SK_{\mathcal{A}}, CT_{\mathcal{B}})$, | | $aux = H_1(\mathsf{sid})$, |
| $K_{\mathcal{A}_2} = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, lpk_{\mathcal{B}}, W_{\mathcal{A}}, w_{\mathcal{A}}, aux)$, | | $K_{\mathcal{B}_2} = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, lsk_{\mathcal{B}}, W_{\mathcal{A}}, aux)$, |
| $K_{\mathcal{A}_3} = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, lsk_{\mathcal{A}}, W_{\mathcal{B}}, aux)$, | | $K_{\mathcal{B}_3} = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, lpk_{\mathcal{A}}, W_{\mathcal{B}}, w_{\mathcal{B}}, aux)$, |
| $s_{\mathcal{A}} = Ext3(K_{\mathcal{A}_1} \oplus K_{\mathcal{A}_2} \oplus K_{\mathcal{A}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}})$, | | $s_{\mathcal{B}} = Ext3(K_{\mathcal{B}_1} \oplus K_{\mathcal{B}_2} \oplus K_{\mathcal{B}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}})$, |
| $SK_{\mathcal{A}} = \widetilde{F}_{s_{\mathcal{A}}}(\mathsf{sid})$. | | $SK_{\mathcal{B}} = \widetilde{F}_{s_{\mathcal{B}}}(\mathsf{sid})$. |

**Fig. 1.** Framework for CLR-eCK secure AKE

1. Select the ephemeral secret key $esk_{\mathcal{B}} \xleftarrow{\$} \{0,1\}^{u(k)}$.
2. Set $\widehat{lsk}_{\mathcal{B}} = \mathsf{Ext}_1(lsk_{\mathcal{B}}, r_{\mathcal{B}_1}), \widehat{esk}_{\mathcal{B}} = \mathsf{Ext}_2(esk_{\mathcal{B}}, r_{\mathcal{B}_2})$.
3. Compute $(w_{\mathcal{B}}, \lambda_{\mathcal{B}}) = \widehat{F}_{\widehat{lsk}_{\mathcal{B}}}(esk_{\mathcal{B}}) + \overline{F}_{\widehat{esk}_{\mathcal{B}}}(r_{\mathcal{B}_1})$.
4. Run $\mathsf{WordG}(\mathsf{param}, \mathcal{L}, w_{\mathcal{B}})$ to obtain a word $W_{\mathcal{B}}$ and compute
$$(CT_{\mathcal{B}}, K_{\mathcal{B}_1}) = \mathcal{KEM}.\mathsf{Enc}(PK_{\mathcal{A}}; \lambda_{\mathcal{B}}).$$
5. Set $(W_{\mathcal{B}}, CT_{\mathcal{B}})$ as the ephemeral public key and send $(\widehat{A}, \widehat{B}, W_{\mathcal{B}}, CT_{\mathcal{B}})$ to $\mathcal{A}$.

**Session Key Output.** When $\mathcal{A}$ receives $(\widehat{A}, \widehat{B}, W_{\mathcal{B}}, CT_{\mathcal{B}})$, $\mathcal{A}$ sets

$$\mathsf{sid} = (\widehat{A}, \widehat{B}, W_{\mathcal{A}}, PK_{\mathcal{A}}, W_{\mathcal{B}}, CT_{\mathcal{B}}), aux = H_1(\mathsf{sid})$$

and computes the session key as follows.

1. Compute $K_{\mathcal{A}_1} = \mathcal{KEM}.\mathsf{Dec}(SK_{\mathcal{A}}, CT_{\mathcal{B}}), K_{\mathcal{A}_2} = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, lpk_{\mathcal{B}}, W_{\mathcal{A}}, w_{\mathcal{A}}, aux)$, $K_{\mathcal{A}_3} = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, lsk_{\mathcal{A}}, W_{\mathcal{B}}, aux)$.
2. Set $s_{\mathcal{A}} = Ext_3(K_{\mathcal{A}_1} \oplus K_{\mathcal{A}_2} \oplus K_{\mathcal{A}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}})$.
3. Compute $SK_{\mathcal{A}} = \widetilde{F}_{s_{\mathcal{A}}}(\mathsf{sid})$.

Similarly, party $\mathcal{B}$ sets

$$\mathsf{sid} = (\widehat{A}, \widehat{B}, W_{\mathcal{A}}, PK_{\mathcal{A}}, W_{\mathcal{B}}, CT_{\mathcal{B}}), aux = H_1(\mathsf{sid})$$

and then computes the session key as follows.

1. Compute $K_{\mathcal{B}_2} = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, lsk_{\mathcal{B}}, W_{\mathcal{A}}, aux), K_{\mathcal{B}_3} = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, lpk_{\mathcal{A}}, W_{\mathcal{B}}, w_{\mathcal{B}}, aux)$.
2. Set $s_{\mathcal{B}} = Ext_3(K_{\mathcal{B}_1} \oplus K_{\mathcal{B}_2} \oplus K_{\mathcal{B}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}})$.
3. Compute $SK_{\mathcal{B}} = \widetilde{F}_{s_{\mathcal{B}}}(\mathsf{sid})$.

**Correctness Analysis.** One can note that $K_{\mathcal{A}_1} = K_{\mathcal{B}_1}$ due to the correctness of $\mathcal{KEM}$. Due to the correctness 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

$$s_{\mathcal{A}} = Ext_3(K_{\mathcal{A}_1} \oplus K_{\mathcal{A}_2} \oplus K_{\mathcal{A}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}}) = Ext_3(K_{\mathcal{B}_1} \oplus K_{\mathcal{B}_2} \oplus K_{\mathcal{B}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}}) = s_{\mathcal{B}},$$

which guarantees that $SK_{\mathcal{A}} = SK_{\mathcal{B}}$.

## 4.2 Security Analysis

**Theorem 1.** *The AKE protocol following the general framework is* $(\lambda_1, \lambda_2)$-$\mathsf{CLR}$-$\mathsf{eCK}$-*secure if the underlying smooth projective hash function is* $2$-$\mathsf{smooth}$*, the underlying KEM is IND-CPA secure,* $H_1$ *is a collision-resistant hash function,* $\widehat{\mathsf{F}}$ *and* $\overline{\mathsf{F}}$ *are PRF families and* $\widetilde{\mathsf{F}}$ *is a* $\pi$*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 $\mathsf{sid}^* = (\widehat{A}, \widehat{B}, W^*_{\mathcal{A}}, PK^*_{\mathcal{A}}, W^*_{\mathcal{B}}, CT^*_{\mathcal{B}})$ be the target session chosen by adversary $\mathcal{M}$. $\mathcal{A}$ is the owner of the session $\mathsf{sid}^*$ 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* $\mathsf{sid}^*$.

We analyse the security based on the reveal and leakage queries that the adversary issues to the target session, its 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 $\lambda_2$-bits of the ephemeral secret key of target session $\mathsf{sid}^*$, we have that

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

Therefore, $(w^*_{\mathcal{A}}, \lambda^*_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(esk^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r_{\mathcal{A}_1}) \overset{c}{\equiv} (w'_{\mathcal{A}}, \lambda'_{\mathcal{A}}) \xleftarrow{\$} \mathcal{W} \times \{0,1\}^{\zeta(k)}$. Similarly, suppose that the adversary obtains at most $\lambda_2$-bits of the ephemeral secret key of matching session $\overline{\mathsf{sid}^*}$, we have that

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

and thus $(w^*_{\mathcal{B}}, \lambda^*_{\mathcal{B}}) = \widehat{F}_{\widehat{lsk}_{\mathcal{B}}}(esk^*_{\mathcal{B}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{B}}}(r_{\mathcal{B}_1}) \overset{c}{\equiv} (w'_{\mathcal{B}}, \lambda'_{\mathcal{B}}) \xleftarrow{\$} \mathcal{W} \times \{0,1\}^{\zeta(k)}$.

– $\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*)$, $\mathsf{EphemeralKeyReveal}(\overline{\mathsf{sid}^*})$, $\mathsf{LongTermKeyLeakage}(\mathcal{A})$, $\mathsf{LongTermKeyLeakage}(\mathcal{B})$. In this sub-case, suppose that the adversary obtains at most $\lambda_1$-bits of the long-term secret key of party $\mathcal{A}$, we have that

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

hence $(w^*_{\mathcal{A}}, \lambda^*_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{A}}}(esk^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r_{\mathcal{A}}) \overset{c}{\equiv} (w'_{\mathcal{A}}, \lambda'_{\mathcal{A}}) \xleftarrow{\$} \mathcal{W} \times \{0,1\}^{\zeta(k)}$. Similarly, suppose that the adversary obtains at most $\lambda_1$-bits of the long-term secret key of party $\mathcal{B}$, we have that

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

and therefore $(w^*_{\mathcal{B}}, \lambda_{\mathcal{B}}*) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{B}}}(esk^*_{\mathcal{B}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{B}}}(r_{\mathcal{B}_1}) \overset{c}{\equiv} (w'_{\mathcal{B}}, \lambda'_{\mathcal{B}}) \xleftarrow{\$} \mathcal{W} \times \{0,1\}^{\zeta(k)}$.

– $\mathsf{LongTermKeyReveal}(\mathcal{A})$, $\mathsf{EphemeralKeyReveal}(\overline{\mathsf{sid}^*})$, $\mathsf{EphemeralKeyLeakage}(\mathsf{sid}^*)$, $\mathsf{LongTermKeyLeakage}(\mathcal{B})$. In this sub-case, suppose that the adversary obtains at most $\lambda_2$-bits of the ephemeral secret key of target session $\mathsf{sid}^*$, at most $\lambda_1$-bits of the long-term secret key of party $\mathcal{B}$, then based on the Equation (1),(4), we have that $(w^*_{\mathcal{A}}, \lambda^*_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{A}}}(esk^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r_{\mathcal{A}_1}) \overset{c}{\equiv} (w'_{\mathcal{A}}, \lambda'_{\mathcal{A}}) \xleftarrow{\$} \mathcal{W} \times \{0,1\}^{\zeta(k)}$ and $(w^*_{\mathcal{B}}, \lambda^*_{\mathcal{B}}) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{B}}}(esk^*_{\mathcal{B}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{B}}}(r_{\mathcal{B}_1}) \overset{c}{\equiv} (w'_{\mathcal{B}}, \lambda'_{\mathcal{B}}) \xleftarrow{\$} \mathcal{W} \times \{0,1\}^{\zeta(k)}$.

– $\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*)$, $\mathsf{LongTermKeyReveal}(\mathcal{B})$, $\mathsf{LongTermKeyLeakage}(\mathcal{A})$, $\mathsf{EphemeralKeyLeakage}(\overline{\mathsf{sid}^*})$. In this sub-case, suppose that the adversary obtains at most $\lambda_1$-bits of the long-term secret key of party $\mathcal{A}$, at most $\lambda_2$-bits of the ephemeral secret key of matching session $\overline{\mathsf{sid}^*}$, then based on Equation (2),(3), we have that $(w^*_{\mathcal{A}}, \lambda^*_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{A}}}(esk^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r_{\mathcal{A}_1}) \overset{c}{\equiv} (w'_{\mathcal{A}}, \lambda'_{\mathcal{A}}) \xleftarrow{\$} \mathcal{W} \times \{0,1\}^{\zeta(k)}$ and $(w^*_{\mathcal{B}}, \lambda^*_{\mathcal{B}}) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{B}}}(esk^*_{\mathcal{B}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{B}}}(r_{\mathcal{B}_1}) \overset{c}{\equiv} (w'_{\mathcal{B}}, \lambda'_{\mathcal{B}}) \xleftarrow{\$} \mathcal{W} \times \{0,1\}^{\zeta(k)}$.

Therefore, regardless of the reveal and leakage queries in the above sub-cases, $(\lambda^*_{\mathcal{A}}, \lambda^*_{\mathcal{B}})$ are uniformly random strings in $\{0,1\}^{\zeta(k)}$ from the view of adversary $\mathcal{M}$. Therefore, $K^*_{\mathcal{A}_1} = K^*_{\mathcal{B}_1}$ is computationally indistinguishable from a random key based on the IND-CPA security of $\mathcal{KEM}$. We then have that the seed $s^*_{\mathcal{A}}$ for the $\pi$PRF 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* $\mathsf{sid}^*$.

In this case, the adversary cannot issue $\mathsf{LongTermKeyReveal}$ query to reveal the long-term secret key of $\mathcal{B}$ but may issue the leakage query $\mathsf{LongTermKeyLeakage}$ to learn some information of $lsk_{\mathcal{B}}$. 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 $\mathsf{WordG}$.

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 a random element in $\mathcal{X} \setminus \mathcal{L}$ is indistinguishable from that in $\mathcal{L}$.

In Game 2, since $W^*_{\mathcal{A}} \in \mathcal{X} \setminus \mathcal{L}$, we have that $K^*_{\mathcal{A}_2}$ is uniformly distributed in $\mathcal{Y}$ due to the smoothness of the SPHF. Suppose that the leakage of $lsk_{\mathcal{B}}$, denoted by $\widetilde{lsk_{\mathcal{B}}}$, is at most $\lambda_1$-bits, then according to *Lemma 1*, we have

$$\widetilde{\mathrm{H}}_\infty(K^*_{\mathcal{A}_2}|\widetilde{lsk_{\mathcal{B}}}) \geq \widetilde{\mathrm{H}}_\infty(K^*_{\mathcal{A}_2}) - \lambda_1 = |\mathcal{Y}| - \lambda_1.$$

Therefore, by using the strong extractor $\mathsf{Ext}_3$, it holds that

$$s^*_{\mathcal{A}} = \mathsf{Ext}_3(K^*_{\mathcal{A}_1} \oplus K^*_{\mathcal{A}_2} \oplus K^*_{\mathcal{A}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}}) \overset{s}{\equiv}_{\epsilon_3} s'_{\mathcal{A}} \overset{\$}{\leftarrow} \{0,1\}^{l_3(k)}.$$

We should note that adversary $\mathcal{M}$ may activate a session with $\mathcal{B}$, which is not matching to the test session $\mathsf{sid}^*$ but is related to $\mathsf{sid}^*$. Specifically, $\mathcal{M}$ can replay $W^*_{\mathcal{A}}$ and send it to $\mathcal{B}$ in another session, denoted by $\mathsf{sid}$, and then issue a $\mathsf{SessionKeyReveal}(\mathsf{sid})$ query to learn the session key output by $\mathcal{B}$. Nevertheless, since $\mathsf{sid} \neq \mathsf{sid}^*$, we have $aux = H_1(\mathsf{sid}) \neq aux^* = H_1(\mathsf{sid}^*)$ based on the collision resistance property of $H_1$. Then due to the 2-smooth property of the underlying smooth projective hash function, we have that $K^*_{\mathcal{A}_2}$ is independent from $K_{\mathcal{B}_2}$ derived by $\mathcal{B}$ in the session $\mathsf{sid}$ and thus $s^*_{\mathcal{A}}$ in the test session $\mathsf{sid}^*$ is pairwisely independent from $s_{\mathcal{B}}$ in $\mathsf{sid}$. Therefore, $SK^*_{\mathcal{A}}$ is computationally indistinguishable from a truly random value from $\mathcal{M}$'s view given that $\tilde{F}$ is a secure $\pi$PRF.

**Simulation for Non-test Session.** For the two cases given above, we need to also simulate the non-test sessions and answer the queries issued by the adversary correctly. It is easy to see that the simulation can be done easily since in both cases the simulator can know both $lsk_{\mathcal{A}}$ and $lsk_{\mathcal{B}}$ as the hardness of the subset membership problem does not rely on the secrecy of the hashing keys. Hence, the simulator can answer $\mathsf{LongTermKeyReveal}$ and $\mathsf{LongTermKeyLeakage}$ queries without any problem. To simulate a non-test session, the simulator can just follow the protocol by generating an ephemeral secret key and computing the ephemeral public key honestly. Similarly, the simulator can also answer the $\mathsf{SessionKeyReveal}$ query for any non-test session simulated by itself easily. This completes the proof.

## 4.3 Instantiations of the Proposed Protocol

In this section, we present several instantiations of the proposed $\mathsf{CLR\text{-}eCK}$-secure AKE protocol based on different assumptions.

**Instantiation based on DDH Assumption.**
We first present the Diffie-Hellman language $\mathcal{L}_{\mathsf{DH}}$ and its corresponding 2-smooth SPHF based on the result of [15].

**Diffie-Hellman Language.** Let $\mathbb{G}$ be a group of primer order $p$ and $g_1, g_2 \in \mathbb{G}$ two random generators of $\mathbb{G}$. Define the domain $\mathcal{X}$ as

$$\mathcal{X} = \{(u_1, u_2)|\exists r_1, r_2 \in \mathbb{Z}_p^2, s.t., u_1 = g_1^{r_1}, u_2 = g_2^{r_2}\}$$

and the Diffie-Hellman Language as

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

One can see that $\mathcal{L}_{\mathsf{DH}} \subset \mathcal{X} = \mathbb{G}^2$ and immediately obtain the following result.

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

**SPHF based on $\mathcal{L}_{\mathsf{DH}}$.** Let $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ denote a collision-resistant hash function. The construction is as follows.

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

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

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

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

- *Projective*. For a word $W = (u_1, u_2) = (g_1^r, g_2^r) \in \mathcal{L}_{\mathsf{DH}}$ we have

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

- *Smooth* (2-smooth). Suppose $g_2 = g_1^\theta$. Given $\mathsf{hp}_1 = g_1^{\alpha_1} g_2^{\alpha_2}, \mathsf{hp}_2 = g_1^{\beta_1} g_2^{\beta_2}$, the hashing key $(\alpha_1, \alpha_2, \beta_1, \beta_2)$ is constrained by the following equations

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

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

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 $r_1 \neq r_2, r_1' \neq r_2'$, and $aux_1 = d_1 = H_1(W_1, aux_1'), aux_2 = d_2 = H_1(W_2, aux_2')$, then the hash values $\mathsf{hv}_1$ of $W_1$ and $\mathsf{hv}_2$ 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 constrain $(\alpha_1, \alpha_2, \beta_1, \beta_2)$ 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

$$\mathbf{A} = \begin{bmatrix} 1 & \theta & 0 & 0 \\ 0 & 0 & 1 & \theta \\ r_1 & \theta r_2 & r_1 d_1 & \theta r_2 d_1 \\ r'_1 & \theta r'_2 & r'_1 d_2 & \theta r'_2 d_2 \end{bmatrix}.$$

If $(W_1, aux_1) \neq (W_2, aux_2)$ where $aux_1 = d_1 = H_1(W_1, aux'_1), aux_2 = d_2 = H_1(W_2, aux'_2)$, we have $d_1 \neq d_2$ since $H_1$ is collision resistant. 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 from the equation (7). Therefore, we have that $\mathsf{hv}_2$ is independent from $\mathsf{hv}_1$ and randomly distributed in $\mathbb{G}$.

**A Concrete AKE Protocol based on DDH.** We can instantiate our generic construction using the $\mathcal{SPHF}_{\mathsf{DH}}$ describe above together with an IND-CPA secure KEM. The Diffie-Hellman key exchange, i.e., the KEM underlying the ElGamal encryption, is a natural candidate for our purpose. The resulting AKE protocol is presented in Fig. 2. It is worth noting that since the Diffie-Hellman KEM allows the protocol responder $\mathcal{B}$ to generate a ciphertext $CT_{\mathcal{B}}$ without knowing the encryption key $PK_{\mathcal{A}}$ from $\mathcal{A}$, the concrete AKE protocol presented in Fig. 2 requires only one round.

**Parameters of The DDH based Protocol.** In the DDH based protocol presented in Fig. 2, $\mathbb{G}$ denotes a cyclic group of primer order $p$ and $g_1, g_2$ are random generators of $\mathbb{G}$. Then for the $\mathcal{SPHF}_{\mathsf{DH}}$, we have that $\mathcal{Y} = \mathbb{G}, \mathcal{HK} = \mathbb{Z}_p^4, \mathcal{HP} = \mathbb{G}^2, \mathcal{AUX} = \mathbb{Z}_p, \mathcal{W} = \mathbb{Z}_p$. Choose a collision-resistant hash function $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ and strong extractors as follows: $\mathsf{Ext}_1 : \mathbb{Z}_p^4 \times \{0,1\}^{t_1(k)} \to \{0,1\}^{l_1(k)}$ an average-case $(4 \cdot \log p - \lambda_1, \epsilon_1)$-strong extractor, $\mathsf{Ext}_2 : \{0,1\}^{u(k)} \times \{0,1\}^{t_2(k)} \to \{0,1\}^{l_2(k)}$ an average-case $(u(k) - \lambda_2, \epsilon_2)$-strong extractor and $\mathsf{Ext}_3 : \mathbb{G} \times \{0,1\}^{t_3(k)} \to \{0,1\}^{l_3(k)}$ an average-case $(\log p - \lambda_1, \epsilon_3)$-strong extractor. Choose $\widehat{F} \leftarrow \widehat{\mathsf{F}}^{k,\Sigma_{\widehat{\mathsf{F}}},\mathcal{D}_{\widehat{\mathsf{F}}},\mathcal{R}_{\widehat{\mathsf{F}}}}$, $\overline{F} \leftarrow \overline{\mathsf{F}}^{k,\Sigma_{\overline{\mathsf{F}}},\mathcal{D}_{\overline{\mathsf{F}}},\mathcal{R}_{\overline{\mathsf{F}}}}$ and $\widetilde{F} \leftarrow \widetilde{\mathsf{F}}^{k,\Sigma_{\widetilde{\mathsf{F}}},\mathcal{D}_{\widetilde{\mathsf{F}}},\mathcal{R}_{\widetilde{\mathsf{F}}}}$.

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

**Corollary 1.** *The concrete AKE protocol in Fig. 2 is* $(\lambda_1, \lambda_2)$-CLR-eCK-*secure under the DDH assumption, where* $\lambda_1 \leq \min\{4 \log p - 2\log(1/\epsilon_1) - l_1(k), \log p - 2\log(1/\epsilon_3) - l_3(k)\}, \lambda_2 \leq u(k) - 2\log(1/\epsilon_2) - l_2(k)$.

## Instantiation based on DCR Assumption

Below we present another language $\mathcal{L}_{\mathsf{DCR}}$ and it corresponding 2-smooth SPHF.

**Decision Composite Residuosity Assumption.** Let $p, q, p', q'$ be distinct odd primes such that $p = 2p' + 1$ and $q = 2q' + 1$ (i.e., $p$ and $q$ are safe primes) where $p'$ and $q'$ are $k$ bits in length. Let $N = pq$, $N' = p'q'$ and $G_N, G_{N'}$ be subgroups of $\mathbb{Z}_{N^2}^*$ of order $N$ and $N'$, respectively. Let $G = \{x^N : x \in \mathbb{Z}_{N^2}^*\}$. The Decision Composite Residuosity (DCR) assumption says that given only $N$, elements randomly chosen from $\mathbb{Z}_{N^2}^*$ are computationally indistinguishable from those randomly chosen from $G$.

Let $T$ denote the subgroup of $\mathbb{Z}_{N^2}^*$ generated by $-1 \pmod{N^2}$. Define

$$\mathcal{X} = G_N \cdot G_{N'} \cdot T \qquad \text{and} \qquad \mathcal{L}_{\mathsf{DCR}} = G_{N'} \cdot T.$$

$\mathcal{A}$ $\mathcal{B}$

**Long-Term Key Generation**

$$\mathsf{hk} = (\alpha_1, \alpha_2, \beta_1, \beta_2) \xleftarrow{\$} \mathbb{Z}_p^4,$$
$$\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2) = (g_1^{\alpha_1} g_2^{\alpha_2}, g_1^{\beta_1} g_2^{\beta_2}) \in \mathbb{G}^2,$$
$$r_1 \xleftarrow{\$} \{0,1\}^{t_1(k)}, r_2 \xleftarrow{\$} \{0,1\}^{t_2(k)}, t \xleftarrow{\$} \{0,1\}^{t_3(k)},$$
$$lsk_{\mathcal{A}} = \mathsf{hk}, lpk_{\mathcal{A}} = (\mathsf{hp}, r_1, r_2, t).$$

$$\mathsf{hk}' = (\alpha_1', \alpha_2', \beta_1', \beta_2') \xleftarrow{\$} \mathbb{Z}_p^4,$$
$$\mathsf{hp}' = (\mathsf{hp}_1', \mathsf{hp}_2') = (g_1^{\alpha_1'} g_2^{\alpha_2'}, g_1^{\beta_1'} g_2^{\beta_2'}) \in \mathbb{G}^2,$$
$$r_1' \xleftarrow{\$} \{0,1\}^{t_1(k)}, r_2' \xleftarrow{\$} \{0,1\}^{t_2(k)}, t' \xleftarrow{\$} \{0,1\}^{t_3(k)},$$
$$lsk_{\mathcal{B}} = \mathsf{hk}', lpk_{\mathcal{B}} = (\mathsf{hp}', r_1', r_2', t').$$

**Session Execution**

$$e \xleftarrow{\$} \{0,1\}^{u(k)},$$
$$\widehat{lsk}_{\mathcal{A}} = \mathsf{Ext}_1(lsk_{\mathcal{A}}, r_1),$$
$$\widehat{esk}_{\mathcal{A}} = \mathsf{Ext}_2(e, r_2),$$
$$(r, x) = \widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(e) + \overline{F}_{\widehat{esk}_{\mathcal{A}}}(r_1),$$
$$W = (u_1, u_2) = (g_1^r, g_2^r), X = g^x,$$

$$e' \xleftarrow{\$} \{0,1\}^{u(k)},$$
$$\widehat{lsk}_{\mathcal{B}} = \mathsf{Ext}_1(lsk_{\mathcal{B}}, r_1'),$$
$$\widehat{esk}_{\mathcal{B}} = \mathsf{Ext}_2(e', r_2'),$$
$$(r', y) = \widehat{F}_{\widehat{lsk}_{\mathcal{B}}}(e') + \overline{F}_{\widehat{esk}_{\mathcal{B}}}(r_1'),$$
$$W' = (u_1', u_2') = (g_1^{r'}, g_2^{r'}), Y = g^y,$$

$$\xrightarrow{(\widehat{B}, \widehat{A}, W, X)}$$

$$\xleftarrow{(\widehat{A}, \widehat{B}, W', Y)}$$

**Session Key Ouput**

$$\text{Set } \mathsf{sid} = (\widehat{A}, \widehat{B}, W, X, W', Y)$$
$$d = H_1(\mathsf{sid}), K_{\mathcal{A}_1} = Y^x,$$
$$K_{\mathcal{A}_2} = \mathsf{hp}_1'^{r} \mathsf{hp}_2'^{dr}, K_{\mathcal{A}_3} = u_1'^{\alpha_1 + d\beta_1} u_2'^{\alpha_2 + d\beta_2},$$
$$s_{\mathcal{A}} = Ext3(K_{\mathcal{A}_1} \oplus K_{\mathcal{A}_2} \oplus K_{\mathcal{A}_3}, t \oplus t'),$$
$$SK_{\mathcal{A}} = \widetilde{F}_{s_{\mathcal{A}}}(\mathsf{sid}).$$

$$\text{Set } \mathsf{sid} = (\widehat{A}, \widehat{B}, W, X, W', Y)$$
$$d = H_1(\mathsf{sid}), K_{\mathcal{B}_1} = X^y,$$
$$K_{\mathcal{B}_2} = u_1^{\alpha_1' + d\beta_1'} u_2^{\alpha_2' + d\beta_2'}, K_{\mathcal{B}_3} = \mathsf{hp}_1^{r'} \mathsf{hp}_2^{dr'},$$
$$s_{\mathcal{B}} = Ext3(K_{\mathcal{B}_1} \oplus K_{\mathcal{B}_2} \oplus K_{\mathcal{B}_3}, t \oplus t'),$$
$$SK_{\mathcal{B}} = \widetilde{F}_{s_{\mathcal{B}}}(\mathsf{sid}).$$

**Fig. 2.** CLR-eCK secure AKE Protocol Based on DDH Assumption

**Theorem 4 ([15]).** *The subset membership problem over $\mathcal{L}_{\mathsf{DCR}}$ and $\mathcal{X}$ is hard under the DCR assumption.*

**SPHF based on $\mathcal{L}_{\mathsf{DCR}}$.** Let $g$ denote a random generator of $\mathcal{L}_{\mathsf{DCR}}$, $\mathcal{W} = \{0, 1, \cdots, N/2\}$, $\mathcal{K} = \{0, 1, \cdots, N^2/2\}$ and $H_1 : \{0,1\}^* \to \mathcal{W}$ a collision-resistant hash function. Define a map $f : \mathbb{Z}_{N^2} \to \mathbb{Z}_N$ as follows:

$$\forall x = a + bN \ (\mathrm{mod}\ N^2) \text{ where } 0 \leq a, b < N,\ f(x) = b \ (\mathrm{mod}\ N).$$

The construction of the SPHF is as follows.

- $\mathsf{SPHFSetup}(1^\lambda)$: $\mathsf{param} = (g, N)$;
- $\mathsf{HashKG}(\mathcal{L}_{\mathsf{DCR}}, \mathsf{param})$: $\mathsf{hk} = (\alpha, \beta) \xleftarrow{\$} \mathcal{K}^2$;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}_{\mathsf{DCR}}, \mathsf{param}))$: $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2) = (g^\alpha, g^\beta) \in \mathcal{L}_{\mathsf{DCR}}^2$;
- $\mathsf{WordG}((\mathcal{L}_{\mathsf{DCR}}, \mathsf{param}), w \in \mathcal{W})$: $W = g^w$;
- $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}_{\mathsf{DCR}}, \mathsf{param}), W = g^w, aux = d = H_1(W, aux'))$: $\mathsf{hv} = f(W^{\alpha + d\beta}) \in \mathbb{Z}_N$;
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}_{\mathsf{DCR}}, \mathsf{param}), W = g^w, w, aux = d = H_1(W, aux'))$: $\mathsf{hv}' = f(\mathsf{hp}_1^w \mathsf{hp}_2^{dw}) \in \mathbb{Z}_N$.

$$\mathcal{A} \qquad\qquad\qquad\qquad \mathcal{B}$$

**Long-Term Key Generation**

$$\begin{array}{cc}
\mathsf{hk} = (\alpha, \beta) \overset{\$}{\leftarrow} \mathcal{K}^2, & \mathsf{hk}' = (\alpha', \beta') \overset{\$}{\leftarrow} \mathcal{K}^2, \\
\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2) = (g^{\alpha}, g^{\beta}) \in \mathcal{L}^2_{\mathsf{DCR}}, & \mathsf{hp}' = (\mathsf{hp}'_1, \mathsf{hp}'_2) = (g^{\alpha'}, g^{\beta'}) \in \mathcal{L}^2_{\mathsf{DCR}}, \\
r_1 \overset{\$}{\leftarrow} \{0,1\}^{t_1(k)}, r_2 \overset{\$}{\leftarrow} \{0,1\}^{t_2(k)}, t \overset{\$}{\leftarrow} \{0,1\}^{t_3(k)}, & r'_1 \overset{\$}{\leftarrow} \{0,1\}^{t_1(k)}, r'_2 \overset{\$}{\leftarrow} \{0,1\}^{t_2(k)}, t' \overset{\$}{\leftarrow} \{0,1\}^{t_3(k)}, \\
lsk_{\mathcal{A}} = \mathsf{hk}, lpk_{\mathcal{A}} = (\mathsf{hp}, r_1, r_2, t). & lsk_{\mathcal{B}} = \mathsf{hk}', lpk_{\mathcal{B}} = (\mathsf{hp}', r'_1, r'_2, t').
\end{array}$$

**Session Execution**

$$\begin{array}{cc}
e \overset{\$}{\leftarrow} \{0,1\}^{u(k)}, & e' \overset{\$}{\leftarrow} \{0,1\}^{u(k)}, \\
\widehat{lsk_{\mathcal{A}}} = \mathsf{Ext}_1(lsk_{\mathcal{A}}, r_1), & \widehat{lsk_{\mathcal{B}}} = \mathsf{Ext}_1(lsk_{\mathcal{B}}, r'_1), \\
\widehat{esk_{\mathcal{A}}} = \mathsf{Ext}_2(e, r_2), & \widehat{esk_{\mathcal{B}}} = \mathsf{Ext}_2(e', r'_2), \\
(r, w) = \widehat{F}_{\widehat{lsk_{\mathcal{A}}}}(e) + \overline{F}_{\widehat{esk_{\mathcal{A}}}}(r_1), & (r', w') = \widehat{F}_{\widehat{lsk_{\mathcal{B}}}}(e') + \overline{F}_{\widehat{esk_{\mathcal{B}}}}(r'_1), \\
W = g^{w}, X = g^{r}, & W' = g^{w'}, X' = g^{r'},
\end{array}$$

$$\xrightarrow{(\widehat{B}, \widehat{A}, W, X)}$$

$$\xleftarrow{(\widehat{A}, \widehat{B}, W', X')}$$

**Session Key Ouput**

$$\begin{array}{cc}
\text{Set sid} = (\widehat{A}, \widehat{B}, W, X, W', X') & \text{Set sid} = (\widehat{A}, \widehat{B}, W, X, W', X') \\
d = H_1(\mathsf{sid}), K_{\mathcal{A}_1} = f(X'^{r}) \in \mathbb{Z}_N, & d = H_1(\mathsf{sid}), K_{\mathcal{B}_1} = f(X^{r'}) \in \mathbb{Z}_N, \\
K_{\mathcal{A}_2} = f(\mathsf{hp}_1'^{w}\mathsf{hp}_2'^{dw}) \in \mathbb{Z}_N, K_{\mathcal{A}_3} = f(W'^{\alpha + d\beta}) \in \mathbb{Z}_N, & K_{\mathcal{B}_2} = f(W^{\alpha' + d\beta'}) \in \mathbb{Z}_N, K_{\mathcal{B}_3} = f(\mathsf{hp}_1^{w'}\mathsf{hp}_2^{dw'}) \in \mathbb{Z}_N, \\
s_{\mathcal{A}} = Ext_3(K_{\mathcal{A}_1} \oplus K_{\mathcal{A}_2} \oplus K_{\mathcal{A}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}}), & s_{\mathcal{B}} = Ext_3(K_{\mathcal{B}_1} \oplus K_{\mathcal{B}_2} \oplus K_{\mathcal{B}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}}), \\
SK_{\mathcal{A}} = \widetilde{F}_{s_{\mathcal{A}}}(\mathsf{sid}). & SK_{\mathcal{B}} = \widetilde{F}_{s_{\mathcal{B}}}(\mathsf{sid}).
\end{array}$$

**Fig. 3.** CLR-eCK secure AKE Protocol Based on DCR Assumption

The following result can be immediately obtained from [15].

**Theorem 5.** *$\mathcal{SPHF}_{\mathsf{DCR}}$ is projective and 2*-smooth.

**KEM.** We can also easily obtain an IND-CPA secure KEM scheme based on the DCR assumption as follows where $g, \mathcal{K}, \mathcal{W}, f$ are as defined above:

- KeyGen: randomly choose $x \in \mathcal{K}$ and return $(SK = x, PK = g^x)$;
- Enc$(PK)$: randomly choose $w \in \mathcal{W}$ and return $(CT = g^w, K = f(PK^w))$;
- Dec$(SK, CT)$: return $K = f(W^x)$.

**A Concrete AKE Protocol based on DCR.** We can instantiate our generic construction using the $\mathcal{SPHF}_{\mathsf{DCR}}$ and the KEM scheme describe above. The resulting AKE protocol is presented in Fig. 3. Same as the previous DDH based protocol, the DCR based protocol also requires only one round.

**Parameters of The DCR-based Protocol.** In the DCR based protocol presented in Fig. 3, $N = pq$ where $p, q$ are both $k + 1$ bits safe primes. The generator $g$ can be picked by choosing

$\mu \overset{\$}{\leftarrow} \mathbb{Z}^*_{N^2}$ and setting $g = -\mu^{2N}$. For the $\mathcal{SPHF}_{\mathsf{DCR}}$, we have that $\mathcal{Y} = \mathbb{Z}_N, \mathcal{HK} = \mathcal{K}^2$ where $\mathcal{K} = \{0,1,\cdots,N^2/2\}$, $\mathcal{HP} = \mathcal{L}^2_{\mathsf{DCR}}, \mathcal{AUX} = \mathcal{W} = \{0,1,\cdots,N/2\}$. Choose a collision-resistant hash function $H_1 : \{0,1\}^* \to \mathcal{W}$ and strong extractors as follows: $\mathsf{Ext}_1 : \mathcal{K}^2 \times \{0,1\}^{t_1(k)} \to \{0,1\}^{l_1(k)}$ an average-case $(2 \cdot \log(N^2/2) - \lambda_1, \epsilon_1)$-strong extractor, $\mathsf{Ext}_2 : \{0,1\}^{u(k)} \times \{0,1\}^{t_2(k)} \to \{0,1\}^{l_2(k)}$ an average-case $(u(k) - \lambda_2, \epsilon_2)$-strong extractor and $\mathsf{Ext}_3 : \mathbb{Z}_N \times \{0,1\}^{t_3(k)} \to \{0,1\}^{l_3(k)}$ an average-case $(\log N - \lambda_1, \epsilon_3)$-strong extractor. Choose $\widehat{F} \leftarrow \widehat{\mathsf{F}}^{k,\Sigma_{\widehat{\mathsf{F}}},\mathcal{D}_{\widehat{\mathsf{F}}},\mathcal{R}_{\widehat{\mathsf{F}}}}$, $\overline{F} \leftarrow \overline{\mathsf{F}}^{k,\Sigma_{\overline{\mathsf{F}}},\mathcal{D}_{\overline{\mathsf{F}}},\mathcal{R}_{\overline{\mathsf{F}}}}$ and $\widetilde{F} \leftarrow \widetilde{\mathsf{F}}^{k,\Sigma_{\widetilde{\mathsf{F}}},\mathcal{D}_{\widetilde{\mathsf{F}}},\mathcal{R}_{\widetilde{\mathsf{F}}}}$. The map $f : \mathbb{Z}_{N^2} \to \mathbb{Z}_N$ is defined as above.

Based on **Theorem 1**, **Theorem 4** and **Theorem 5**, we have the following result for the concrete AKE protocol based on DCR.

**Corollary 2.** *The concrete AKE protocol in Fig. 3 is* $(\lambda_1, \lambda_2)$-CLR-eCK-*secure under the D-CR assumption, where* $\lambda_1 \leq \min\{2 \cdot \log(N^2/2) - 2\log(1/\epsilon_1) - l_1(k), \log N - 2\log(1/\epsilon_3) - l_3(k)\}, \lambda_2 \leq u(k) - 2\log(1/\epsilon_2) - l_2(k)$.

## 5 Symmetric-key Based Construction

In this section, we adapt our generic construction in the symmetric-key setting.

### 5.1 Adapting The Security Model

Before we present the new construction, we need to first adapt the security model to the symmetric-key setting: since two users $\mathcal{A}$ and $\mathcal{B}$ share the same long-term secret key, if there is any restriction to the adversary in querying one of the users' long-term secret key, the restriction must be applied to the long-term secret keys of both users. Specifically, the following changes on the security model are needed:

- In the *fresh session* definition of the eCK model given in Definition 7, if the matching session $\overline{\mathsf{sid}^*}$ of the test session $\mathsf{sid}^*$ does not exist, then the adversary is not allowed to issue either $\mathsf{LongTermKeyReveal}(\mathcal{A})$ or $\mathsf{LongTermKeyReveal}(\mathcal{B})$.
- In the *fresh session* definition of the $(\lambda_1, \lambda_2)$-Leakage CLR-eCK model given in Definition 9,
  - the total output length of all the $\mathsf{LongTermKeyReveal}(\mathcal{A})$ and $\mathsf{LongTermKeyReveal}(\mathcal{B})$ queries is at most $\lambda_1$;
  - before the adversary makes either $\mathsf{LongTermKeyReveal}(\mathcal{A})$ or $\mathsf{LongTermKeyReveal}(\mathcal{B})$ query, the adversary must commit ephemeral key leakage function sets $\mathcal{F}_2$ (with regards to the test session $\mathsf{sid}^*$) and $\mathcal{F}'_2$ (with regards to the matching session $\overline{\mathsf{sid}^*}$, if it exists).

### 5.2 Symmetric-key Based Construction

Our symmetric-key based CLR-eCK secure AKE scheme is presented in Fig. 4. The construction is under the common reference string (CRS) model where $r_1$ and $r_2$ are two randomly chosen public strings.

The security proof can be easily obtained by following the proof of the public-key based construction (i.e., by considering two cases based on whether $\overline{\mathsf{sid}^*}$ exists or not). We should note that in this construction we only require $\widehat{F}, \overline{F}, \widetilde{F}, \check{F}$ to be normal PRFs.

**Common Reference String**

$$r_1 \overset{\$}{\leftarrow} \{0,1\}^{t_1(k)}, r_2 \overset{\$}{\leftarrow} \{0,1\}^{t_2(k)}$$

**Session Execution**

$$\begin{array}{ccc}
\mathcal{A} \ \ (lsk_{\mathcal{A}}) & & \mathcal{B} \ \ (lsk_{\mathcal{B}}) \\
esk_{\mathcal{A}} \overset{\$}{\leftarrow} \{0,1\}^{u(k)}, & & esk_{\mathcal{B}} \overset{\$}{\leftarrow} \{0,1\}^{u(k)}, \\
\widehat{lsk}_{\mathcal{A}} = \mathsf{Ext}_1(lsk_{\mathcal{A}}, r_1), & & \widehat{lsk}_{\mathcal{B}} = \mathsf{Ext}_1(lsk_{\mathcal{B}}, r_1), \\
\widehat{esk}_{\mathcal{A}} = \mathsf{Ext}_2(esk_{\mathcal{A}}, r_2), & & \widehat{esk}_{\mathcal{B}} = \mathsf{Ext}_2(esk_{\mathcal{B}}, r_2), \\
\lambda_{\mathcal{A}} = \widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(esk_{\mathcal{A}}) + \overline{F}_{\widehat{esk}_{\mathcal{A}}}(r_1), & & \lambda_{\mathcal{B}} = \widehat{F}_{\widehat{lsk}_{\mathcal{B}}}(esk_{\mathcal{B}}) + \overline{F}_{\widehat{esk}_{\mathcal{B}}}(r_1), \\
(PK_{\mathcal{A}}, SK_{\mathcal{A}}) = \mathcal{KEM}.\mathsf{KeyGen}(1^k; \lambda_{\mathcal{A}}) & & \\
 & \xrightarrow{(\mathcal{B}, \mathcal{A}, PK_{\mathcal{A}})} & \\
 & & (CT_{\mathcal{B}}, K_{\mathcal{B}_1}) = \mathcal{KEM}.\mathsf{Enc}(PK_{\mathcal{A}}; \lambda_{\mathcal{B}}) \\
 & \xleftarrow{(\mathcal{A}, \mathcal{B}, CT_{\mathcal{B}})} &
\end{array}$$

**Session Key Ouput**

$$\begin{array}{cc}
\text{Set sid} = (\mathcal{A}, \mathcal{B}, PK_{\mathcal{A}}, CT_{\mathcal{B}}) & \text{Set sid} = (\mathcal{A}, \mathcal{B}, PK_{\mathcal{A}}, CT_{\mathcal{B}}) \\
aux = H_1(\mathsf{sid}), K_{\mathcal{A}_1} = \mathcal{KEM}.\mathsf{Dec}(SK_{\mathcal{A}}, CT_{\mathcal{B}}), & aux = H_1(\mathsf{sid}), \\
SK_{\mathcal{A}} = \widetilde{F}_{K_{\mathcal{A}_1}}(aux) \oplus \check{F}_{\widehat{lsk}_{\mathcal{A}}}(aux). & SK_{\mathcal{B}} = \widetilde{F}_{K_{\mathcal{B}_1}}(aux) \oplus \check{F}_{\widehat{lsk}_{\mathcal{B}}}(aux)
\end{array}$$

**Fig. 4.** Symmetric-key Based (i.e., $lsk_{\mathcal{A}} = lsk_{\mathcal{B}}$) CLR-eCK secure AKE

## 5.3 Protocol Instantiation

Similar to the public-key based generic construction, our symmetric-key based construction given in Fig. 4 can also be instantiated under various standard assumptions. It is easy to see that the latter actually has more options for the instantiation since we only need to provide a concrete KEM instantiation that is IND-CPA secure. If we use the same KEM instantiations based on the DDH or DCR assumption given in Sec. 4 then we will also obtain a one-round protocol where the $\mathcal{A}$ and $\mathcal{B}$ can exchange their messages simultaneously.

From our symmetric-key based construction, we are also able to derive CLR-eCK secure AKE protocols that can resist attacks from quantum computers. For instance, we can use the efficient IND-CPA secure KEM scheme based on the Ring Learning With Errors (RLWE) assumption by Peikert [36]. The resulting AKE scheme will have comparable performance with those instantiated under DDH and DCR assumptions [11] except that the RLWE based scheme is two-round (i.e., $\mathcal{B}$ needs to wait for the first message from $\mathcal{A}$ before sending his own message).

## 6 Conclusion

In this paper, we revisited leakage-resilient authenticated key exchange (AKE). 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 challenge-dependent leakage of both long-term and ephemeral secret keys. We also proposed two generic frameworks, one under public-key setting and the other under symmetric-key setting, for constructing efficient AKE protocols that are resilient to key exposure and leakage attacks, and presented several efficient instantiations of the general frameworks under various standard assumptions.

**Acknowledgements.** We would like to thank Janaka Alawatugoda and the anonymous reviewers of CT-RSA'16 for their invaluable comments on a previous version of this paper. The work of Guomin Yang is supported by the Australian Research Council Discovery Early Career Researcher Award (Grant No. DE150101116). The work of Yi Mu is supported by the National Natural Science Foundation of China (Grant No. 61170298).

## 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., Stebila, D., Boyd, C.: Modelling after-the-fact leakage for key exchange. In: ASIACCS. pp. 207–216 (2014)
5. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: CRYPTO. pp. 36–54 (2009)
6. 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)
7. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: CRYPTO. pp. 232–249 (1993)
8. 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)
9. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: CRYPTO. pp. 513–525 (1997)
10. Bitansky, N., Canetti, R., Halevi, S.: Leakage-tolerant interactive protocols. In: TCC. pp. 266–284 (2012)
11. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: IEEE Symposium on Security and Privacy. pp. 553–570 (2015)
12. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: EUROCRYPT. pp. 453–474 (2001)
13. Chen, R., Mu, Y., Yang, G., Susilo, W., Guo, F.: Strongly leakage-resilient authenticated key exchange. In: CT-RSA. pp. 19–36 (2016)
14. Choo, K.R., Boyd, C., Hitchcock, Y.: Examining indistinguishability-based proof models for key establishment protocols. In: ASIACRYPT. pp. 585–604 (2005)
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. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Information Theory 22(6), 644–654 (1976)
18. 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)
19. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: STOC. pp. 621–630 (2009)
20. 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)
21. Faust, S., Hazay, C., Nielsen, J.B., Nordholt, P.S., Zottarel, A.: Signature schemes secure against hard-to-invert leakage. In: ASIACRYPT. pp. 98–115 (2012)
22. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Cryptographic Hardware and Embedded Systems. pp. 251–261. No. Generators (2001)
23. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: EUROCRYPT. pp. 524–543 (2003)
24. 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)
25. Halevi, S., Kalai, Y.T.: Smooth projective hashing and two-message oblivious transfer. J. Cryptology 25(1), 158–193 (2012)
26. Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: TCC. pp. 107–124 (2011)
27. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: ASIACRYPT. pp. 703–720 (2009)
28. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: TCC. pp. 293–310 (2011)
29. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: CRYPTO. pp. 388–397 (1999)

30. 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)
31. LaMacchia, B.A., Lauter, K.E., Mityagin, A.: Stronger security of authenticated key exchange. In: Provable Security. pp. 1–16 (2007)
32. Marvin, R.: Google admits an android crypto prng flaw led to bitcoin heist (august 2013). `http://sdt.bz/64008`.
33. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: TCC. pp. 278–296 (2004)
34. Moriyama, D., Okamoto, T.: Leakage resilient eck-secure key exchange protocol without random oracles. In: ASIACCS
35. Okamoto, T.: Authenticated key exchange and key encapsulation in the standard model. In: ASIACRYPT. pp. 474–484 (2007)
36. Peikert, C.: Lattice cryptography for the internet. In: PQCrypto. pp. 197–219 (2014)
37. Quisquater, J., Samyde, D.: Electromagnetic attack. In: Encyclopedia of Cryptography and Security, 2nd Ed., pp. 382–385 (2011)
38. 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`.
39. 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)
40. 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)
41. Zetter, K.: How a crypto 'backdoor' pitted the tech world against the nsa. `http://www.wired.com/threatlevel/2013/09/nsa-backdoor/all/`.