Standard Model Leakage-Resilient Authenticated Key Exchange using Inner-product Extractors

Janaka Alawatugoda* Tatsuaki Okamoto†

*Department of Computer Engineering, University of Peradeniya, Peradeniya 20400, Sri Lanka
alawatugoda@eng.pdn.ac.lk

†Cryptography and Information Security Laboratories, NTT Research, Inc., Sunnyvale, CA 94085, USA
tatsuaki.okamoto@ntt-research.com

Abstract

With the development of side-channel attacks, a necessity arises to invent authenticated key exchange protocols in a leakage-resilient manner. Constructing authenticated key exchange protocols using existing cryptographic schemes is an effective method, as such construction can be instantiated with any appropriate scheme in a way that the formal security argument remains valid. In parallel, constructing authenticated key exchange protocols that are proven to be secure in the standard model is more preferred as they rely on real-world assumptions. In this paper, we present a Diffie-Hellman-style construction of a leakage-resilient authenticated key exchange protocol, that can be instantiated with any CCLA2-secure public-key encryption scheme and a function from the pseudo-random function family. Our protocol is proven to be secure in the standard model assuming the hardness of the decisional Diffie-Hellman problem. Furthermore, it is resilient to continuous partial leakage of long-term secret keys, that happens even after the session key is established, while satisfying the security features defined by the eCK security model.

Keywords: Leakage-resilient cryptography Authenticated key exchange eCK model CAFL-eCK model Standard model

1 Introduction

Since the mid-twentieth century, Information age (also known as digital age or computer age) begins with the rapid development of information technology. In the information age, the security of information arises as an essential requirement. Authenticated key exchange (AKE) protocols play a crucial role in the context of information security. AKE protocols enable two or more parties to authentically exchange a secret key (often called as session key) by communicating over a public communication channel. The session key is used to encrypt data that are exchanged between the parties, within a particular period of time known as session.

During the past couple of decades, many AKE protocols are invented, and various security models are proposed to analyze their security. A security model is a formal statement that addresses potential attack scenarios. The Bellare-Rogaway models (BR93 [BR93], BR95 [BR95]), the Canetti-Krawczyk (CK) model [CK01], and the extended Canetti-Krawczyk (eCK) model [LLM07] are the renowned conventional security models. Among them, the eCK model can be considered as a well-known security model that captures the most demanding security features of AKE protocols [Ala17a].

With the advancement of side-channel attacks [Koc96, MDS02, BB03, HMF07], the discussion on countermeasures against side-channel attacks become popular [HR07, AJR11, HAR13]. Thus, a necessity arises to invent AKE protocols in a leakage-resilient manner. The aforementioned security models do not address the partial leakage of long-term secret keys. Therefore, they are not suitable to analyze the security of AKE protocols against side-channel attacks. Thus, new security models are proposed to address the side-channel attacks, in addition to the attack scenarios addressed in the conventional security models [ASB14, ABS14, MO11, CMY+16]. New AKE protocols are invented in a way that their security can be proven in the new security models

Janaka Alawatugoda acknowledges the NTT Corporation, Japan for giving him the opportunity to pursue his internship at the NTT Musashino R&D Center in April 2015.

[CAR17, YMSW13, ASB14, ABS14, MO11, CMY+16, ASB15, Ala17b, YCM+19, CMY+17], and they are commonly known as leakage-resilient AKE protocols. Thus, there is a formal guarantee that the leakage-resilient AKE protocols are robust against side-channel attacks and various other attack scenarios that are addressed in the new security models.

Constructing AKE protocols using existing cryptographic building blocks is an effective way, because such constructions can be instantiated with any appropriate building block such that the formal security argument remains valid. This is also beneficial for cipher suits such as Transport Layer Security (TLS), as they can use existing implementations of cryptographic building blocks with a minimal implementation cost to convert them to a AKE protocol. There are several proven-secure AKE protocol constructions that are transformations of existing cryptographic building blocks; some of them are non-leakage-resilient AKE protocols [BCNP09, Ala17a] while the others are leakage-resilient AKE protocols [ASB14, ABS14].

Our Contribution. In this paper, our aim is to advance the constructions of leakage-resilient AKE protocols that are proven to be secure in the standard model. We use a variant of generic after-the-fact leakage eCK model, namely continuous after-the-fact leakage eCK (CAFL-eCK) model [ASB14], as a reasonably strong and realistic security model for security analysis, because the CAFL-eCK model addresses continuous partial leakage of long-term secret keys in addition to the attack scenarios modeled in the eCK model. The section 3 gives a detailed description of the CAFL-eCK model.

We construct a Diffie-Hellman-style $\boldsymbol{\lambda}-$ CAFL-eCK-secure AKE protocol, namely protocol P1.v2. As the $\boldsymbol{\lambda}-$ CAFL-eCK model defines, our protocol is resilient to continuous partial leakage of long-term secret keys (under a leakage bound $\boldsymbol{\lambda}$), which happens even after the session key is established (hence, after-the-fact leakage), while satisfying the security features defined by the eCK model. The protocol P1.v2 is a transformation of a CCLA2-secure public-key encryption scheme into a CAFL-eCK-secure AKE protocol, that can be instantiated with any CCLA2-secure public-key encryption scheme and a function from the pseudo-random function family. The protocol P1.v2 uses the leakage-resilient storage scheme and the leakage-resilient refreshing protocol of Dziembowski and Faust [DF11] (refer to the section 2.2) as the core building block of the construction. The storage scheme encodes a secret value into two matrices, such that their inner product gives back the secret value. Given the leakage information from the two matrices independently, an adversary cannot obtain any information about the encoded secret value. Therefore, unless an attacker obtains leakage information from the inner product of the two matrices, she cannot derive the secret value.

In protocol P1.v2, the leakage-resilient storage scheme is used to encode the Diffie-Hellman long-term secret keys. The two matrices of the encoded long-term secret keys are stored in the memory. For all the exponentiation computations that use a Diffie-Hellman long-term secret key, component-wise exponentiation and multiplication operations are performed using the two matrices; the inner product is never computed. The leakage-resilient refreshing protocol is used to refresh the Diffie-Hellman long-term secret keys to make the protocol P1.v2 robust against continuous partial leakage attacks. The protocol P1.v2 can be considered as an application that extends the use of leakage-resilient storage scheme and the leakage-resilient refreshing protocol of Dziembowski and Faust.

The security of the protocol P1.v2 is proven in the standard model, assuming the hardness of the decisional Diffie-Hellman (DDH) problem. Table 1 compares the protocol P1.v2 with couple of selected AKE protocols, by means of the leakage model, after-the-fact leakage feature, the base security model and the proof model. The protocol P1.v2 costs one multi-exponentiation and one encryption scheme key generation operation for the initial key setup at a protocol principal. Then, for an execution of a session at a protocol principal, the protocol P1.v2 costs $(3n + 2)$ exponentiation operations, one encryption operation, one decryption operation and one pseudo-random function operation. Note that $n \in \mathbb{N}$ is the statistical security parameter, that is a function of the security parameter k. Moreover, the protocol P1.v2 maintains a leakage bound of $1/n - o(1)$ for a Diffie-Hellman long-term secret key. The leakage bound for a decryption secret key of the underlying leakage-resilient public-key encryption scheme is specific to the particular public-key encryption scheme.

Being a transformation of a CCLA2-secure public-key encryption scheme into a CAFL-eCK-secure AKE protocol based on the standard model assumptions, the protocol P1.v2 is a good AKE protocol candidate for future TLS protocol suits, that can be implemented and deployed with minimal implementation cost, given existing implementations of a CCLA2-secure public-key encryption scheme and a function from the pseudo-random function family.

Protocol	Leakage Model	After-the-fact	Base Model	Proof Model
NAXOS [LLM07]	None	No	eCK	Random oracle
Moriyama and Okamoto [MO11]	Bounded	No	eCK	Standard
Alawatugoda et al. [ASB14]	Bounded	Yes	eCK	Standard
Alawatugoda et al. [ASB15]	Continuous	Yes	eCK	Random Oracle
Chen et al. [CMY+17]	Auxiliary input[1]	Yes	eCK	Standard
Chen et al. [CMY+16]	Auxiliary input[1]	Yes	eCK	Standard
Protocol P1.v2 (This Paper)	Continuous	Yes	eCK	Standard

[1] Auxiliary input model requires that it is computationally hard for an adversary to recover the secret key given the leakage, and therefore it is a generalization of the bounded leakage model [CMY+17]. For our protocol construction we consider continuous leakage model rather than the bounded leakage model.

Table 1: Comparison of (selected) AKE protocols

2 Preliminaries

2.1 Leakage Model

We frequently borrow notions for the leakage model from Dziembowski and Faust [DF11].

Leakage Game. Assume that secret key is stored in the memory as two encoded parts, say L and $R \in \{0,1\}^s$, and an adversary $\mathcal{A}$ wants to learn information from L and R. For a positive integer λ, we define λ-*leakage game* between an adaptive adversary $\mathcal{A}$, called λ-limited adversary, and a leakage oracle $\Lambda(L,R)$ as follows: the adversary queries two adaptively-chosen leakage functions (f_1, f_2) to the oracle $\Lambda(L,R)$. Then the oracle replies with $f_1(L)$ and $f_2(R)$ with restriction that the adversary cannot learn more than λ-bits from each L and R at the end of the game. We denote $Out(\mathcal{A}, \Lambda(L,R))$ by the output of this game. Moreover, the information from L and R are leaked independently from each other.

Leakage from Computations. We follow the axiom "only computations leak information" and assume that only computations involving L or R leak their information to the adversary. This can be described in the following setting.

Consider a two-party protocol between the parties P_L and P_R. Initially, the party P_L (respectively, P_R) is given the input L (resp. R) and at the end of the protocol each party have the results L' and R', respectively. The execution of the protocol proceeds in rounds. In each round, one party (owner) computes a message and send it to the other. The message may be computed from owner's input (eg. L or R), randomness chosen by the owner, and the received message from the earlier rounds. In the setting that only computation leaks information, the computations carried out by P_L (resp. P_R) with the initial state L (resp. R) in each round may leak information on L (resp. R) independently from R (resp. L). Precisely, the adversary $\mathcal{A}$ adaptively chooses two leakage functions (f_1^j, f_2^j) and obtains $f_1^j(L)$ and $f_2^j(R)$ from each j-th round in the execution of the protocol. At this time, the adversary is allowed to play the λ-leakage game with access to the leakage oracle $\Lambda((L, \rho_L, M_L); (R, \rho_R, M_R))$, where ρ_x is the randomness used by $x \in \{L, R\}$ and M_x is the previously received message by user $x \in \{L, R\}$ in the execution of the protocol. We sometimes omit ρ_x or M_x when they are obviously null from the context.

2.2 Leakage-resilient Storage Scheme and it's Refreshing Protocol

In Dziembowski and Faust [DF11], they used a leakage-resilient storage scheme to encode the secret key into two parts and a refreshing protocol to construct public-key cryptosystems secure against continuous partial leakage attacks.

2.2.1 Leakage-resilient Storage

A leakage-resilient storage (LRS) $\Phi = (\mathrm{Encode}, \mathrm{Decode})$ is a scheme to encode a message in $\mathcal{M}$, where $\mathrm{Encode} : \mathcal{M} \to \mathcal{L} \times \mathcal{R}$ is a probabilistic polynomial-time function and $\mathrm{Decode} : \mathcal{L} \times \mathcal{R} \to \mathcal{M}$ is a deterministic, polynomial-time function satisfying $\mathrm{Decode}(\mathrm{Encode}(S)) = S$ for any $S \in \mathcal{M}$.

An LRS Φ is called (λ, ϵ)-secure if for any $S, S' \in \mathcal{M}$ and any λ-limited adversary $\mathcal{A}$, we have

$$\Delta(Out(\mathcal{A}, \Lambda(L, R)), Out(\mathcal{A}, \Lambda(L', R'))) \leq \epsilon,$$

where $(L, R) := \mathrm{Encode}(S)$ and $(L', R') := \mathrm{Encode}(S')$. Here, Δ denotes the statistical distance.

In Dziembowski and Faust [DF11], they use the fact that the LRS from inner product is (λ, ϵ)-secure for some λ and $\epsilon > 0$. Precisely, for a field $\mathbb{F}$, an LRS $\Phi_{\mathbb{F}}^{n,m} = (\mathrm{Encode}_{\mathbb{F}}^{n,m}, \mathrm{Decode}_{\mathbb{F}}^{n,m})$ is defined as follows:

- $\mathrm{Encode}_{\mathbb{F}}^{n,m}(S)$ chooses $L \leftarrow \mathbb{F}^n \backslash \{0^n\}$ and samples $R \leftarrow \mathbb{F}^{n \times m}$ such that $L \cdot R = S$.
- $\mathrm{Decode}_{\mathbb{F}}^{n,m}(L, R)$ computes $L \cdot R = S$.

Corollary 2.0.1 ([DF11]). *Suppose $|\mathbb{F}| = \Omega(n)$ and $m < n/20$, then the LRS $\Phi_{\mathbb{F}}^{n,m}$ is a $(0.3|\mathbb{F}^n|, \epsilon)$-secure LRS, where ϵ is negligible in the statistical security parameter $n \in \mathbb{N}$.*

2.2.2 Refreshing Protocol

It is obvious that the above LRS scheme is not secure, if we allow an adversary to obtain leakage information continuously from L and R. To prevent this obstacle, Dziembowski and Faust [DF11] introduced a refreshing protocol Refresh of an LRS Φ. The main idea is to refresh the encoded secret (L, R) securely to a newly encoded value (L', R') so that it does not reveal enough information to recover the secret key even when an λ-limited adversary can observe the leakage continuously from the refreshing protocol.

Precisely, a refreshing protocol $(L', R') \leftarrow \mathrm{Refresh}(L, R)$ is a two-party protocol between P_L and P_R holding L and R respectively. At the end of the protocol, each party outputs L' and R' respectively satisfying $\mathrm{Decode}(L, R) = \mathrm{Decode}(L', R')$.

Let Refresh a refreshing protocol, $\Phi = (\mathrm{Encode}, \mathrm{Decode})$ an LRS, $\mathcal{A}$ an λ-limited adversary, $\ell \in \mathbb{N}$ and $S \in \mathcal{M}$. We consider the following experiments $\mathrm{Exp}_{(\mathrm{Refresh},\Phi)}(\mathcal{A}, S, \ell)$:

- For a secret S, encode it by $(L^0, R^0) \leftarrow \mathrm{Encode}(S)$.
- For $i = 1$ to ℓ, run $\mathcal{A}$ against the refreshing protocol: $\mathcal{A} \leftrightarrows (\mathrm{Refresh}(L_{i-1}, R_{i-1}) \to (L_i, R_i))$.
- Return $b \in \{0, 1\}$ output by $\mathcal{A}$.

As described in the previous paragraph, the leakage from each execution of the protocol is obtained via the leakage oracle $\Lambda((L_i, \rho_{L_i}, M_{L_i}); (R_i, \rho_{R_i}, M_{R_i}))$. Finally, the security of the refreshing protocol is defined by indistinguishability notion.

Definition 2.1 (A $(\ell, \lambda, \epsilon_1)$-secure refreshing protocol [DF11]). *For a LRS Φ with message space $\mathcal{M}$ and $\ell \in \mathbb{N}$ number of leakage rounds, a refreshing protocol* Refresh *is $(\ell, \lambda, \epsilon_1)$-secure, if for any λ-limited adversary $\mathcal{A}$ and any two secrets $S, S' \in \mathcal{M}$, we have*

$$\Delta(\mathrm{Exp}_{(\mathrm{Refresh},\Phi)}(\mathcal{A}, S, \ell), \mathrm{Exp}_{(\mathrm{Refresh},\Phi)}(\mathcal{A}, S', \ell)) \leq \epsilon_1.$$

Theorem 2.1 ([DF11]). *Let $m/3 \leq n$, $n \geq 16$ and $\ell \in \mathbb{N}$. Let n, m and $\mathbb{F}$ be such that $\Phi_{\mathbb{F}}^{n,m}$ is (λ, ϵ)-secure for some $\lambda, \epsilon > 0$. Then the protocol $\mathrm{Refresh}_{\mathbb{F}}^{n,m}$ is a $(\ell, \lambda/2 - 1, \epsilon')$-secure refreshing protocol for $\Phi_{\mathbb{F}}^{n,m}$ with $\epsilon' = 2\ell|\mathbb{F}|^m(3|\mathbb{F}|^m\epsilon + m|\mathbb{F}|^{-n-1})$.*

Definition 2.2 (A $(\ell, \lambda, \epsilon_1)$-secure refreshing protocol with auxiliary information [DF11]). *Let $Z \subset \mathcal{M}^m$ be an $m' < m$ dimensional affine subspace defined by W and Q. For a LRS Φ with message space $\mathcal{M}$ and $\ell \in \mathbb{N}$, a refreshing protocol* Refresh *is $(\ell, \lambda, \epsilon_1)$-secure with auxiliary information (W, Q), if for any λ-limited adversary $\mathcal{A}$ and any two secrets $S, S' \in \mathcal{M}$, we have,*

$$\Delta(\mathrm{Exp}^{\mathrm{aux}}_{(\mathrm{Refresh},\Phi)}(\mathcal{A}, S, \ell, W, Q), \mathrm{Exp}^{\mathrm{aux}}_{(\mathrm{Refresh},\Phi)}(\mathcal{A}, S', \ell, W, Q)) \leq \epsilon_1.$$

Theorem 2.2 (Generalization of Theorem 1 [DF11]). *Let $m/4 \leq n$, $n \geq 16$ and $\ell \in \mathbb{N}$. Let (W, Q) be as above defining an $m' < m$ dimensional affine subspace Z. Let n, m' and $\mathbb{F}$ be such that $\Phi_{\mathbb{F}}^{n,m'}$ is (λ, ϵ)-secure for some $\lambda, \epsilon > 0$. Then the protocol $\mathrm{Refresh}_{\mathbb{F}}^{n,m}$ is a $(\ell, \lambda/2 - 1, \epsilon')$-secure refreshing protocol with auxiliary information defined by (W, Q) for $\Phi_{\mathbb{F}}^{n,m}$ with $\epsilon' = 2\ell|\mathbb{F}|^m(3|\mathbb{F}|^{2m}\epsilon + m|\mathbb{F}|^{-n-1})$.*

Corollary 2.2.1 ([DF11]). *Suppose $|\mathbb{F}| = \Omega(n)$ and $m = o(n)$, then the $\mathrm{Refresh}_{\mathbb{F}}^{n,m}$ is a $(\ell, 0.15n \log |\mathbb{F}|) - 1, \epsilon_1)$-secure refreshing protocol for $\Phi_{\mathbb{F}}^{n,m}$, where ℓ is a polynomial and ϵ_1 is negligible in the statistical security parameter $n \in \mathbb{N}$.*

2.3 Leakage-resilient CCA2-secure Public-key Encryption Scheme

For security parameter k, a public-key encryption scheme $\mathrm{PKE} = (\mathrm{KG}, \mathrm{Enc}, \mathrm{Dec})$ consists of the following algorithms.

- $(pk, sk) \leftarrow \mathrm{KG}(1^k)$: It outputs a public/secret key pair.
- $c \leftarrow \mathrm{Enc}(pk, m)$: A probabilistic polynomial-time algorithm that outputs $c = \mathrm{Enc}(pk, m)$ with inputs a message m and the public key pk.
- $m = \mathrm{Dec}(sk, c)$: A deterministic algorithm that decrypts the ciphertext c together with the secret key sk. We always have $m = \mathrm{Dec}(sk, \mathrm{Enc}(pk, m))$.

The strongest security notion for public-key encryption scheme is security against *adaptively-chosen-ciphertext attacks* (CCA2-secure). In CCA2 attack against a public-key encryption scheme, the adversary who is given pk picks two messages m_0, m_1 and guesses $b \in \{0, 1\}$ on input $c^* = \mathrm{Enc}(pk, m_b)$ while the adversary is allowed to ask for the decryption of chosen-ciphertexts prior and after to seeing c^*.

In the setting that computation leaks information, it is natural to consider leakage information from the queries to the decryption oracle. This yields the following extension of CCA2-security against leakage attacks.

Definition 2.3 ([DF11], Security against Chosen Ciphertext Leakage Attacks (CCLA2)). *Let k be the security parameter. A public-key encryption scheme $\mathrm{PKE} = (\mathrm{KG}, \mathrm{Enc}, \mathrm{Dec})$ is CCLA2-secure if for any λ'-limited adversary $\mathcal{A}$ the probability that the experiment below outputs 1 is at most $1/2 + \mathtt{negl}(k)$.*

1. *The challenger runs $(pk, sk) \leftarrow \mathrm{KG}(1^k)$ and sends pk to $\mathcal{A}$.*
2. *The adversary $\mathcal{A}$ asks for the decryption of a ciphertext c learning at most λ'-bits of the current secret sk for each query: $\mathcal{A} \leftrightarrows (\mathrm{Dec}(sk, c) \rightarrow sk')$. Set $sk := sk'$ for the next round.*
3. *$\mathcal{A}$ outputs two messages, m_0, m_1, and sends them to the challenger.*
4. *The challenger computes $c^* \leftarrow \mathrm{Enc}(pk, m_b)$ for $b \in \{0, 1\}$ and gives it to $\mathcal{A}$.*
5. *Repeat $\mathcal{A} \leftrightarrows (\mathrm{Dec}(sk, c) \rightarrow sk')$ for $c \neq c^*$ learning at most λ'-bits of the current secret sk. Set $sk := sk'$ for the next round.*
6. *$\mathcal{A}$ outputs $b' \in \{0, 1\}$. If $b = b'$, then output 1, otherwise output 0.*

2.4 Diffie-Hellman Assumptions

We now describe two Diffie-Hellman assumptions which form the basis of security for many cryptographic primitives. Let k be the security parameter, $\mathcal{G}$ be a group generation algorithm and $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^k)$, where $\mathbb{G}$ is a cyclic group of prime order q and g is an arbitrary generator of $\mathbb{G}$.

Definition 2.4 (Computational Diffie-Hellman (CDH) Assumption). *We say that computational Diffie-Hellman assumption holds in $\mathbb{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, the probability of solving the* CDH *problem in $\mathbb{G}$ given as:*

$$Pr_{\mathbb{G},q}^{\mathrm{CDH}}(\mathcal{A}) = \Pr\left(\mathcal{A}(g, q, g^a, g^b) = g^{ab}\right)$$

is negligible for a given security parameter k.

Definition 2.5 (Decisional Diffie-Hellman (DDH) Assumption)**.** *Consider the following two distributions:* $\mathcal{DH}_{\mathbb{G}} = \{(g, g^a, g^b, g^{ab}); a, b \leftarrow \mathbb{Z}_q\}$ *and* $\mathcal{R}_{\mathbb{G}} = \{(g, g^a, g^b, g^c); a, b, c \leftarrow \mathbb{Z}_q\}$ *. It is said that* DDH *assumption holds in* $\mathbb{G}$ *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$*, the advantage in distinguishing the two distributions* $\mathcal{DH}$ *and* $\mathcal{R}$ *given as:*

$$\mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G},q}(\mathcal{A}) = \Big|\Pr[\mathcal{A}(\mathcal{DH}_{\mathbb{G}}) = 1] - \Pr[\mathcal{A}(\mathcal{R}_{\mathbb{G}}) = 1]\Big|$$

is negligible for a given security parameter k*.*

2.5 Pseudo-Random Functions

We now describe the security definition of pseudo-random functions according to Katz and Lindell [KL07].

Definition 2.6 (Pseudo-Random Functions)**.** *Let* $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *be an efficient, length preserving, keyed function. We say* F *is a* pseudo-random function *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* ϵ_{PRF} *in the security parameter* k *such that:*

$$\Big|\Pr[\mathcal{A}^{F(key,\cdot)}(1^k) = 1] - \Pr[\mathcal{A}^{f_{rnd}(\cdot)}(1^k) = 1]\Big| \leq \epsilon_{\mathrm{PRF}},$$

where $key \in \{0,1\}^k$ *is chosen uniformly at random an* f_{rnd} *is chosen uniformly at random from the set of functions mapping* k*-bit strings to* k*-bit strings.*

3 Continuous After-the-fact Leakage eCK Model [ASB14]

In security experiments for public-key cryptosystems, the challenge to the adversary is, given a ciphertext, distinguish the corresponding plaintext. In key exchange security models, the challenge to the adversary is to identify the real session key of a chosen session from a random session key [BR93, CK01, LLM07]. Leakage which happens after the challenge is given to the adversary is considered as after-the-fact leakage. In leakage models for public-key cryptosystems, after-the-fact leakage is the leakage which happens after the challenge ciphertext is given, whereas in leakage-resilient key exchange security models, after-the-fact leakage is the leakage which happens after the session key is established.

The generic after-the-fact leakage eCK ((·)AFL-eCK) model is equipped with an adversary-chosen, polynomial-time, adaptive leakage function $\mathbf{f}$, in addition to the adversarial capabilities modeled in the eCK model. Therefore, the (·)AFL-eCK model captures all the attacks captured by the eCK model, and captures the partial leakage of long-term secret keys due to side-channel attacks.

In the eCK model, in sessions where the adversary does not modify the communication between parties (passive sessions), the adversary is allowed to reveal both ephemeral secrets, long-term secrets, or one of each from two different parties, whereas in sessions where the adversary may forge the communication of one of the parties (active sessions), the adversary is allowed to reveal the long-term or ephemeral secret of the other party. The security challenge is to distinguish the real session key from a random session key, in an adversary-chosen protocol session.

The generic (·)AFL-eCK model can be instantiated in two different ways which leads to two security models. Namely, *bounded* after-the-fact leakage eCK (BAFL-eCK) model and *continuous* after-the-fact leakage eCK (CAFL-eCK) model. The BAFL-eCK model allows the adversary to obtain a bounded amount of leakage of the long-term secret keys of the protocol principals, as well as reveal session keys, long-term secret keys and ephemeral keys. Differently, the CAFL-eCK model allows the adversary to continuously obtain arbitrarily large amount of leakage of the long-term secret keys of the protocol principals, enforcing the restriction that the amount of leakage per observation is bounded. In this model, continuous leakage of the ephemeral secret keys is not allowed; it is justifiable because the ephemeral secret keys are one-time secrets that are less likely to be leaked continuously over number of computations.

Below we revisit the definitions of the CAFL-eCK model, and we also recall the definitions of the eCK model as a comparison to the CAFL-eCK definitions.

3.1 Partner Sessions in the CAFL-eCK Model

Definition 3.1 (Partner sessions in the CAFL-eCK model)**.** *Two oracles $\Pi^{s}_{U,V}$ and $\Pi^{s'}_{U',V'}$ are said to be partners if all of the following hold:*

1. *both $\Pi^{s}_{U,V}$ and $\Pi^{s'}_{U',V'}$ have computed session keys;*
2. *messages sent from $\Pi^{s}_{U,V}$ and messages received by $\Pi^{s'}_{U',V'}$ are identical;*
3. *messages sent from $\Pi^{s'}_{U',V'}$ and messages received by $\Pi^{s}_{U,V}$ are identical;*
4. *$U' = V$ and $V' = U$;*
5. *Exactly one of U and V is the initiator and the other is the responder.*

The protocol is said to be correct if two partner oracles compute identical session keys.

Note that the definition of partner sessions is the same as in the eCK model.

3.2 Leakage in the CAFL-eCK Model

A realistic way in which side-channel attacks can be mounted against key exchange protocols seems to be to obtain the leakage information from the protocol computations which use the secret keys. Following the previously used premise, "only computation leaks information", leakage is modeled where any computation takes place using secret keys. In conventional security models, by issuing a `Send` query, the adversary will get a protocol message which is computed according to the normal protocol computations. Sending an adversary-chosen, polynomial-time adaptive leakage function with the `Send` query reflects the premise "only computation leaks information".

A tuple of t adaptively-chosen polynomial-time leakage functions $\mathbf{f} = (f^j_1, f^j_2, \ldots, f^j_t)$ are introduced; j indicates the j-th leakage occurrence (leakage happens at the j-th session of the protocol) and the size t of the tuple is protocol-specific. A key exchange protocol may use more than one cryptographic primitive where each primitive uses a distinct secret key. Hence, it is necessary to address the leakage of secret keys from each of these primitives. On the other hand, some cryptographic primitives which have been used to construct an AKE protocol, may have stored the secret key by encoding it into number of portions. Hence, it is necessary to address the leakage of each of these portions of the encoded secret key. Note that the adversary is only allowed to obtain the leakage from each portion of the encoded secret key independently. This prevents the adversary from trivially deriving the secret key using the leakage from the portions of the encoded secret key.

3.3 Adversarial Powers of the CAFL-eCK Model

The adversary $\mathcal{A}$ controls the whole network. $\mathcal{A}$ interacts with a set of oracles which represent protocol instances. The following query allows the adversary to run the protocol.

- `Send`$(U, V, s, m, \mathbf{f})$ query: The oracle $\Pi^{s}_{U,V}$, computes the next protocol message according to the protocol specification and sends it to the adversary $\mathcal{A}$, along with the leakage $\mathbf{f}(sk_U)$. $\mathcal{A}$ can also use this query to activate a new protocol instance as an initiator with blank m.

In the eCK model `Send` query is same as the above except the leakage function $\mathbf{f}$.

The following set of queries allow the adversary $\mathcal{A}$ to compromise certain session specific ephemeral secrets and long-term secrets from the protocol principals.

- `SessionKeyReveal`(U, V, s) query: $\mathcal{A}$ is given the session key of the oracle $\Pi^{s}_{U,V}$.
- `EphemeralKeyReveal`(U, V, s) query: $\mathcal{A}$ is given the ephemeral keys (per-session randomness) of the oracle $\Pi^{s}_{U,V}$.
- `Corrupt`(U) query: $\mathcal{A}$ is given the long-term secrets of the principal U. Then $\mathcal{A}$ may set up long-term secrets at principal U at will. This query does not reveal any session keys or ephemeral keys to $\mathcal{A}$.

`SessionKeyReveal`, `EphemeralKeyReveal` and `Corrupt` (Long-term key reveal) queries are the same in the eCK model.

Once the oracle $\Pi^s_{U,V}$ has accepted a session key, asking the following query the adversary $\mathcal{A}$ attempt to distinguish it from a random session key. The `Test` query is used to formalize the notion of the semantic security of a key exchange protocol.

- `Test`(U, s) query: When $\mathcal{A}$ asks the `Test` query, the challenger first chooses a random bit $b \leftarrow \{0, 1\}$ and if $b = 1$ then the actual session key is returned to $\mathcal{A}$, otherwise a random string chosen from the same session key space is returned to $\mathcal{A}$. This query is only allowed to be asked once across all sessions.

The `Test` query is the same in the eCK model.

3.4 Freshness Definition of the CAFL-eCK Model

Definition 3.2 ($\boldsymbol{\lambda}-$CAFL-eCK-freshness)**.** *Let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_t)$ be a vector of t elements (same size as $\mathbf{f}$ in* `Send` *query). Each λ_i denotes the leakage bound of each underlying primitive or each portion of an encoded secret key. An oracle $\Pi^s_{U,V}$ is said to be $\boldsymbol{\lambda}-$*CAFL-eCK*-fresh if and only if:*

1. *The oracle $\Pi^s_{U,V}$ or its partner, $\Pi^{s'}_{V,U}$ (if it exists) has not been asked a* `SessionKeyReveal`*.*
2. *If the partner $\Pi^{s'}_{V,U}$ exists, none of the following combinations have been asked:*
   (a) `Corrupt`(U) *and* `EphemeralKeyReveal`(U, V, s)*.*
   (b) `Corrupt`(V) *and* `EphemeralKeyReveal`(V, U, s')*.*
3. *If the partner $\Pi^{s'}_{V,U}$ does not exist, none of the following combinations have been asked:*
   (a) `Corrupt`(V)*.*
   (b) `Corrupt`(U) *and* `EphemeralKeyReveal`(U, V, s)*.*
4. *For each* `Send`$(U, \cdot, \cdot, \cdot, \mathbf{f})$ *query, size of the output of $|f^j_i(sk_{U\,i})| \leq \lambda_i$.*
5. *For each* `Send`$(V, \cdot, \cdot, \cdot, \mathbf{f})$ *queries, size of the output of $|f^j_i(sk_{V\,i})| \leq \lambda_i$.*

The eCK-freshness is slightly different from the $\boldsymbol{\lambda}-$CAFL-eCK-freshness by stripping off points 4 and 5.

3.5 Security Game and Security Definition of the CAFL-eCK Model

Definition 3.3 ($\boldsymbol{\lambda}-$CAFL-eCK security game)**.** *Security of a key exchange protocol in the* CAFL-eCK *model is defined using the following security game, which is played by the adversary $\mathcal{A}$ against the protocol challenger.*

- ***Stage 1:*** *$\mathcal{A}$ may ask any of* `Send`*,* `SessionKeyReveal`*,* `EphemeralKeyReveal` *and* `Corrupt` *queries to any oracle at will.*
- ***Stage 2:*** *$\mathcal{A}$ chooses a $\boldsymbol{\lambda}-$*CAFL-eCK*-fresh oracle and asks a* `Test` *query. The challenger chooses a random bit $b \leftarrow \{0, 1\}$, and if $b = 1$ then the actual session key is returned to $\mathcal{A}$, otherwise a random string chosen from the same session key space is returned to $\mathcal{A}$.*
- ***Stage 3:*** *$\mathcal{A}$ continues asking* `Send`*,* `SessionKeyReveal`*,* `EphemeralKeyReveal` *and* `Corrupt` *queries. $\mathcal{A}$ may not ask a query that violates the $\boldsymbol{\lambda}-$*CAFL-eCK*-freshness of the test session.*
- ***Stage 4:*** *At some point $\mathcal{A}$ outputs the bit $b' \leftarrow \{0, 1\}$ which is its guess of the value b on the test session. $\mathcal{A}$ wins if $b' = b$.*

The eCK security game is same as the above, except that in Stage 2 and Stage 3 eCK-fresh oracles are chosen instead of $\boldsymbol{\lambda}-$CAFL-eCK-fresh oracles. $\text{Succ}_{\mathcal{A}}$ is the event that the adversary $\mathcal{A}$ wins the security game in Definition 3.3.

Definition 3.4 ($\boldsymbol{\lambda}-$CAFL-eCK security)**.** *A protocol π is said to be $\boldsymbol{\lambda}-$*CAFL-eCK *secure if there is no adversary $\mathcal{A}$ that can win the $\boldsymbol{\lambda}-$*CAFL-eCK *security game with significant advantage. The advantage of an adversary $\mathcal{A}$ is defined as* $\text{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\pi}(\mathcal{A}) = |2\Pr(Succ_{\mathcal{A}}) - 1|$.

3.6 Practical Interpretation of Security of CAFL-eCK Model

We review the relationship between the CAFL-eCK model and real world attack scenarios.

- **Active adversarial capabilities:** `Send` queries address the powers of an active adversary who can control the message flow over the network. In the previous security models, this property is addressed by introducing the *send* query.
- **Side-channel attacks:** Leakage functions are embedded with the `Send` query. Thus, assuming that the leakage happens when computations take place in principals, a wide variety of side-channel attacks such as timing attacks, EM emission based attacks, power analysis attacks, which are based on *continuous partial leakage of long-term secrets* are addressed. This property is not addressed in the earlier security models such as the BR models, the CK model, the eCK model.
- **Malware attacks:** `EphemeralKeyReveal` queries cover the malware attacks which steal stored ephemeral keys, given that the long-term keys may be securely stored separately from the ephemeral keys in places such as smart cards or hardware security modules. Separately, `Corrupt` queries address malware attacks which steal the long-term secret keys of protocol principals.
- **Weak random number generators:** Due to weak random number generators, the adversary may correctly determine the produced random number. `EphemeralKeyReveal` query addresses situations where the adversary can get the ephemeral secrets.

4 Construction of a $\boldsymbol{\lambda}$ – CAFL-eCK-secure Protocol

In this section we present a construction of a new $\boldsymbol{\lambda}$ – CAFL-eCK-secure AKE protocol. Our new protocol construction has a similar structure to the protocol P1 of Alawatugoda [Ala17a]. The Protocol P1 is eCK-secure AKE protocol, that is constructed using a CCA2-secure public-key encryption scheme and a function from the pseudo-random function family, and based on the DDH hardness assumption.

4.1 Construction Details

We name our new protocol construction as Protocol P1.v2. The protocol P1.v2 shown in Table 2 is a Diffie-Hellman-style [DH76] AKE protocol. Let k be the security parameter and group $\mathbb{G}$ is generated using a group generation algorithm $\mathcal{G}$ that takes k as an input, where $\mathbb{G}$ is a group of prime order q with generators g_1, g_2 such that $g_2 = g_1^{\alpha}$.

In Protocol P1.v2, we use a CCLA2-secure leakage-resilient public-key encryption scheme PKE = (KG, Enc, Dec) (refer to the section 2.3) to encrypt protocol messages between parties. Given the security parameter k, KG generates a pair of secret/public keys for the leakage-resilient encryption scheme PKE. Let sk^0_{Alice}, pk_{Alice} be the secret/public keys of the PKE of Alice, and sk^0_{Bob}, pk_{Bob} be the secret/public keys of the PKE of Bob, at the initial key setup. Moreover, for the protocol P1.v2, we use the $(\ell, \lambda/2, \epsilon_1)$-secure leakage-resilient refreshing protocol $\text{Refresh}^{n,2}_{\mathbb{Z}_q}$ (refer to the section 2.2) of the (λ, ϵ)-secure LRS scheme $\Phi^{n,2}_{\mathbb{Z}_q}$. Let a_1, a_2 and b_1, b_2 be the Diffie-Hellman long-term secret keys of Alice and Bob respectively. Then, $A \leftarrow g_1^{a_1} g_2^{a_2} g_2^{a_1} g_1^{a_2}$ and $B \leftarrow g_1^{b_1} g_2^{b_2} g_2^{b_1} g_1^{b_2}$ be the corresponding public keys of Alice and Bob respectively. As the secret key is of form (x_1, x_2), it is information theoretically hidden against unbounded adversaries as well. As in Dziembowski and Faust [DF11], according to the point of view of Alice, we encode the secret key (a_1, a_2) as $(a^0_L, a^0_R) \leftarrow \text{Encode}^{n,2}_{\mathbb{Z}_q}(a_1, a_2)$ using an LRS scheme $\Phi^{n,2}_{\mathbb{Z}_q}$, such that $a^0_L = \{a^0_{L0}, a^0_{L1}, \ldots, a^0_{Ln}\} \in \mathbb{Z}^n_q$ and $a^0_R = (a^0_{R_1}, a^0_{R_2}) = \big\{\{a^0_{R_1 0}, a^0_{R_1 1}, \ldots, a^0_{R_1 n}\}, \{a^0_{R_2 0}, a^0_{R_2 1}, \ldots, a^0_{R_2 n}\}\big\} \in \mathbb{Z}^{n\times 2}_q$. Note that the inner product of the two matrices a^0_L and a^0_R give the secret key (a_1, a_2). After encoding the secret key (a_1, a_2) into two matrices (a^0_L, a^0_R), (a_1, a_2) is securely erased from the memory, and the encoded values (a^j_L, a^j_R) will be used for protocol computations. This procedure similarly applies at Bob as well. In the protocol P1.v2, the initial key setup (that is long-term public/secret key generation and encoding the Diffie-Hellman long-term secret keys) is done offline to preserve higher security.

Let x, X and y, Y be the Diffie-Hellman ephemeral secret and public keys of Alice and Bob respectively, of the current session. After exchanging the protocol messages ($\bar{X} \leftarrow \text{Enc}_{pk_{Bob}}(X)$ and $\bar{Y} \leftarrow \text{Enc}_{pk_{Alice}}(Y)$),

both principals decrypt the incoming messages. Then Alice and Bob compute the Diffie-Hellman ephemeral shared value of the current session (values Z_1 and Z'_1 respectively in Alice and Bob). At the computations of Z_1 and Z'_1, we do not consider the partial leakage of ephemeral secret keys x and y, that happens continuously over the computations, because they are one-time secrets that are only used for a single session. We can reasonably assume that an adversary cannot obtain enough partial leakage through a side-channel attack over computations to distinguish a ephemeral secret value, due to its short life time. The decryption operations implicitly refresh the current j-th session's long-term secret keys of PKE (sk^j_{Alice} becomes sk^{j+1}_{Alice} and sk^j_{Bob} becomes sk^{j+1}_{Bob}). The current j-th session's encoded secret keys, (a^j_L, a^j_R) and (b^j_L, b^j_R) are refreshed, such that $(a^{j+1}_L, a^{j+1}_R) \leftarrow \text{Refresh}^{n,2}_{\mathbb{Z}_q}(a^j_L, a^j_R)$ and $(b^{j+1}_L, b^{j+1}_R) \leftarrow \text{Refresh}^{n,2}_{\mathbb{Z}_q}(b^j_L, b^j_R)$, after the computation of the Diffie-Hellman long-term shared values (values Z_2 and Z'_2 respectively in Alice and Bob) of the current session.

Finally, protocol principals compute the session key K using the function PRF from the pseudo-random function family (refer to the section 2.5). Inputs to the PRF are the Diffie-Hellman ephemeral shared values (Z_1 at Alice and Z'_1 at Bob) and the Diffie-Hellman long-term shared values (Z_2 at Alice and Z'_2 at Bob) of the current session, together with the two protocol messages that are exchanged between Alice and Bob ($Alice\|\bar{X}\|Bob\|\bar{Y}$).

4.2 Leakage-Resilient Exponentiation Computation from Encoded Secrets

It is good to explain details about the computation of the Diffie-Hellman long-term shared values of the current session, namely Z_2 at Alice and Z'_2 at Bob. Because this exponentiation computation is performed using the encoded values of the long-term secret key. Note that encoding the secret key is the key mechanism to achieve the leakage resiliency of the protocol P1.v2. In order to understand this computation, following we explain how it is happened at the protocol principal Alice.

Let $\mathbb{G}$ be a group of prime order q with generators g_1 and g_2. Let $a_1, a_2 \leftarrow \mathbb{Z}_q$ be the Diffie-Hellman long-term secret key of Alice, and B be the Diffie-Hellman long-term public key of Bob. We need to compute the value Z_2 which is equivalent to $B^{a_1} \cdot B^{a_2}$ at Alice, but instead of a_1 and a_2 we have to use the encoded long-term secret keys of Alice. Let $a^j_L = \{a^j_{L0}, a^j_{L1}, \ldots, a^j_{Ln}\} \in \mathbb{Z}^n_q$ and $a^j_R = (a^j_{R_1}, a^j_{R_2}) = \{\{a^j_{R_10}, a^j_{R_11}, \ldots, a^j_{R_1n}\}, \{a^j_{R_20}, a^j_{R_21}, \ldots, a^j_{R_2n}\}\} \in \mathbb{Z}^{n\times 2}_q$ are the encoded long-term secret values of Alice at the current j-th session. Now we compute $T_1 = \{T_{10}, T_{11}, \ldots, T_{1n}\} = B^{a^j_{R_1}} \odot B^{a^j_{R_2}} \odot$, where $\odot$ denotes component-wise multiplication of vectors. Therefore, $T_{1i} = B^{a^j_{R_1 i}} \odot B^{a^j_{R_2 i}}$. Next, we compute $T_2 = \{T_{20}, T_{21}, \ldots, T_{2n}\} = T_1^{a^j_L}$. It gives, $T_2 = \{{T_{10}}^{a^j_{L1}}, {T_{11}}^{a^j_{L2}}, \ldots, {T_{1n}}^{a^j_{Ln}}\}$. Finally, we compute $Z_2 = \prod_{i=0}^{n} T_{2i}$. The protocol P1.v2 shown in Table 2, this computation is emphasized in a box, that is indicated as $\texttt{exp}(B, (a^j_L, a^j_R))$. This procedure similarly applies at Bob to compute Z'_2.

4.3 Security Analysis of the Protocol P1.v2

There is a leakage bound defined by the LRS scheme $\Phi^{n,2}_{\mathbb{Z}_q}$, that we call λ, and there is a leakage bound defined by the CCLA2-secure leakage-resilient public-key encryption scheme PKE, that we call λ'. Thus, we define a leakage bound $\boldsymbol{\lambda}$ for protocol P1.v2, such that $\boldsymbol{\lambda} = (\lambda, \lambda')$. According to the CAFL-eCK model, the adversary $\mathcal{A}$ is allowed to query leakage of the Diffie-Hellman long-term secret keys as well as the decryption keys of the PKE under the leakage bounds λ and λ' respectively, using separate leakage functions to each of the building blocks. Let $\mathbf{f}^j = (f^j_\Phi, f^j_{\text{PKE}})$ a tuple of leakage functions issues with a `Send` query, where j indicates the j-th leakage occurrence (j-th protocol session). The leakage function f^j_Φ is computed over the Diffie-Hellman long-term secret keys, and the leakage function f^j_{PKE} is computed over the decryption keys of a protocol principal at the current j-th session, to produce the leakage information. This leakage information is bounded by the corresponding leakage bound. Note that the Diffie-Hellman long-term secret keys and the decryption keys are encoded into portions. The adversary is allowed to obtain the leakage from each of these portions independently. The details about how the leakage is modelled from the encoded secrets is explained in section 2.1.

Alice (Initiator)	$\mathbb{G}, q, g_1, g_2 \leftarrow \mathcal{G}(1^k)$	*Bob* (Responder)
	Initial Key Setup (Offline)	
$a_1, a_2 \leftarrow \mathbb{Z}_q^2$ $A \leftarrow g_1^{a_1} g_2^{a_2} g_2^{a_1} g_1^{a_2}$ $(a_L^0, a_R^0) \leftarrow \text{Encode}_{\mathbb{Z}_q}^{n,2}(a_1, a_2)$ erase a_1, a_2 $sk_{Alice}^0, pk_{Alice} \leftarrow \text{KG}(1^k)$		$b_1, b_2 \leftarrow \mathbb{Z}_q^2$ $B \leftarrow g_1^{b_1} g_2^{b_2} g_2^{b_1} g_1^{b_2}$ $(b_L^0, b_R^0) \leftarrow \text{Encode}_{\mathbb{Z}_q}^{n,2}(b_1, b_2)$ erase b_1, b_2 $sk_{Bob}^0, pk_{Bob} \leftarrow \text{KG}(1^k)$
	Protocol Execution	
$x \leftarrow \mathbb{Z}_q, X \leftarrow g_1^x$ $\bar{X} \leftarrow \text{Enc}_{pk_{Bob}}(X)$	$\xrightarrow{Alice, \bar{X}}$ $\xleftarrow{Bob, \bar{Y}}$	$y \leftarrow \mathbb{Z}_q, Y \leftarrow g_1^y$ $\bar{Y} \leftarrow \text{Enc}_{pk_{Alice}}(Y)$
$Y, sk_{Alice}^{j+1} \leftarrow \text{Dec}_{sk_{Alice}^j}(\bar{Y})$ $Z_1 \leftarrow Y^x$		$X, sk_{Bob}^{j+1} \leftarrow \text{Dec}_{sk_{Bob}^j}(\bar{X})$ $Z_1' \leftarrow X^y$
$\texttt{exp}(B,(a_L^j,a_R^j))$ $T_1 \leftarrow B^{a_{R_1}^j} \odot B^{a_{R_2}^j}, T_2 \leftarrow T_1^{a_L^j}$ $Z_2 \leftarrow \prod_{i=0}^{n} T_{2i}$		$\texttt{exp}(A,(b_L^j,b_R^j))$ $T_3 \leftarrow A^{b_{R_1}^j} \odot A^{b_{R_2}^j}, T_4 \leftarrow T_3^{b_L^j}$ $Z_2' \leftarrow \prod_{i=0}^{n} T_{4i}$
$(a_L^{j+1}, a_R^{j+1}) \leftarrow \text{Refresh}_{\mathbb{Z}_q}^{n,1}(a_L^j, a_R^j)$		$(b_L^{j+1}, b_R^{j+1}) \leftarrow \text{Refresh}_{\mathbb{Z}_q}^{n,1}(b_L^j, b_R^j)$
$K \leftarrow \text{PRF}(Z_1, Alice\|\bar{X}\|Bob\|\bar{Y}) \oplus$ $\text{PRF}(Z_2, Alice\|\bar{X}\|Bob\|\bar{Y})$		$K \leftarrow \text{PRF}(Z_1', Alice\|\bar{X}\|Bob\|\bar{Y}) \oplus$ $\text{PRF}(Z_2', Alice\|\bar{X}\|Bob\|\bar{Y})$
	K is the session key	

Table 2: Protocol P1.v2

First, we show that the leakage of Diffie-Hellman long-term secret keys from a single $\texttt{exp}(\cdot,\cdot)$ computation of a protocol session can be simulated using a polynomial-time simulator $\mathcal{S}$, which has access to a leakage oracle $\Lambda(L^*, R^*)$, and an auxiliary information aux. The following lemma is the analogous version of Lemma 7 in Dziembowski and Faust [DF11].

Lemma 4.1. *The Diffie-Hellman long-term secret key of a protocol principal is $a_1, a_2 \leftarrow \mathbb{Z}_q^2$ and the corresponding public-key is $A \leftarrow g_1^{a_1} g_2^{a_2} g_2^{a_1} g_2^{a_2}$. Let B be a Diffie-Hellman long-term public key of another protocol principal. The the secret key is encoded such that $(L, R = (R_1, R_2)) \leftarrow \text{Encode}_{\mathbb{Z}_q}^{n,2}(a_1, a_2)$. Let λ be the leakage bound of each portion of the encoded secret key. Then, for a polynomial-time λ-limited adversary $\mathcal{A}$, there exists a polynomial-time λ-limited simulator $\mathcal{S}$ given access to the leakage oracle $\Lambda(L^*, R^*)$ such that for any $b \in \{0, 1\}$*

$$\Pr[Out(\mathcal{S}(A, \text{aux}), \Lambda(L^*, R^*)) = b] = \Pr[(\mathcal{A}(A) \leftrightarrows (\texttt{exp}(B, (L, R))) = b],$$

where $(L^*, R^*) \leftarrow \text{Encode}_{\mathbb{Z}_q}^{n,1}(a_1)$ and $\text{aux} = (R_1 + R_2)$. Note that $\texttt{exp}(B, (L, R))$ denotes the exponentiation computations of B, that are computed using the two encoded portions L and R, as emphasized in the protocol diagram in Table 2.

The proof is similar to the proof of Lemma 7 of Dziembowski and Faust [DF11].

Proof. (Sketch) The simulation is described as follows:

1. The simulator $\mathcal{S}$ is given the auxiliary information $\text{aux} = (R_1 + R_2)$, and given access to the leakage oracle $\Lambda(L^*, R^*)$. She sets $L = L^*$ and $R_1 = R^*$.

2. $\mathcal{S}$ simulates the leakage of L and R_1 using the leakage oracle $\Lambda(L^*, R^*)$.

3. The leakage of R_2 can be described from the relation $R_2 = \text{aux} - R^*$. Therefore, $\mathcal{S}$ can perfectly simulate the leakage of R_2 using the leakage oracle $\Lambda(L^*, R^*)$ and the auxiliary value aux.

□

Above we have shown that the leakage of Diffie-Hellman long-term secret keys from a single protocol session can be perfectly simulated. Then, we will use this observation to prove that the leakage from several protocol sessions will also not help the adversary to learn about the encoded Diffie-Hellman long-term secret keys. For this, we require to refresh the encoded secret key periodically. Therefore, after the exponentiation computations (denoted by `exp`), the refreshing protocol $\text{Refresh}^{n,2}_{\mathbb{Z}_q}$ is executed on L and R to output L' and R' respectively. Then, we set (L', R') as the secret key for the next protocol session. We denote the aforementioned combination of operations by `expRef`, that is `exp` followed by $\text{Refresh}^{n,2}_{\mathbb{Z}_q}$. Now we show that an adversary cannot learn enough information on the Diffie-Hellman long-term secret key, even she can issue arbitrary ℓ number of leakage queries to the protocol principal. Recall that according to the CAFL-eCK model leakage queries are issued with `Send` queries. The following is the analogous version of Lemma 8 in Dziembowski and Faust [DF11].

Lemma 4.2. *The Diffie-Hellman long-term secret key of a protocol principal is $a_1, a_2 \leftarrow \mathbb{Z}_q^2$ and the corresponding public-key is $A \leftarrow g_1^{a_1} g_1^{a_2} g_2^{a_1} g_2^{a_2}$. Let $\ell, n \in \mathbb{N}$ and suppose that $\Phi^{n,2}_{\mathbb{Z}_q}$ is a (λ, ϵ)-secure LRS where λ is the leakage bound of each portion of the encoded secret key. Let W be the vector that defines auxiliary information. Then for every $S = (a_1, a_2), S' = (a'_1, a'_2)$ that satisfies $g_1^{a_1} g_2^{a_2} g_2^{a_1} g_1^{a_2} = g_1^{a'_1} g_2^{a'_2} g_2^{a'_1} g_1^{a'_2}$, and any $(\lambda/2 - 1)$-limited adversary $\mathcal{A}$, we have,*

$$\Delta\big(\text{Exp}^{\text{aux}}_{\texttt{expRef}((B,(L,R))}(\mathcal{A}, S, \ell, W, A), \text{Exp}^{\text{aux}}_{\texttt{expRef}(B,(L,R))}(\mathcal{A}, S', \ell, W, A)\big) \leq \epsilon_1 \ ,$$

where ϵ_1 is negligible in the statistical security parameter n.

The proof is the same as the proof of Lemma 8 of Dziembowski and Faust [DF11].

Proof. (Sketch) Note that (W, A) defines a 1-dimensional subspace $Z \subset (\mathbb{Z}_q)^2$ that contains all the pairs (a_1, a_2) that correspond to the public key A. For any $S, S' \in Z$ and any $(\lambda/2 - 1)$-limited adversary $\mathcal{A}$ we have by the Theorem 2 (Generalization of Theorem 1) of Dziembowski and Faust [DF11], for refreshing of $\Phi^{n,2}_{\mathbb{Z}_q}$ that,

$$\Delta(\text{Exp}^{\text{aux}}_{\texttt{expRef}(B,(L,R))}(\mathcal{A}, S, \ell, W, A), \text{Exp}^{\text{aux}}_{\texttt{expRef}(B,(L,R))}(\mathcal{A}, S', \ell, W, A)) \leq 2\ell p^6(3\epsilon + 2p^{-n-5}) \ . \tag{1}$$

The above experiment addresses only the leakage from the refreshing operation of (L, R). In order to combine this with leakage from `exp`, we use the leakage simulation from Lemma 4.1. We can apply this lemma since (1)–equation 1 is shown by reduction to the (λ, ϵ)-security of $\Phi^{n,2}_{\mathbb{Z}_q}$ and (2)–aux is known to the leakage simulator. This completes the proof of Lemma 4.2, by proving that ϵ_1 is negligible in the statistical security parameter n. □

Using Lemma 4.1 and Lemma 4.2, we prove that continuous, λ-limited leakage of the two encodings of a Diffie-Hellman long-term secret key, that happens due to `expRef` computations ($\texttt{expRef} + \text{Refresh}^{n,2}_{\mathbb{Z}_q}$), will not reveal the Diffie-Hellman long-term secret key to an adversary. Using the aforementioned result, now we prove that our construction, protocol P1.v2, is $\boldsymbol{\lambda}$ − CAFL-eCK-secure in the standard model. Here we consider the λ'-limited leakage happens from the secret keys of the encryption scheme PKE, during the decryption operation. It completes our security analysis for the protocol P1.v2. Since the protocol P1.v2 has a similar structure to the Protocol P1 of Alawatugoda, the security analysis of the protocol P1.v2 also has a similar security analysis to the Protocol P1, by means of different cases to be analyzed.

Theorem 4.3. *If $\mathbb{G}$ is a group of a prime order q with generator g_1, g_2 such that $g_2 = g_1^{\alpha}$, where the* (DDH) *assumption holds, the underlying public-key encryption scheme* PKE *is* CCLA2*-secure against any λ'-limited adversary, the function* PRF *is a function from the pseudo-random function family, and the refreshing protocol* $\text{Refresh}^{n,2}_{\mathbb{Z}_q}$ *is $(\ell, \lambda/2, \epsilon_1)$-secure refreshing protocol of a (λ, ϵ)-secure leakage-resilient storage scheme $\Phi^{n,2}_{\mathbb{Z}_q}$, then the protocol* P1.v2 *is secure in the* CAFL-eCK *model.*

Let $\mathcal{U} = \{U_1, \ldots, U_{N_P}\}$ be a set of N_P parties. Each party U_i owns at most N_s number of protocol sessions. Let $\mathcal{A}$ be any adversary against the CAFL-eCK *challenger of the protocol* P1.v2*. Then, the advantage of $\mathcal{A}$ against the* CAFL-eCK *security challenge of the protocol* P1.v2*,* $\mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2}}$ *is:*

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2}}(\mathcal{A}) \leq N_P^2 N_s^2 \cdot \max\Big(\big(2\epsilon_1 + \mathrm{Adv}^{\text{DDH}}_{\mathbb{G},q}(\mathcal{C}) + \epsilon_{\text{PRF}}\big), \big(\epsilon_{\text{PRF}} + \mathrm{Adv}^{\text{CCLA2}}_{\text{PKE}}(\mathcal{D})\big)\Big) \ .$$

where $\mathcal{C}$ is the algorithm against a DDH *challenger and $\mathcal{D}$ is the λ'-limited adversary against the* CCLA2 *challenger of the underlying public-key encryption scheme* PKE*.*

Proof. We split the proof of Theorem 4.3 into two main cases and sub cases as mentioned below:

1. A partner to the test session exists.
   (a) Adversary corrupts both the owner and the partner principals to the test session - Case **1a**
   (b) Adversary corrupts neither the owner nor the partner principal to the test session - Case **1b**
   (c) Adversary corrupts the owner to the test session, but does not corrupt the partner to the test session - Case **1c**
   (d) Adversary corrupts the partner to the test session, but does not corrupt the owner to the test session - Case **1d**
2. A partner to the test session does not exist: the adversary is not allowed to corrupt the peer to the target session.
   (a) Adversary corrupts the owner to the test session - Case **2a**
   (b) Adversary does not corrupt the owner to the test session - Case **2b**

Note that the secret keys (a_1, a_2) and (b_1, b_2) are securely erased from the memory at the initial key setup phase. Therefore, the encoded values (a_L^j, a_R^j) and (b_L^j, b_R^j) are used to answer the `Corrupt` queries.

Case 1a: Adversary corrupts both the owner and partner principals to the test session.

Game 1: This is the original game. When `Test` query is asked the game 1 challenger will choose a random bit $b \leftarrow \{0, 1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$, otherwise a random value chosen from the same session-key space is given. Hence,

$$\mathrm{Adv}_{\text{Game 1}}(\mathcal{A}) = \mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2,Case 1a}}(\mathcal{A}) \ . \tag{2}$$

Game 2: Same as game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \leftarrow \{U_1, ..., U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \leftarrow \{1, ...N_s\}$ are chosen, where N_P is the number of protocol principals and N_s is the number of sessions on a principal. The session $\Pi^{s^*}_{U^*,V^*}$ is chosen as the target session and the session $\Pi^{t^*}_{V^*,U^*}$ is chosen as the partner to the target session. If the test session is not the session $\Pi^{s^*}_{U^*,V^*}$ or partner to the session is not $\Pi^{t^*}_{V^*,U^*}$, the game 2 challenger aborts the game. Unless the incorrect choice happens, the game 2 is identical to the game 1. Hence,

$$\mathrm{Adv}_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} \mathrm{Adv}_{\text{Game 1}}(\mathcal{A}) \ . \tag{3}$$

Game 3: Same as game 2 with the following exception: the game 3 challenger randomly chooses $z \leftarrow \mathbb{Z}_q$ and computes K according to the protocol description, using $Z_1 = g_1^z$. When the adversary asks the `Test`(U^*, V^*, s^*) query, the game 3 challenger will answer with the K.

We construct an algorithm $\mathcal{C}$ against a DDH challenger, using the adversary $\mathcal{A}$ as a sub routine. The game 3 challenger sets the long-term secret/public key pairs (Diffie-Hellman and encryption key pairs) of all the protocol principals. Note that the adversary $\mathcal{A}$ can also set long-term keys at corrupted principals.

The algorithm $\mathcal{C}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$, such that $\mathcal{A}$ is interacting with either game 2 or game 3. The DDH challenger sends values (g_1^x, g_1^y, g_1^z), such that either $z = xy$ or $z \leftarrow \mathbb{Z}_q$, as the inputs to the algorithm $\mathcal{C}$. The game 3 challenger sets X of the target session $(\Pi^{s^*}_{U^*,V^*})$ as g_1^x, Y of the target session $(\Pi^{s^*}_{U^*,V^*})$ as g_1^y, and computes the K according to the protocol specification, using g_1^z as Z_1, upon receiving the $\texttt{Test}(U^*, V^*, s^*)$ query. The game 3 challenger can answer all the other queries normally.

If $\mathcal{C}$'s input is a Diffie-Hellman triple, simulation constructed by the game 3 challenger is identical to game 2, otherwise it is identical to game 3. If $\mathcal{A}$ can distinguish the difference between games, then $\mathcal{C}$ can answer the DDH challenge. Hence,

$$|\mathrm{Adv}_{\text{Game 2}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 3}}(\mathcal{A})| \leq \mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G},q}(\mathcal{C}) \ . \tag{4}$$

Game 4: Same as game 3 with the following exception: the game 4 challenger randomly chooses $K \leftarrow \{0,1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the $\texttt{Test}(U^*, V^*, s^*)$ query.

The game 4 challenger sets the Diffie-Hellman long-term secret/public key pairs and the encryption key pairs of all the protocol principals. Therefore, the challenger can answer all the queries normally.

If K is computed using the real PRF with a hidden key, the simulation is identical to game 3, whereas if K is chosen randomly from the session key space, the simulation constructed is identical to game 4. Hence,

$$|\mathrm{Adv}_{\text{Game 3}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 4}}(\mathcal{A})| \leq \epsilon_{\mathrm{PRF}} \ . \tag{5}$$

Semantic security of the session key in Game 4: Since the session key K of $\Pi^{s^*}_{U^*,V^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in game 4. Hence,

$$\mathrm{Adv}_{\text{Game 4}}(\mathcal{A}) = 0 \ . \tag{6}$$

Using equations (2)–(6) we find,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2,Case 1a}}(\mathcal{A}) \leq {N_P}^2 {N_s}^2 \Big(\mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G},q}(\mathcal{C}) + \epsilon_{\mathrm{PRF}} \Big) \ .$$

Case 1b: Adversary corrupts neither the owner nor the partner principals to the test session.

Game 1: This is the original game. When $\texttt{Test}$ query is asked the game 1 challenger will choose a random bit $b \leftarrow \{0,1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$, otherwise a random value chosen from the same session-key space is given. Hence,

$$\mathrm{Adv}_{\text{Game 1}}(\mathcal{A}) = \mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2,Case 1b}}(\mathcal{A}). \tag{7}$$

Game 2: Same as game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \leftarrow \{U_1, ..., U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \leftarrow \{1, ...N_s\}$ are chosen, where N_P is the number of protocol principals and N_s is the number of sessions on a principal. The session $\Pi^{s^*}_{U^*,V^*}$ is chosen as the target session and the session $\Pi^{t^*}_{V^*,U^*}$ is chosen as the partner to the target session. If the test session is not the session $\Pi^{s^*}_{U^*,V^*}$ or partner to the session is not $\Pi^{t^*}_{V^*,U^*}$, the game 2 challenger aborts the game. Unless the incorrect choice happens, the game 2 is identical to the game 1. Hence,

$$\mathrm{Adv}_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} \mathrm{Adv}_{\text{Game 1}}(\mathcal{A}). \tag{8}$$

Game 3: Let (a_1, a_2) be the Diffie-Hellman long-term secret keys of U^*, that are set according to the protocol specification. Let A be the Diffie-Hellman long-term public key, that is computed from (a_1, a_2), according to the protocol specification. Moreover, the game 3 challenger sets the other Diffie-Hellman long-term secret/public key pairs and all the encryption key pairs of protocol principals. This is as game 2 with the following changes:

1. Sample $(a_1', a_2') \leftarrow \{(o,r)|{g_1}^o {g_2}^r {g_2}^o {g_1}^r = {g_1}^{a_1}{g_2}^{a_2}{g_1}^{a_2}{g_2}^{a_1}\}$ at random and sets $\mathrm{Encode}_{\mathbb{Z}_q}^{n,2}(a_1', a_2')$ as the Diffie-Hellman long-term secret keys of U^*.
2. Compute the answers to the protocol queries and the leakage queries at U^* with $\mathrm{Encode}_{\mathbb{Z}_q}^{n,2}(a_1', a_2')$ until the adversary asks for the $\mathtt{Test}(U^*, V^*, s^*)$ query.
3. Compute the answer for the $\mathtt{Test}(U^*, V^*, s^*)$ query by using (a_1, a_2) as in the previous game.
4. Answer the protocol queries and leakage queries at U^* with $\mathrm{Encode}_{\mathbb{Z}_q}^{n,2}(a_1', a_2')$ further.

Other than the aforementioned change, the game 3 challenger can answer all the other queries normally. It is easy to see that the distance between game 2 and game 3 is negligible by the Lemma 4.2. Thus,

$$|\mathrm{Adv}_{\text{Game }2}(\mathcal{A}) - \mathrm{Adv}_{\text{Game }3}(\mathcal{A})| \leq \epsilon_1 \ . \tag{9}$$

Game 4: Let (b_1, b_2) be the Diffie-Hellman long-term secret keys of V^*, that are set according to the protocol specification. Let B be the Diffie-Hellman long-term public key, that is computed from (b_1, b_2), according to the protocol specification. Moreover, the game 4 challenger sets the other Diffie-Hellman long-term secret/public key pairs and all the encryption key pairs of protocol principals. This is as game 3 with the following changes:

1. Sample $(b_1', b_2') \leftarrow \{(o,r)|{g_1}^o {g_2}^r {g_2}^o {g_1}^r = {g_1}^{b_1}{g_2}^{b_2}{g_1}^{b_2}{g_2}^{b_1}\}$ at random and sets $\mathrm{Encode}_{\mathbb{Z}_q}^{n,2}(b_1', b_2')$ as the Diffie-Hellman long-term secret keys of V^*.
2. Compute the answers to the protocol queries and the leakage queries at V^* with $\mathrm{Encode}_{\mathbb{Z}_q}^{n,2}(b_1', b_2')$ until the adversary asks for the $\mathtt{Test}(U^*, V^*, s^*)$ query.
3. Compute the answer for the $\mathtt{Test}(U^*, V^*, s^*)$ query by using (b_1, b_2) as in the previous game.
4. Answer the relevant protocol queries and leakage queries at V^* with $\mathrm{Encode}_{\mathbb{Z}_q}^{n,2}(b_1', b_2')$ further.

Other than the aforementioned change, the game 4 challenger can answer all the other queries normally. It is easy to see that the distance between game 3 and game 4 is negligible by the Lemma 4.2. Thus,

$$|\mathrm{Adv}_{\text{Game }3}(\mathcal{A}) - \mathrm{Adv}_{\text{Game }4}(\mathcal{A})| \leq \epsilon_1 \ . \tag{10}$$

Game 5: Same as game 4 with the following exception: the game 5 challenger randomly chooses $\delta \leftarrow \mathbb{Z}_q$ and computes K according to the protocol description, using g_1^δ for the computation of Z_2. When the adversary asks the $\mathtt{Test}(U^*, V^*, s^*)$ query, the game 5 challenger will answer with K.

We construct an algorithm $\mathcal{C}$ against a DDH challenger, using the adversary $\mathcal{A}$ as a sub routine. Recall that $g_2 = g_1^\alpha$ and the game 5 challenger knows α. The DDH challenger sends values $(g_1^\beta, g_1^\gamma, g_1^\delta)$ such that either $\delta = \beta\gamma$ or $\delta \leftarrow \mathbb{Z}_q$, as the inputs to the algorithm $\mathcal{C}$. The game 5 challenger sets the encryption key pairs of all the protocol principals, and sets the Diffie-Hellman long-term secret/public key pairs of the other protocol principals, except for U^* and V^*. For U^* and V^*, sets only one Diffie-Hellman long-term secret value at each principal; precisely, a_2 at U^* and b_2 at V^*. Then, compute the Diffie-Hellman long-term public value of U^* as $g_1^\beta g_2^{a_2} g_1^{a_2} (g_1^\beta)^\alpha$, and the Diffie-Hellman long-term public value of V^* as $g_1^\gamma g_2^{b_2} g_1^{b_2} (g_1^\gamma)^\alpha$. The leakage of U^* and V^* are computed as in game 3 and game 4. We explain it again here briefly: The game 5 challenger samples $(a_1', a_2') \leftarrow \{(o,r)|{g_1}^o {g_2}^r {g_2}^o {g_1}^r = {g_1}^\beta {g_2}^{a_2}{g_1}^{a_2}{g_1}^{\alpha\beta}\}$ at random and sets $\mathrm{Encode}_{\mathbb{Z}_q}^{n,2}(a_1', a_2')$ at U^*, and $(b_1', b_2') \leftarrow \{(o,r)|{g_1}^o {g_2}^r {g_2}^o {g_1}^r = {g_1}^\gamma {g_2}^{b_2}{g_1}^{b_2}{g_1}^{\alpha\gamma}\}$ at random and sets $\mathrm{Encode}_{\mathbb{Z}_q}^{n,2}(b_1', b_2')$ at V^* as the Diffie-Hellman long-term secret keys. Then, the leakage queries to U^* and V^* are answered using them. The game 5 challenger can answer all the other queries normally: it computes session keys according to the protocol description, using g_1^δ for the computation of Z_2, at the sessions involving both U^* and V^*.

If $\mathcal{C}$'s input is a Diffie-Hellman triple, simulation constructed by the game 5 challenger is identical to game 4, otherwise it is identical to game 5. If $\mathcal{A}$ can distinguish the difference between games, then $\mathcal{C}$ can answer the DDH challenge. Hence,

$$|\mathrm{Adv}_{\text{Game }4}(\mathcal{A}) - \mathrm{Adv}_{\text{Game }5}(\mathcal{A})| \leq \mathrm{Adv}_{\mathbb{G},q}^{\mathrm{DDH}}(\mathcal{C}). \tag{11}$$

Game 6: Same as game 5 with the following exception: the game 6 challenger randomly chooses $K \leftarrow \{0,1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the $\texttt{Test}(U^*, V^*, s^*)$ query.

The game 6 challenger sets the Diffie-Hellman long-term secret/public key pairs and the encryption key pairs of all the protocol principals, as in the previous game. Therefore, the challenger can answer all the queries normally.

If K is computed using the real PRF with a hidden key, the simulation is identical to game 5, whereas if K is chosen randomly from the session key space, the simulation constructed is identical to game 6. Hence,

$$|\mathrm{Adv}_{\text{Game 5}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 6}}(\mathcal{A})| \leq \epsilon_{\text{PRF}} \ . \tag{12}$$

Semantic security of the session key in Game 6: Since the session key K of $\Pi^{s^*}_{U^*,V^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in game 6. Hence,

$$\mathrm{Adv}_{\text{Game 6}}(\mathcal{A}) = 0. \tag{13}$$

Using equations (7)–(13) we find,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2,Case 1b}}(\mathcal{A}) \leq {N_P}^2 {N_s}^2 \Big(2\epsilon_1 + \mathrm{Adv}^{\text{DDH}}_{\mathbb{G},q}(\mathcal{C}) + \epsilon_{\text{PRF}}\Big).$$

Case 1c: Adversary corrupts the owner to the test session, but does not corrupt the partner.

Game 1: This is the original game. When $\texttt{Test}$ query is asked the game 1 challenger will choose a random bit $b \leftarrow \{0,1\}$. If $b = 1$, the real session key is given to $\mathcal{A}$, otherwise a random value chosen from the same session-key space is given. Hence,

$$\mathrm{Adv}_{\text{Game 1}}(\mathcal{A}) = \mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2,Case 1c}}(\mathcal{A}). \tag{14}$$

Game 2: Same as game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \leftarrow \{U_1, ..., U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \leftarrow \{1, ...N_s\}$ are chosen, where N_P is the number of protocol principals and N_s is the number of sessions on a principal. The session $\Pi^{s^*}_{U^*,V^*}$ is chosen as the target session and the session $\Pi^{t^*}_{V^*,U^*}$ is chosen as the partner to the target session. If the test session is not the session $\Pi^{s^*}_{U^*,V^*}$ or partner to the session is not $\Pi^{t^*}_{V^*,U^*}$, the game 2 challenger aborts the game. Unless the incorrect choice happens, the game 2 is identical to game 1. Hence,

$$\mathrm{Adv}_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} \mathrm{Adv}_{\text{Game 1}}(\mathcal{A}). \tag{15}$$

Game 3: Same as game 2 with the following exception: the game 3 challenger randomly chooses C from the ciphertext space as encryption of the value X of the session $\Pi^{s^*}_{U^*,V^*}$, and sends it to the session $\Pi^{t^*}_{V^*,U^*}$ as having come from the session $\Pi^{s^*}_{U^*,V^*}$.

We introduce an algorithm $\mathcal{D}$ that is constructed using the adversary $\mathcal{A}$. If $\mathcal{A}$ can distinguish the difference between game 2 and game 3, then $\mathcal{D}$ can be used against the CCLA2 challenger of underlying public-key cryptosystem, PKE. The game 3 challenger uses the public key of the CCLA2 challenger as the encryption public key of the protocol principal V^*, and sets the other long-term public/secret key pairs (Diffie-Hellman and encryption keys) for protocol principals. $\mathcal{D}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$, such that it is interacting with either game 2 or game 3. $\mathcal{D}$ picks two random strings, $X_0, X_1 \leftarrow \mathbb{Z}_q$ and passes them to the CCLA2 challenger. From the CCLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C such that $C \leftarrow \mathrm{Enc}(pk_{V^*}, X_\theta)$ where $X_\theta = X_0$ or $X_\theta = X_1$. The game 3 challenger uses X_1 as the decryption of C when answering the $\texttt{Test}(U^*, V^*, s^*)$ query. With the aid of the CCLA2 challenger the game 3 challenger can compute the leakage of the decryption key of the principal V^*, and the corresponding decryptions at V^*, to answer the queries of $\mathcal{A}$. The game 3 challenger can answer all the other queries normally. Upon receiving the $\texttt{Test}(U^*, V^*, s^*)$ query, X_1 is used to compute the Z_1 and K is computed using the Z_1. Note that, the CCLA2 challenger is allowed to answer the leakage queries and the decryption queries even after issuing the

challenge ciphertext C. Therefore, the game 3 challenger can answer all the queries normally, even after the $\texttt{Test}(U^*, V^*, s^*)$ query is answered. Recall that the leakage bound of the PKE is λ'.

If the value C is the encryption of the value X_1, the simulation constructed by the game 3 challenger is identical to the game 2, otherwise it is identical to game 3. If $\mathcal{A}$ can distinguish the difference between games, then $\mathcal{D}$ can answer the CCLA2 challenge successfully. Hence,

$$|\mathrm{Adv}_{\text{Game 2}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 3}}(\mathcal{A})| \leq \mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{PKE}}(\mathcal{D}). \tag{16}$$

Game 4: Same as game 3 with the following exception: the game 4 challenger randomly chooses $K \leftarrow \{0,1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the $\texttt{Test}(U^*, V^*, s^*)$ query.

The game 4 challenger sets the Diffie-Hellman long-term secret/public key pairs and the encryption key pairs of all the protocol principals, as in the previous game. Therefore, the challenger can answer all the queries normally.

If K is computed using the real PRF with a hidden key, the simulation is identical to game 3, whereas if K is chosen randomly from the session key space, the simulation constructed is identical to game 4. Hence,

$$|\mathrm{Adv}_{\text{Game 3}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 4}}(\mathcal{A})| \leq \epsilon_{\mathrm{PRF}} \ . \tag{17}$$

Semantic security of the session key in Game 4: Since the session key K of $\Pi^{s^*}_{U^*,V^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in game 4. Hence,

$$\mathrm{Adv}_{\text{Game 4}}(\mathcal{A}) = 0. \tag{18}$$

Using equations (14)–(18) we find,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2,Case 1c}}(\mathcal{A}) \leq N_P^2 {N_s}^2 \Big(\epsilon_{\mathrm{PRF}} + \mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{PKE}}(\mathcal{D})\Big).$$

Case 1d: Adversary corrupts the partner to the test session, but does not corrupt the owner.

The analysis of this case is similar the analysis of case 1c. The only difference with the case 1c is that we have to set the public key of the CCLA2 challenger at U^*. Therefore, we get,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2,Case 1d}}(\mathcal{A}) \leq N_P^2 {N_s}^2 \Big(\epsilon_{\mathrm{PRF}} + \mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{PKE}}(\mathcal{D})\Big).$$

Case 2a: Adversary corrupts the owner to the test session.

There is no partner existing to the test session. The adversary $\mathcal{A}$ computes the corresponding protocol message as the partner principal and sends to the owner (U^*) of the test session. The analysis of this case is the same as the analysis of case 1c. Therefore, we get,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2,Case 2a}}(\mathcal{A}) \leq N_P^2 {N_s}^2 \Big(\epsilon_{\mathrm{PRF}} + \mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{PKE}}(\mathcal{D})\Big).$$

Case 2b: Adversary does not corrupt the owner to the test session.

There is no partner existing to the test session. The adversary $\mathcal{A}$ computes the corresponding protocol message as the partner principal and sends to the owner (U^*) of the test session. The analysis of this case is the same as the analysis of case 1b. Therefore, we get,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2,Case 2b}}(\mathcal{A}) \leq N_P^2 {N_s}^2 \Big(2\epsilon_1 + \mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G},q}(\mathcal{C}) + \epsilon_{\mathrm{PRF}}\Big).$$

Combining all the above cases: According to the analysis we can see the adversary $\mathcal{A}$'s advantage of winning against the CAFL-eCK challenger of the protocol P1.v2 is:

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}_{\text{P1.v2}}(\mathcal{A}) \leq N_P^2 N_s^2 \cdot \max\Big(\big(2\epsilon_1 + \mathrm{Adv}^{\text{DDH}}_{\mathbb{G},q}(\mathcal{C}) + \epsilon_{\text{PRF}}\big), \big(\epsilon_{\text{PRF}} + \mathrm{Adv}^{\text{CCLA2}}_{\text{PKE}}(\mathcal{D})\big)\Big) \ .$$

□

4.4 Practical Information of the Protocol P1.v2

Now we discuss about the practical information about the protocol P1.v2.

4.4.1 Leakage Bound

The leakage bound $\boldsymbol{\lambda}$ of the protocol P1.v2 is (λ, λ'). Trivially, the leakage bound λ of the Diffie-Hellman long-term secret keys is as same as the leakage bound of the underlying refreshing protocol $\text{Refresh}^{n,2}_{\mathbb{Z}_q}$. As shown in Corollary 1 of Dziembowski and Faust [DF11], the leakage bound of the LRS $\Phi^{n,2}_{\mathbb{Z}_q}$ is $0.3|\mathbb{Z}_q|^n$ bits, for a statistical security parameter n. As shown in Corollary 2 of Dziembowski and Faust [DF11], the leakage bound of the refreshing protocol $\text{Refresh}^{n,2}_{\mathbb{Z}_q}$ is $0.15|\mathbb{Z}_q|^n - 1 = 0.15n \log q - 1$ bits. Therefore, per leakage query $0.15n \log q - 1$ bits of leakage is allowed from each of the two encodes of the secret key, independently from each other. For a security parameter k, $\log q \geq k$ and n is a function of k. Thus, the λ is $1/n - o(1)$ of a Diffie-Hellman long-term secret key. The leakage bound λ' for a decryption secret key of the underlying leakage-resilient public-key encryption scheme PKE is specific to the particular public-key encryption scheme.

4.4.2 Computation Cost

Now we look at the number of operations to be performed at a protocol principal for a single protocol execution. We count the number of exponentiation operations only. Note that we ignored the multiplication and addition operations since they are significantly lighter than the exponentiation operations. Table 3 shows the details of performance analysis. Recall that $n \in \mathbb{N}$ is the statistical security parameter, that is a function of the security parameter k.

	Operation (at initiator or responder)	**Computation Cost**
Initial Key Setup		
(Offline)	Diffie-Hellman long-term key pair generation	1ME
	Encryption scheme key pair generation	1KG
	Encoding Diffie-Hellman long-term secret key	0
Protocol Execution		
	$\bar{X}/\bar{Y}$ computation	1E, 1Enc
	Z_1/Z_1' computation	1Dec, 1E
	Z_2/Z_2' computation	$(3n)$E
	K computation	1PRF
	Refreshing	0

Key: E – exponentiation operation; ME – multi-exponentiation; KG – key generation; Enc – encryption operation; Dec – decryption operation; PRF – pseudo-random function operation

Table 3: Overall computation cost at a protocol principal

5 Conclusions and Future Works

In this paper, we have presented a Diffie-Hellman-style construction of a $\boldsymbol{\lambda}-$CAFL-eCK-secure AKE protocol. Our protocol is proven to be secure in the standard model and based on the hardness assumption of the DDH problem. Our protocol can be instantiated with any CCLA2-secure public-key encryption scheme and a function from the pseudo-random function family. Hence, our protocol is a good AKE protocol candidate for future TLS protocol suits, that can be implemented and deployed with minimal implementation cost, given existing implementations of a CCLA2-secure public-key encryption scheme and a function from the pseudo-random function family.

Encoding the secret key into two portions and allowing only independent leakage from them seems somewhat strong assumption. Therefore, as a future work, it is better to advance the knowledge towards stripping off this strong assumption, while preserving the security features of this protocol in the standard model. Further, reducing the computation cost and reaching to the optimum leakage bound of $1 - o(1)$ should be valuable directions to work on. On the other hand, research on quantum-safe leakage-resilient AKE constructions are worthwhile as the quantum computers will be a reality in the near future.

References

[ABS14] Janaka Alawatugoda, Colin Boyd, and Douglas Stebila. Continuous after-the-fact leakage-resilient key exchange. In *Information Security and Privacy - 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings*, pages 258–273, 2014.

[AJR11] J. Alawatugoda, D. Jayasinghe, and R. Ragel. Countermeasures against Bernstein's remote cache timing attack. In *6th IEEE International Conference on Industrial and Information Systems (ICIIS)*, pages 43 –48, Aug. 2011.

[Ala17a] Janaka Alawatugoda. Generic construction of an eck -secure key exchange protocol in the standard model. *Int. J. Inf. Sec.*, 16(5):541–557, 2017.

[Ala17b] Janaka Alawatugoda. On the leakage-resilient key exchange. *J. Math. Cryptol.*, 11(4):215–269, 2017.

[ASB14] Janaka Alawatugoda, Douglas Stebila, and Colin Boyd. Modelling after-the-fact leakage for key exchange. In *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, pages 207–216, 2014.

[ASB15] Janaka Alawatugoda, Douglas Stebila, and Colin Boyd. Continuous after-the-fact leakage-resilient eck-secure key exchange. In *Cryptography and Coding - 15th IMA International Conference, IMACC 2015, Oxford, UK, December 15-17, 2015. Proceedings*, pages 277–294, 2015.

[BB03] David Brumley and Dan Boneh. Remote timing attacks are practical. In *USENIX Security Symposium*, pages 1–14, 2003.

[BCNP09] Colin Boyd, Yvonne Cliff, Juan Manuel González Nieto, and Kenneth G. Paterson. One-round key exchange in the standard model. *International Journal of Advanced Computer Technology*, pages 181–199, 2009.

[BR93] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *CRYPTO*, pages 232–249, 1993.

[BR95] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution - the three party case. pages 57–66. ACM Press, 1995.

[CAR17] Suvradip Chakraborty, Janaka Alawatugoda, and C. Pandu Rangan. Leakage-resilient non-interactive key exchange in the continuous-memory leakage setting. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li, editors, *Provable Security - 11th International Conference, ProvSec 2017, Xi'an, China, October 23-25, 2017, Proceedings*, volume 10592 of *Lecture Notes in Computer Science*, pages 167–187. Springer, 2017.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT*, pages 453–474, 2001.

[CMY+16] Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo, and Fuchun Guo. Strongly leakage-resilient authenticated key exchange. In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, volume 9610 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 2016.

[CMY+17] Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo, and Fuchun Guo. Strong authenticated key exchange with auxiliary inputs. *Des. Codes Cryptogr.*, 85(1):145–173, 2017.

[DF11] Stefan Dziembowski and Sebastian Faust. Leakage-resilient cryptography from the inner-product extractor. In *ASIACRYPT*, pages 702–721, 2011.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, pages 644 – 654, 1976.

[HAR13] U. Herath, J. Alawatugoda, and R. Ragel. Software implementation level countermeasures against the cache timing attack on advanced encryption standard. In *2013 IEEE 8th International Conference on Industrial and Information Systems*, pages 75–80, 2013.

[HMF07] Michael Hutter, Stefan Mangard, and Martin Feldhofer. Power and EM attacks on passive 13.56MHz RFID devices. In *CHES*, pages 320–333, 2007.

[HR07] I. Herath and R.G. Ragel. Side channel attacks: Measures and countermeasures. In *14th Annual Conference of the IET Sri Lanka Network*, 2007.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* Chapman and Hall/CRC Press, 2007.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO*, pages 104–113, 1996.

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

[MDS02] T.S. Messerges, E.A. Dabbish, and R.H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, pages 541–552, 2002.

[MO11] Daisuke Moriyama and Tatsuaki Okamoto. Leakage resilient eCK-secure key exchange protocol without random oracles. In *ASIACCS*, pages 441–447, 2011.

[YCM+19] Guomin Yang, Rongmao Chen, Yi Mu, Willy Susilo, Fuchun Guo, and Jie Li. Strongly leakage resilient authenticated key exchange, revisited. *Des. Codes Cryptogr.*, 87(12):2885–2911, 2019.

[YMSW13] Guomin Yang, Yi Mu, Willy Susilo, and Duncan S. Wong. Leakage resilient authenticated key exchange secure in the auxiliary input model. In *ISPEC*, pages 204–217, 2013.