12-2019

Strongly leakage resilient authenticated key exchange, revisited

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Rongmao CHEN

Yi MU

Willy SUSILO

GUO Fuchun

See next page for additional authors

Designs, Codes and Cryptography (2019) 87:2885–2911
https://doi.org/10.1007/s10623-019-00656-3

Strongly leakage resilient authenticated key exchange, revisited

Guomin Yang[2] · Rongmao Chen[1] · Yi Mu[2] · Willy Susilo[2] · Fuchun Guo[2] · Jie Li[1]

Received: 6 September 2018 / Revised: 4 March 2019 / Accepted: 11 June 2019 / Published online: 22 June 2019

Abstract
Authenticated Key Exchange (AKE) protocols allow two (or multiple) parties to authenticate each other and agree on a common secret key, which is essential for establishing a secure communication channel over a public network. AKE protocols form a central component in many network security standards such as IPSec, TLS/SSL, and SSH. However, it has been demonstrated that many standardized AKE protocols are vulnerable to side-channel and key leakage attacks. In order to defend against such attacks, leakage resilient (LR-) AKE protocols have been proposed in the literature. Nevertheless, most of 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 the use of poor randomness sources or insecure pseudo-random number generators (PRNGs). In this paper, we revisit the strongly leakage resilient AKE protocol (CT-RSA'16) that aimed to resist challenge-dependent leakage on both long-term and ephemeral secret keys. We show that there is a security issue in the design of the protocol and propose an improved version that can fix the problem. In addition, we extend the protocol to a more general framework that can be efficiently instantiated under various assumptions, including hybrid instantiations that can resist key leakage attacks while preserving session key security against future quantum machines.

Keywords Authenticated key exchange · Key leakage · Weak randomness

Mathematics Subject Classification 94A60 · 14G50

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

Communicated by C. Blundo.

This paper is an extended version of a preliminary work published in the proceedings of CT-RSA 2016 [11]. In this version, we fix a security flaw in the protocol presented in [11] and extends the protocol to a more general framework.

Extended author information available on the last page of the article

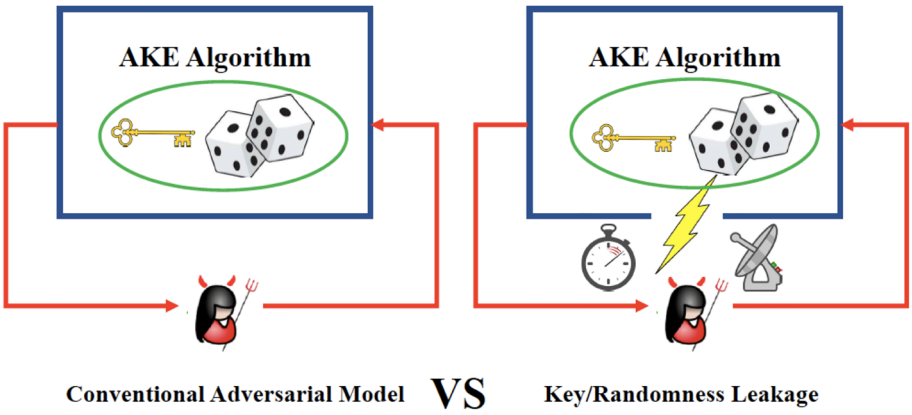

Fig. 1 Side-channel/secret-leakage attacks against AKE

methods, a shared secret key needs to be established among those parties. Since Diffie and Hellman [16] 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) [10,20] and the Internet Key Exchange (a.k.a. SIGMA) protocol [27] for IPSec, 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 user 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 [7,23,26,35] and memory attacks [1,24] voided the aforementioned assumptions underlying the traditional AKE designs. In such attacks (illustrated in Fig. 1), 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 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 [10] and eCK [28]). In particular, we are interested in the situation that one of those keys is completely exposed to the adversary while the other key is partially leaked.

1.1 Capturing key leakage in AKE

There have been a few works on the modeling and construction of leakage-resilient AKE [3,4,19,32]. However, there are some limitations in these early works. The general approach for modeling secret leakage in a security protocol is that in the adversarial game, in addition to the normal black-box interaction with an honest party executing the protocol, the attacker can also learn some partial information of a user secret via an abstract leakage function f which simulates the real key leakage attacks by different means (e.g., timing, electromagnetic radiation, etc.). Meanwhile, 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. 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 the adversarial game defined in the security model. The previous security definitions for leakage-resilient AKE, e.g., [4,19,32, 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 [32]) 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 [25], Alawatugoda et al. [3] 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.

Moreover, the aforementioned leakage-resilient security notions for AKE 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 [30,37,40].

Motivated by the limitations in the existing LR-AKE models and constructions, Chen et al. [11] proposed 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, and a new AKE protocol that aimed to achieve their security goals. Subsequently, they also extended their work from the relative/bounded leakage setting to the auxiliary input setting [12]. However, the model under the auxiliary input setting presented in [12] is non-adaptive, which means the adversary is required to commit all the leakage functions at the beginning of the adversarial game. On the other hand, the model under the relative leakage setting given in [11] considers an adaptive adversary. Hence, the two models given in [11] and [12] are incomparable.

1.2 Our results

In this paper, we revisit the CLR-eCK secure AKE construction presented in [11] and show that there is a security problem in the design of the protocol. Specifically, we show that the protocol is insecure when the adversary is allowed to make some challenge-dependent leakage queries that are permitted in the model given in [11]. We then present an improved protocol that can fix the problem.[1]

In addition, we present a more generic framework for the construction of CLR-eCK secure AKE protocols. Compared with the original construction given in [11], the new framework allows more flexible instantiations under various assumptions. Specifically, we show that the

[1] It is worth noting that the identified problem is not applicable to [12] due to the reason that the leakage model considered in [12] is non-adaptive.

Table 1 Comparison with existing leakage-resilient AKE protocols

Protocols	Round	Relative leakage[b]		Security	AKE models
		lsk	esk		
eSIG-DH [4]	3	$(1-\varepsilon)$	0	w/ RO	BRM-CK [4]
Enc-DH [19]	3	$(1-\varepsilon)$	0	w/o RO	LR-CK [19]
MO [32]	2	$(1/4-\varepsilon)$	0	w/o RO	LR-eCK [32]
π [3]	2	$(1/n-\varepsilon)$	0	w/o RO	BAFL-eCK [3]
Our framework	2[a]	$(1/4-\varepsilon)$	$(1-\varepsilon)$	w/o RO	CLR-eCK

[a] In some cases, our protocol could be only one-round (e.g., Sects. 4.4.1 and 4.4.2).
[b] The "Relative Leakage" column indicates the leakage ratio of a secret key. We use lsk to denote the long-term secret key and esk the ephemeral secret key. In [3], the secret key is split into n parts

extended framework allows concrete instantiations under a sole number theoretic assumption, as well as hybrid instantiations that can resist key leakage attacks and provide forward secrecy against (future) quantum machines. A comparison between our framework and existing leakage-resilient AKE protocols is given in Table 1.

1.3 Related work

Traditional AKE security notions Bellare and Rogaway (BR) [5] 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 [10], 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. [28] 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 [13,15] for some detailed analysis and comparisons among these models.

Modeling leakage resilience The modeling of leakage attacks in an abstract way was first proposed by Micali and Reyzin [31]. Inspired by the cold boot attack presented by Halderman et al. [24], Akavia et al. [1] 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. [4] 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. [18] where the leakage function can be any one-way function (i.e., there is no bound on the leakage size).

Leakage-resilient AKE Alwen et al. [4] 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 [19], Dodis et al. also proposed leakage-resilient AKE constructions

based on different primitives. Their first construction is similar as [4], 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 [32] proposed another leakage-resilient model for AKE. Their proposed notion, named λ-leakage resilient eCK where λ denotes the leakage size, is an extension of the eCK model by incorporating the relative leakage introduced in [1]. 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]. Alawatugoda et al. [3] modeled after-the-fact leakage for AKE protocols and proposed a construction that implicitly assumes the long-term secret key has split-state. In [11,12], Chen et al. proposed the challenge-dependent leakage-resilient eCK models under different leakage settings. Specifically, they proposed a model with adaptive adversaries under the relative leakage setting in [11], and another (incomparable) model with non-adaptive adversaries under the auxiliary input setting in [12].

AKE under bad randomness Yang et al. [38] extended the CK model to consider AKE under bad randomness. They considered two bad randomness scenarios, namely randomness reset and adversarial-controlled randomness, and pointed out that no AKE protocol can be secure against reset-and-replay attacks. They then introduced some generic methods for enhancing the security of some existing AKE protocols such as SIG-DH under the bad randomness setting. Whereas [38] considered the worst-case randomness failure, they didn't consider leakages of the long-term secret key. Noting that the result of [38] only holds for stateless AKE protocols, Feltz et al.[22] proposed novel stateful AKE protocols that can provide resilience even against the worst case randomness failure. In particular, they constructed variants of the NAXOS protocol [28] that are secure against reset-and-replay attacks.

2 Preliminaries

2.1 Notation

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

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

Computational indistinguishability Let $\mathcal{V}_0$ and $\mathcal{V}_1$ be two probability distributions over a finite set Ω where $|\Omega| \geq 2^k$ 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 γ. 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

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. [17], 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. [17].

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

Definition 1 (*Average-case strong extractor*) [17] 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, ϵ)-*strong extractor* if for all pairs of random variables (X, I) such that $X \in \{0,1\}^{n(k)}$ and $\widetilde{\mathrm{H}}_\infty(X|I) \geq m$, it holds that

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

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

2.3 Pseudo-random function

PRF Let $k \in \mathbb{N}$ be a security parameter and F be 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 F is a *pseudo-random function* (PRF) family if

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

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

πPRF [33]. 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 σ_{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 (πPRF) Define $\widetilde{\mathsf{F}}(\rho_j) = \mathsf{F}_{\sigma_{i_j}}^{k,\sum,\mathcal{D},\mathcal{R}}(\rho_j)$ for $i_j \in I_\sum$, $\rho_j \in \mathcal{D}$. We say that F is a πPRF family if

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

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

2.4 Smooth projective hash function

Syntax A smooth projective hash function (SPHF) [14] 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 param and the description of an $\mathcal{NP}$ language $\mathcal{L}$;
- $\mathsf{WordG}(\mathcal{L}, \mathsf{param}, w)$: a deterministic algorithm that generates a word $W \in \mathcal{L}$ from the witness w;
- $\mathsf{HashKG}(\mathcal{L}, \mathsf{param})$: a probabilistic algorithm that generates a (secret) hashing key hk;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}))$: a deterministic algorithm that derives the (public) projection key hp from hk;
- $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W, aux)$: 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:

$$\begin{aligned}\mathcal{V}_1 = \{&(\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, aux, \mathsf{hv})| \\ &\mathsf{hv} = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W, aux)\} \\ \mathcal{V}_2 = \{&(\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, aux, \mathsf{hv})|\mathsf{hv} \xleftarrow{\$} \mathcal{Y}\}.\end{aligned}$$

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:

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

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 $W \xleftarrow{\$} \mathcal{L}$ is computationally indistinguishable from the distribution $W' \xleftarrow{\$} \mathcal{X}\backslash\mathcal{L}$.

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 encapsulation and decapsulation key pair (EK, DK);
- $\mathsf{Enc}(EK)$: a probabilistic algorithm that generates a ciphertext CT and a session key K encapsulated in CT;
- $\mathsf{Dec}(DK, CT)$: 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

$$\{EK, CT, K\} \stackrel{c}{\equiv} \{EK, CT, R\}$$

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

3 Strongly leakage-resilient AKE security model

3.1 AKE protocol

A two-party AKE protocol is run between two parties. 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*. 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 $(\mathcal{A}, B, epk_{\mathcal{A}}, epk_{\mathcal{B}})$ in this paper, and the session $(\mathcal{B}, A, epk_{\mathcal{B}}, epk_{\mathcal{A}})$ 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 et al. [28] based on the CK model by Canetti and Krawczyk [10]. 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 $\mathcal{M}$ controls all the communications among the parties and is also responsible for activating all the protocol sessions. Under the public-key setting, $\mathcal{M}$ is also given 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 PK on behalf of a party 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 PK.
- $\mathsf{LongTermKeyReveal}(\mathsf{pid})$. This query allows the adversary to learn the long-term secret key of an honest party 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 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 $b \xleftarrow{\$} \{0, 1\}$. If $b = 1$, the challenger returns $SK^* \leftarrow \mathsf{SessionKeyReveal}(\mathsf{sid}^*)$. Otherwise, the challenger sends $\mathcal{M}$ a random key $R^* \xleftarrow{\$} \{0, 1\}^{|SK^*|}$.

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

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

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

Definition 8 (*eCK security*) Let the test session 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

The *Challenge-Dependent Leakage-Resilient eCK* (CLR-eCK) model [11] 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.

- LongTermKeyLeakage(f_1, 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.
- EphemeralKeyLeakage(f_2, sid). This query allows $\mathcal{M}$ to learn $f_2(esk)$ where f_2 denotes the leakage function and esk denotes the ephemeral secret key used by an honest user in the session sid.

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

Under the relative leakage setting, 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 $\mathcal{F}_{\mathsf{bbd\text{-}l}}$ for the long-term secret key and $\mathcal{F}_{\mathsf{bbd\text{-}ll}}$ for the ephemeral secret key as follows. $\mathcal{F}_{\mathsf{bbd\text{-}l}}(k)$ is defined as the class of all polynomial-time computable functions: $f : \{0,1\}^{|lsk|} \rightarrow \{0,1\}^{\leq \lambda_1(k)}$, where $\lambda_1(k) < |lsk|$. $\mathcal{F}_{\mathsf{bbd\text{-}ll}}(k)$ 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 $f_1 \in \mathcal{F}_{\mathsf{bbd\text{-}l}}$ and $f_2 \in \mathcal{F}_{\mathsf{bbd\text{-}ll}}$.

Another necessary restriction is related to the challenge-dependent leakage security of AKE protocols. Consider a test session sid^* which is owned by party $\mathcal{A}$ with peer $\mathcal{B}$. Note that for a 2-pass AKE protocol, the session key of sid^* is determined by $(\mathcal{A}, B, lsk_{\mathcal{A}}, esk^*_{\mathcal{A}}, lpk_{\mathcal{B}}, epk^*_{\mathcal{B}})$ 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 LongTermKeyLeakage(f_1, $\mathcal{A}$) and EphemeralKeyLeakage(f_2, sid^*) as follows.

- $\mathcal{M}$ is allowed to query an arbitrary leakage function $f_1 \in \mathcal{F}_{\mathsf{bbd\text{-}l}}$ before it obtains the ephemeral secret key $esk^*_{\mathcal{A}}$ via an EphemeralKeyReveal(sid^*) query; however, after obtaining $esk^*_{\mathcal{A}}$, $\mathcal{M}$ can only query the leakage functions $f_1 \in \mathcal{F}_1 \subset \mathcal{F}_{\mathsf{bbd\text{-}l}}$ where $\mathcal{F}_1$ is a leakage functions set chosen and submitted by $\mathcal{M}$ before issuing EphemeralKeyReveal(sid^*).
- $\mathcal{M}$ is allowed to query an arbitrary leakage function $f_2 \in \mathcal{F}_{\mathsf{bbd\text{-}ll}}$ before it obtains the long-term secret key $lsk_{\mathcal{A}}$ via a LongTermKeyReveal($\mathcal{A}$) query; however, after obtaining $lsk_{\mathcal{A}}$, $\mathcal{M}$ can only query the leakage functions $f_2 \in \mathcal{F}_2 \subset \mathcal{F}_{\mathsf{bbd\text{-}ll}}$ where $\mathcal{F}_2$ is a set of leakage functions chosen and submitted by $\mathcal{M}$ before it issues LongTermKeyReveal($\mathcal{A}$).
- If the matching session $\overline{\mathsf{sid}^*}$ of the test session exists, the above restrictions also apply to the leakage queries LongTermKeyLeakage(f_1, $\mathcal{B}$) and EphemeralKeyLeakage (f_2, $\overline{\mathsf{sid}^*}$).

Remarks One can see that the above 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 $(lpk_{\mathcal{A}}, lpk_{\mathcal{B}}, epk^*_{\mathcal{A}}, epk^*_{\mathcal{B}})$. The model essentially gives $\mathcal{M}$ the most freedom in making key reveal and key leakage queries.

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

Definition 9 (*Session freshness in the* (λ_1, λ_2)*-leakage* CLR-eCK *model*) Let sid be a completed session owned by an honest party $\mathcal{A}$ with peer $\mathcal{B}$, who is also honest. Let $\overline{\mathsf{sid}}$ denote the matching session of sid, if it exists. Session sid is said to be fresh in the CLR-eCK model if the following conditions hold:

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

We now present the notion of CLR-eCK security.

Definition 10 (CLR-eCK *security*) Let the test session sid^* be fresh according to the above definition. We define the advantage of $\mathcal{M}$ in the CLR-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 (λ_1, λ_2)-CLR-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.

4 CLR-eCK-secure AKE

In this section, we present a generic construction of CLR-eCK-secure AKE under the public-key setting. Our construction fixes a security flaw in the construction given in [11] and is more generic (i.e., allowing various instantiations).

4.1 General framework

Figure 2 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.

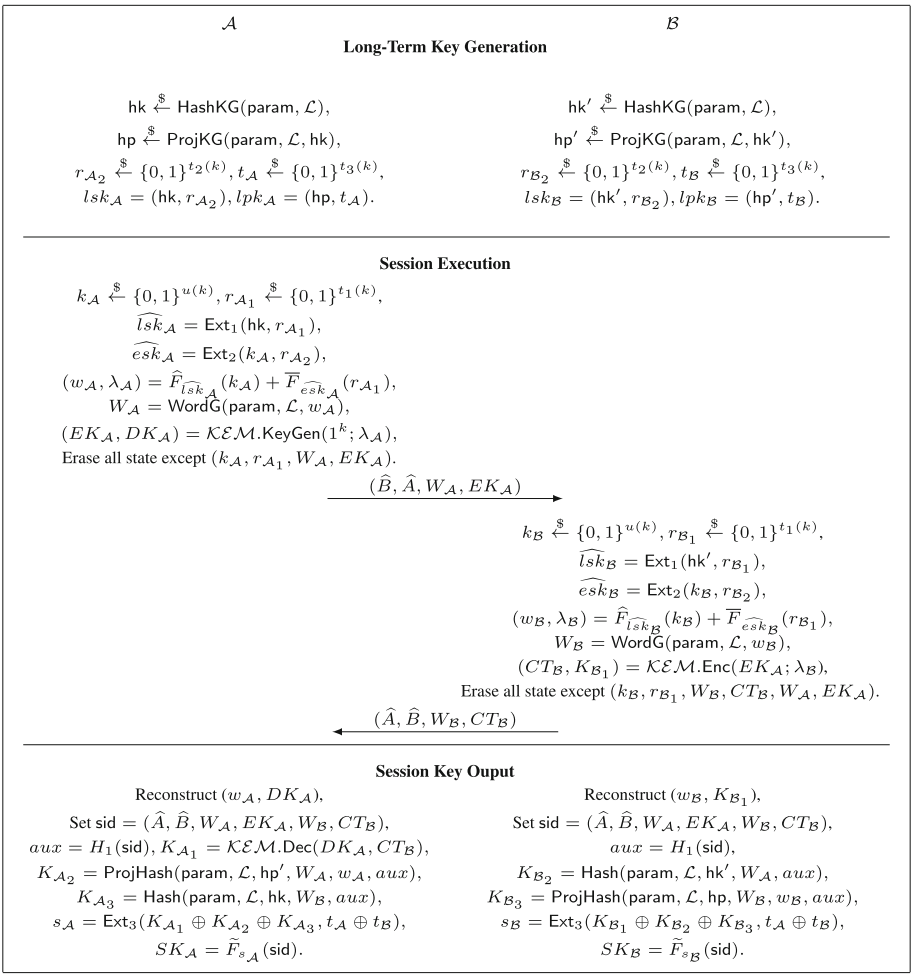

Fig. 2 CLR-eCK secure AKE framework

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 $(u(k) - \lambda_2, \epsilon_2)$-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 πPRF family as follows.[2]

$$\widehat{\mathsf{F}} : \textstyle\sum_{\widehat{\mathsf{F}}} = \{0,1\}^{l_1(k)}, \mathcal{D}_{\widehat{\mathsf{F}}} = \{0,1\}^{u(k)}, \mathcal{R}_{\widehat{\mathsf{F}}} = \mathcal{W} \times \{0,1\}^{\zeta(k)},$$
$$\overline{\mathsf{F}} : \textstyle\sum_{\overline{\mathsf{F}}} = \{0,1\}^{l_2(k)}, \mathcal{D}_{\overline{\mathsf{F}}} = \{0,1\}^{t_1(k)}, \mathcal{R}_{\overline{\mathsf{F}}} = \mathcal{W} \times \{0,1\}^{\zeta(k)},$$
$$\widetilde{\mathsf{F}} : \textstyle\sum_{\widetilde{\mathsf{F}}} = \{0,1\}^{l_3(k)}, \mathcal{D}_{\widetilde{\mathsf{F}}} = \Lambda(k)^2 \times \mathcal{L}^2 \times \mathcal{PK} \times \mathcal{C} \times \{0,1\}^{2t_3(k)}, \mathcal{R}_{\widetilde{\mathsf{F}}} = \{0,1\}^{l_4(k)}.$$

[2] 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 $\zeta(k)$.

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

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

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 $k_{\mathcal{A}} \xleftarrow{\$} \{0,1\}^{u(k)}$, $r_{\mathcal{A}_1} \xleftarrow{\$} \{0,1\}^{t_1(k)}$.
2. Set $\widehat{lsk}_{\mathcal{A}} = \mathsf{Ext}_1(lsk_{\mathcal{A}}, r_{\mathcal{A}_1})$, $\widehat{esk}_{\mathcal{A}} = \mathsf{Ext}_2(k_{\mathcal{A}}, r_{\mathcal{A}_2})$.
3. Compute $(w_{\mathcal{A}}, \lambda_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(k_{\mathcal{A}}) + \overline{F}_{\widehat{esk}_{\mathcal{A}}}(r_{\mathcal{A}_1})$.
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 $(EK_{\mathcal{A}}, DK_{\mathcal{A}})$.
5. Erase all state except $(k_{\mathcal{A}}, r_{\mathcal{A}_1}, W_{\mathcal{A}}, EK_{\mathcal{A}})$, set $(W_{\mathcal{A}}, EK_{\mathcal{A}})$ as the ephemeral public key and send $(\widehat{B}, \widehat{A}, W_{\mathcal{A}}, EK_{\mathcal{A}})$ to $\mathcal{B}$.

– $(\mathcal{B} \rightharpoonup \mathcal{A})$. Upon receiving the message from $\mathcal{A}$, $\mathcal{B}$ executes the following steps.

1. Select the ephemeral secret key $k_{\mathcal{B}} \xleftarrow{\$} \{0,1\}^{u(k)}$, $r_{\mathcal{B}_1} \xleftarrow{\$} \{0,1\}^{t_1(k)}$.
2. Set $\widehat{lsk}_{\mathcal{B}} = \mathsf{Ext}_1(lsk_{\mathcal{B}}, r_{\mathcal{B}_1})$, $\widehat{esk}_{\mathcal{B}} = \mathsf{Ext}_2(k_{\mathcal{B}}, r_{\mathcal{B}_2})$.
3. Compute $(w_{\mathcal{B}}, \lambda_{\mathcal{B}}) = \widehat{F}_{\widehat{lsk}_{\mathcal{B}}}(k_{\mathcal{B}}) + \overline{F}_{\widehat{esk}_{\mathcal{B}}}(r_{\mathcal{B}_1})$.
4. Run $\mathsf{WordG}(\mathsf{param}, \mathcal{L}, w_{\mathcal{B}})$ to obtain a word $W_{\mathcal{B}}$ and compute
$$(CT_{\mathcal{B}}, K_{\mathcal{B}_1}) = \mathcal{KEM}.\mathsf{Enc}(EK_{\mathcal{A}}; \lambda_{\mathcal{B}}).$$
5. Erase all state except $(k_{\mathcal{B}}, r_{\mathcal{B}_1}, W_{\mathcal{B}}, CT_{\mathcal{B}}, W_{\mathcal{A}}, EK_{\mathcal{A}})$, 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 $(\widehat{A}, \widehat{B}, W_{\mathcal{B}}, CT_{\mathcal{B}})$, $\mathcal{A}$ computes the session key as follows.

1. Reconstruct $(w_{\mathcal{A}}, DK_{\mathcal{A}})$ from $(lsk_{\mathcal{A}}, lpk_{\mathcal{A}}, k_{\mathcal{A}})$.
2. Sets $\mathsf{sid} = (\widehat{A}, \widehat{B}, W_{\mathcal{A}}, EK_{\mathcal{A}}, W_{\mathcal{B}}, CT_{\mathcal{B}}), aux = H_1(\mathsf{sid})$.
3. Compute $K_{\mathcal{A}_1} = \mathcal{KEM}.\mathsf{Dec}(DK_{\mathcal{A}}, CT_{\mathcal{B}})$, $K_{\mathcal{A}_2} = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, hp', W_{\mathcal{A}}, w_{\mathcal{A}}, aux)$, $K_{\mathcal{A}_3} = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, hk, W_{\mathcal{B}}, aux)$.
4. Set $s_{\mathcal{A}} = Ext_3(K_{\mathcal{A}_1} \oplus K_{\mathcal{A}_2} \oplus K_{\mathcal{A}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}})$.
5. Compute $SK_{\mathcal{A}} = \widetilde{F}_{s_{\mathcal{A}}}(\mathsf{sid})$.

Similarly, party $\mathcal{B}$ computes the session key as follows.

1. Reconstruct $(w_{\mathcal{B}}, K_{\mathcal{B}_1})$ from $(lsk_{\mathcal{B}}, lpk_{\mathcal{B}}, k_{\mathcal{B}}, r_{\mathcal{B}_1}, EK_{\mathcal{A}})$.
2. Set $\mathsf{sid} = (\widehat{A}, \widehat{B}, W_{\mathcal{A}}, EK_{\mathcal{A}}, W_{\mathcal{B}}, CT_{\mathcal{B}}), aux = H_1(\mathsf{sid})$.
3. Compute $K_{\mathcal{B}_2} = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, hk', W_{\mathcal{A}}, aux)$, $K_{\mathcal{B}_3} = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, hp, W_{\mathcal{B}}, w_{\mathcal{B}}, aux)$.
4. Set $s_{\mathcal{B}} = Ext_3(K_{\mathcal{B}_1} \oplus K_{\mathcal{B}_2} \oplus K_{\mathcal{B}_3}, t_{\mathcal{A}} \oplus t_{\mathcal{B}})$.
5. Compute $SK_{\mathcal{B}} = \widetilde{F}_{s_{\mathcal{B}}}(\mathsf{sid})$.

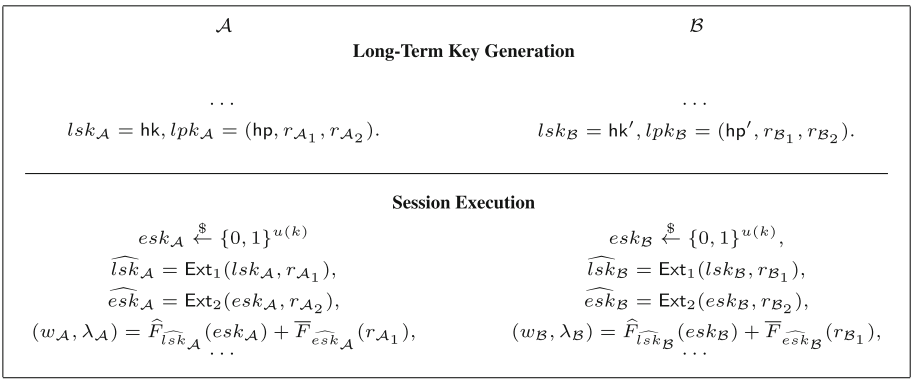

Fig. 3 The AKE framework proposed in [11]

Correctness analysis One can easily see that $K_{\mathcal{A}_1} = K_{\mathcal{B}_1}$ due to the correctness of $\mathcal{KEM}$. Due to the correctness of SPHF, we have

$$\begin{aligned} K_{\mathcal{A}_2} &= \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, hp', W_{\mathcal{A}}, w_{\mathcal{A}}, aux) \\ &= \mathsf{Hash}(\mathsf{param}, \mathcal{L}, hk', W_{\mathcal{A}}, aux) = K_{\mathcal{B}_2}, \\ K_{\mathcal{A}_3} &= \mathsf{Hash}(\mathsf{param}, \mathcal{L}, hk, W_{\mathcal{B}}, aux) \\ &= \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, hp, W_{\mathcal{B}}, w_{\mathcal{B}}, aux) = K_{\mathcal{B}_3}. \end{aligned}$$

Therefore, we can obtain that

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

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

4.2 Fixing a security issue in the construction in [11]

One of the differences between the construction presented in [11] and the one in Fig. 2 is that in [11], the keys $(r_{\mathcal{A}_1}, r_{\mathcal{A}_2})$ and $(r_{\mathcal{B}_1}, r_{\mathcal{B}_2})$ of the strong randomness extractors are given in the long-term public keys of $\mathcal{A}$ and $\mathcal{B}$ respectively (shown in Fig. 3).

As a result, the protocol given in [11] cannot be proved secure under the CLR-eCK model when the adversary performs the following queries in the game:

1. Based on the knowledge of $r_{\mathcal{A}_1}$ given in $\mathcal{A}$'s long-term public key, the adversary adds a function $f(\cdot) = \mathsf{Ext}_1(\cdot, r_{\mathcal{A}_1})_{[1,\cdots,\lambda_1(k)]}$ in the committed set of leakage functions $\mathcal{F}_1$.
2. The adversary makes an ephemeral key reveal query to learn $esk_{\mathcal{A}}$ of the test session, which allows the adversary to derive $\widehat{esk}_{\mathcal{A}}$ in the test session.
3. The adversary makes a leakage query on $f(lsk_{\mathcal{A}})$ to obtain $\lambda_1(k)$ bits of the $\widehat{lsk}_{\mathcal{A}}$ in the test session.

Since the adversary has the full knowledge of $\widehat{esk}_{\mathcal{A}}$ and also partial knowledge of $\widehat{lsk}_{\mathcal{A}}$, we cannot guarantee that $(w_{\mathcal{A}}, \lambda_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(esk_{\mathcal{A}}) + \overline{F}_{\widehat{esk}_{\mathcal{A}}}(r_{\mathcal{A}_1})$ is random and unknown to the adversary.

Fixing the problem In the modified construction given in Fig. 2, $r_{\mathcal{A}_2}$ ($r_{\mathcal{B}_2}$, respectively) is now part of the user long-term private key of $\mathcal{A}$ ($\mathcal{B}$, respectively), while $r_{\mathcal{A}_1}$ ($r_{\mathcal{B}_1}$, respectively) is part of the ephemeral secret key randomly chosen by $\mathcal{A}$ ($\mathcal{B}$, respectively) in each session. Such a modification can prevent the above problem due to the following reason:

1. If the adversary does not reveal the ephemeral secret key, which contains $(esk_{\mathcal{A}}, r_{\mathcal{A}_1})$, of the test session, then *before* the adversary reveals the long-term secret key, which contains $(hk, r_{\mathcal{A}_2})$, the adversary needs to commit the set $\mathcal{F}_2$ of leakage functions on $(esk_{\mathcal{A}}, r_{\mathcal{A}_1})$, and at that time, the adversary can only learn at most $\lambda_1(k)$ bits of information about $lsk_{\mathcal{A}} = (hk, r_{\mathcal{A}_2})$. If we set $t_2(k) \gg \lambda_1(k)$ (e.g., $t_2(k) - \lambda_1(k) = k$), then the function $\mathsf{Ext}_2(\cdot, r_{\mathcal{A}_2})$ is excluded from the set $\mathcal{F}_2$ of leakage functions the adversary can make after obtaining $lsk_{\mathcal{A}}$.
2. On the other hand, if the adversary reveals the ephemeral secret key $(esk_{\mathcal{A}}, r_{\mathcal{A}_1})$, then before that the adversary needs to commit the set of leakage functions $\mathcal{F}_1$ for $lsk_{\mathcal{A}}$, and at that time the adversary can only learn at most $\lambda_2(k)$ bits of information about $(esk_{\mathcal{A}}, r_{\mathcal{A}_1})$. If we set $t_1(k) \gg \lambda_2(k)$ (e.g., $t_1(k) - \lambda_2(k) = k$), then the function $\mathsf{Ext}_1(\cdot, r_{\mathcal{A}_1})$ is excluded from the set $\mathcal{F}_1$ of leakage functions the adversary can make after obtaining $(esk_{\mathcal{A}}, r_{\mathcal{A}_1})$.

4.3 Security analysis

Theorem 1 *The AKE protocol following the general framework is* (λ_1, λ_2)*-CLR-eCK-secure if the underlying smooth projective hash function is 2-smooth, the 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* π*PRF family. Here* $\lambda_1 \leq min\{|\mathcal{HK}| - 2\log(1/\epsilon_1) - l_1(k), |\mathcal{Y}| - 2\log(1/\epsilon_3) - l_3(k), t_2(k) - k\}$, $\lambda_2 \leq min\{u(k) - 2\log(1/\epsilon_2) - l_2(k), t_1(k) - k\}$.

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 $\mathcal{M}$. Wlog, assume $\mathcal{A}$ is the owner of the session sid^* and $\mathcal{B}$ is the peer. We then analyze the security of the AKE protocol in the following two disjoint cases.

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

We analyse the security based on the reveal and leakage queries that the adversary issues to the target session, its matching session and the corresponding parties.

– $\mathsf{LongTermKeyReveal}(\mathcal{A})$, $\mathsf{LongTermKeyReveal}(\mathcal{B})$.
In this sub-case, suppose that the adversary obtains at most λ_2 bits of the ephemeral secret key of target session sid^*. Before the adversary reveals the long-term secret key of $\mathcal{A}$, which contains $(hk, r_{\mathcal{A}_2})$, the adversary needs to commit the set $\mathcal{F}_2$ of leakage functions on $(k^*_{\mathcal{A}}, r^*_{\mathcal{A}_1})$, and at that time, the adversary can only learn at most λ_1 bits of information about $lsk_{\mathcal{A}} = (hk, r_{\mathcal{A}_2})$. Since $\lambda_1 \leq t_2(k) - k$, $\mathsf{Ext}_2(\cdot, r_{\mathcal{A}_2})$ is excluded from the set of leakage functions the adversary can make on the ephemeral secret key. Since at most λ_2 bits of $k^*_{\mathcal{A}}$ is leaked, based on the security of the strong randomness extractor Ext_2, we have that

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

Therefore, $(w^*_{\mathcal{A}}, \lambda^*_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}_{\mathcal{A}}}(k^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r^*_{\mathcal{A}_1}) \stackrel{c}{\equiv} (w'_{\mathcal{A}}, \lambda'_{\mathcal{A}}) \stackrel{\$}{\leftarrow} \mathcal{W} \times \{0,1\}^{\zeta(k)}$. Similarly, suppose that the adversary obtains at most λ_2 bits of the ephemeral secret key of matching session $\overline{\mathsf{sid}^*}$, we have that

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

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

– $\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*)$, $\mathsf{EphemeralKeyReveal}(\overline{\mathsf{sid}^*})$.
In this sub-case, before the adversary reveals the entire ephemeral secret key of sid^*, the adversary must first commit the set $\mathcal{F}_1$ of leakage functions on $lsk_{\mathcal{A}}$. Suppose that the adversary first obtains at most λ_2 bits of the ephemeral secret key of sid^* before committing the set $\mathcal{F}_1$, due to the condition that $\lambda_2 \leq t_1(k) - k$, $\mathsf{Ext}_1(\cdot, r^*_{\mathcal{A}_1})$ is excluded from the set $\mathcal{F}_1$. Since the adversary obtains at most λ_1-bits of the long-term secret key of $\mathcal{A}$, we have that

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

hence $(w^*_{\mathcal{A}}, \lambda^*_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{A}}}(k^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r^*_{\mathcal{A}_1}) \stackrel{c}{\equiv} (w'_{\mathcal{A}}, \lambda'_{\mathcal{A}}) \stackrel{\$}{\leftarrow} \mathcal{W} \times \{0,1\}^{\zeta(k)}$. Similarly, suppose that the adversary obtains at most λ_1 bits of the long-term secret key of party $\mathcal{B}$, we have that

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

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

– $\mathsf{LongTermKeyReveal}(\mathcal{A})$, $\mathsf{EphemeralKeyReveal}(\overline{\mathsf{sid}^*})$
In this sub-case, following the same analysis with regards to Eqs. (1) and (4), we have that

$$(w^*_{\mathcal{A}}, \lambda^*_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{A}}}(k^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r^*_{\mathcal{A}_1}) \stackrel{c}{\equiv} (w'_{\mathcal{A}}, \lambda'_{\mathcal{A}}) \stackrel{\$}{\leftarrow} \mathcal{W} \times \{0,1\}^{\zeta(k)}$$

and

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

– $\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*)$, $\mathsf{LongTermKeyReveal}(\mathcal{B})$. In this sub-case, following the same analysis with regards to Eqs. (2) and (3), we have that

$$(w^*_{\mathcal{A}}, \lambda^*_{\mathcal{A}}) = \widehat{F}_{\widehat{lsk}^*_{\mathcal{A}}}(k^*_{\mathcal{A}}) + \overline{F}_{\widehat{esk}^*_{\mathcal{A}}}(r^*_{\mathcal{A}_1}) \stackrel{c}{\equiv} (w'_{\mathcal{A}}, \lambda'_{\mathcal{A}}) \stackrel{\$}{\leftarrow} \mathcal{W} \times \{0,1\}^{\zeta(k)}$$

and

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

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 πPRF function is uniformly distributed and unknown to the adversary and thus the derived session key $SK^*_{\mathcal{A}}$ is computationally indistinguishable from a random string. It is worth noting that in this case we only require $\tilde{F}$ to be a normal PRF.

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

In this case, the adversary cannot issue LongTermKeyReveal query to reveal the long-term secret key of $\mathcal{B}$ but may issue the leakage query 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 $(w^*_{\mathcal{A}}, \lambda^*_{\mathcal{A}})$ by random elements from the respective spaces.
- Game 2: Replace

$$K^*_{\mathcal{A}_2} = \mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, hp', W^*_{\mathcal{A}}, w^*_{\mathcal{A}}, aux^*)$$

  by $K^*_{\mathcal{A}_2} = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, hk', W^*_{\mathcal{A}}, aux^*)$.
- Game 3: Choose $W^*_{\mathcal{A}} \in \mathcal{X} \setminus \mathcal{L}$ instead of deriving it from $\mathcal{L}$ through the algorithm WordG.

By following the same analysis as in Eqs. (1) and (3), we can see that Game 1 is indistinguishable from the original game. Game 2 is identical to Game 1 from the view of adversary $\mathcal{M}$ due to the fact that $\mathsf{ProjHash}(\mathsf{param}, \mathcal{L}, hp', W^*_{\mathcal{A}}, w^*_{\mathcal{A}}, aux^*) = \mathsf{Hash}(\mathsf{param}, \mathcal{L}, hk', W^*_{\mathcal{A}}, aux^*)$, and Game 3 is indistinguishable from Game 2 (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 3, 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 λ_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 Ext_3, it holds that

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

In this case, $\tilde{F}$ is required to be a secure πPRF in order to ensure $SK^*_{\mathcal{A}}$ is indistinguishable from a random key. $\mathcal{M}$ can replay $W^*_{\mathcal{A}}$ and send it to $\mathcal{B}$ in another session, denoted by sid, and then issue a SessionKeyReveal (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 sid and thus $s^*_{\mathcal{A}}$ in the test session sid^* is pairwisely independent from $s_{\mathcal{B}}$ in 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 πPRF.

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 security of strong randomness extractor and the hardness of the subset membership problem does not rely on the secrecy of lsk. Hence, the simulator can answer LongTermKeyReveal and 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 SessionKeyReveal query for any non-test session simulated by itself. □

4.4 Instantiations of the proposed protocol

In this section, we present several instantiations of the proposed framework based on different assumptions.

4.4.1 Instantiation based on DDH assumption

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 $\mathcal{L}_{DH}$ 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 *$\mathcal{SPHF}_{DH}$ 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

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

- *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 hv_1 of W_1 and 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 Eq. (8) is independent from the Eq. (7). Therefore, we have that hv_2 is independent from hv_1 and randomly distributed in $\mathbb{G}$.

□

A concrete AKE protocol based on DDH We can instantiate our generic construction using the $\mathcal{SPHF}_{\mathsf{DH}}$ described 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. 4. 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. 4, $\mathbb{G}$ 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 $\widehat{F} \leftarrow \widehat{\mathsf{F}}^{k, \sum_{\widehat{\mathsf{F}}}, \mathcal{D}_{\widehat{\mathsf{F}}}, \mathcal{R}_{\widehat{\mathsf{F}}}}$, $\overline{F} \leftarrow \overline{\mathsf{F}}^{k, \sum_{\overline{\mathsf{F}}}, \mathcal{D}_{\overline{\mathsf{F}}}, \mathcal{R}_{\overline{\mathsf{F}}}}$ and $\widetilde{F} \leftarrow \widetilde{\mathsf{F}}^{k, \sum_{\widetilde{\mathsf{F}}}, \mathcal{D}_{\widetilde{\mathsf{F}}}, \mathcal{R}_{\widetilde{\mathsf{F}}}}$. The system parameter is $(\mathbb{G}, p, g_1, g_2, g, H_1, \mathsf{Ext}_1, \mathsf{Ext}_2, \mathsf{Ext}_3, \widehat{F}, \overline{F}, \widetilde{F})$.

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

Corollary 1 *The concrete AKE protocol in Fig. 4 is* (λ_1, λ_2)*-CLR-eCK-secure under the DDH assumption, where* $\lambda_1 \leq \min\{4\log p - 2\log(1/\epsilon_1) - l_1(k), \log p - 2\log(1/\epsilon_3) - l_3(k), t_2(k) - k\}$, $\lambda_2 \leq \min\{u(k) - 2\log(1/\epsilon_2) - l_2(k), t_1(k) - k\}$.

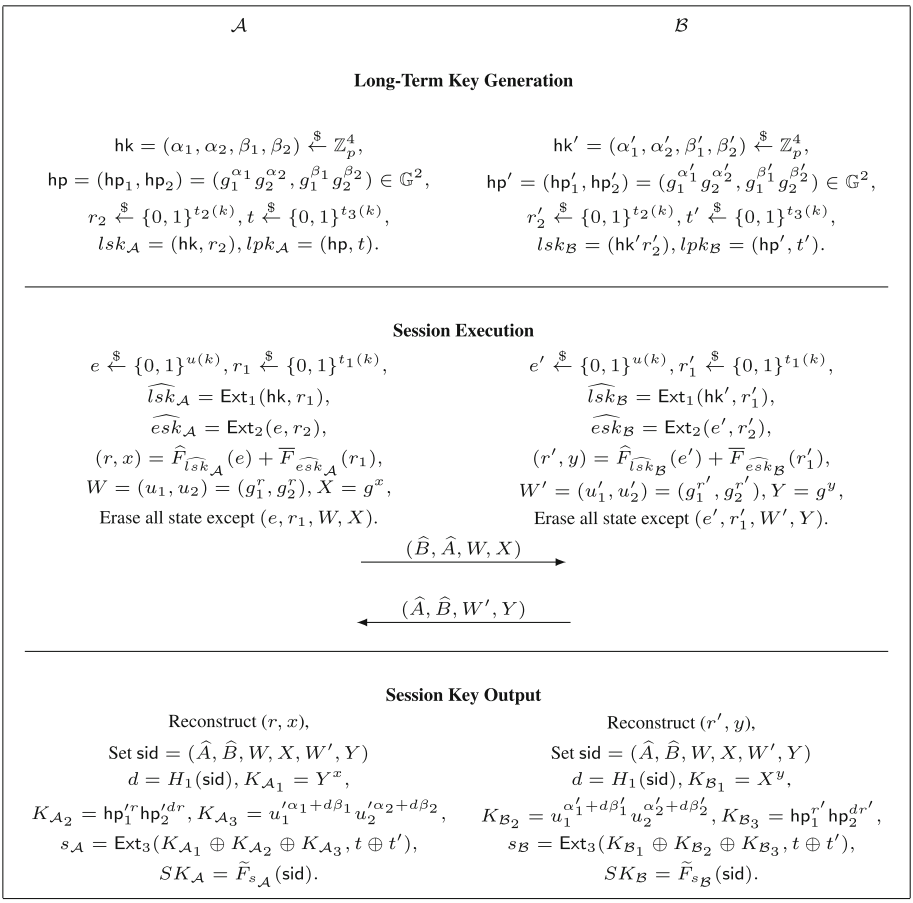

Fig. 4 CLR-eCK secure AKE protocol based on DDH assumption

4.4.2 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

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

Theorem 4 [14] *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:

$$\forall x = a + bN \pmod{N^2},$$

where $0 \leq a, b < N$, $f(x) = b \pmod{N}$.

The construction of the SPHF is as follows.

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

The following result can be immediately obtained from [14].

Theorem 5 *$\mathcal{SPHF}_{DCR}$ 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 $g, \mathcal{K}, W, f$ are as defined above:

- KeyGen: randomly choose $x \in \mathcal{K}$ and return $(SK = x, PK = g^x)$;
- $\mathsf{Enc}(PK)$: randomly choose $w \in \mathcal{W}$ and return $(CT = g^w, K = f(PK^w))$;
- $\mathsf{Dec}(SK, CT)$: 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. 5, where $N = pq$ such that p and q are both $k + 1$ bits safe primes and the generator g can be picked by choosing $\mu \xleftarrow{\$} \mathbb{Z}^*_{N^2}$ and setting $g = -\mu^{2N}$. Same as the previous DDH based protocol, the DCR based protocol also requires only one round.

Parameters of The DCR-based protocol 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 $\widehat{F} \leftarrow \widehat{\mathsf{F}}^{k, \sum_{\widehat{\mathsf{F}}}, \mathcal{D}_{\widehat{\mathsf{F}}}, \mathcal{R}_{\widehat{\mathsf{F}}}}$, $\overline{F} \leftarrow \overline{\mathsf{F}}^{k, \sum_{\overline{\mathsf{F}}}, \mathcal{D}_{\overline{\mathsf{F}}}, \mathcal{R}_{\overline{\mathsf{F}}}}$ and $\widetilde{F} \leftarrow \widetilde{\mathsf{F}}^{k, \sum_{\widetilde{\mathsf{F}}}, \mathcal{D}_{\widetilde{\mathsf{F}}}, \mathcal{R}_{\widetilde{\mathsf{F}}}}$. The map $f : \mathbb{Z}_{N^2} \to \mathbb{Z}_N$ is defined as above.

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

Corollary 2 *The concrete AKE protocol in Fig. 5 is (λ_1, λ_2)-CLR-eCK-secure under the DCR assumption, where $\lambda_1 \leq \min\{2 \cdot \log(N^2/2) - 2\log(1/\epsilon_1) - l_1(k), \log N - 2\log(1/\epsilon_3) - l_3(k), t_2(k) - k\}$, $\lambda_2 \leq \min\{u(k) - 2\log(1/\epsilon_2) - l_2(k), t_1(k) - k\}$.*

4.4.3 Hybrid instantiations

The instantiations given above are based a sole number theoretic assumption. Since a secure SPHF implies a secure KEM scheme, we are able to instantiate the proposed generic construction with any number theoretic assumption that allows instantiation of the SPHF. On

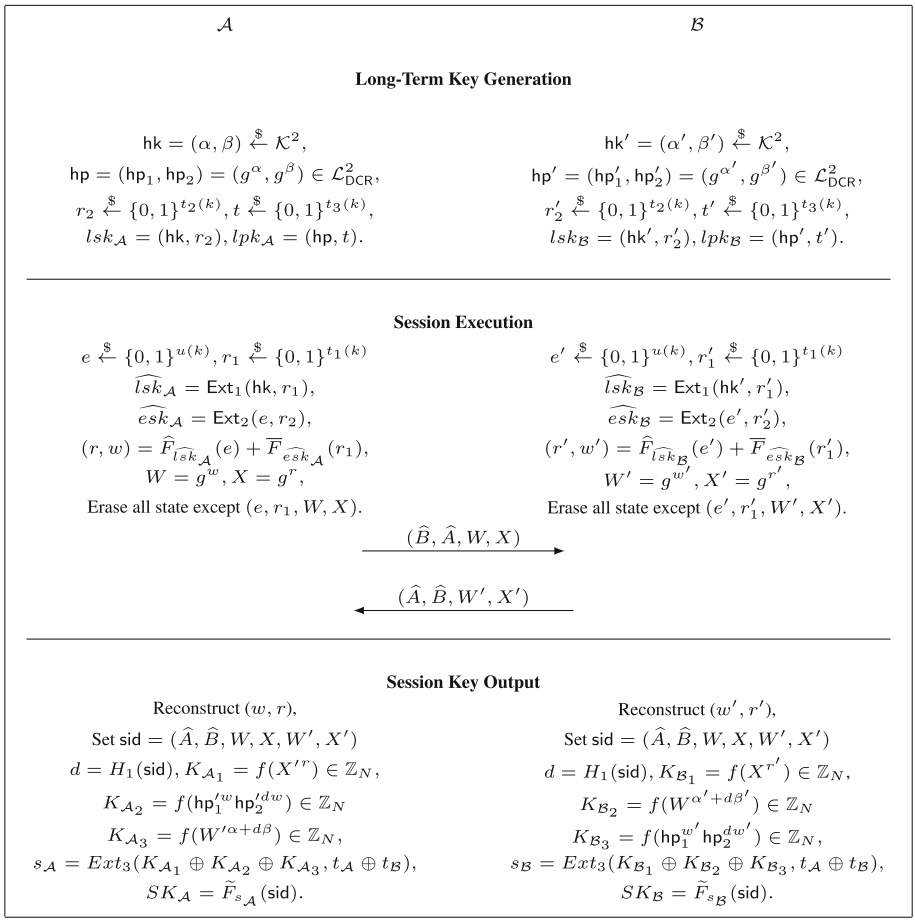

Fig. 5 CLR-eCK secure AKE protocol based on DCR assumption

the other hand, our generic construction can also be instantiated with multiple assumptions (i.e., via a hybrid instantiation) that could accommodate the needs in different applications.

One particular interesting setting of the hybrid instantiation is to combine an SPHF based on a traditional assumption (e.g., DDH, DCR or QR) and a KEM that is based on a hard problem even for quantum computers (e.g., (Ring) Learning With Errors (LWE) [29,34,36]). Such a construction method has also been considered by some existing works (e.g., [8,9]) which performs key exchange with explicit authentication based on digital signature schemes. As highlighted in [8,9], using a quantum-resistant key exchange mechanism (corresponding to the KEM in our construction) together with some non-quantum-resistant authentication mechanism (corresponding to the SPHF in our construction which enforces implicit authentication) can achieve better performance (in comparison with a pure quantum-resistant scheme) while ensuring that the secure communications performed today would remain secure against a quantum computer that might be built in the future.

By applying the hybrid instantiation approach mentioned above, our proposed construction allows efficient instantiations of AKE schemes that can *resist the* (*present*) *key leakage attacks while ensuring the secrecy of current communications against the* (*future*) *quantum machines*.

5 Performance analysis

To evaluate the performance of our proposed CLR-eCK secure AKE constructions, we implement both the DDH-based and DCR-based instantiations.

5.1 Experimental setup

We simulate an interactive session between two parties over the Internet, and test each protocol under 80-, 112-, and 128-bit security levels. We use SHA-256 as the cryptographic hash function and choose multiplicative groups for implementation. Specifically, for the DDH-based protocol, we choose group $\mathbb{G}$ with a primer order p as a subgroup of Z_q^* where q is a prime of size 1024, 2048, and 3072 bits for 80-, 112-, and 128-bit security respectively, and p has the size of 160, 224, and 256 bits correspondingly. As for the DCR-based instantiation, to generate the group $\mathbb{Z}_{N^2}^*$, we choose the p' (and q') as 512-, 1024-, and 1536-bit prime for 80-, 112-, and 128-bit security respectively, and thus the group element from $\mathbb{Z}_{N^2}^*$ is of 2048, 4096, and 6144 bits correspondingly.

The implementation is built on ubuntu 16.04 running on 3.4 GHz Intel Core i7-6700 CPU with access to 8 GB DDR4-2133 SDRAM providing 15 GB/s read and write speed. We used python 3.5 as programming language and Charm [2] as the library for implementation.

5.2 Computation efficiency

We divide the protocols into three stages, i.e. KG (key generation stage), SE (session execution stage), and SKD (session key derivation). The stage KG does not include the system parameter setup. It is worth noting that we mainly focus on the computation time of the algorithms involved in the protocol at each stage, and do not consider the network transmission delay. Moreover, since both the DDH-based protocol and the DCR-based protocol are symmetric in terms of computation, we only test one party for the computation efficiency analysis.

We run each protocol for 100 times to test the computation efficiency of each protocol. Figures 6 and 7 show the average time of each stage for the DDH-based protocol and DCR-based protocol, respectively. One can note that even at 128-bit security level, each stage costs less than 100 ms and thus both protocols are computationally efficient. Particularly, for the DDH-based protocol, the session execution time of each party is less than 6 ms and thus is very efficient. Regarding the DCR-based protocol, since the underlying group $\mathbb{Z}_{N^2}$ is larger than that of the DDH-based protocol, its execution time is higher. Specifically, the session key derivation in the DCR-based protocol for 128-bit security level takes about 42 ms while that in the DDH-based protocol of the same security level is only about 5 ms.

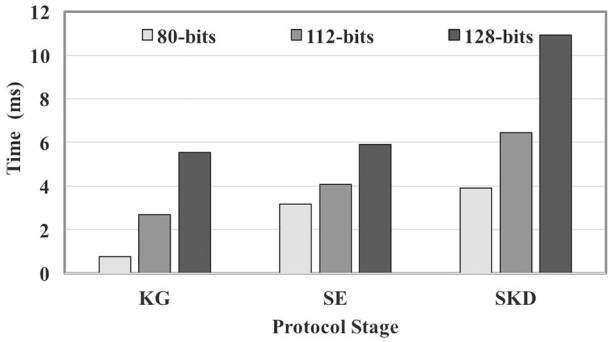

Fig. 6 DDH-based protocol computation cost

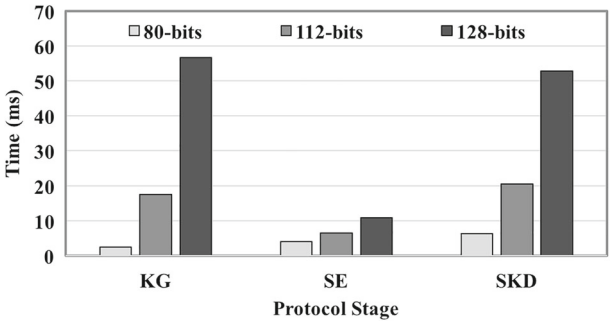

Fig. 7 DCR-based protocol computation cost

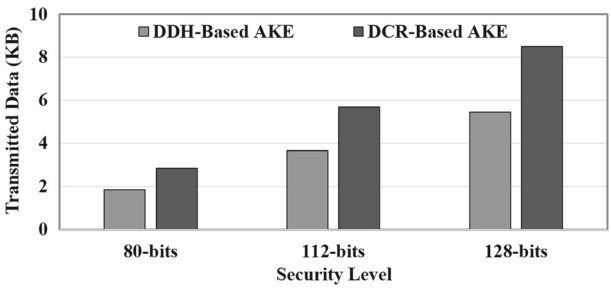

Fig. 8 DDH/DCR-based protocol transmission cost

5.3 Communication efficiency

Beside the computation cost, we also evaluate the communication cost of both AKE protocols. In our experiment, we consider the transmitted data size as the total amount of data that a party sends and receives during a session.

As depicted in Fig. 8, the transmitted data grows as the security level increases. In particular, the transmitted data of DDH-based protocol is below 2 KB when the security level is 80-bit and increases to around 5.5 KB for 128-bit security. Compared to the DDH-based protocol, the DCR-based protocol transmits more data as the group is larger for the same

security level. Specifically, the transmitted data size is 8.5 KB for the DCR-based protocol for 128-bit security.

6 Conclusion

In this paper, we revisited strongly leakage-resilient authenticated key exchange (AKE). We demonstrated that there is a security issue in the AKE protocol presented in [11] which aimed to achieve challenge dependent leakage on both long-term and ephemeral secret keys. We then presented an improved and extended construction that not only fixed the problem but also allowed more instantiations under various assumptions, including hybrid instantiations that are secure under key leakage attacks and can protect current private communications against future quantum computers.

Acknowledgements We would like to thank Janaka Alawatugoda for his comments on a preliminary version of this paper. The work of Guomin Yang is supported by the Australian Research Council Discovery Early Career Researcher Award (Grant No. DE150101116) and the National Natural Science Foundation of China (Grant No. 61472308). The work of Rongmao Chen is supported by the National Natural Science Foundation of China (Grant No. 61702541), the Young Elite Scientists Sponsorship Program by CAST (Grant No. 2017QNRC001), and the Science Research Plan Program by NUDT (Grant No. ZK17-03-46). The work of Yi Mu is supported by the National Natural Science Foundation of China (Grant No.61872087).

References

1. Akavia A., Goldwasser S., Vaikuntanathan V.: Simultaneous hardcore bits and cryptography against memory attacks. In: TCC, pp. 474–495 (2009).
2. Akinyele J.A., Garman C., Miers I., Pagano M.W., Rushanan M., Green M., Rubin A.D.: Charm: a framework for rapidly prototyping cryptosystems. J. Cryptogr. Eng. **3**(2), 111–128 (2013).
3. Alawatugoda J., Stebila D., Boyd C.: Modelling after-the-fact leakage for key exchange. In: ASIACCS, pp. 207–216 (2014).
4. Alwen J., Dodis Y., Wichs D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: CRYPTO, pp. 36–54 (2009).
5. Bellare M., Rogaway P.: Entity authentication and key distribution. In: CRYPTO, pp. 232–249 (1993).
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. Biham E., Shamir A.: Differential fault analysis of secret key cryptosystems. In: CRYPTO, pp. 513–525 (1997).
8. 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).
9. Bos J.W., Costello C., Ducas L., Mironov I., Naehrig M., Nikolaenko V., Raghunathan A., Stebila D.: Frodo: take off the ring! practical, quantum-secure key exchange from LWE. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 Oct 2016, pp. 1006–1018 (2016).
10. Canetti R., Krawczyk H.: Analysis of key-exchange protocols and their use for building secure channels. In: EUROCRYPT, pp. 453–474 (2001).
11. Chen R., Mu Y., Yang G., Susilo W., Guo F.: Strongly leakage-resilient authenticated key exchange. In: CT-RSA, pp. 19–36 (2016).
12. Chen R., Mu Y., Yang G., Susilo W., Guo F.: Strong authenticated key exchange with auxiliary inputs. Des. Codes Cryptogr. **85**(1), 145–173 (2017).
13. Choo K.R., Boyd C., Hitchcock Y.: Examining indistinguishability-based proof models for key establishment protocols. In: ASIACRYPT, pp. 585–604 (2005).
14. Cramer R., Shoup V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: EUROCRYPT, pp. 45–64 (2002).

15. 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).
16. Diffie W., Hellman M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976).
17. 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).
18. Dodis Y., Kalai Y.T., Lovett S.: On cryptography with auxiliary input. In: STOC, pp. 621–630 (2009).
19. 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).
20. Entity authentication mechanisms-part3: Entity authentication using asymmetric techniques. ISO/IEC IS 9789-3 (1993).
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. Feltz M., Cremers C.: On the limits of authenticated key exchange security with an application to bad randomness. IACR Cryptol. ePrint Arch. **2014**, 369 (2014).
23. Gandolfi K., Mourtel C., Olivier F.: Electromagnetic analysis: concrete results. In: Cryptographic Hardware and Embedded Systems, Generators, pp. 251–261 (2001).
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., Lin H.: After-the-fact leakage in public-key encryption. In: TCC, pp. 107–124 (2011).
26. Kocher P.C., Jaffe J., Jun B.: Differential power analysis. In: CRYPTO, pp. 388–397 (1999).
27. Krawczyk H.: SIGMA: the 'sign-and-mac' approach to authenticated Diffie-Hellman and its use in the ike-protocols. In: CRYPTO, pp. 400–425 (2003).
28. LaMacchia B.A., Lauter K.E., Mityagin A.: Stronger security of authenticated key exchange. In: Provable Security, pp. 1–16 (2007).
29. Lyubashevsky V., Peikert C., Regev O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology—EUROCRYPT 2010, pp. 1–23 (2010).
30. Marvin R.: Google admits an android crypto prng flaw led to bitcoin heist (Aug 2013). http://sdt.bz/64008.
31. Micali S., Reyzin L.: Physically observable cryptography (extended abstract). In: TCC, pp. 278–296 (2004).
32. Moriyama D., Okamoto T.: Leakage resilient eck-secure key exchange protocol without random oracles. In: ASIACCS, pp. 441–447 (2011).
33. Okamoto T.: Authenticated key exchange and key encapsulation in the standard model. In: ASIACRYPT, pp. 474–484 (2007).
34. Peikert C.: Lattice cryptography for the internet. In: PQCrypto, pp. 197–219 (2014).
35. Quisquater J., Samyde D.: Electromagnetic attack. In: Encyclopedia of Cryptography and Security, 2nd ed., pp. 382–385 (2011).
36. Regev O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 84–93 (2005).
37. 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.
38. Yang G., Duan S., Wong D.S., Tan C.H., Wang H.: Authenticated key exchange under bad randomness. In: Financial Cryptography and Data Security, pp. 113–126 (2011).
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. Zetter K.: How a crypto 'backdoor' pitted the tech world against the nsa. http://www.wired.com/threatlevel/2013/09/nsa-backdoor/all/.

Affiliations

Guomin Yang[2] · Rongmao Chen[1] · Yi Mu[2] · Willy Susilo[2] · Fuchun Guo[2] · Jie Li[1]

✉ Rongmao Chen
chromao@nudt.edu.cn

Guomin Yang
gyang@uow.edu.au

Yi Mu
ymu@uow.edu.au

Willy Susilo
wsusilo@uow.edu.au

Fuchun Guo
fuchun@uow.edu.au

Jie Li
lijie13d@nudt.edu.cn

1 College of Computer, National University of Defense Technology, Changsha, China

2 School of Computing and Information Technology, University of Wollongong, Wollongong, Australia