On the Leakage-Resilient Key Exchange

Janaka Alawatugoda

Department of Computer Engineering
University of Peradeniya, Peradeniya 20400
Sri Lanka
janaka.alawatugoda@qut.edu.au

Abstract

Typically, secure channels are constructed from an authenticated key exchange (AKE) protocol, which authenticates the communicating parties based on long-term public keys and establishes secret session keys. In this paper we address the partial leakage of long-term secret keys of key exchange protocol participants due to various side-channel attacks. Security models for two-party authenticated key exchange protocols have developed over time to provide security even when the adversary learns certain secret values. This paper combines and extends the advances of security modelling for AKE protocols addressing more granular partial leakage of long-term secrets of protocol participants.

Keywords: Public Key Cryptography, Key Exchange Protocols, Leakage-Resilient Cryptography

This technical report is based on the PhD thesis of Alawatugoda [8], which combines and extends some of the previous papers on leakage-resilient key exchange [43, 4, 6, 7].

Contents

1 Introduction

As the Internet developed, more and more individuals and organizations connect their computers and private networks to the Internet. Since the Internet is a publicly available resource, the security of the information exchanged via the Internet is not guaranteed. Therefore, ensuring the security of the information becomes an important task. For many years, confidentiality, integrity and availability (known as the CIA triad) are known as the core principles of the information security [39].

Cryptography is engaged with communication systems to enforce the security of the information by establishing a *secure channel* for communication. A secure channel assures that no third party can see or modify the actual messages that are being transferred. Particularly, in the real world scenario the TLS/SSL protocol suite is used for this purpose. First, the TLS/SSL handshake protocol (key exchange protocol) exchanges a *secret session key*. Thereafter the TLS/SSL record protocol uses that secret session key to *encrypt* the messages. During the handshake protocol, both parties agree on an algorithm to encrypt data in the TLS/SSL record layer.

With the development of side-channel attacks, a necessity arises to develop cryptosystems in a leakage-resilient manner. Being one of the important cryptographic primitives, key exchange protocols were considered to be constructed in a leakage-resilient manner. Even though most of the current key exchange security models like Bellare-Rogaway (BR) model [9], Canetti-Krawczyk (CK) model [15], extended Canetti-Krawczyk (eCK) model [36] address different adversarial capabilities, they do not address the partial leakage of long-term secret parameters due to the side-channel attacks. Those do not suffice for analyzing the security of existing key exchange protocols in a leaky environment. In order to address leakage resilience of key exchange protocols, it is necessary to construct key exchange security models which allow the adversary to obtain partial leakage of secret parameters of protocol participants. The ultimate goal is to come up with a security model which addresses strong leakage features, with no additional restrictions than existing strong security models such as eCK. Then construct leakage-resilient key exchange protocols, which are proven secure in the new security model. This way it is possible to construct strong key exchange protocols, which are resilient to side-channel attacks.

This paper combines and extends results from few of the previous papers on leakage-resilient key exchange [43, 4, 6, 7], and fix some flaws in security proofs. We will discuss in detail about,

- Side-channel attacks and countermeasures.
- Various leakage models which could possibly be incorporated with security models/games to address side-channel attacks.
- Practical interpretation of various leakage security models for key exchange.
- Leakage-resilient protocol constructions and fix some flows in their security proofs.
- Compare and contrast previous security models and key exchange protocols.

1.1 Side-Channel Attacks and Countermeasures

The revolutionary idea of side-channel attacks was first introduced by Kocher [33] by presenting timing attacks on Diffie-Hellman, RSA, DSS and other implementations. Although the cryptographic schemes are designed in such a way that they are hard to break computationally, leaking information from the implemented system may give sufficient power to the adversary to break the system by recovering the secret parameters such as secret keys. There are various kinds of side channels available such as timing information, electromagnetic radiation, acoustic signals, visual or light signals and leaking information about power consumption. Since it is very difficult to fully stop the information leakage from cryptosystems, side-channel attacks become a huge threat for the security of cryptosystems [26]. Moreover, it is very useful to study different types of side-channel attacks in the literature, to understand how they extract information and proceed to extract the secrets from the targeted cryptosystem. In the following, we concentrate on timing attacks, power analysis attacks and electromagnetic emission based attacks, to understand the nature of side channel attacks.

Timing Attacks. Timing attacks enable the attacker to reveal secrets from the cryptosystem by measuring the amount of time taken for different computations. Cryptosystems take different amount of time for different operations. If the attacker can get the different timing information from the targeted cryptosystem, the attacker may be able to determine what is happening inside the system.

As mentioned before, Kocher [33] stated that by measuring the amount of time required for secret key operations, attackers might be able to find fixed Diffe-Hellman exponents, factor RSA keys and break other cryptosystems. He also stated that these kinds of attacks are computationally inexpensive and often need only a known ciphertext. Actual systems are potentially at risk whenever attackers can get accurate timing measurements, especially when the amount of time required to perform an operation is variable and dependent on a secret key or protected value. Bernstein demonstrated an attack on the implementation of Advanced Encryption Standard (AES) algorithm and showed that the standard AES algorithm is vulnerable for cache-timing attacks [12]. Even though the brute-force search takes thousands of years to attack AES, Bernstein's attack could be completed within days as long as the adversary has the power to study the cache-timing information from the targeted system.

Power Analysis Attacks. Power analysis attacks are initiated by measuring the power consumption of a cryptographic device which is used to implement the cryptosystem. Messerges et al. [40] applied this idea to attack an actual smart card, which uses Data Encryption Standard (DES). They examined the noise characteristics of the power signals and developed an approach to model the signal-to-noise ratio. They showed that the signal-to-noise ratio can be significantly improved using a multiple-bit attack. It is not difficult to attach a device which can measure the power consumption of an ATM machine and sending that information to the adversary. By analyzing the information an adversary can reveal the secret keys of smart cards used on the ATM.

Electromagnetic (EM)-Emission-based Attacks. EM-emission-based attacks are another possible type of side-channel attack. Electric circuits generate EM radiation as they operate. Attackers can use these radiation emission and analyze them to extract the secrets of the cryptosystem. Hutter et al. [27] discussed the effectiveness of EM emission based attacks on electronic passports and contact-less payment systems which use passive 13.56 MHz radio frequency identification devices. They stated that these devices can be successfully attacked with less than 1000 EM traces. Unlike power analysis attacks, EM emission based attacks can be performed remotely. Receivers can be used to catch emission radiation and more powerful receivers can get information from distant systems.

Cold Boot Attacks. A cold boot attack is a type of side-channel attack in which an attacker physically accesses a system and retrieves secret keys from a running operating system after a cold reboot (boot process in which the system starts from a powerless state) to restart the system. This attack relies on the data remanence property of DRAM and SRAM to retrieve memory contents that remain readable in the seconds to minutes after power has been removed.

1.1.1 Local versus Remote Side-Channel Attacks.

Side-channel attacks can be mounted locally (local attacks) or remotely (remote attacks) from the targeted system. Local attacks need some kind of physical access to the target system or proximity. For instance, side-channel attacks such as power analysis attacks, EM-emission based attacks, cold boot attacks need physical access to the targeted system or proximity to capture the leakage. Alternatively, remote attacks can be mounted from a long distance. For instance, the side-channel attacks such as timing attacks can be mounted from remotely, as the adversary can measure response time or ciphertext lengths from a distance system.

1.1.2 Countermeasures for Side-Channel Attacks.

Considering the countermeasures of side-channel attacks, mainly there are two approaches. One is a hardware based approach where the focus is to design hardware that minimizes the leakage of secret information. Bernstein has proposed ideas to design CPUs which provides protection against timing attacks on cache memory, as well as many ideas to mask leaking timing information by software based AES implementations [12]. Besides this there are other software based countermeasures which are mostly focusing on masking the leaking information. Alawatugoda et al. [5] have presented three methods for masking leaking timing information: injecting some randomness to the leaking cache-timing information, dedicating cache portions to fetch data from different memory portions and pre-fetching from the memory before the algorithm accesses the required memory portions and hence inject random timing information and change the cache access pattern respectively. Obviously, those types of countermeasures protect systems only against some specific attacks that are known at the moment. Those countermeasures are known as ad-hoc solutions.

Above we discussed a few known side-channel attacks and possible countermeasures against them. There may be many unknown side-channel attacks as well. Therefore, it is important to defend against both known and unknown side-channel attacks.

1.2 Leakage-Resilient Cryptography

As discussed above side-channels leak some amount of information about the secret parameters to the adversary. The basic idea of leakage-resilient cryptography is, even though some leaking information is visible to the adversary, the security of the cryptographic scheme remains. Trying to stop the leakage is nearly impossible because electronic devices have their physical limitations.

Even though a cryptographic scheme may be proven secure in a strong security model which does not address leakage attacks, it is not possible to say anything about the security of the cryptographic scheme in an environment where the adversary is capable of obtaining leakage information. In order to analyze the leakage resiliency of cryptographic schemes we need to construct security models where the adversary is given capability of obtaining leakage information.

Different leakage models have been introduced to capture side-channel attacks. They provide leakage based information to the adversary under different constraints. In order to achieve leakage resilience in a particular leakage model, the scheme should be proven secure even when the adversary is capable of accessing leakage information in that particular leakage model.

Continuous Leakage Model. In the pioneering work of Micali and Reyzin [41], a general framework was introduced to model the leakage that occurs when computation takes place on secret parameters. This framework relies on the assumption that "only computation leaks information". Further, Micali and Reyzin mentioned that leakage only occurs from the secret memory portions which are actively involved in computations, and the amount of leakage per occurrence is less than the size of the corresponding secret memory portion, hence bounded by a leakage parameter λ. The adversary is allowed to obtain leakage from an arbitrarily large number of computations, hence the overall leakage amount is *unbounded* and it can be larger than the size of the secret key. This leakage model addresses side-channel attacks such as timing attacks, power analysis attacks and EM emission based attacks, which obtain leakage of secret values whenever computations take place on them.

This model is suitable to analyze *stateful* leakage-resilient cryptographic schemes [23, 45], where at the end of each i^{th} execution round a new secret key state sk_{i+1} is computed using the current secret key state sk_i. sk_{i+1} is going to be used as the secret parameter of the next execution round $i+1$. Before the i^{th} round an attacker chooses (adaptively) a leakage function f_i and after the execution of the round, it receives $f_i(sk_i)$, under the constrain that $|f_i(sk_i)| \leq \lambda$.

Bounded Leakage Model. Inspired by "cold boot" attacks Akavia, Goldwasser and Vaikuntanathan constructed a general framework to model memory attacks [3]. The adversary can adaptively choose an efficiently computable arbitrary leakage function, f_i and send it to the leakage oracle. The leakage oracle gives $f_i(sk)$ to the adversary where sk is the secret key. The only restriction comes here is that $\Sigma|f_i(sk)| \leq \lambda$, where λ is the leakage parameter, which is smaller than the size of sk.

This model is suitable to analyze *stateless* leakage-resilient cryptographic schemes which are not using new secret states for each round.

After-the-fact Leakage. Leakage which happens after the challenge is given to the adversary is considered as after-the-fact leakage. 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 [9, 15, 36]. 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.

Earlier leakage models only consider the leakage which happens before the challenge is given (before-the-fact leakage). Hence the adversary is not allowed to obtain leakage after the challenge is given. Recent leakage models facilitate more granular leakage by allowing the adversary to issue leakage functions, and obtain leakage even after the challenge is given, either under the bounded or continuous leakage assumptions as explained above.

For leakage-resilient public-key encryption there are three properties which may be important differentiators for the different models. One is whether the model allows access to decryption of chosen ciphertexts before (CCA1) and after (CCA2) the challenge is known. The second is whether the leakage allowed to the adversary is *continuous* or *bounded*. The third is whether the leakage is allowed only before the challenge ciphertext is known or also after the fact.

In earlier models, such as that of Naor and Segev [44], it was expected that although the adversary is given access to the decryption oracle (CCA2), the adversary cannot be allowed to obtain leakage after the challenge ciphertext is given. This is because the adversary can encode the decryption algorithm and challenge ciphertext with the leakage function and by revealing a few bits of the decrypted value of the challenge ciphertext trivially win the challenge. Subsequently, Halevi and Lin [25] introduced after-the-fact leakage-resilient semantic security (CPLA2) on public-key cryptosystems, in the bounded leakage model. In their security experiment, the adversary is not allowed to access the decryption oracle. Dziembowski and Faust [22] defined an adaptively-chosen ciphertext after-the-fact leakage (CCLA2) in which the adversary is allowed to access the decryption oracle adaptively and obtain leakage information even after the challenge ciphertext is given. Furthermore, they allow continuous leakage, so the total leakage amount is unbounded.

2 Prelimineries

Here we describe several background concepts that help to understand the paper.

2.1 Computational Assumptions

2.1.1 Computational Diffie-Hellman (CDH) Assumption

Let k be the security parameter, $\mathcal{G}$ be a group generation algorithm and $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^k)$, where g is the generator and q is the order of $\mathbb{G}$. The CDH problem is to compute g^{ab} given a random instance (g, g^a, g^b) for $a, b \in \mathbb{Z}_q$.

We say that computational Diffie-Hellman assumption holds in $\mathbb{G}$ if for all PPT algorithms $\mathcal{A}$, the probability of solving the CDH problem in $\mathbb{G}$ given as,

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

is negligible for a given security parameter k.

2.1.2 Decisional Diffie-Hellman (DDH) Assumption

Let k be the security parameter, $\mathcal{G}$ be a group generation algorithm and $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^k)$, where g is the generator and q is the order of $\mathbb{G}$. Consider the following two distributions:

$$\mathcal{DH}_{\mathbb{G}} = \{(g, g^a, g^b, g^{ab}); a, b \overset{\$}{\leftarrow} \mathbb{Z}_q\}$$

and

$$\mathcal{R}_{\mathbb{G}} = \{(g, g^a, g^b, g^c); a, b, c \overset{\$}{\leftarrow} \mathbb{Z}_q\} \ .$$

It is said that DDH assumption holds in $\mathbb{G}$ if for all PPT algorithms $\mathcal{A}$, the advantage in distinguishing the two distributions $\mathcal{DH}$ and $\mathcal{R}$ given as,

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

is negligible for a given security parameter k.

2.1.3 Gap Diffie-Hellman (GDH) Assumption

Let k be the security parameter, $\mathcal{G}$ be a group generation algorithm and $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^k)$, where g is the generator and q is the order of $\mathbb{G}$. Given a random instance (g, g^a, g^b) for $a, b \in \mathbb{Z}_q$, the GDH problem is to find g^{ab} given an oracle $\mathcal{O}$ that solves the Decisional Diffie-Hellman problem in $\mathbb{G}$.

It is said that GDH assumption holds in $\mathbb{G}$ if for all PPT algorithms $\mathcal{A}$, the probability of solving the GDH problem is given as,

$$\Pr^{\text{GDH}}_{g,q}(\mathcal{A}) = \Pr\left(\mathcal{A}(\mathbb{G}, g, q, \mathcal{O}, g^a, g^b) = g^{ab}\right)$$

is negligible for a given security parameter k.

2.1.4 Oracle Diffie-Hellman (ODH) Assumption [1]

Let k be the security parameter, $\mathcal{G}$ be a group generation algorithm and $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^k)$, where g is the generator and q is the order of $\mathbb{G}$, and H be arbitrary efficiently computable function. A PPT adversary $\mathcal{R}$ is interacting with the Oracle Diffie-Hellman challenger, which is defined using the following game:

- $u, v \xleftarrow{\$} \mathbb{Z}_q$
- $Z_0 \leftarrow \mathrm{H}(g^{uv}); Z_1 \xleftarrow{\$} \{0,1\}^k$
- $b \xleftarrow{\$} \{0,1\}$
- $b' \leftarrow \mathcal{R}^{\mathcal{O}^{\mathrm{ODH}}}(g^u, g^v, Z_b)$
- $\mathcal{R}$ wins if $b' = b$

<u>$\mathcal{O}^{\mathrm{ODH}}$ **Oracle**</u>

- If $X = g^u$, return $\perp$
- Else return $\mathrm{H}(X^v)$

It is said that ODH assumption holds in $\mathbb{G}$ if for all PPT algorithms $\mathcal{R}$, the the advantage of winning the ODH challenge is negligible in k.

2.2 Cryptographic Tools

2.2.1 Key Derivation Functions

We review the definitions of key derivation functions by Krawczyk [35]. Secure and efficient key derivation functions are available in the literature, for example based on HMAC [35].

Definition 2.1 (Key Derivation Function)**.** Let k be the security parameter. A key derivation function KDF is an efficient algorithm that accepts as input four arguments: a value σ sampled from a source of keying material Σ, a length value ℓ and two additional arguments, a salt value r defined over a set of possible salt values and a context variable c, both of which are optional i.e., can be set to a null. The KDF output is a string of ℓ bits.

Definition 2.2 (Source of Key Material)**.** A source of keying material Σ is a two-valued (σ, κ) probability distribution generated by an efficient probabilistic algorithm, where σ is the secret source key material to be input to the KDF and κ is some public knowledge about σ or its distribution.

Definition 2.3 (Security of key derivation function with respect to a source of key material)**.** A key derivation function KDF is said to be secure with respect to a source of key material Σ, if no feasible attacker $\mathcal{B}$ can win the following distinguishing game with probability significantly better than 1/2. In other words the advantage: $\mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B})$ of winning the following game is negligible in k.

1. $(\sigma, \kappa) \xleftarrow{\$} \Sigma(1^k)$. (Both the probability distribution as well as the generating algorithm have been referred to Σ)
2. A salt value r is chosen at random from the set of possible salt values defined by KDF (r may be set to a constant or a null value if so defined by KDF).
3. The attacker $\mathcal{B}$ is provided with κ and r.
4. $\mathcal{B}$ chooses arbitrary value ℓ and c.
5. A bit $b \xleftarrow{\$} \{0,1\}$ is chosen at random. If $b = 0$, attacker $\mathcal{B}$ is provided with the output of $\mathrm{KDF}(\sigma, r, \ell, c)$ else $\mathcal{B}$ is given a random string of ℓ bits.
6. $\mathcal{B}$ outputs a bit $b' \leftarrow \{0,1\}$. $\mathcal{B}$ wins if $b' = b$.

2.2.2 Pseudo Random Functions

Following we review the security definition of pseudo random function according to Katz and Lindell [29].

Definition 2.4 (Pseudo Random Functions)**.** Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient, length-preserving, keyed function. F is a *pseudo random function* if for all PPT distinguishers J, there is a negligible function *negl* such that:

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

where the first probability is taken over uniform choice of $key \in \{0,1\}^k$ and the randomness of $\mathcal{J}$, and the second probability is taken over uniform choice of f_{rnd} and randomness of $\mathcal{J}$, and $\mathcal{J}$ is not given a key key.

2.3 Leakage-Resilient Primitives

2.3.1 Leakage-Resilient Storage

We review the definitions of leakage-resilient storage according to Dziembowski and Faust [22]. The idea behind their construction is to split the storage of elements into two parts using a randomized encoding function. As long as leakage is limited from each of its two parts, no adversary can learn useful information about an encoded element. The key observation of Dziembowski and Faust is then to show how such encodings can be *refreshed* in a leakage-resilient way so that the new parts can be re-used. To construct a continuous leakage-resilient primitive the relevant secrets are split, used separately, and then refreshed between any two usages.

Definition 2.5 (Dziembowski-Faust leakage-resilient storage scheme)**.** For any $m, n \in \mathbb{N}$, the storage scheme $\Lambda^{n,m}_{\mathbb{Z}^*_q} = (\text{Encode}^{n,m}_{\mathbb{Z}^*_q}, \text{Decode}^{n,m}_{\mathbb{Z}^*_q})$ efficiently stores elements $s \in (\mathbb{Z}^*_q)^m$ where:

- $\text{Encode}^{n,m}_{\mathbb{Z}^*_q}(s) : s_L \xleftarrow{\$} (\mathbb{Z}^*_q)^n \backslash \{(0^n)\}$, then $s_R \leftarrow (\mathbb{Z}^*_q)^{n \times m}$ such that $s_L \cdot s_R = s$ and outputs (s_L, s_R).
- $\text{Decode}^{n,m}_{\mathbb{Z}^*_q}(s_L, s_R)$: outputs $s_L \cdot s_R$.

In the model we expect an adversary to see the results of a leakage function applied to s_L and s_R. This may happen each time computation occurs.

Definition 2.6 ($\boldsymbol{\lambda}$-limited adversary)**.** If the amount of leakage obtained by the adversary from each of s_L and s_R is limited to $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$ bits in total respectively, the adversary is known as a $\boldsymbol{\lambda}$*-limited adversary.*

Definition 2.7 (($\boldsymbol{\lambda}_\Lambda, \epsilon_1$)-secure leakage-resilient storage scheme)**.** We say $\Lambda = (\text{Encode}, \text{Decode})$ is $(\boldsymbol{\lambda}_\Lambda, \epsilon_1)$-secure leakage-resilient, if for any $s_0, s_1 \xleftarrow{\$} \mathcal{M}$ and any $\boldsymbol{\lambda}_\Lambda$-limited adversary $\mathcal{C}$, the leakage from $\text{Encode}(s_0) = (s_{0L}, s_{0R})$ and $\text{Encode}(s_1) = (s_{1L}, s_{1R})$ are statistically ϵ_1-close. For an adversary-chosen leakage function $\mathbf{f} = (f_1, f_2)$, and a secret s such that $\text{Encode}(s) = (s_L, s_R)$, the leakage is denoted as $\big(f_1(s_L), f_2(s_R)\big)$.

Lemma 2.1 ([22])**.** *Suppose that* $m < n/20$*. Then* $\Lambda^{n,m}_{\mathbb{Z}^*_q} = (\text{Encode}^{n,m}_{\mathbb{Z}^*_q}, \text{Decode}^{n,m}_{\mathbb{Z}^*_q})$ *is* $(\boldsymbol{\lambda}, negl(n))$*-secure for some negligible function* $negl$ *and* $\boldsymbol{\lambda} = (0.3 \cdot n \log q, 0.3 \cdot n \log q)$*.*

The encoding function can be used to design different leakage resilient schemes with bounded leakage. The next step is to define how to *refresh* the encoding so that a continuous leakage is also possible to defend against.

Definition 2.8 (Refreshing of Leakage-Resilient Storage [22])**.** Let $(L', R') \leftarrow \text{Refresh}^{n,m}_{\mathbb{Z}^*_q}(L, R)$ be a refreshing protocol that works as follows:

- Input : (L, R) such that $L \in (\mathbb{Z}^*_q)^n$ and $R \in (\mathbb{Z}^*_q)^{n \times m}$.
- Refreshing R :
  1. $A \xleftarrow{\$} (\mathbb{Z}^*_q)^n \backslash \{(0^n)\}$ and $B \leftarrow$ non singular $(\mathbb{Z}^*_q)^{n \times m}$ such that $A \cdot B = (0^m)$.
  2. $M \leftarrow$ full rank $(\mathbb{Z}^*_q)^{n \times n}$ such that $L \cdot M = A$.
  3. $X \leftarrow M \cdot B$ and $R' \leftarrow R + X$.
- Refreshing L :
  1. $\tilde{A} \xleftarrow{\$} (\mathbb{Z}^*_q)^n \backslash \{(0^n)\}$ and $\tilde{B} \leftarrow$ full rank $(\mathbb{Z}^*_q)^{n \times m}$ such that $\tilde{A} \cdot \tilde{B} = (0^m)$.
  2. $\tilde{M} \leftarrow$ non-singular $(\mathbb{Z}^*_q)^{n \times n}$ such that $\tilde{M} \cdot R' = \tilde{B}$.
  3. $Y \leftarrow \tilde{A} \cdot \tilde{M}$ and $L' \leftarrow L + Y$.
- Output : (L', R')

Let $\Lambda = (\text{Encode}, \text{Decode})$ be a $(\boldsymbol{\lambda}_\Lambda, \epsilon_1)$-secure leakage-resilient storage scheme and Refresh be a refreshing protocol. We consider the following experiment Exp, which runs Refresh for ℓ rounds and lets the adversary obtain leakage in each round. For refreshing protocol Refresh, a $\boldsymbol{\lambda}_{\text{Refresh}}$-limited adversary $\mathcal{B}$, $\ell \in \mathbb{N}$ and $s \xleftarrow{\$} \mathcal{M}$, we denote the following experiment by $\text{Exp}_{(\text{Refresh},\Lambda)}(\mathcal{B}, s, \ell)$:

1. For a secret s, the initial encoding is generated as $(s_L^0, s_R^0) \leftarrow \text{Encode}(s)$.
2. For $j = 1$ to ℓ run $\mathcal{B}$ against the j^{th} round of the refreshing protocol.
3. Return whatever $\mathcal{B}$ outputs.

We require that the adversary $\mathcal{B}$ outputs a single bit $b \in \{0, 1\}$ upon performing the experiment Exp using $s \xleftarrow{\$} \{s_0, s_1\} \in \mathcal{M}$. Now we define leakage-resilient security of a refreshing protocol.

Definition 2.9 ($(\ell, \boldsymbol{\lambda}_{\text{Refresh}}, \epsilon_2)$-secure Leakage-Resilient Refreshing Protocol)**.** For a $(\boldsymbol{\lambda}_\Lambda, \epsilon_1)$-secure leakage-resilient storage scheme $\Lambda = (\text{Encode}, \text{Decode})$ with message space $\mathcal{M}$, Refresh is $(\ell, \boldsymbol{\lambda}_{\text{Refresh}}, \epsilon_2)$-secure leakage-resilient, if for every $\boldsymbol{\lambda}_{\text{Refresh}}$-limited adversary $\mathcal{B}$ and any two secrets $s_0, s_1 \in \mathcal{M}$, the statistical distance between $\text{Exp}_{(\text{Refresh},\Lambda)}(\mathcal{B}, s_0, \ell)$ and $\text{Exp}_{(\text{Refresh},\Lambda)}(\mathcal{B}, s_1, \ell)$ is bounded by ϵ_2.

Theorem 2.2 ([22])**.** *Let $m/3 \leq n, n \geq 16$ and $\ell \in \mathbb{N}$. Let n, m and $\mathbb{Z}_q^*$ be such that $\Lambda_{\mathbb{Z}_q^*}^{n,m}$ is $(\boldsymbol{\lambda}, \epsilon)$-secure leakage-resilient storage scheme (Definition 2.5 and Definition 2.7). Then the refreshing protocol $\text{Refresh}_{\mathbb{Z}_q^*}^{n,m}$ (Definition 2.8) is a $(\ell, \boldsymbol{\lambda}/2, \epsilon')$-secure leakage-resilient refreshing protocol for $\Lambda_{\mathbb{Z}_q^*}^{n,m}$ (Definition 2.9) with $\epsilon' = 2\ell p(3p^m\epsilon + mp^{-n-1})$.*

2.3.2 Adaptively Chosen Ciphertext After-the-fact Leakage Secure (CCLA2) Public-Key Cryptosystems

Dziembowski and Faust [22] constructed an adaptively chosen ciphertext after-the-fact leakage-resilient public-key cryptosystem, which is secure against continuous leakage.

Definition 2.10 (Security Against Adaptively Chosen Ciphertext After-the-fact Leakage Attacks (CCLA2))**.** This is a modification of the IND-CCA2 security notion. Let $k \in \mathbb{N}$ be the security parameter, λ be the leakage parameter and f_i be arbitrary, efficiently computable adaptive leakage functions. Let $\text{PKE} = (\text{KG}, \text{Enc}, \text{Dec})$ be a public-key encryption scheme, we define $\text{Adv}_{\text{PKE}}^{\text{CCLA2}}(\mathcal{D})$ as the advantage of any PPT adversary $\mathcal{D}$, winning the following game:

1. $(sk, pk) \xleftarrow{\$} \text{KG}(1^k)$.
2. $(m_0, m_1, state) \leftarrow \mathcal{D}^{\text{Dec}(sk,\cdot,f_i)}(pk)$ such that $|m_0| = |m_1|$
3. $b \xleftarrow{\$} \{0, 1\}$
4. $C \leftarrow \text{Enc}(pk, m_b)$
5. $b' \xleftarrow{\$} \mathcal{D}^{\text{Dec}_{\neq C}(sk,\cdot,f_i)}(C, state)$
6. Output b'. $\mathcal{D}$ wins if $b' = b$

Decryption Oracle

- $\text{Dec}(sk, c, f_i) \to (sk', m)$ where m is the corresponding plaintext of the ciphertext c and sk' is the update of the secret key sk.
- compute $f_i(sk)$ whenever $|f_i(sk)| \leq \lambda$
- Update the secret state sk to sk'
- returns $(m, f_i(sk))$ to $\mathcal{A}$

PKE is CCLA2-secure, if $\text{Adv}_{\text{PKE}}^{\text{CCLA2}}(\mathcal{D})$ is negligible in k.

2.3.3 After-the-fact Leakage-resilient Semantically Secure (CPLA2) Public-Key Cryptosystems

Splitting the secret key into arbitrarily number of parts is known as the *split-state* model. The leakage is allowed from the splits of the secret key independently. The secret key is split into an arbitrary number $\tilde{n}$ parts such that $s = (s_1, \ldots, s_{\tilde{n}})$. The tuple leakage function $\mathbf{f} = (f_{1j}, \ldots, f_{\tilde{n}j})$ is an adversary chosen efficiently computable adaptive tuple leakage function, which consists of $\tilde{n}$ arbitrary leakage functions, and j indicates the j^{th} leakage occurrence. Each leakage function f_{ij} leaks $f_{ij}(s_i)$ from each s_i split of the secret key individually.

Halevi and Lin [25] constructed a generic after-the-fact leakage-resilient semantically secure public-key cryptosystem, which is secure against bounded leakage in the split-state model. It can be instantiated with the DDH-based leakage-resilient public-key cryptosystem of Naor and Segev [44].

Definition 2.11 (After-the-fact Leakage-resilient Semantic Security (CPLA2))**.** This is a modification of the IND-CPA security notion. Let $k \in \mathbb{N}$ be the security parameter and $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_{pre}, \boldsymbol{\lambda}_{post})$ be a tuple of two vectors, where $\boldsymbol{\lambda}_{pre} = (\lambda_{pre_1}, \ldots, \lambda_{pre_{\tilde{n}}})$ is the leakage bound vector before the challenge ciphertext is issued, and $\boldsymbol{\lambda}_{post} = (\lambda_{post_1}, \ldots, \lambda_{post_{\tilde{n}}})$ is the leakage bound vector after the challenge ciphertext is issued. Let $\mathbf{f}$ be the leakage function as described above. Let $\mathrm{PKE} = (\mathrm{KG}, \mathrm{Enc}, \mathrm{Dec})$ be a public-key cryptosystem, we define $\mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D})$ as the advantage of any probabilistic polynomial time (PPT) adversary $\mathcal{D}$, winning the following game:

1. $(s, p) \xleftarrow{\$} \mathrm{KeyGen}(1^k)$.
2. $(m_0, m_1) \leftarrow \mathcal{D}^{\mathrm{Leak}(\mathbf{f})}(p)$ such that $|m_0| = |m_1|$, for $i = 1, \ldots, \tilde{n}$, $\mathrm{Leak}(\mathbf{f})$ returns $f_{ij}(s_i)$ if $\sum_j |f_{ij}(s_i)| \leq \lambda_{pre_i}$.
3. $b \xleftarrow{\$} \{0, 1\}$.
4. $C \xleftarrow{\$} \mathrm{Enc}(pk, m_b)$.
5. $b' \leftarrow \mathcal{D}^{\mathrm{Leak}(\mathbf{f})}(p, C)$ for $i = 1$ to $i = \tilde{n}$, $\mathrm{Leak}(\mathbf{f})$ returns $f_{ij}(s_i)$ if $\sum_j |f_{ij}(s_i)| \leq \lambda_{post_i}$.
6. $\mathcal{D}$ wins if $b' = b$.

PKE is CPLA2-secure, if $\mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D})$ is negligible in k.

2.3.4 Unforgeability Against Chosen Message Leakage Secure UFCMLA Signature Schemes

Katz et al. [30] constructed an unforgeability against chosen message leakage attacks secure signature scheme in bounded leakage model. It contains signing and verification operations based on NIZK proofs.

Definition 2.12 (Unforgeability Against Chosen Message Leakage Attacks (UFCMLA))**.** This is a modification of the UFCMA security notion. Let $k \in \mathbb{N}$ be the security parameter, λ be the leakage parameter and f_i be arbitrary efficiently computable adaptive leakage functions. Let $\mathrm{SIG} = (\mathrm{KG}, \mathrm{Sign}, \mathrm{Vfy})$ be a signature scheme, we define $\mathrm{Adv}_{\mathrm{SIG}}^{\mathrm{UFCMLA}}(\mathcal{E})$ as the advantage of any PPT adversary $\mathcal{E}$, winning the following game:

1. $r \xleftarrow{\$} \{0, 1\}^*$
2. $(sk, vk) \xleftarrow{\$} \mathrm{KG}(1^k, r); st \leftarrow r$
3. $(m^*, \sigma^*) \leftarrow \mathcal{E}^{\mathcal{O}^{\mathrm{UFCMLA}}(\cdot,\cdot)}(vk)$
4. If $\mathrm{Vfy}(vk, m^*, \sigma^*) =$ "true" and m^* is not been previously signed, then $\mathcal{E}$ wins.

$\mathcal{O}^{\mathrm{UFCMLA}}(m, f_i)$

- $r_i \xleftarrow{\$} \{0, 1\}^*$
- $\sigma \xleftarrow{\$} \mathrm{Sign}(sk, m, r_i)$
- $st \leftarrow st \cup r_i$
- compute $\gamma \leftarrow f_i(st)$ whenever $\sum_i |f_i(st)| \leq \lambda$
- Return (σ, γ) to $\mathcal{E}$

SIG is UFCMLA-secure, if $\mathrm{Adv}_{\mathrm{SIG}}^{\mathrm{UFCMLA}}(\mathcal{E})$ is negligible in k.

If we think about the split-state setting for the UFCMLA security notion, above notion is same as the case where $\tilde{n} = 1$, because the signing key sk has not been split.

3 Key Exchange Security Models

In 1976, Diffie and Hellman introduced a key exchange primitive [19], which enables two parties to exchange a secret key (session key) by communicating over a public channel. Users *Alice* and *Bob* agree on a group $\mathbb{G}$ of prime order q and on a generator g of this group. This is done before executing the rest of the protocol, and g and q are assumed to be public. Alice picks a random integer $a \xleftarrow{\$} \mathbb{Z}_q$ and computes $A \leftarrow g^a$ and sends it to Bob. Then Bob picks a random integer $b \xleftarrow{\$} \mathbb{Z}_q$ and computes $B \leftarrow g^b$ and sends it to Alice. After that, Alice computes $B^a = (g^b)^a = s \in \mathbb{G}$ and Bob computes $A^b = (g^a)^b = s \in \mathbb{G}$. Thus, both Alice and Bob end up with the same value $s \in \mathbb{G}$. An eavesdropper who watches this communication can see A and B values, but should be unable to determine the values of s (assuming CDH holds).

Many key exchange protocols have been created based on the Diffie-Hellman key exchange primitive [13, 20, 28]. In these key exchange protocols, different types of keys may be used to compute session keys: long-term secret keys are the static secrets belong to the protocol participants which are often used to add authentication to the session key, ephemeral keys are the session specific secrets belong to protocol participants which are used to add freshness to the session key. There are number of known attacks against key exchange protocols:

Implicit Key Authentication. If a protocol provides a guarantee that no party apart from the protocol participants can compute the session key, that key exchange protocol is said to provide *implicit key authentication.* If a key exchange protocol provides implicit key authentication that protocol is said to be an *authenticated key exchange* protocol.

Key Confirmation. If a key exchange protocol provides a guarantee that each party is assured that all other participants possess the session key, that key exchange protocol is said to provide *key confirmation.*

Known Key Security. The knowledge of a session key should not enable the adversary to learn the session keys in other sessions; all session keys should not be depended on the session keys of other sessions.

Unknown Key Share (UKS) Security. It should not happen that a party A shares a session key with some party B, but believing that it is sharing the session key with some one else C. That means public keys and identities of the parties should be certified and confirmed or incorporated into protocol execution.

Key Compromise Impersonation (KCI) Resistance. Knowing the long-term secret key of a party A should not enable the adversary to impersonate other honest parties to A.

Forward Secrecy. An adversary who knows the long-term secret keys of parties should not be able to compute the session keys of past sessions between those two particular parties.

In order to analyze the security of key exchange protocols, a formal methodology is needed. Therefore, key exchange security models have been created. A security model is a formal security statement of certain security features. Generally, security models are designed to reflect real world adversarial capabilities, addressing the known attacks (mentioned earlier). At the same time, it is natural to design security models with theoretical adversaries which have more capabilities than real world adversaries, because that way it is possible to address more powerful attacks which may exist in the future. Following is the general structure of a security model.

- **Definition of the algorithm:** Inputs, outputs and abstract description of the algorithm.
- **Adversary capabilities:** How the adversary can interact with the system and which information the adversary is allowed to learn, usually in the form of queries. As a usual practice the adversary is made as strong as possible by giving more capabilities to the adversary.
- **Security game:** The way in which the adversary perform queries.
- **Security goal:** The requirement for the adversary to win the security game.

In a security model, there is a predefined list of queries that an adversary can perform (adversary capabilities). Those queries reveal information such as session keys, ephemeral keys, long-term secret keys etc. Even after performing the queries, within the constraints defined in the security model, if the adversary's advantage of distinguishing the real session key from a random key chosen from the same distribution is negligible, the protocol is said to be secure in the particular security model. The session in which the adversary tries to distinguish the real session key from a random key, is known as the *target session.*

The Bellare-Rogaway models (BR93 [9], BR95 [11]), the Canetti-Krawczyk (CK) model [15], and the extended Canetti-Krawczyk (eCK) model [36] are a few such security models, and protocol designers use them to analyze the security of key exchange protocols. We briefly look at a few of the earlier models such as the BR models and the CK model, and then discuss the eCK model in detail, as it is a more recent and widely used security model.

The Bellare-Rogaway Models. The BR93 model [9] is the first formal security model for key exchange protocols, in which the adversary is defined as a probabilistic polynomial time machine that initiates and controls all the communications between the protocol participants. Moreover, the adversary is allowed to *reveal* the session keys, learn the complete internal state of the protocol participants by *corrupting* the protocol participants, and overwrite the long-term secret key of the corrupted participant with any value of her choice. The Bellare-Rogaway models defined a notion for partner sessions. The definition of partner sessions is used in the security definition, to restrict the adversary's *reveal* and *corrupt* operations to sessions that are not partners of the target session. In the BR models security is defined on the notions of partner sessions and the indistinguishability of session keys.

The BR93 model defines the partnership using the notion of matching conversations, where a conversation is defined to be the sequence of messages sent and received by a particular protocol instance (session) of a protocol participant. In the BR95 model, the partnership is defined using the notion of a *partner function*, which uses the transcript containing the record of all messages sent and received by a particular protocol instance (session) of a protocol participant, to determine the partner sessions. However, such partner definitions can easily go wrong [16].

Canetti-Krawczyk Model. In the CK model, the adversary is allowed to initiate and control all the communications between the protocol participants, reveal the session keys, learn the complete internal state of the protocol participants by corrupting the protocol participants and overwrite the long-term secret key of the corrupted participant with any value of her choice, as in the BR models, and additionally reveal the internal state of the protocol participants (but not the long-term secret keys). The partner sessions are defined using the notion of matching session identifiers (SIDs) and partner identifiers (PIDs). There is no formal definition of how SIDs should be defined and the values of SIDs are not specified in the CK model. It is assumed that the SIDs are known by the protocol participants before the protocol begins. Such an assumption may not be practical, because it requires some form of communication between the protocol participants before the protocol begins. The definition of partner sessions is used in the security definition, to restrict the adversary's session key reveal, session state reveal and corrupt operations to sessions that are not partners of the target session. In the CK model security is defined on the notions of partner sessions and the indistinguishability of session keys, which is similar to the security definition of the BR models.

3.1 Extended Canetti-Krawczyk Model (eCK) [36]

Parties and Long-term Keys. Let $\mathcal{U} = \{U_1, \ldots, U_{N_P}\}$ be a set of N_P parties. Each party U_i where $i \in [1, N_P]$ has a pair of long-term public and secret keys, (pk_{U_i}, sk_{U_i}). Each party U_i owns at most N_S number of protocol sessions.

Sessions. Each party may run multiple instances of the protocol concurrently or sequentially; we use the term *principal* to refer a party involved in a protocol instance, and the term *session* to identify a protocol instance at a principal. The notation $\Pi^s_{U,V}$ represents the s^{th} session at the owner principal U, with intended partner principal V. The principal which sends the first protocol message of a session is the *initiator* of the session, and the principal which responds to the first protocol message is the *responder* of the session. A session $\Pi^s_{U,V}$ enters an *accepted* state when it computes a session key. Note that a session may terminate without ever entering into the accepted state. The information of whether a session has terminated with or without acceptance is public.

Partnering. Legitimate execution of a key exchange protocol between two principals U and V makes two partnering sessions owned by U and V respectively. Two sessions $\Pi^s_{U,V}$ 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 sessions compute identical session keys.

Adversarial Powers. The adversary $\mathcal{A}$ is a probabilistic polynomial time algorithm in the security parameter k, that has the control over the whole network. $\mathcal{A}$ interacts with set of sessions which represent protocol instances. $\mathcal{A}$ can adaptively ask following queries.

- `Send` (U, V, s, m) query- This query allows $\mathcal{A}$ to run the protocol. It sends the message m to the session $\prod_{U,V}^{s}$ as coming from the session $\prod_{V,U}^{s'}$. $\prod_{U,V}^{s}$ will return to $\mathcal{A}$ the next message according to the protocol conversation so far or decision on whether to accept or reject the session. $\mathcal{A}$ can also use this query to initiate a new protocol instance with blank m. This query captures capabilities of active adversary, who can initiate sessions and modify or delay protocol messages.
- `Session-Key reveal` (U, V, s) query- If a session $\prod_{U,V}^{s}$ has accepted and holds a session key, $\mathcal{A}$ gets the session key of $\prod_{U,V}^{s}$. A session can only accept a session key once. This query captures the known key attacks.
- `Ephemeral-Key reveal` (U, V, s) query- Gives all the ephemeral keys (per session randomness) of the session $\prod_{U,V}^{s}$ to $\mathcal{A}$.
- `Corrupt` (U) query- $\mathcal{A}$ gets all the long-term secrets of the principal U. But this query does not reveal any session keys to $\mathcal{A}$. This query captures the key compromise impersonation (KCI) attacks, unknown key share (UKS) attacks and forward secrecy.
- `Test` (U, s) query- Once a session $\prod_{U,V}^{s}$ has accepted and holds a session key, $\mathcal{A}$ can attempt to distinguish it from a random key. When $\mathcal{A}$ asks the `Test` query, the session $\prod_{U,V}^{s}$ first chooses a random bit $b \in \{0, 1\}$ and if $b = 1$, the actual session key is returned to $\mathcal{A}$, otherwise a random session key is chosen uniformly at random from the same session key distribution, and is returned to $\mathcal{A}$. This query is only allowed to be asked once.

Freshness. A session $\prod_{U,V}^{s}$ is said to be *fresh* if and only if all of the following hold:

1. The session $\prod_{U,V}^{s}$ and its partner (if it exists), $\prod_{V,U}^{s'}$ have not been asked the `Session- Key reveal` query.
2. If partner $\prod_{V,U}^{s'}$ exists none of the following combinations have been asked:
   (a) `Corrupt`(U) and `Ephemeral-Key reveal`(U, V, s)
   (b) `Corrupt`(V) and `Ephemeral-Key reveal`(V, U, s')
3. If partner $\prod_{V,U}^{s'}$ does not exist none of the following combinations have been asked
   (a) `Corrupt`(V)
   (b) `Corrupt`(U) and `Ephemeral-Key reveal`(U, V, s)

Security Game.

- **Stage 0:** The challenger generates the keys by using the security parameter k.
- **Stage 1:** $\mathcal{A}$ is executed and may ask any of `Send`, `Session-Key reveal`, `Ephemeral-Key reveal` and `Corrupt` queries to any session at will.
- **Stage 2:** At some point $\mathcal{A}$ chooses a fresh session and asks the `Test` query.
- **Stage 3:** $\mathcal{A}$ may continue asking `Send`, `Session-Key reveal`, `Ephemeral- Key reveal` and `Corrupt` queries. The only condition is that $\mathcal{A}$ cannot violate the freshness of the test session.
- **Stage 4:** At some point $\mathcal{A}$ outputs the bit $b' \in \{0, 1\}$ which is its guess of the value b on the test session. $\mathcal{A}$ wins if $b' = b$.

Definition of Security. Let $\text{Succ}_{\mathcal{A}}$ be the event that the adversary $\mathcal{A}$ wins the eCK game.

Definition 3.1. A protocol (π) is said to be secure in the eCK model if there is no PPT adversary $\mathcal{A}$ who can win the eCK game with non-negligible advantage in the security parameter k. The advantage of an adversary $\mathcal{A}$ is defined as $\text{Adv}_{\pi}^{\text{eCK}}(\mathcal{A}) = |2\text{Pr}(\text{Succ}_{\mathcal{A}}) - 1|$.

In the BR models and the CK model, the adversary is not allowed to learn the long-term secret key of the owner of the target session, before it expires. Therefore, those models are not capable of addressing the key compromise impersonation attacks, whereas the eCK model allows the adversary to learn the long-term secret key of the owner of the target session. Therefore the eCK model addresses the KCI attacks. Moreover, the BR models and the CK model do not allow the adversary to reveal the session states or ephemeral keys of the target session or its partner session. Therefore, those models are not capable of addressing the ephemeral key leakage attacks, whereas the eCK model allows the adversary to reveal both of the ephemeral keys of the target session, as long as the owner and the partner principals to the target session are not corrupted. Therefore, the eCK model addresses the ephemeral key reveal attacks. In the CK model, after the target session has expired, the adversary is allowed to learn the long-term secret keys of the protocol participants of the target session, regardless of whether the adversary actively interfered with the target session, whereas the eCK model only allows the adversary to learn the long-term secret keys of both protocol participants of the target session when the adversary has not actively interfered with the target session. Therefore, the CK model addresses the perfect forward secrecy, while the eCK model only addresses the weak perfect forward secrecy.

The essential difference between the eCK [36] and the CK model is that the eCK model substitutes the adversarial operation of revealing the complete internal state of the protocol participants with a new operation to reveal the ephemeral secret key, which reveals the randomness used in the specified session. The important point to note is that the ephemeral key does not include the session state that has been computed using the long-term secret of the protocol participant. This is not the case in the CK model, in which the adversary is allowed access to all the inputs (including the randomness, but excluding the long-term secret itself) and the results of all the computations done as part of a session. Among the other models the eCK model is clearly defined to capture most of the demanding security features of key exchange protocols, and thus widely used as a strong security model to analyze the security of key exchange protocols. We chose the eCK model for detailed discussion, because the eCK model is used as the base of the leakage security models which are discussed in this paper.

3.2 eCK-Secure Key Exchange Protocols

The initial effort of constructing the eCK-secure key exchange protocols is combining the long-term secret key and the ephemeral secret key using a random oracle function [10] to obtain a pseudo ephemeral value. This trick is first introduced by LaMacchia et al. [36] in their protocol named NAXOS, and now it is widely known as the NAXOS trick. A "psuedo" ephemeral key $\widetilde{esk}$ is computed as the random oracle function of the long-term key lsk and the actual ephemeral key esk: $\widetilde{esk} \leftarrow \text{H}(esk, lsk)$. The value $\widetilde{esk}$ is never stored, and thus in the eCK model the adversary must learn both esk and lsk in order to be able to compute $\widetilde{esk}$. Note however, that in the NAXOS protocol, the initiator must compute $\widetilde{esk} = \text{H}(esk, lsk)$ twice: once when sending its Diffie–Hellman ephemeral public key $g^{\widetilde{esk}}$, and once when computing the Diffie–Hellman shared secrets from the received values. This is to avoid storing a single value that, when compromised, can be used to compute the session key. There are some key exchange protocols created using the NAXOS trick [36, 46].

NAXOS Protocol. The NAXOS protocol [36] was originally published with the eCK model, and it is proven secure in the eCK model. Table 1 shows the NAXOS protocol. Let $\mathbb{G}$ be a group of prime order q with generator g. Here a and $\bar{A}$ are the long-term secret and public keys of A, while x and X are the ephemeral secret and public keys of A. The important feature we can see is that the long-term secret key and the ephemeral secret key are combined using a hash function H_1. The trick of combining the long-term secret key and the ephemeral secret key is known as "NAXOS trick". In this protocol both hash functions H_1 and H are modeled as random oracles.

CMQV Protocol. Using the MQV [37], the HMQV [34] and the NAXOS protocols Ustaoglu [46] created a new key exchange protocol called "Combined MQV" (CMQV). Table 2 shows the CMQV protocol. Let $\mathbb{G}$ be a group of prime order q with generator g. Here a and $\bar{A}$ are the long-term secret and

A		B
$a \xleftarrow{\$} \mathbb{Z}_q$		$b \xleftarrow{\$} \mathbb{Z}_q$
$\bar{A} \leftarrow g^a$		$\bar{B} \leftarrow g^b$
$x \xleftarrow{\$} \mathbb{Z}_q$		$y \xleftarrow{\$} \mathbb{Z}_q$
$\bar{x} \leftarrow \mathrm{H}_1(x, a)$		$\bar{y} \leftarrow \mathrm{H}_1(y, b)$
$X = g^{\bar{x}}$	$\xrightarrow{X}$	$Y = g^{\bar{y}}$
	$\xleftarrow{Y}$	
$\kappa = \mathrm{H}(Y^a, \bar{B}^{\mathrm{H}_1(x,a)}, Y^{\mathrm{H}_1(x,a)}, A, B)$		$\kappa = \mathrm{H}(\bar{A}^{\mathrm{H}_1(y,b)}, X^b, X^{\mathrm{H}_1(y,b)}, A, B)$
	κ is the session key	

Table 1: NAXOS protocol

public keys of A, while x and X are the ephemeral secret and public keys of A. The main advantages of the CMQV protocol are, better efficiency than the NAXOS protocol and proven-security in the eCK model. The CMQV protocol uses the NAXOS trick to achieve the eCK security. As for the NAXOS protocol, the CMQV security proof also uses the random oracle assumption, because hash functions H_1, H_2 and H are modeled as random oracles.

A		B
$a \xleftarrow{\$} \mathbb{Z}_q$		$b \xleftarrow{\$} \mathbb{Z}_q$
$\bar{A} \leftarrow g^a$		$\bar{B} \leftarrow g^b$
$x \xleftarrow{\$} \mathbb{Z}_q$		$y \xleftarrow{\$} \mathbb{Z}_q$
$\bar{x} = \mathrm{H}_1(x, a)$		$\bar{y} = \mathrm{H}_1(y, b)$
$X = g^{\bar{x}}$		$Y = g^{\bar{y}}$
	$\xrightarrow{(B,A,X)}$	
	$\xleftarrow{(A,B,X,Y)}$	
$E = \mathrm{H}_2(Y, A, B)$		$D = \mathrm{H}_2(X, A, B)$
$\sigma = (Y\bar{B}^E)^{\bar{x}+Da}$		$\sigma = (X\bar{A}^D)^{\bar{y}+Eb}$
$\kappa = \mathrm{H}(\sigma, X, Y, A, B)$		$\kappa = \mathrm{H}(\sigma, X, Y, A, B)$
	κ is the session key	

Table 2: Two-pass CMQV protocol

eCK **Security without NAXOS Trick.** Recently, some researchers worked on constructing eCK-secure key exchange protocols without NAXOS trick [42, 47, 32, 7]. The motivation for such research can be explained as follows: The eCK model addresses the leakage of the ephemeral secret key. It is unnatural to assume that the ephemeral secret key is leaked, while the exponent of the ephemeral public key (eg:- the pseudo ephemeral value in the NAXOS protocol) remains safe, without leaking. Therefore, it seems that there is an unnatural and indirect assumption of a leakage-free exponentiation computation or leakage-free random source, in the eCK-security proof of the NAXOS-style key exchange protocols. Therefore, eliminating the NAXOS trick and still preserving the eCK security would be more realistic.

Table 3 shows the key exchange protocol P1 [7], which is a Diffie-Hellman-type, NAXOS-free and eCK-secure key exchange protocol. The reason to present this protocol is that, it is used as a building block for a leakage-resilient key exchange protocol which is presented in section 5. Let $\mathbb{G}$ be a group of prime order q with generator g. Here a and A are the long-term secret and public keys of Alice, while x and X are the ephemeral secret and public keys of Alice. After exchanging the public values both principals compute a Diffie-Hellman-type shared secret, and then compute the session key using a random oracle H.

In order to compute the session key, the protocol P1 combines four components ($Z_1 \leftarrow B^a$, $Z_3 \leftarrow Y^a$, $Z_4 \leftarrow Y^x$, $Z_2 \leftarrow B^x$) using the random oracle function H. These four components cannot be recovered by the attacker without both the ephemeral and long-term secret keys of at least one protocol principal.

3.3 eCK-type Leakage Security Models for Key Exchange: Moriyama-Okamoto Model

Earlier key exchange security models, such as the Bellare–Rogaway [9], Canetti–Krawczyk [15], and extended Canetti–Krawczyk (eCK) [36] models, aim to capture security against an adversary who can

Alice (Initiator)		Bob (Responder)
$a \xleftarrow{\$} \mathbb{Z}_q^*, A \leftarrow g^a$		$b \xleftarrow{\$} \mathbb{Z}_q^*, B \leftarrow g^b$
$x \xleftarrow{\$} \mathbb{Z}_q^*, X \leftarrow g^x$	$\xrightarrow{Alice, X}$	$y \xleftarrow{\$} \mathbb{Z}_q^*, Y \leftarrow g^y$
	$\xleftarrow{Bob, Y}$	
$Z_1 \leftarrow B^a, Z_2 \leftarrow B^x$		$Z_1' \leftarrow A^b, Z_2' \leftarrow X^b$
$Z_3 \leftarrow Y^a, Z_4 \leftarrow Y^x$		$Z_3' \leftarrow A^y, Z_4' \leftarrow X^y$
$K \leftarrow \mathrm{H}(Z_1, Z_2, Z_3, Z_4, Alice, X, Bob, Y)$		$K \leftarrow \mathrm{H}(Z_1', Z_2', Z_3', Z_4', Alice, X, Bob, Y)$
	K is the session key	

Table 3: Concrete construction of Protocol P1

fully compromise some, but not all secret keys. For example, in the eCK model, a session key should be secure even if the adversary has compromised either the long-term or ephemeral key at the client, and either the long-term or ephemeral key at the server, but not all of the values at one party. This is not a very granular form of leakage, and thus is not fully suitable for modelling side-channel attacks.

Moriyama and Okamoto have presented a suitable security model to capture leakage, and a proven secure protocol in that model [43]. The security model introduced by Moriyama and Okamoto is based on the eCK model. The Moriyama-Okamoto model allows the adversary to obtain leakage of a long-term secret key sk, of a protocol principal U, by issuing adaptively-chosen arbitrary leakage functions f_i and specifying the identity of the protocol principal U. Hence, in addition to the adversarial powers in the eCK, model Moriyama-Okamoto model provides:

- `StaticKeyLeakage(`f, U`)` query: From this query the adversary obtains $f_i(sk)$ where sk denotes the long-term secret key of the principal U.

Further, it is important to study the constraints in the Moriyama-Okamoto model. We can identify two main limitations in Moriyama-Okamoto model.

1. The adversary is allowed to obtain leakage from long-term secrets of protocol participants; this leakage is bounded by some parameter λ.
2. The model does not allow the adversary to obtain leakage after the test session is activated.

Since the Moriyama-Okamoto model is restricted to the bounded leakage, it can only address the side-channel attacks such as cold boot attacks (to some extent), which happen due to the leakage of bounded amount of information from the secret memory. The Moriyama-Okamoto model allows the adversary to reveal either the long-term secret key or the ephemeral secret key of the target session (same as in the eCK model). Additionally, the the Moriyama-Okamoto model allows bounded amount of leakage of the long-term secret key with the ephemeral secret key reveal from the target session. Thus the Moriyama-Okamoto model addresses the cold boot attacks to some extent by addressing following situations: the attacker reveals either (i) the long-term secret key, (ii) the ephemeral secret key, or (iii) the ephemeral secret key and part of the long-term secret key of the target session.

Differently, side-channel attacks which happen due to the continuous leakage of secret keys, such as timing attacks or power analysis attacks can not be modelled using the Moriyama-Okamoto model. Because the Moriyama-Okamoto model does not address continuous leakage of the long-term secret keys, which happens whenever computations use the long-term secret keys. Further, restricting the leakage to occur only before the target session is activated is not a natural restriction. Therefore, although the Moriyama-Okamoto model addresses side-channel attacks for some extent, there is some gap between the Moriyama-Okamoto model and real world side-channel attacks.

Those two limitations are considered when defining the freshness of a session in the Moriyam-Okamoto model.

3.3.1 Moriyama-Okamoto Freshness

A session $\prod_{U,V}^s$ is λ-leakage fresh if the following conditions hold:

- $\prod_{U,V}^s$ is a fresh session in the sense of the eCK model.
- Before the adversary activates the session $\prod_{U,V}^s$, the total amount of leakage that the adversary obtains from each partner principal of the session $\prod_{U,V}^s$: U and V, is bounded by the leakage parameter λ.

- After the session $\prod_{U,V}^{s}$ is activated, no leakage is allowed from the partner principals of the session $\prod_{U,V}^{s}$.

Apart from the freshness condition, partnering and the security game are the same as in the eCK model.

4 Continuous After-the-fact Leakage in Restricted-eCK Model

As mentioned above, there are two central limitations in the Moriyama–Okamoto model as we pointed out in section 3.3. The first restriction is troublesome because, in practice, ongoing executions of a protocol may reveal a small amount of leakage each time, and we would like to provide security against this "continuous" leakage. The latter restriction is problematic because we would like to provide security of one session, even if some leakage happens in subsequent sessions.

The above limitations thus restrict the adversary in the Moriyama–Okamoto model. In this section, we present a model and a protocol of Alawatugoda et al. [4], that allows the adversary to adaptively obtain an unbounded amount of total *continuous leakage*, albeit with the restriction that the amount of leakage obtained in each leakage request is limited; this addresses the first restriction of previous models. Secondly, the adversary is allowed to obtain leakage after the test session is activated, addressing the second restriction of the Moriyama–Okamoto model. The model we present in this section enforces restrictions on the freshness condition, in addition to the eCK-style freshness condition defined in the Moriyama–Okamoto model. We explain the additional restrictions on the freshness condition further in this section.

4.1 Continuous After-the-fact Leakage (CAFL) Model [4]

In the CAFL model, the adversary is allowed to adaptively obtain partial leakage on the long-term secret keys even after the test session is activated, as well as reveal session keys, long-term keys, and ephemeral keys.

4.1.1 Modelling Leakage

This key exchange security model considers *continuous leakage of the long-term secret keys* of protocol principals, because long-term secret keys are not one-time secrets, but they last for multiple protocol sessions. Leakage of long-term secret key from one session affects the security of another session which uses the same long-term secret key. Considering side-channel attacks which can be mounted against key exchange protocols, the most realistic way to obtain the leakage information of long-term secret keys is from the protocol computations which use long-term secret keys. Hence, following the premise "only computation leaks information" [41], the leakage is modelled to occur where computation takes place using secret keys. By issuing a `Send` query, the adversary will get a protocol message which is computed according to the normal protocol computations. Therefore, the instance of a `Send` query would be the appropriate instance to address the leakage which occurs due to a computation which uses a long-term secret key. Thus, sending an adversary-chosen leakage function, f, with the `Send` query would reflect the premise "only computation leaks information". The leakage function f is an efficiently computable, adaptive leakage function.

Further, the amount of leakage of a secret key is bounded by a leakage parameter λ, per computation. The adversary is allowed to obtain leakage from many computations continuously. Hence, the overall leakage amount is unbounded.

4.1.2 Adversarial Powers

The adversary (a probabilistic algorithm) controls all interaction and communication between parties. In particular, the adversary initiates sessions at parties and delivers protocol messages; it can create, change, delete, or reorder messages. The adversary can also compromise certain short-term and long-term secrets. Notably, whenever the party performs an operation using its long-term key, the adversary obtains some leakage information about the long-term key.

The following query allows the adversary $\mathcal{A}$ to run the protocol, modelling normal communication.

- `Send`(U, V, s, m, f) query: The session $\Pi_{U,V}^{s}$, computes the next protocol message according to the protocol specification on receipt of m, and sends it to the adversary $\mathcal{A}$, along with the leakage

$f(sk_U)$ as described in Section 4.1.1. $\mathcal{A}$ can also use this query to activate a new protocol instance as an initiator with blank m.

The following 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 session $\Pi^s_{U,V}$, if the session $\Pi^s_{U,V}$ is in the accepted state.
- `EphemeralKeyReveal`(U, V, s) query: $\mathcal{A}$ is given the ephemeral keys of the session $\Pi^s_{U,V}$.
- `Corrupt`(U) query: $\mathcal{A}$ is given the long-term secrets of the principal U. This query does not reveal any session keys or ephemeral keys to $\mathcal{A}$.

Remark 1 (`Corrupt` query vs Leakage queries)**.** By issuing a `Corrupt` query, the adversary gets the party's entire long-term secret key. Separately, by issuing leakage queries (using leakage function f embedded with the `Send` query) the adversary gets λ-bounded amount of leakage information about the long-term secret key. It may seem paradoxical to consider `Corrupt` and Leakage queries at the same time. But there are good reasons to consider both.

- A *non-leakage* version of CAFL model (`Send` query without f) addresses KCI attacks, because the adversary is allowed to corrupt the owner of the test session before the activation of the test session. In the CAFL model, we allow the adversary to obtain leakage from the partner of the test session, in addition to allowing the adversary to corrupt the owner of the test session.
- A *non-leakage* version of CAFL model (`Send` query without f) addresses partial weak forward secrecy, because the adversary is allowed to corrupt either of the protocol principals, but not both, after the test session is activated. In the CAFL model, we allow the adversary to obtain leakage from the uncorrupted principal, in addition to allowing the adversary to corrupt one of the protocol principals.

Hence, the CAFL model allows the adversary to obtain more information than a *non-leakage* version of CAFL model.

4.1.3 Defining Security

In this section we give formal definitions for partner sessions, freshness of a session and security in the CAFL model.

Definition 4.1 (Partner sessions in CAFL model)**.** As in the eCK model, two sessions $\Pi^s_{U,V}$ and $\Pi^{s'}_{U',V'}$ are said to be partners if:

1. $\Pi^s_{U,V}$ and $\Pi^{s'}_{U',V'}$ have computed session keys and
2. Sent messages from $\Pi^s_{U,V}$ = Received messages to $\Pi^{s'}_{U',V'}$ and
3. Sent messages from $\Pi^{s'}_{U',V'}$ = Received messages to $\Pi^s_{U,V}$ and
4. $U' = V$ and $V' = U$ and
5. If U is the initiator then V is the responder, or vice versa.

A protocol is said to be *correct* if two partner sessions compute identical session keys in the presence of a passive adversary.

We now define what it means for a session to be $\lambda -$ CAFL-*fresh* in the CAFL model.

Definition 4.2 ($\lambda -$ CAFL-freshness)**.** Let λ be the leakage bound per occurrence. A session $\Pi^s_{U,V}$ is said to be $\lambda -$ CAFL-*fresh* if and only if:

1. The session $\Pi^s_{U,V}$ 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 `Corrupt`(V).
   (b) `Corrupt`(U) and `EphemeralKeyReveal`(U, V, s).

   (c) `Corrupt`(V) and `EphemeralKeyReveal`(V, U, s').

   (d) `EphemeralKeyReveal`(U, V, s) 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) `EphemeralKeyReveal`(U, V, s).

4. For each `Send`$(U, \cdot, \cdot, \cdot, f)$ query, the output of f is at most λ bits.

5. For each `Send`$(V, \cdot, \cdot, \cdot, f)$ query, the output of f is at most λ bits.

Limitations of $\lambda -$ CAFL-freshness. When the adversary asks `Corrupt` and `EphemeralKeyReveal` queries, there are two `Corrupt`–`EphemeralKeyReveal` query combinations which trivially expose the session key of a session, in a scenario that a partner to that particular session exists.

1. `Corrupt`(U) and `EphemeralKeyReveal`(U, V, s).

2. `Corrupt`(V) and `EphemeralKeyReveal`(V, U, s').

As in the other models we have compared with [36, 43] CAFL does not allow above combinations in the freshness condition, as they trivially expose the session key of sessions $\Pi^s_{U,V}$ and $\Pi^{s'}_{V,U}$. Differently, in the other models we have compared with, there are four `Corrupt`–`EphemeralKeyReveal` query combinations which do not trivially expose the session key a session, in a scenario that a partner to that particular session exists.

1. `Corrupt`(U) and `Corrupt`(V).

2. `Corrupt`(U) and `EphemeralKeyReveal`(V, U, s).

3. `Corrupt`(V) and `EphemeralKeyReveal`(U, V, s').

4. `EphemeralKeyReveal`(V, U, s) and `EphemeralKeyReveal`(U, V, s').

All the models we consider [36, 43] allow above combinations in the freshness condition, whereas the CAFL model does not allow the query combinations 1 and 4 in the freshness condition.

When the adversary asks `EphemeralKeyReveal` and `Corrupt` queries, there are two query combinations which trivially expose the session key of a session, in a scenario that a partner to that particular session does not exist.

1. `Corrupt`(V).

2. `Corrupt`(U) and `EphemeralKeyReveal`(U, V, s).

As in the other models we have compared with [36, 43] the CAFL does not allow above combinations in the freshness condition, as they trivially expose the session key of sessions $\Pi^s_{U,V}$ and $\Pi^{s'}_{V,U}$. Weakening that condition, the CAFL model does not allow following two query combinations in the freshness condition, when a partner to the test session does not exist.

1. `Corrupt`(V).

2. `EphemeralKeyReveal`(U, V, s). (instead of `EphemeralKeyReveal`(U, V, s) and `Corrupt`(U) as in other models)

Thus, the freshness of a non-leakage variant of the CAFL model (without conditions 4 and 5) is weaker than the eCK-freshness definition, because of the restriction enforced in the conditions (2)-a and (2)-d. Differently, the $\lambda -$ CAFL-freshness allows partial leakage of the long-term secret key of a protocol principal, even when the partner principal is corrupted or `EphemeralKeyReveal` query is asked to the partner session. In some sense that is stronger than the eCK-freshness definition, because according to the eCK-freshness, once `EphemeralKeyReveal` query have been asked to a session, revealing the long-term secret key of the partner is not allowed. Hence, although the freshness of a non-leakage variant of the CAFL model is weaker than the eCK-freshness in some sense, $\lambda -$ CAFL-freshness achieved an improvement over eCK-freshness by means of partial leakage of long-term secrets.

We explain why the additional restrictions are introduced (restriction enforced in the conditions (2)-a and (2)-d) to the freshness condition of the CAFL model, more than in the freshness condition of the eCK model as follows: Their aim was to construct a simple leakage-resilient two-pass key exchange protocol, using a leakage-resilient public-key encryption scheme, in which each of the principals randomly chooses its ephemeral secret key, encrypts it with the public key of the intended partner principal, and sends the encrypted message to the intended partner principal. Defining eCK-style freshness makes it impossible to prove the security of the simple two-pass key exchange protocol, because corrupting both principals to the target session or revealing the ephemeral key from both sessions to the target session will trivially expose the session key. Therefore, additional restrictions are enforced to the $\lambda -$ CAFL-freshness condition, but allow the partial leakage of long-term secret keys, as the aim is to model the side-channel attacks.

Security of a key exchange protocol in the CAFL model is defined using the a security game (similar to the security game in the eCK model), which is played by a probabilistic polynomial time adversary $\mathcal{A}$ against the protocol challenger. $\mathrm{Succ}_{\mathcal{A}}$ is the event that $\mathcal{A}$ wins the security game. The security is defined as follows:

Definition 4.3 ($\lambda-$CAFL-security)**.** A protocol π is said to be $\lambda-$CAFL-secure if there is no probabilistic polynomial time algorithm $\mathcal{A}$ that can win the security game with non-negligible advantage. The advantage of an adversary $\mathcal{A}$ is defined as $\mathrm{Adv}_{\pi}^{\lambda-\mathrm{CAFL}}(\mathcal{A}) = |2\Pr(\mathrm{Succ}_{\mathcal{A}}) - 1|$.

4.1.4 Practical Interpretation of Security of CAFL Model

We review the relationship between the CAFL 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 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 and the Moriyama-Okamoto model.
- **Cold boot attacks:** The CAFL model allows the adversary to reveal either the long-term secret key (`Corrupt` query) or the ephemeral secret key (`EphemeralKeyReveal` query) of the target session (same as in the eCK model and the Moriyama-Okamoto model). Thus these queries address the cold boot attacks to some extent, where the cold boot attacks reveal either (i) the long-term secret key or (ii) the ephemeral secret key of the target session. Note that the Moriyama-Okamoto model addresses the cold boot attacks by additionally covering the situation, where the attacker reveals (iii) the ephemeral secret key and part of the long-term secret key of the target session. Thus, the Moriyama-Okamoto model is more suitable to model cold boot attacks.
- **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. In the previous security models, this property is addressed by introducing the *ephemeral-key reveal, session-state reveal* and *corrupt* queries.
- **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. In the previous security models, this property is addressed by introducing the *ephemeral-key reveal query* or the *session-state reveal* query.
- **Known key attacks:** `SessionKeyReveal` query covers the attacks which can be mounted by knowing past session keys. In the previous security models, this property is addressed by introducing the *session key reveal* query.

- **Key compromise impersonation attacks:** $\lambda -$ CAFL-freshness allows the adversary to corrupt the owner of the test session before the activation of the test session. Hence, the CAFL model security protects against the key compromise impersonation attacks. In the eCK model and the Moriyama-Okamoto model, this property is addressed by introducing the *long-term key reveal* query to the owner of the target session, before the session is completed. Earlier models such as the BR models and the CK model do not allow the adversary to reveal the long-term secret key of the owner of the target session before it is expired, and hence do not address this property.

- **Partial weak forward secrecy:** $\lambda -$ CAFL-freshness allows the adversary to corrupt either of the protocol principals, after the test session is activated. Hence, the CAFL model addresses partial weak forward secrecy. The eCK model and the Moriyama-Okamoto model allow the adversary to reveal the long-term secret keys of both protocol principals of the target session after the target session is activated, as long as the adversary is passive. Hence they address weak forward secrecy. The CK model allows the adversary to reveal the long-term secret keys of both protocol principals of the target session, after the session is expired but regardless of whether the adversary is passive or active. Therefore, the CK model address perfect forward secrecy.

4.2 Constructing CAFL-secure Key Exchange Protocols

Table 4 shows the generic construction of protocol $\pi 1$, which is CAFL-secure. The protocol $\pi 1$ is a key agreement protocol, in which each of the principals randomly chooses its ephemeral secret key, encrypts it with the public key of the intended partner principal, and sends the encrypted message to the intended partner principal. After exchanging the ephemeral secrets both principals compute the session key with ephemeral secrets, identities of the two principals and the protocol message sequence, using a pseudo random function. Updating the secret keys of protocol principals is an essential ingredient in achieving CAFL security. For this generic protocol construction, the underlying public-key encryption scheme is chosen to be a continuous leakage-resilient public-key encryption scheme, which updates the secret key after each decryption operation. This public-key encryption scheme is used to achieve the continuous leakage resiliency of the key exchange protocol.

The generic CAFL-secure protocol construction that is initially presented in Alawatugoda et al. [4], is vulnerable against active adversary. Following we explain the reason: In that protocol construction, the inputs to the key derivation function KDF contain the two ephemeral values, r_A and r_B, and the identities of the initiator and the responder, A and B, respectively: $\text{KDF}(r_A||r_B, \perp, k, A||B)$. Assume that the target session is in A, if the adversary corrupts A, decrypts the protocol message from C_B, then re-encrypt the corresponding plaintext again using pk_A, that makes a different plaintext C'_B, due to probabilistic encryption. Then if the adversary sends C'_B to A, instead of C_B, and executes the rest of the protocol, it results that A's session and B's session are not matching, because the message C_B computed by B is different from the message C'_B received by A, but compute the same session key. Thus, the adversary can issue `SessionKeyReveal` query to the session at B and thus trivially learn the session key of the target session. Cremers [18] showed that such attacks can be avoided by using the session identifier in the key derivation step together with other shared secrets. In this paper we re-design the protocol accordingly to ensure that mismatching sessions do not compute same session keys. Thus, the session key is derived using a pseudo random function (two calls to the PRF) as $K \leftarrow \text{PRF}(r_A, A||C_A||B||C_B) \oplus \text{PRF}(r_B, A||C_A||B||C_B)$, such that it contains the session identifier $A||C_A||B||C_B$ as an input to the pseudo random function.

Here we use a multiple-call pseudo random function PRF, instead of a key derivation function KDF, to ensure that the adversary chosen r_A xor r_B, can be used in the session key derivation, when the presence of an active adversary. Since the input $\sigma = r_A||r_B$ to the KDF should be uniformly random in the security definition of the KDF, using adversary chosen r_A xor r_B in the σ of KDF input is not allowed.

4.2.1 Protocol Construction

In Table 4, we show the construction of protocol $\pi 1$ with the fixture to the mentioned problem. KG, Enc and Dec are the key generation, encryption and decryption algorithms of the underlying adaptively chosen ciphertext after-the-fact leakage (CCLA2) secure public-key cryptosystem PKE (Section 2.3.2). PRF is a pseudo random function (Section 2.2.2) which generates the session key of length k.

A (Initiator)		B (Responder)
	Initial Setup	
$sk_A, pk_A \leftarrow \mathrm{KG}(1^k)$		$sk_B, pk_B \leftarrow \mathrm{KG}(1^k)$
	Protocol Execution	
$r_A \leftarrow \{0,1\}^k$		$r_B \leftarrow \{0,1\}^k$
$C_A \leftarrow \mathrm{Enc}(pk_B, r_A)$	$\xrightarrow{A,C_A}$	$(sk'_B, r_A) \leftarrow \mathrm{Dec}(sk_B, C_A)$
		$sk_B \leftarrow sk'_B$
$(sk'_A, r_B) \leftarrow \mathrm{Dec}(sk_A, C_B)$	$\xleftarrow{B,C_B}$	$C_B \leftarrow \mathrm{Enc}(pk_A, r_B)$
$sk_A \leftarrow sk'_A$		
$K \leftarrow \mathrm{PRF}(r_A, A\|\|C_A\|\|B\|\|C_B)$		$K \leftarrow \mathrm{PRF}(r_A, A\|\|C_A\|\|B\|\|C_B)$
$\oplus\mathrm{PRF}(r_B, A\|\|C_A\|\|B\|\|C_B)$		$\oplus\mathrm{PRF}(r_B, A\|\|C_A\|\|B\|\|C_B)$
	K is the session key	

Table 4: Generic CAFL-secure protocol construction: Protocol $\pi 1$

4.2.2 Security of the Protocol $\pi 1$ in the CAFL Model

Theorem 4.1. *The protocol $\pi 1$ is $\lambda -$ CAFL-secure, whenever the underlying public-key cryptosystem* PKE *is* CCLA2*-secure and* PRF *is a pseudo random function.*

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 PPT adversary against the key exchange protocol $\pi 1$. Then the advantage of $\mathcal{A}$ against the CAFL*-security of the protocol $\pi 1$, $\mathrm{Adv}_{\pi 1}^{\lambda-\mathrm{CAFL}}$ is:*

$$\mathrm{Adv}_{\pi 1}^{\lambda-\mathrm{CAFL}}(\mathcal{A}) \leq N_P^2 N_s^2 \big(\mathrm{Adv}_{PKE}^{\mathrm{CCLA2}}(\mathcal{D}) + \mathrm{Adv}_{PRF}(\mathcal{B})\big).$$

where $\mathcal{B}, \mathcal{D}$ are efficient algorithms constructed using the adversary $\mathcal{A}$, against the underlying pseudo random function, PRF, and the public-key cryptosystem, PKE, *respectively.*

The detailed proof of this theorem is in Appendix A.

4.2.3 Leakage Tolerance of the CAFL-secure Protocol $\pi 1$

In the created protocol, a principal simply encrypts a randomly-chosen ephemeral key using a CCLA2-secure public key encryption scheme, and sends it to the partner principal. Therefore, the leakage tolerance is exactly same as the leakage tolerance of the underlying CCLA2-secure public key encryption scheme.

Dziembowski and Faust. [22] constructed a CCLA2-secure public-key cryptosystem, where the secret key $sk = (x_1, x_2) \in (\mathbb{Z}_q^*)^2$ is split into two parts ℓ_{sk}, r_{sk} such that $\ell_{sk} \overset{\$}{\leftarrow} (\mathbb{Z}_q^*)^n$ at random and $r_{sk} \leftarrow (\mathbb{Z}_q^*)^{n\times 2}$ holding $\ell_{sk} \cdot r_{sk} = sk$, where n is the statistical security parameter. They proved their public-key cryptosystem is CCLA2-secure for $\lambda = 0.15 \cdot n \cdot \log q - 1$. So if we consider $n = 20$ and $\log(q-1)$ to be 1024, we can allow $\lambda = 3072$ bits of leakage, from each split per occurrence. Considering only the most expensive computations, the computation cost of Enc and Dec is 5 exponentiations for each.

5 Bounded/Continuous After-the-fact Leakage eCK Model

The continuous after-the-fact leakage key exchange security model (CAFL) mentioned in the section 4 enforces more restrictions to the freshness definition, more than in the eCK model [36] or Moriyama-Okamoto model [43]: it does not allow to reveal the ephemeral keys of both principals as to corrupt both protocol principals of the target session etc. So there is a necessity to accommodate a reasonable security model which addresses more granular leakage and at the same time does not enforce more restrictions than currently existing key exchange security models.

In this section, we present a *generic leakage-security model* for key exchange protocols, which can be instantiated as a *bounded* leakage variant as well as a *continuous* leakage variant [6]. In the bounded leakage variant, the total amount of leakage is bounded, whereas in the continuous leakage variant, a protocol execution may reveal a small amount of leakage each time. Further, the adversary is allowed to obtain the leakage even after the session key is established for the session in which the adversary tries to distinguish the real session key from a random session key. The leakage functions are arbitrary polynomial time functions with output length restrictions.

5.1 After-the-fact Leakage-eCK ((·)AFL-eCK) Model [6]

The generic after-the-fact leakage eCK ((·)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 both instantiations of the generic (·)AFL-eCK model the partnering definition and the adversarial powers are same. The freshness conditions differ according to the leakage allowed. So it is possible to define the partnering and adversarial powers in the generic (·)AFL-eCK model and define the freshness separately in each BAFL-eCK and CAFL-eCK models.

5.1.1 Modelling Leakage

Considering side-channel attacks which can be mounted against key exchange protocols, the most realistic way is to obtain the leakage information of secret keys from the protocol computations which use secret keys for computations. Following the premise "only computation leaks information", the leakage is modelled in a place where a computation takes place on secret keys. After issuing a `Send` query, the adversary will get a protocol message which is computed according to the normal protocol computations. So sending an adversary-chosen adaptive leakage function with the `Send` query reflects the premise "only computation leaks information".

A tuple of $\tilde{n}$ adaptively chosen efficiently computable leakage functions $\mathbf{f} = (f_{1j}, f_{2j}, \ldots, f_{\tilde{n}j})$ are introduced; the size $\tilde{n}$ of the tuple is *protocol-specific*, and j indicates the j^{th} leakage occurrence. A key exchange protocol may use more than one cryptographic primitive and each primitive uses a distinct secret key or secret state (in signature schemes). Hence, it is needed to address the leakage of secret keys or secret states from each of those primitives. Also, some cryptographic primitives which have been used to construct a key exchange protocol may be stateful cryptographic primitives. The execution of a stateful cryptographic primitive is split into a number of sequential stages and each of these stages uses one part of the secret key. The tuple of leakage functions $\mathbf{f} = (f_{1j}, f_{2j}, \ldots, f_{\tilde{n}j})$ leaks information from the secret key of each of the underlying primitives or each split of the secret keys at occurrence j. There exists a leakage parameter $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{\tilde{n}})$ where each λ_i bounds the leakage for the corresponding primitive as key split.

5.1.2 Adversarial Powers

The adversary $\mathcal{A}$ is a probabilistic polynomial time (PPT) algorithm that controls the whole network. $\mathcal{A}$ interacts with a set of sessions which represent protocol instances. The following query allows the adversary $\mathcal{A}$ to run the protocol.

- $\texttt{Send}(U, V, s, m, \mathbf{f})$ query: The session $\Pi^s_{U,V}$, 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.

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.

- $\texttt{SessionKeyReveal}(U, V, s)$ query: $\mathcal{A}$ is given the session key of the session $\Pi^s_{U,V}$.
- $\texttt{EphemeralKeyReveal}(U, V, s)$ query: $\mathcal{A}$ is given the ephemeral keys (per-session randomness) of the session $\Pi^s_{U,V}$.
- $\texttt{Corrupt}(U)$ query: $\mathcal{A}$ is given the long-term secrets of the principal U. This query does not reveal any session keys or ephemeral keys to $\mathcal{A}$.

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

- $\texttt{Test}(U, s)$ query: When $\mathcal{A}$ asks the `Test` query, the challenger first chooses a random bit $b \overset{\$}{\leftarrow} \{0, 1\}$ 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.

Remark 2 (`Corrupt` query vs Leakage queries). By issuing a `Corrupt` query, the adversary gets the party's entire long-term secret key. Separately, by issuing leakage queries (using a tuple leakage function $\mathbf{f}$ embedded with the `Send` query) the adversary gets respectively $\boldsymbol{\lambda}$-bounded leakage information about the long-term secret key(s). It may seem paradoxical to consider `Corrupt` and Leakage queries at the same time. But there is a good reason to consider both.

The eCK model addresses KCI attacks, because the adversary is allowed to corrupt the owner of the test session before the activation of the test session. In the generic $(\cdot)$AFL-eCK model, we allow the adversary to obtain leakage from the partner of the test session, in addition to allowing the adversary to corrupt the owner of the test session.

Hence, the generic $(\cdot)$AFL-eCK model allows the adversary to obtain more information than the eCK model. Moreover, none of the existing security models such as BR, CK, CK_{HMQV}, eCK allow a `Send` query with a tuple leakage function $\mathbf{f}$. Hence, we can see that $(\cdot)$AFL-eCK allows the adversary to obtain leakage information which none of the existing security models allow.

5.1.3 Bounded After-the-fact Leakage-eCK (BAFL-eCK) Model

In the BAFL-eCK model the total amount of leakage of each secret key of the underlying cryptographic primitives or each split of the secret key of the underlying stateful cryptographic primitives are bounded by leakage parameters. The leakage parameters are *primitive-specific*.

If the total leakage bound of the i^{th} cryptographic primitive (or the total leakage bound of the i^{th} state of the stateful cryptographic primitive) is λ_i and the leakage function f_{ij} outputs leakage bits of the secret key of the i^{th} cryptographic primitive (or leakage bits of the i^{th} split of the secret key), then for leakage resilience of i^{th} cryptographic primitive (or the stateful cryptographic primitive), we need that $\sum_j |f_{ij}(s_i)| \leq \lambda_i$.

Definition 5.1 ($\boldsymbol{\lambda}$ – BAFL-eCK-freshness)**.** Let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{\tilde{n}})$ be a vector of $\tilde{n}$ elements (same size as $\mathbf{f}$ in `Send` query). A session $\Pi^s_{U,V}$ is said to be $\boldsymbol{\lambda}$ – BAFL-eCK-fresh if and only if:

1. The session $\Pi^s_{U,V}$ 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 all `Send`$(U, \cdot, \cdot, \cdot, \mathbf{f})$ queries, $\sum_j |f_{ij}(sk_{U_i})| \leq \lambda_i$.
5. For all `Send`$(V, \cdot, \cdot, \cdot, \mathbf{f})$ queries, $\sum_j |f_{ij}(sk_{V_i})| \leq \lambda_i$.

5.1.4 Continuous After-the-fact Leakage-eCK (CAFL-eCK) Model

In the CAFL-eCK model, continuous leakage of each secret key of the underlying cryptographic primitives or each split of the secret key of the underlying stateful cryptographic primitives is allowed. The only restriction is that the amount of leakage per occurrence is bounded by leakage parameters. The leakage parameters are *primitive-specific*.

If the leakage bound of the i^{th} cryptographic primitive is λ_i per leakage occurrence and the leakage function f_{ij} outputs leakage bits of the secret key of the i^{th} cryptographic primitive, then for leakage resilience of i^{th} cryptographic primitive we need that $|f_{ij}(sk_i)| \leq \lambda_i$, per leakage occurrence. If the leakage bound of the i^{th} state of the stateful cryptographic primitive is λ_i per leakage occurrence and the leakage function f_{ij} outputs leakage bits of the i^{th} split of the secret key, then for leakage resilience of the stateful cryptographic primitive we need that $|f_{ij}(sk_i)| \leq \lambda_i$, per leakage occurrence.

Definition 5.2 ($\boldsymbol{\lambda}$ – CAFL-eCK-freshness)**.** Let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{\tilde{n}})$ be a vector of $\tilde{n}$ elements (same size as $\mathbf{f}$ in `Send` query). A session $\Pi^s_{U,V}$ is said to be $\boldsymbol{\lambda}$ – CAFL-eCK-fresh if and only if: Conditions (1)-(3) of Definition 5.1 hold, and

4. For each $\texttt{Send}(U, \cdot, \cdot, \cdot, \mathbf{f})$ query, size of the output of $|f_{ij}(sk_{U\,i})| \leq \lambda_i$.
5. For each $\texttt{Send}(V, \cdot, \cdot, \cdot, \mathbf{f})$ queries, size of the output of $|f_{ij}(sk_{V\,i})| \leq \lambda_i$.

5.1.5 Defining Security

In this section we give formal definitions for partner sessions and security in the $(\cdot)$AFL-eCK model.

Definition 5.3 (Partner sessions in generic $(\cdot)$AFL-eCK model)**.** Two sessions $\Pi^s_{U,V}$ 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 sessions compute and accept identical session keys.

Security of a key exchange protocol in the $\boldsymbol{\lambda}$ – BAFL-eCK model is defined using the a security game (similar to the security game in the eCK model). If we consider $\boldsymbol{\lambda}$ – BAFL-eCK-freshness, the security game is BAFL-eCK, otherwise if we consider $\boldsymbol{\lambda}$ – CAFL-eCK-freshness, it is CAFL-eCK security game.

$\mathrm{Succ}_{\mathcal{A}}$ is the event that the adversary $\mathcal{A}$ wins the security game. The security is defined as follows:

Definition 5.4 ($\boldsymbol{\lambda}$ – $(\cdot)$AFL-eCK-security)**.** A protocol π is said to be $\boldsymbol{\lambda}$ – $(\cdot)$AFL-eCK-secure if there is no PPT algorithm $\mathcal{A}$ that can win the $\boldsymbol{\lambda}$ – $(\cdot)$AFL-eCK security game with non-negligible advantage. The advantage of an adversary $\mathcal{A}$ is defined as $\mathrm{Adv}^{\boldsymbol{\lambda}-(\cdot)\text{AFL-eCK}}_{\pi}(\mathcal{A}) = |2\Pr(\mathrm{Succ}_{\mathcal{A}}) - 1|$.

5.1.6 Practical Interpretation of Security of AFL-eCK Model

The generic $(\cdot)$AFL-eCK model addresses the real world attack scenarios which were discussed in section 4.1.4, with the following differences.

- **Cold boot attacks:** The $(\cdot)$AFL-eCK model allows the adversary to reveal either the long-term secret key (`Corrupt` query) or the ephemeral secret key (`EphemeralKeyReveal` query) of the target session. The bounded leakage instantiation of the $(\cdot)$AFL-eCK model, BAFL-eCK model, allows bounded amount of leakage of the long-term secret key with the ephemeral key reveal. Thus these queries address the cold boot attacks to some extent, where the cold boot attacks reveal either (i) the long-term secret key, (ii) the ephemeral secret key, or (iii) the ephemeral secret key and part of the long-term secret key of a protocol principal, which is same as in the Moriyama-Okamoto model. The improvement of the BAFL-eCK model is that it allows the adversary to obtain the partial leakage of long-term secret key even after the test session is established, which is not allowed in the Moriyama-Okamoto model.

- **Weak forward secrecy:** $(\cdot)$AFL-eCK-freshness allows the adversary to corrupt both of the protocol principals of the target session, after the test session is activated, as long as the adversary is passive. Hence, the $(\cdot)$AFL-eCK model addresses weak forward secrecy, as for the eCK model and the Moriyama-Okamoto model.

- **eCK security:** The generic $(\cdot)$AFL-eCK model is a leakage-resilient version of the eCK model [36], hence, the generic $(\cdot)$AFL-eCK model captures all possible attacks from ephemeral and long-term key compromises. More precisely, in sessions where the adversary does not modify the communication between parties (passive sessions), the adversary is allowed to reveal both ephemeral secrets, both 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 key of the other party.

The main reason to introduce a generic security model, $(\cdot)$AFL-eCK model, and then present two instantiations (BAFL-eCK model and CAFL-eCK model) is to offer more flexibility to construct leakage-resilient key exchange protocols. The generic $(\cdot)$AFL-eCK model gives a reasonable security framework for key exchange protocols capturing a wide range of practical attacks including side-channel attacks. The only difference between the two instantiations is the leakage allowance (bounded or continuous). If we need to implement a key exchange protocol which is resilient to cold boot attacks we use the BAFL-eCK model as the security framework, whereas if we need to implement a key exchange protocol which is secure against continuous-leakage side-channel attacks such as timing, power analysis and EM radiation, we use the CAFL-eCK model as the security framework.

5.2 Generic Construction of $(\cdot)$AFL-eCK-secure Key Exchange Protocol

The motivation of LaMacchia et al. [36] in designing the eCK model was that an adversary should have to compromise both the long-term and ephemeral secret keys of a party in order to recover the session key. In their NAXOS protocol, the main way this is accomplished is using what is now called the NAXOS trick.

Leakage-Resilient NAXOS Trick [6]. Moving to the leakage-resilient setting requires rethinking the NAXOS trick. In the model "only computation leaks information", we must consider leakage at any place the long-term secret key is used. Thus, some kind of *leakage-resilient NAXOS trick* is needed. As noted above, the initiator must not store the pseudo-ephemeral value, $\widetilde{esk}$, and instead must apply the NAXOS trick twice for each session. The hash function H is replaced with a new leakage-resilient NAXOS trick to compute the pseudo-ephemeral value. The requirement is, given the long-term secret key and a particular ephemeral key, the NAXOS trick should always compute the same pseudo-ephemeral value, such that without knowing both the long-term and ephemeral keys the adversary is unable to compute the pseudo-ephemeral value. Moreover, the NAXOS trick computation should be resilient to the leakage of the long-term secret key, which happens even after the test session is activated.

A leakage-resilient NAXOS trick is achieved by using the *decryption* function of a CPLA2-secure public-key cryptosystem [25]. Since decryption is deterministic, given the long-term secret key and a randomly chosen ciphertext, it will output the corresponding plaintext. So one can randomly choose an ephemeral key and use it as the ciphertext to the decryption function, and obtain the corresponding plaintext (output of the decryption function) as the pseudo-ephemeral value. Without knowing both the long-term and ephemeral keys, it is infeasible to compute the pesudo-ephemeral value. Thus, a leakage-resilient NAXOS trick can be achieved and the pseudo-ephemeral value can be computed. Further, *bounded* or *continuous* leakage-resilient key exchange protocol can be constructed, if the underlying public-key cryptosystem is bounded or continuous leakage-resilient.

Pair Generation Indistinguishability [6]. Using a decryption algorithm of a CPLA2-secure public-key cryptosystem does not work for our requirement unless the public-key cryptosystem has a special property: any randomly chosen ciphertext should be decrypted without rejection. A randomly chosen ciphertext can be rejected with a significant probability if NIZK proofs have been used for CPLA2-secure public-key cryptosystems. In NIZK proofs, the party which creates a ciphertext should provide a proof of knowledge of the plaintext, and the party which decrypts the ciphertext first verifies the proof, then only if the proof is correct it decrypts the ciphertext, otherwise rejects. Use of a CPLA2-secure public-key cryptosystem without the special property would allow the adversary to break the protocol with a significant probability, whenever a randomly chosen ciphertext is rejected. The special property is defined as *pair generation indistinguishability.*

Definition 5.5 (Pair Generation Indistinguishability)**.** Let PKE $= (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ be a public-key cryptosystem. For $(p, s) \xleftarrow{\$} \mathrm{KeyGen}(1^k)$, let $D_1^{(p,s)}$, $D_2^{(p,s)}$ be two distributions such that $D_1^{(p,s)} = \{(m, c) : m \xleftarrow{\$} M, c \xleftarrow{\$} \mathrm{Enc}(p, m)\}$ and $D_2^{(p,s)} = \{(m, c) : c \xleftarrow{\$} \mathrm{C}, m \leftarrow \mathrm{Dec}(s, c)\}$ where M is the message space and C is the ciphertext space. For $\epsilon \geq 0$, the public-key cryptosystem PKE is *ϵ-pair-generation-indistinguishable* ($\epsilon -$ PG-IND) if for all $(p, s) \xleftarrow{\$} \mathrm{KeyGen}(1^k)$, $\mathrm{SD}(D_1^{(p,s)}, D_2^{(p,s)}) \leq \epsilon$.

Recall that the statistical distance, SD, between two distributions X and Y over a domain U is defined as $\mathrm{SD}(X, Y) = \frac{1}{2} \sum_{u \in U} \big| \Pr[X = u] - \Pr[Y = u] \big|$.

The notion of pair generation indistinguishability shares some resemblance with the pseudorandom decapsulation notion introduced by Abdalla et al. [2], where the notion was needed for the construction of verifiable random functions from identity-based key encapsulation schemes. They presented a methodology to construct verifiable random functions (VRFs) from a class of identity based key encapsulation mechanisms (IB-KEM) that is called VRF suitable. An IB-KEM is VRF suitable if it provides an unique decryption (i.e. given a ciphertext C produced with respect to an identity ID, all the secret keys corresponding to identity ID$'$, decrypt to the same value, even if ID $=$ ID$'$) and it satisfies an additional property that is called *pseudorandom decapsulation*. Pseudorandom decapsulation means that if one decrypts a ciphertext C, produced with respect to an identity ID, using the decryption key corresponding to any other identity ID$'$ the resulting value looks random to a polynomially bounded observer. Both the pair generation indistinguisbability and the pseudorandom decapsulation notions are similar, except that the pseudorandom decapsulation is in the identity-based setting whereas the pair generation indistinguishability is in the public key setting.

We show a $0-$PG-IND public-key cryptosystem available in the literature. Naor and Segev [44] described the framework of a hash proof system [17] as a key-encapsulation mechanism using the notion of Kiltz et al. [31]. Let K be the symmetric key space, C be the valid ciphertext space and M be the message space. Both K and C are the same size and elements of M are μ-bit strings. The leakage-resilient public-key cryptosystem of Naor and Segev encrypts an arbitrary message $m \xleftarrow{\$} \mathrm{M}$ as (Ψ, c, seed), where $c \xleftarrow{\$} \mathrm{C}$ with the corresponding witness ω (of the fact that c is indeed a valid ciphertext from C), $\mathit{seed} \xleftarrow{\$} \{0,1\}^t$ is a random seed and $\Psi = \mathit{Ext}(\mathit{Pub}(p, c, \omega), \mathit{seed}) \oplus m$. $\mathit{Ext} : \mathrm{K} \times \{0,1\}^t \rightarrow \{0,1\}^\mu$ is a public average-case strong extractor function [21], p is the public key and Pub is the deterministic public evaluation function of the underlying key-encapsulation mechanism. Pub receives as input a public key p, a valid ciphertext $c \in \mathrm{C}$ and the corresponding witness ω, and outputs an encapsulated key in K. Whenever a random (Ψ, c, seed) is sampled, the decryption, $m \leftarrow \Psi \oplus \mathit{Ext}(\mathit{Priv}(s, c), \mathit{seed})$ corresponds to a random $m \in \mathrm{M}$. Priv is a private evaluation algorithm of the underlying key-encapsulation mechanism, receives as input the secret key s (of the public key p) and a valid ciphertext c, and outputs an encapsulated key in K. Thus, the leakage-resilient public-key cryptosystem of Naor and Segev is $0-$PG-IND. The generic CPLA2-secure public-key cryptosystem of Halevi and Lin [25] can be instantiated using the leakage-resilient public-key cryptosystem of Naor and Segev Hence, instantiation of the generic CPLA2-secure public-key cryptosystem of Halevi and Lin is also $0-$PG-IND.

Authenticating Protocol Messages. After computing the pseudo-ephemeral value by the NAXOS trick, a principal computes a Diffie-Hellman exponentiation and sends it to the other protocol principal. If that value is sent alone, the protocol is not secure because there is no authentication for the protocol messages, and hence an attacker can simply replace the original protocol message with its own value. In order to prevent this, it is necessary to provide authenticity to the protocol messages. There are unforgeable against chosen message leakage (UFCMLA) secure signature schemes available in the literature [30, 24, 38, 14], which can be used to sign the protocol messages and provide authenticity. Further, the key exchange protocol is *bounded* or *continuous* leakage-resilient, if the underlying signature scheme is bounded or continuous leakage-resilient.

5.2.1 Weakening the $(\cdot)$AFL-eCK Model

When we consider the `EphemeralKeyReveal` query in the $(\cdot)$AFL-eCK model, it allows the adversary to learn the randomness used in the session, including the randomness used in signing. Full leakage of the randomness is not allowed in leakage-resilient signature schemes. We notice that Alawatugoda et al. [6] missed this fact. In order to use available leakage-resilient signature schemes in the protocol instantiation, we will assume that the `EphemeralKeyReveal` query will not reveal the randomness used to compute the signature. Therefore, in this generic protocol construction, the security model we consider is slightly weaker than the actual $(\cdot)$AFL-eCK model, as it does not reveal the randomness used for signing with the `EphemeralKeyReveal` query. We name the weaker model as w$(\cdot)$AFL-eCK model.

We found that the generic protocol construction of Alawatugoda et al. [6], is vulnerable against active adversary. Following we explain the reason: In that protocol construction, the inputs to the key derivation function contain the Diffie-Hellman shared secret, and the identities of the initiator and the responder, A and B, respectively: KDF$(g^{\widetilde{r_A}\widetilde{r_B}}, \perp, k, A||B)$. Assume that the target session is in B, if the adversary corrupts B and gets the signing key of B, re-sign the protocol message X_B computing the new signature σ'_B, that makes a different signature from σ_B, due to probabilistic signing algorithm. Then if the adversary sends B, A, X_B, σ'_B to B, instead of B, A, X_B, σ_B, and executes the rest of the protocol, it results that A's session and B's session are not matching, because the message B, A, X_B, σ_B computed by B is different from the message B, A, X_B, σ'_B received by A, but compute the same session key. Thus, the adversary can issue `SessionKeyReveal` query to the session at A and thus trivially learn the session key of the target session. Cremers [18] showed that such attacks can be avoided by using the session identifier in the key derivation step together with other shared secrets. Thus, in this paper we re-design the protocol accordingly to ensure that mismatching sessions do not compute same session keys. Thus, the session key is derived using a pseudo random function as $K \leftarrow \mathrm{PRF}(ms, A||X_A||\sigma_A||B||X_B||\sigma_B)$, such that it contains the session identifier $A||X_A||\sigma_A||B||X_B||\sigma_B$ as an input to the pseudo random function. The shared secret ms is derived as $ms \leftarrow \mathrm{KDF}(X_B^{\widetilde{r_A}}, \perp, k, \perp)$. The σ input to the KDF is the Diffie-Hellman shared secret value $X_B^{\widetilde{r_A}}$, and it is a uniformly random element of the group, and therefore we can use KDF in the simulation without any problem.

5.2.2 Protocol Construction

In Table 5, we show the construction of protocol π with the fixture to the mentioned problem. KeyGen, Enc and Dec are the key generation, encryption and decryption algorithms of the underlying CPLA2-secure (Section 2.3.3), $\epsilon -$ PG-IND-public-key cryptosystem PKE with ciphertext space $\widehat{\mathrm{C}}$. Moreover, we choose the message space M of the underlying public-key encryption scheme PKE to be equal to $\mathbb{Z}_q^*$. KG, Sign and Vfy are the key generation, signature generation and signature verification algorithms of the underlying leakage-resilient signature scheme SIG (Section 2.3.4). The protocol π is a Diffie-Hellman-type [19] key agreement protocol where $\mathbb{G}$ is a group of prime order q with generator g. After exchanging the public values both principals compute a Diffie-Hellman-type shared secret value, KDF is a secure key derivation function (Section 2.2.1) which generates a shared secret key (ms) using the Diffie-Hellman-type shared secret key, and PRF is a pseudo random function (Section 2.2.2) that is used to compute the session key using that shared key, ms, and the protocol message sequence. The computations which leak information are underlined.

Remark 3. In Table 5, let $\widehat{\mathrm{C}}$ be the ciphertext space: in a setting like Naor and Segev [44], the random r values are not just chosen from C, but from $\widehat{\mathrm{C}} = \{0,1\}^\mu \times \mathrm{C} \times \{0,1\}^t$, which gives random $r \xleftarrow{\$} \widehat{\mathrm{C}}$ in the form (Ψ, c, seed).

A (Initiator)		B (Responder)
	Initial Setup	
$sk_A, vk_A \xleftarrow{\$} \mathrm{KG}(1^k)$		$sk_B, vk_B \xleftarrow{\$} \mathrm{KG}(1^k)$
$s_A, p_A \xleftarrow{\$} \mathrm{KeyGen}(1^k)$		$s_B, p_B \xleftarrow{\$} \mathrm{KeyGen}(1^k)$
	Protocol Execution	
$r_A \xleftarrow{\$} \widehat{\mathrm{C}}$		**If** $\mathrm{Vfy}(vk_A, X_A, \sigma_A) =$ "true" $\{$
$\widetilde{r_A} \leftarrow \underline{\mathrm{Dec}}(s_A, r_A)$		$r_B \xleftarrow{\$} \widehat{\mathrm{C}}$
$X_A \leftarrow g^{\widetilde{r_A}}$		$\widetilde{r_B} \leftarrow \underline{\mathrm{Dec}}(s_B, r_B)$
$\sigma_A \xleftarrow{\$} \underline{\mathrm{Sign}}(sk_A, (A, B, X_A))$	$\xrightarrow{A,B,X_A,\sigma_A}$	$X_B \leftarrow g^{\widetilde{r_B}}$
	$\xleftarrow{B,A,X_B,\sigma_B}$	$\sigma_B \xleftarrow{\$} \underline{\mathrm{Sign}}(sk_B, (B, A, X_B))$
1-1 **If** $\mathrm{Vfy}(vk_B, (B, A, X_B), \sigma_B) =$ "true" $\{$		
$\widetilde{r_A} \leftarrow \underline{\mathrm{Dec}}(s_A, r_A)$		
$ms \leftarrow \mathrm{KDF}(X_B^{\widetilde{r_A}}, \perp, k, \perp)$		$ms \leftarrow \mathrm{KDF}(X_A^{\widetilde{r_B}}, \perp, k, \perp)$
$K \leftarrow \mathrm{PRF}(ms, A\Vert X_A\Vert \sigma_A\Vert B\Vert X_B\Vert \sigma_B)$		$K \leftarrow \mathrm{PRF}(ms, A\Vert X_A\Vert \sigma_A\Vert B\Vert X_B\Vert \sigma_B)$
$\}$		$\}$
	K is the session key	

Table 5: Generic w(·)AFL-eCK-secure protocol construction: Protocol π

5.2.3 Security of the Protocol π in the w(·)AFL-eCK Model

We prove the security of the generic protocol π in the w(·)AFL-eCK model. If the underlying primitives are secure in the bounded or continuous leakage model, the protocol π is BAFL-eCK-secure or CAFL-eCK-secure respectively (with the restriction that `EphemeralKeyReveal` query does not reveal the randomness used in the signature computation).

Theorem 5.1. *The protocol* π *is* $\mathrm{Adv}_\pi^{\boldsymbol{\lambda}-w(\cdot)\text{AFL-eCK}}$*-secure, whenever the underlying public-key cryptosystem* PKE *is* CPLA2*-secure and* $\epsilon -$ PG*-IND, the key derivation function* KDF *is secure with respect to an uniformly random source key material, the signature scheme* SIG *is* UFCMLA*-secure,* PRF *is a pseudo random function, the* DDH *and the* ODH *[1] assumptions hold.*

Let $\mathcal{U} = \{U_1, \dots, U_{N_P}\}$ *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 PPT adversary against the protocol* π*. Then the advantage of* $\mathcal{A}$ *against* $\boldsymbol{\lambda} - w(\cdot)$AFL-eCK*-security of protocol* π*,* $\mathrm{Adv}_\pi^{\boldsymbol{\lambda}-w(\cdot)\text{AFL-eCK}}$ *is:*

$$\begin{aligned}\mathrm{Adv}_{\pi}^{\boldsymbol{\lambda}-w(\cdot)\text{AFL-eCK}}(\mathcal{A}) \leq \max \Big[& N_P^2 N_s^2 \big[\big(\mathrm{Adv}_{q,g}^{DDH}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) \big) + \frac{1}{q} \big], \\ & N_P^2 N_s^2 \big[\big(\mathrm{Adv}_{q,g}^{DDH}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) + 2\mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D}) + 2\epsilon \big) + \frac{1}{q} \big], \\ & N_P^2 N_s^2 \big[\big(\mathrm{Adv}_{q,g}^{\mathrm{ODH}}(\mathcal{R}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) + 2\mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D}) + 2\epsilon \big) + \frac{1}{q} \big], N_P \mathrm{Adv}_{\mathrm{SIG}}^{\mathrm{UFCMLA}}(\mathcal{E}) \Big].\end{aligned}$$

where $\mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}, \mathcal{J}, \mathcal{R}$ are efficient algorithms constructed using the adversary $\mathcal{A}$, against the underlying key derivation function, KDF, DDH *problem, public-key cryptosystem,* PKE*, the signature scheme,* SIG*, pseudo random function,* PRF*, and Oracle Diffie-Hellman problem respectively. The* PKE *is $\epsilon-$*PG-*IND.*

The detailed proof of this theorem is in Appendix B.

5.2.4 Leakage Tolerance of the w(·)AFL-eCK-secure Protocol π: wBAFL-eCK-Secure Instantiation

In the presented protocol, a principal uses a decryption function of a CPLA2-secure $\epsilon-$PG-IND-public-key cryptosystem to compute the NAXOS value in a leakage-resilient manner and sets the Diffie-Hellman exponent as the decrypted message. Then, the principal uses a UFCMLA-secure signature scheme to authenticate the message. Therefore, the leakage tolerance from the secret key used to compute the NAXOS value is exactly same as the leakage tolerance of the underlying CPLA2-secure public key encryption scheme, and the leakage tolerance from the secret key used to compute the signature is exactly same as the leakage tolerance of the underlying UFCMLA-secure signature scheme.

Halevi and Lin [25] constructed a generic CPLA2-secure public-key cryptosystem which is secure against bounded leakage and also satisfies pair generation indistinguishability. It can be instantiated with the DDH-based leakage-resilient public-key encryption scheme of Naor and Segev [44] with decryption cost of 4 exponentiations, and for a key length k the leakage is bounded by $(1-o(1))k$. Katz and Vaikuntanathan [30] constructed an UFCMLA-secure signature scheme in the bounded leakage model, where a signature can be generated with cost of 2 exponentiations, and verified with cost of 4 exponentiations (with a simple NIZK proof). The signature scheme of Katz and Vaikuntanathan contains signing and verification operations based on NIZK protocols. For a key length k, the signature scheme tolerates leakage of $(1-k^t) \cdot k$, for any constant $t < 1$. Hence, this protocol can be instantiated with the above mentioned leakage-resilient signature scheme and the public-key encryption scheme, and achieve leakage tolerance according to the leakage parameters specified in the above mentioned cryptographic constructions.

5.3 Concrete CAFL-eCK-secure Key Exchange Protocol [7]

In section 5.2 we presented a generic construction for a protocol which is proven secure in the w(·)AFL-eCK security model. However, when it comes to a concrete construction, the presented generic protocol can only be instantiated in a way that is secure in the *bounded* version of the security model and the model we considered is slightly weaker than the desired (·)AFL-eCK model. Up to now there are no suitable cryptographic primitives which can be used to instantiate the generic protocol in the continuous leakage variant of the security model, as well as in the desired (·)AFL-eCK model. Now we present a concrete protocol construction [7] which is proven secure in the continuous leakage instantiation of the generic (·)AFL-eCK, namely CAFL-eCK model. Moreover, this construction does not require a weaker variant of the model for the security proof.

Moving to the leakage-resilient setting of the eCK-style secure key exchange requires rethinking the NAXOS trick. We have presented a generic construction of a weak after-the-fact leakage eCK (w(·)AFL-eCK)-secure key exchange protocol in Section 5.2, which uses a leakage-resilient NAXOS trick. The leakage-resilient NAXOS trick is obtained using a decryption function of an after-the-fact leakage-resilient public key encryption scheme. A concrete construction of a wBAFL-eCK-secure protocol is possible since there exists a bounded after-the-fact leakage-resilient public key encryption scheme which can be used to obtain the required leakage-resilient NAXOS trick, but it is not currently possible to construct a CAFL-eCK-secure protocol since there is currently no continuous after-the-fact leakage-resilient public-key encryption scheme available. Therefore, an attempt to construct a CAFL-eCK-secure key exchange protocol using the leakage-resilient NAXOS approach is not possible at this stage.

In section 3.2 we presented a eCK-secure protocol construction [7], which does not use the NAXOS trick, namely the protocol P1. The protocol P1 is based on Diffie-Hellman key exchange, which requires exponentiation computations. Moving to the leakage-resilient setting requires rethinking the exponentiation computation in a leakage-resilient manner. Since there exist leakage-resilient encoding schemes and leakage-resilient refreshing protocols for them (Definition 2.5 and 2.8), the aim is to compute the required exponentiations in a leakage-resilient manner using the available leakage-resilient storage and refreshing schemes.

5.3.1 Leakage-Resilient Construction of Protocol P2

Protocol P1 is an eCK-secure key exchange protocol. The eCK model considers an environment where partial information leakage does not take place. Following the concept that only computation leaks information, it is assumed that the leakage of long-term secret keys happens when computations are performed using them. Then, instead of the *non-leakage* eCK model which is used for the security proof of protocol P1, the CAFL-eCK model is used, which additionally allows the adversary to obtain continuous leakage of long-term secret keys.

The idea is to perform the computations which use long-term secret keys (exponentiation operations) in such a way that the resulting leakage from the long-term secrets should not leak sufficient information to reveal them to the adversary. To overcome that challenge a leakage-resilient storage scheme and a leakage-resilient refreshing protocol are used, and the architecture of the protocol P1 is modified, in such a way that the secret keys s are encoded into two portions s_L, s_R, Exponentiations are computed using two portions s_L, s_R instead of directly using s, and the two portions s_L, s_R are being refreshed continuously.

Obtaining Leakage Resiliency by Encoding Secrets. In this setting a secret s is encoded using an Encode function of a leakage-resilient storage scheme $\Lambda = (\mathrm{Encode}, \mathrm{Decode})$. So the secret s is encoded as $(s_L, s_R) \leftarrow \mathrm{Encode}(s)$. The leakage-resilient storage scheme randomly chooses s_L and then computes s_R such that $s_L \cdot s_R = s$. A tuple leakage parameter $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$ is defined as follows: $\boldsymbol{\lambda}$-limited adversary $\mathcal{A}$ sends a leakage function $\mathbf{f} = (f_{1j}, f_{2j})$ and obtains at most λ_1, λ_2 amount of leakage from each of the two encodings of the secret s respectively: $f_{1j}(s_L)$ and $f_{2j}(s_R)$.

As mentioned in Definition 2.8, the leakage-resilient storage scheme can continuously refresh the encodings of the secret. Therefore, after executing the refreshing protocol it outputs new random-looking encodings of the same secret. So for the $\boldsymbol{\lambda}$-limited adversary again the situation is as before. Thus, refreshing the encodings will help to obtain leakage resilience over a number of protocol executions.

The computation of exponentiations is also split into two parts. Let $\mathbb{G}$ be a group of prime order q with generator g. Let $s \xleftarrow{\$} \mathbb{Z}_q^*$ be a long-term secret key and $E = g^e$ be a received ephemeral value. Then, the value Z needs to be computed as $Z \leftarrow E^s$. In the leakage-resilient setting, in the initial setup the secret key is encoded as $s_L, s_R \leftarrow \mathrm{Encode}^{n,1}_{\mathbb{Z}_q^*}(s)$. So the vector $s_L = (s_{L1}, \cdots, s_{Ln})$ and the vector $s_R = (s_{R1}, \cdots, s_{Rn})$ are such that $s = s_{L1}s_{R1} + \cdots + s_{Ln}s_{Rn}$. Then the computation of E^s can be performed as two component-wise computations as follows: compute the intermediate vector $T \leftarrow E^{s_L} = (E^{s_{L1}}, \cdots, E^{s_{Ln}})$ and then compute the element $Z \leftarrow T^{s_R} = E^{s_{L1}s_{R1}} E^{s_{L2}s_{R2}} \cdots E^{s_{L1}s_{R1}} = E^{s_{L1}s_{R1} + \cdots + s_{Ln}s_{Rn}} = E^s$.

5.3.2 Protocol Construction

Using the above ideas, by encoding the secret using a leakage-resilient storage scheme, and refreshing the encoded secret using a refreshing protocol, it is possible to hide the secret from a $\boldsymbol{\lambda}$-limited adversary. Further, it is possible to successfully compute the exponentiation using the encoded secrets. A CAFL-eCK-secure key exchange protocol is constructed, using an eCK-secure key exchange protocol as an underlying primitive.

Let $\Lambda^{n,1}_{\mathbb{Z}_q^*} = (\mathrm{Encode}^{n,1}_{\mathbb{Z}_q^*}, \mathrm{Decode}^{n,1}_{\mathbb{Z}_q^*})$ be the leakage-resilient storage scheme which is used to encode secret keys and $\mathrm{Refresh}^{n,1}_{\mathbb{Z}_q^*}$ be the $(\ell, \boldsymbol{\lambda}, \epsilon)$-secure leakage-resilient refreshing protocol of $\Lambda^{n,1}_{\mathbb{Z}_q^*}$.

As we can see, the obvious way of key generation (initial setup) in a protocol principal of this protocol is as follows: first pick $a \xleftarrow{\$} \mathbb{Z}_q^*$ as the long-term secret key, then encode the secret key as $(a_L^0, a_R^0) \leftarrow \mathrm{Encode}^{n,1}_{\mathbb{Z}_q^*}(a)$, then compute the long-term public key $A = g^a$ using the two encodings (a_L^0, a_R^0), and finally erase a from the memory. The potential threat to that key generation mechanism is that even though the long-term secret key a is erased from the memory, it might not be properly erased and can be leaked to the adversary during the key generation. In order to avoid such a vulnerability, two values $a_L^0 \xleftarrow{\$} (\mathbb{Z}_q^*)^n \backslash \{(0^n)\}$, $a_R^0 \xleftarrow{\$} (\mathbb{Z}_q^*)^{n\times 1} \backslash \{(0^{n\times 1})\}$ are picked at random and use them as the encodings of the long-term secret key a of a protocol principal. As explained earlier, a_L^0, a_R^0 are used to compute the corresponding long-term public key A in two steps as $a' \leftarrow g^{a_L^0}$ and $A \leftarrow a'^{a_R^0}$. Thus, it is possible to avoid exposing the un-encoded secret key a at any point of time in the key generation and hence avoid leaking directly from a at the key generation step. Further, the random vector a_L^0 is multiplied with the random vector a_R^0, such that $a = a_L^0 \cdot a_R^0$, which will give a random integer a in the group $\mathbb{Z}_q^*$. Therefore, this approach is same as picking $a \xleftarrow{\$} \mathbb{Z}_q^*$ at first and then encode, but in the reverse order. During the protocol execution both a_L^0, a_R^0 are continuously refreshed and refreshed encodings a_L^j, a_R^j are used to exponentiation computations.

Table 6 shows the protocol P2 of Alawatugoda et al. [7]. Leakage of a long-term secret key does not happen directly from the long-term secret key itself, but from the two encodings of the long-term secret key (the leakage function $\mathbf{f} = (f_{1j}, f_{2j})$ directs to the each individual encoding). During the exponentiation computations and the refreshing operation collectively at most $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$ leakage is allowed to the adversary from each of the two portions independently. Then, the two portions of the encoded long-term secret key are refreshed and in the next protocol session another $\boldsymbol{\lambda}$-bounded leakage is allowed. Thus, continuous leakage is allowed.

Alice (Initiator)		Bob (Responder)
	Initial Setup	
$a_L^0 \xleftarrow{\$} (\mathbb{Z}_q^*)^n \backslash \{(0^n)\}, a_R^0 \xleftarrow{\$} (\mathbb{Z}_q^*)^{n\times 1} \backslash \{(0^{n\times 1})\}$		$b_L^0 \xleftarrow{\$} (\mathbb{Z}_q^*)^n \backslash \{(0^n)\}, b_R^0 \xleftarrow{\$} (\mathbb{Z}_q^*)^{n\times 1} \backslash \{(0^{n\times 1})\}$
$a' \leftarrow g^{a_L^0}, A \leftarrow (a')^{a_R^0}$		$b' \leftarrow g^{b_L^j}, B \leftarrow (b')^{b_R^0}$
	Protocol Execution	
$x \xleftarrow{\$} \mathbb{Z}_q^*, X \leftarrow g^x$	$\xrightarrow{Alice, X}$ $\xleftarrow{Bob, Y}$	$y \xleftarrow{\$} \mathbb{Z}_q^*, Y \leftarrow g^y$
$T_1 \leftarrow B^{a_L^j}, Z_1 \leftarrow T_1^{a_R^j}$		$T_3 \leftarrow A^{b_L^j}, Z_1' \leftarrow T_3^{b_R^j}$
$Z_2 \leftarrow B^x$		$T_4 \leftarrow X^{b_L^j}, Z_2' \leftarrow T_4^{b_R^j}$
$T_2 \leftarrow Y^{a_L^j}, Z_3 \leftarrow T_2^{a_R^j}$		$Z_3' \leftarrow A^y$
$Z_4 \leftarrow Y^x$		$Z_4' \leftarrow X^y$
$(a_L^{j+1}, a_R^{j+1}) \leftarrow \mathrm{Refresh}_{\mathbb{Z}_q^*}^{n,1}(a_L^j, a_R^j)$		$(b_L^{j+1}, b_R^{j+1}) \leftarrow \mathrm{Refresh}_{\mathbb{Z}_q^*}^{n,1}(b_L^j, b_R^j)$
$K \leftarrow \mathrm{H}(Z_1, Z_2, Z_3, Z_4, Alice, X, Bob, Y)$		$K \leftarrow \mathrm{H}(Z_1', Z_2', Z_3', Z_4', Alice, X, Bob, Y)$
	K is the session key	

Table 6: Concrete construction of Protocol P2

5.3.3 Security of the Protocol P2 in the CAFL-eCK Model

Theorem 5.2. *[7] If the underlying refreshing protocol* $\mathrm{Refresh}_{\mathbb{Z}_q^*}^{n,1}$ *is $(\ell, \boldsymbol{\lambda}, \epsilon)$-secure leakage-resilient refreshing protocol of the leakage-resilient storage scheme $\Lambda_{\mathbb{Z}_q^*}^{n,1}$ and the underlying key exchange protocol* P1 *is* eCK*-secure key exchange protocol, then the protocol* P2 *is* $\boldsymbol{\lambda} -$ CAFL-eCK*-secure.*

Let $\mathcal{A}$ be any PPT adversary against the key exchange protocol P2*. Then the advantage of $\mathcal{A}$ against the* CAFL-eCK*-security of the protocol* P2*,* $\mathrm{Adv}_{\mathrm{P2}}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}$ *is:*

$$\mathrm{Adv}_{\mathrm{P2}}^{\boldsymbol{\lambda}-\text{CAFL-eCK}}(\mathcal{A}) \leq N_P\Big(\mathrm{Adv}_{\mathrm{P1}}^{\mathrm{eCK}}(\mathcal{A}) + \epsilon\Big) \ . \tag{1}$$

5.3.4 Leakage Tolerance of the Protocol P2

The order of the group $\mathbb{G}$ is q. Let $m = 1$ in the leakage-resilient storage scheme $\Lambda_{\mathbb{Z}_q^*}^{n,1}$. According to the Lemma 2.1, if $m < n/20$, then the leakage parameter for the leakage-resilient storage scheme is $\boldsymbol{\lambda}_\Lambda = (0.3n \log q, 0.3n \log q)$. Let $n = 21$, then $\boldsymbol{\lambda}_\Lambda = (6.3 \log q, 6.3 \log q)$ bits. According to the Theorem 2.2, if $m/3 \leq n$ and $n \geq 16$, the refreshing protocol $\mathrm{Refresh}_{\mathbb{Z}_q^*}^{n,1}$ of the leakage-resilient storage scheme $\Lambda_{\mathbb{Z}_q^*}^{n,1}$ is tolerant to (continuous) leakage up to $\boldsymbol{\lambda}_{\mathrm{Refresh}} = \boldsymbol{\lambda}_\Lambda/2 = (3.15 \log q, 3.15 \log q)$ bits, per occurrence.

When a secret key s (of size $\log q$ bits) of the protocol P2 is encoded into two parts, the left part s_L will be $n \cdot \log q = 21 \log q$ bits and the right part s_R will be $n \cdot 1 \cdot \log q = 21 \log q$ bits. For a tuple leakage function $\mathbf{f} = (f_{1j}, f_{2j})$ (each leakage function $f_{(\cdot)}$ for each of the two parts s_L and s_R), there exists a tuple leakage bound $\boldsymbol{\lambda} = (\lambda, \lambda)$ for each leakage function $f_{(\cdot)}$, such that $\lambda = 3.15 \log q$ bits, per occurrence, which is $\frac{3.15 \log q}{21 \log q} \times 100\% = 15\%$ of the size of a part. The overall leakage amount is unbounded since continuous leakage is allowed.

6 Comparison of Key Exchange Security Models and Protocols

In this paper, we have presented two security models for key exchange protocols, addressing more granular partial leakage of long-term secret keys, namely continuous after-the-fact leakage model (CAFL) and the generic after-the-fact leakage eCK model ((·)AFL-eCK) (and w(·)AFL-eCK) model). Further, we presented generic protocol constructions for each of CAFL and w(·)AFL-eCK) models and a concrete protocol construction for the continuous leakage variant of the (·)AFL-eCK model, the CAFL-eCK model.

6.1 Comparison of Security Models

Table 7 summarizes the adversarial powers of the two instantiations of the generic (·)AFL-eCK model and the CAFL model, in comparison with the adversarial powers of the eCK model [36] and the Moriyama–Okamoto (MO) model [43]. There are four `Corrupt`–`EphemeralKeyReveal` query combinations which do not trivially expose the session key. In the column "Combinations" of Table 7, we mention how many of them are allowed in the corresponding security model. We discussed query combinations in detail in Section 4.1.3. The $*$ indicates that the w(·)AFL-eCK model does not allow the `EphemeralKeyReveal` query to reveal the randomness used in the underlying signature scheme.

Security model	Combinations	Leakage model	After-the-fact
eCK[36]	4/4	No	No
MO[43]	4/4	Bounded	No
CAFL (Section 4)	2/4	Continuous	Yes
(·)AFL-eCK (Section 5)	4/4	Bounded/Continuous	Yes
w(·)AFL-eCK (Section 5)	$4^*/4$	Bounded/Continuous	Yes

Table 7: Key exchange security models with reveal queries and leakage allowed

The eCK model is a non-leakage security model and Moriyama and Okamoto have constructed a leakage security model based on eCK model. The Moriyama-Okamoto model allows bounded leakage, only before the target session is established. The CAFL model, allows continuous leakage even after the target session is established, while enforcing additional restrictions to the eCK-style freshness condition. Therefore, the strength of the CAFL model is not directly comparable with eCK or Moriyama-Okamoto models, but the CAFL model clearly allows more granular partial leakage. The generic (·)AFL-eCK releases the additional restrictions to the freshness condition which had been introduced in the CAFL model. Thus, the two instantiations of the generic (·)AFL-eCK model, namely the BAFL-eCK and CAFL-eCK models are stronger than the eCK, the Moriyama-Okamoto and the CAFL models.

6.2 Comparison of Key Exchange Protocols

Table 8 compares the protocol $\pi 1$ of section 4, the protocol π of section 5 and the protocol P2 of section 5, with the NAXOS protocol [36] and the Moriyama–Okamoto protocol [43].

Protocol	Security Model	Assumptions	Construction
NAXOS [36]	eCK	GDH, RO	Concrete
MO [43]	MO	DDH, HPS, PRF, (λ, ϵ)-Ext	Concrete
$\pi 1$ of Section 4	CAFL	CCLA2-secure PKE, PRF,	Generic
π of Section 5	w(·)AFL-eCK	DDH, ODH, $\epsilon -$ PG-IND and CPLA2-secure PKE, secure KDF, PRF	Generic
P2 of Section 5	CAFL-eCK	GDH, RO	Concrete

Table 8: Comparison of key exchange protocols

The NAXOS protocol is the first concrete protocol which is proven secure in the eCK model. Being an eCK-secure protocol, NAXOS does not provide any security guarantee for side-channel attacks. The Moriyama-Okamoto protocol is the first concrete protocol which is proven secure in a leakage security model, namely the Moriyama-Okamoto model. The Moriyama-Okamoto protocol is resistant to a bounded amount of leakage of long-term secret keys only before the target session is activated. We presented a generic CAFL-secure protocol, which can be instantiated using any suitable leakage-resilient public-key encryption scheme, in a way that it is secure against continuous leakage of long-term secret keys even after the target session is activated. Then we presented a generic w(·)AFL-eCK-secure protocol in a way that it is possible to instantiate a wBAFL-eCK or wCAFL-eCK-secure key exchange protocol using any suitable leakage-resilient public-key encryption scheme and a leakage-resilient signature scheme. Since there are currently no suitable leakage-resilient public-key encryption schemes to instantiate the continuous leakage-resilient variant of the generic protocol, we presented a concrete CAFL-eCK-secure protocol, namely protocol P2, using leakage-resilient storage schemes. Protocol P2 is proven secure in the strongest leakage-security model for key exchange, guaranteeing the eCK-style security as well as tolerance against continuous leakage of long-term secret keys even after the target session is activated.

The generic CAFL-secure protocol of section 4 can be instantiated with the CCLA2-secure public-key encryption scheme of Dziemboski and Faust [22], whereas the generic w(·)AFL-eCK-secure protocol of section 5 can be instantiated as a wBAFL-eCK-secure protocol using the CPLA2-secure pair generation indistinguishable public-key encryption scheme of Halevi and Lin [25] and the UFCMLA-secure signature scheme of Katz and Vaikuntanathan [30]. Table 9 compares these protocol instantiations and the protocol P2 with the NAXOS protocol [36] and the Moriyama-Okamoto (MO) protocol [43], in terms of computation cost and the security model.

Protocol	Initiator Cost	Responder Cost	Security Model
NAXOS [36]	4 **Exp**	4 **Exp**	eCK
MO [43]	8 **Exp**	8 **Exp**	MO
π1 of Section 4 instantiation	10 **Exp**	10 **Exp**	CAFL
π of Section 5 instantiation	12 **Exp**	12 **Exp**	wBAFL-eCK
P2 protocol of Section 5	6 **Exp**	6 **Exp**	CAFL-eCK

Table 9: Security and efficiency comparison of key exchange protocols

Acknowledgements

I would like to acknowledge Dr. Douglas Stebila and Professor Colin Boyd for numerous discussions on key exchange protocols and leakage-resilient cryptography.

References

[1] M. Abdalla, M. Bellare, and P. Rogaway. The oracle diffie-hellman assumptions and an analysis of DHIES. In *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, pages 143–158, 2001.

[2] M. Abdalla, D. Catalano, and D. Fiore. Verifiable random functions from identity-based key encapsulation. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 554–571, 2009.

[3] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptology Conference*, pages 474–495, 2009.

[4] J. Alawatugoda, C. Boyd, and D. 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.

[5] 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.

[6] J. Alawatugoda, D. Stebila, and C. 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.

[7] J. Alawatugoda, D. Stebila, and C. 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.

[8] J. A. B. Alawatugoda. *On the leakage resilience of secure channel establishment.* PhD thesis, Queensland University of Technology, 2015.

[9] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *CRYPTO*, pages 232–249, 1993.

[10] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.

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

[12] D. J. Bernstein. Cache-timing attacks on AES. Technical report, 2005. http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[13] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'00, pages 156–171, Berlin, Heidelberg, 2000. Springer-Verlag.

[14] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. *IACR Cryptology ePrint Archive*, Report 2010/278, 2010.

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

[16] K. R. Choo, C. Boyd, Y. Hitchcock, and G. Maitland. On session identifiers in provably secure protocols: The bellare-rogaway three-party key distribution protocol revisited. In *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers*, pages 351–366, 2004.

[17] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64. Springer, 2002.

[18] C. Cremers. Examining indistinguishability-based security models for key exchange protocols: the case of ck, ck-hmqv, and eck. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011, Hong Kong, China, March 22-24, 2011*, pages 80–91, 2011.

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

[20] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptography*, 2(2):107–125, 1992.

[21] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, pages 523–540, 2004.

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

[23] S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *IEEE Symposium on Foundations of Computer Science*, pages 293–302, 2008.

[24] S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum. Leakage-resilient signatures. *IACR Cryptology ePrint Archive*, Report 2009/282, 2009.

[25] S. Halevi and H. Lin. After-the-fact leakage in public-key encryption. In *Theory of Cryptology Conference*, pages 107–124, 2011.

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

[27] M. Hutter, S. Mangard, and M. Feldhofer. Power and EM attacks on passive 13.56MHz RFID devices. In *CHES*, pages 320–333, 2007.

[28] D. P. Jablon. Strong password-only authenticated key exchange. *SIGCOMM Comput. Commun. Rev.*, 26(5):5–26, Oct. 1996.

[29] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.

[30] J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.

[31] E. Kiltz, K. Pietrzak, M. Stam, and M. Yung. A new randomness extraction paradigm for hybrid encryption. In *EUROCRYPT*, pages 590–609, 2009.

[32] M. Kim, A. Fujioka, and B. Ustaoglu. Strongly secure authenticated key exchange without naxos' approach. In *Advances in Information and Computer Security, 4th International Workshop on Security, IWSEC 2009, Toyama, Japan, October 28-30, 2009, Proceedings*, pages 174–191, 2009.

[33] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. pages 104–113. Springer-Verlag, 1996.

[34] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *CRYPTO*, pages 546–566, 2005.

[35] H. Krawczyk. On extract-then-expand key derivation functions and an HMAC-based KDF. http://webee.technion.ac.il/ hugo/kdf/kdf.pdf, 2008.

[36] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *ProvSec*, pages 1–16, 2007.

[37] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. Technical report, Designs, Codes and Cryptography, 1998.

[38] T. Malkin, I. Teranishi, Y. Vahlis, and M. Yung. Signatures resilient to continual leakage on memory and computation. In *Theory of Cryptology Conference*, pages 89–106, 2011.

[39] J. R. McCumber. Information systems security : A comprehensive model. In *Proceedings of the 14th National Computer Security Conference*, October 1991.

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

[41] S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In *Theory of Cryptology Conference*, pages 278–296, 2004.

[42] D. Moriyama and T. Okamoto. An eck-secure authenticated key exchange protocol without random oracles. In *Provable Security, Third International Conference, ProvSec 2009, Guangzhou, China, November 11-13, 2009. Proceedings*, pages 154–167, 2009.

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

[44] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35. 2009.

[45] K. Pietrzak. A leakage-resilient mode of operation. In *Proceedings of the 28th Annual International Conference on Advances in Cryptology: the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '09, pages 462–482, Berlin, Heidelberg, 2009. Springer-Verlag.

[46] B. Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Des. Codes Cryptography*, 46(3):329–342, 2008.

[47] Z. Yang. Efficient eck-secure authenticated key exchange protocols in the standard model. In *Information and Communications Security - 15th International Conference, ICICS 2013, Beijing, China, November 20-22, 2013. Proceedings*, pages 185–193, 2013.

A Proof of the Theorem 4.1

Proof. Assume that the adversary $\mathcal{A}$ can win the challenge against the protocol $\pi 1$ challenger with advantage $\mathrm{Adv}_{\pi 1}^{\lambda-\mathrm{CAFL}}(\mathcal{A})$. We split the proof into two cases: partner to the test session exists and partner to the test session does not exist.

Case 1: Partner to the test session exists

In this case we consider three sub cases as follows:

1. Adversary corrupts the owner of the test session, but does not corrupt the peer.
2. Adversary corrupts the peer of the test session, but does not corrupt the owner.
3. Adversary corrupts neither the owner nor the partner of the test session

Case 1.1: Adversary corrupts the owner of the test session, but does not corrupt the peer

In this case we consider the situation that $\mathcal{A}$ corrupts the owner of the test session but not the partner.

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

Game 2. Same as Game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \leftarrow \{U_1, \ldots, U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \leftarrow \{1, \ldots, N_s\}$ 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 the partner to the session is not $\Pi^{t^*}_{V^*,U^*}$, the Game 2 challenger aborts the game.

Game 3. Same as Game 2 with the following exception: the Game 3 challenger chooses a random value $r' \xleftarrow{\$} \{0,1\}^k$.

- If the test session is on the initiator, the challenger computes the session key in the test session $K \leftarrow \mathrm{PRF}(r', U^*||C_{U^*}||V^*||C_{V^*}) \oplus \mathrm{PRF}(r_{V^*}, U^*||C_{U^*}||V^*||C_{V^*})$.
- If the test session is on the responder, the challenger computes the session key in the test session $K \leftarrow \mathrm{PRF}(r_{V^*}, V^*||C_{V^*}||U^*||C_{U^*}) \oplus \mathrm{PRF}(r', V^*||C_{V^*}||U^*||C_{U^*})$.

The session key is computed in the same way in the partner to the test session.

Game 4. Same as Game 3 with the following exception: In the `Test` query, in the target session, the Game 4 challenger randomly chooses $K \xleftarrow{\$} \{0,1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the `Test` query. In sessions at V^* which has the same incoming message to V^* as in the target session, the session key is randomly chosen as $K \xleftarrow{\$} \{0,1\}^k$.

Differences between games. In this section the adversary's advantage of distinguishing each game from the previous game is investigated. $\mathrm{Adv}_{\text{Game x}}(\mathcal{A})$ denotes the advantage of the adversary $\mathcal{A}$ of winning Game x.

Game 1 is the original game. Hence,

$$\mathrm{Adv}_{\text{Game 1}}(\mathcal{A}) = \mathrm{Adv}^{\lambda-\text{CAFL}}_{\pi 1,\text{Case 1.1}}(\mathcal{A}). \tag{2}$$

Game 1 and Game 2. The probability of Game 2 to be halted due to incorrect choice of the test session is $1 - \frac{1}{N_P^2 N_s^2}$. Unless the incorrect choice happens, Game 2 is identical to Game 1. Hence,

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

Game 2 and Game 3. We introduce an algorithm $\mathcal{D}$ which 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 the underlying public-key cryptosystem, PKE. The algorithm $\mathcal{D}$ uses the public key of the CCLA2 challenger as the public key of the protocol principal V^* and generates public/secret key pairs for all other protocol principals. $\mathcal{D}$ runs a copy of $\mathcal{A}$ and interacts with $\mathcal{A}$, such that $\mathcal{A}$ is interacting with either Game 2 or Game 3. $\mathcal{D}$ picks two random strings, $r'_0, r'_1 \leftarrow \{0,1\}^k$ and passes them to the CCLA2 challenger. From the CCLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C such that $C \leftarrow \mathrm{Enc}(pk_{V^*}, r')$ where $r' = r'_0$ or $r' = r'_1$. The following describes $\mathcal{D}$'s procedure of answering queries.

- `Send`(U, V, s, m, f) query:
  - $U = U^*, V = V^*, s = s^*$:
    - If U^* is the initiator, $\mathcal{D}$ sends the ciphertext C to $\mathcal{A}$ as the first message of the test session. Upon receiving the second protocol message computes the session key $K \leftarrow \mathrm{PRF}(r'_1, U^* || C_{U^*} || V^* || C_{V^*}) \oplus \mathrm{PRF}(r_{V^*}, U^* || C_{U^*} || V^* || C_{V^*})$.
    - If U^* is the responder, upon receiving the first protocol message sends C to $\mathcal{A}$, and computes the session key $K \leftarrow \mathrm{PRF}(r'_1, V^* || C_{V^*} || U^* || C_{U^*}) \oplus \mathrm{PRF}(r_{V^*}, V^* || C_{V^*} || U^* || C_{U^*})$.
  - $U = U^*, V = V^*, s \neq s^*$: Executes the protocol normally.
  - $U = U^*, V \neq V^*$: Executes the protocol normally.
  - $U = V^*$:
    - If this is the initiator and it is the first message, then executes the protocol normally.
    - If this is the initiator and the second protocol message, or the responder:
      - If C has come as the incoming message uses r'_1 as the decryption of the incoming message. To obtain the corresponding leakage, $\mathcal{D}$ first encrypts r'_1 using pk_{V^*}, gets that ciphertext and access the leakage oracle of CCLA2 challenger with the ciphertext of r'_1.
      - Else uses the decryption oracle to decrypt incoming messages.
  - $U, V \neq U^*$ or V^*: Executes the protocol normally.

    For all other leakage queries $f(sk_{V^*})$, $\mathcal{D}$ obtains the leakage accessing the leakage oracle of the CCLA2 challenger, whereas for all the other leakage queries, $\mathcal{D}$ can compute the leakage by its own, because $\mathcal{D}$ knows all other secret keys.

- `SessionKeyReveal`(U, V, s) query: `SessionKeyReveal` query is not allowed to the target session or the partner of the target session. $\mathcal{D}$ can compute all the session keys by executing the protocol.
  - For sessions involving the principal V^*, and the incoming message to V^* is the same message which has come to V^* in the target session, uses r'_1 as the decryption.
  - For other sessions involving the principal V^*, $\mathcal{D}$ can decrypt the incoming messages to V^* by using the decryption oracle.
  - Otherwise, $\mathcal{D}$ can decrypt all the other incoming messages to protocol principals by its own.

  Then compute the session key using the PRF.

- `EphemeralKeyReveal`(U, V, s) query: For all `EphemeralKeyReveal` queries allowed in the freshness condition, $\mathcal{D}$ can answer correctly, because $\mathcal{D}$ has the ephemeral keys.

- `Corrupt`(U) query: Except for V^*, algorithm $\mathcal{D}$ can answer all other `Corrupt` queries. In this case we consider the situation in which the adversary is not allowed to corrupt the partner principal of the target session, so in fact, $\mathcal{D}$ can answer all legitimate `Corrupt` queries.

- `Test`(U, V, s) query: Answers with the K which is computed at the `Send` query when $U = U^*, V = V^*, s = s^*$.

If r'_1 is the decryption of C in the target session, the simulation constructed by $\mathcal{D}$ is identical to Game 2 whereas if r'_0 is the decryption of C, the simulation constructed by $\mathcal{D}$ is identical to Game 3. If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{D}$ can distinguish whether $C \leftarrow \mathrm{Enc}(pk_{V^*}, r'_0)$ or $C \leftarrow \mathrm{Enc}(pk_{V^*}, r'_1)$.

The algorithm $\mathcal{D}$ plays the CCLA2 game against the public-key cryptosystem PKE according to the Definition 2.10 since $\mathcal{D}$ does not ask the decryption of the challenge ciphertext C. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 2}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 3}(\mathcal{A})| \leq \mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{PKE}}(\mathcal{D}). \tag{4}$$

Game 3 and Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session key value K is computed using the real PRF with a hidden key, or using a random function. The adversary $\mathcal{A}$ is given a K, such that it is computed using the PRF or randomly chosen from the session key space. The following describes $\mathcal{B}$'s procedure of answering queries.

- `Send`(U, V, s, m, f) query:
  - $U = U^*, V = V^*, s = s^*$:
    - If U^* is the initiator, upon receiving the second protocol message computes the session key $K \leftarrow \text{Oracle}^{\text{PRF}}(U^*||C_{U^*}||V^*||C_{V^*}) \oplus \text{PRF}(r_{V^*}, U^*||C_{U^*}||V^*||C_{V^*})$.
    - If U^* is the responder, upon receiving the first protocol message computes the session key $K \leftarrow \text{Oracle}^{\text{PRF}}(U^*||C_{U^*}||V^*||C_{V^*}) \oplus \text{PRF}(r_{V^*}, U^*||C_{U^*}||V^*||C_{V^*})$.
  - $U = U^*, V = V^*, s \neq s^*$: Executes the protocol normally.
  - $U = U^*, V \neq V^*$: Executes the protocol normally.
  - $U = V^*$:
    - If this is the initiator and it is the first message, then executes the protocol normally.
    - If this is the initiator and the second protocol message, or the responder:
      - If the same message that came to V^* in the test session has come as the incoming message, computes the session key using the $\text{Oracle}^{\text{PRF}}$.
      - Otherwise, executes the protocol normally.
  - $U, V \neq U^*$ or V^*: Executes the protocol normally.

    For all leakage queries, $\mathcal{B}$ can compute the leakage by its own, because $\mathcal{B}$ knows all the secret keys.

- `SessionKeyReveal`(U, V, s) query: `SessionKeyReveal` query is not allowed to the target session or its partner. $\mathcal{B}$ can compute all the session keys by executing the protocol.
  - For sessions involving the principal V^*, and the incoming message to V^* is the same message which has come to V^* in the target session, $\mathcal{B}$ uses $\text{Oracle}^{\text{PRF}}$ to compute the session key.
  - For all other sessions, $\mathcal{B}$ computes the session key by using the PRF.

- `EphemeralKeyReveal`(U, V, s) query: $\mathcal{B}$ can answer all `EphemeralKeyReveal` queries, which are allowed by the freshness condition, because $\mathcal{B}$ has the ephemeral keys.

- `Corrupt`(U) query: Except for V^*, algorithm $\mathcal{B}$ can answer all other `Corrupt` queries. In this case we consider the situation in which the adversary is not allowed to corrupt the partner principal of the target session, so in fact, $\mathcal{B}$ can answer all legitimate `Corrupt` queries.

- `Test`(U, V, s) query: Answers with the K which is computed at the `Send` query when $U = U^*, V = V^*, s = s^*$.

If the oracle is using the real PRF with a hidden key, the simulation is identical to Game 3, whereas if the oracle is using a random function, the simulation constructed is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the PRF challenger is real or random. Hence,

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

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,

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

Combining the results above, we find,

$$\text{Adv}^{\lambda-\text{CAFL}}_{\pi 1,\text{Case 1.1}}(\mathcal{A}) \leq N_P^2 N_s^2 \big(\text{Adv}^{\text{CCLA2}}_{\text{PKE}}(\mathcal{D}) + \text{Adv}_{\text{PRF}}(\mathcal{B})\big). \tag{7}$$

Case 1.2: Adversary corrupts the peer of the test session, but does not corrupt the owner.

In this case we consider the situation that $\mathcal{A}$ corrupts the partner of the test session but not the owner. The proof structure and games are similar to the previous case. The differences in this case is that the algorithm $\mathcal{D}$ uses the public key of the CCLA2 challenger as the public key of the protocol principal U^* (difference between Game 2 and Game 3), and Oracle$^{\mathrm{PRF}}$ is used when the incoming message to U^* in the test session is used as the incoming message to U^* in any other sessions (Game 3 and Game 4 analysis). We find,

$$\mathrm{Adv}^{\lambda-\mathrm{CAFL}}_{\pi 1,\mathrm{Case\ 1.2}}(\mathcal{A}) \leq N_P^2 N_s^2 \big(\mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{PKE}}(\mathcal{D}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B})\big). \tag{8}$$

Case 1.3: Adversary corrupts neither the owner nor the partner of the test session

In this case we consider the situation that $\mathcal{A}$ corrupts neither the owner nor the partner of the test session. So $\mathcal{D}$ can set the public key of the CCLA2 challenger as the public key of either U^* or V^*. The proof structure and games are similar to the previous case. We consider two sub cases under this case as follows:

- (a) Adversary does not ask `EphemeralKeyReveal`(V^*, U^*, t^*): simulation and analysis of this case is similar to the Case 1.2, because here $\mathcal{D}$ can set the public key of the CCLA2 challenger as the public key of the protocol principal U^* and proceed with the simulation as in the Case 1.2 (only difference is that here the adversary does not corrupt the partner principal of the test session, as in the Case 1.2, but rest of the simulation is same as in the Case 1.2). Thus,

$$\mathrm{Adv}^{\lambda-\mathrm{CAFL}}_{\pi 1,\mathrm{Case\ 1.3.a}}(\mathcal{A}) \leq N_P^2 N_s^2 \big(\mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{PKE}}(\mathcal{D}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B})\big). \tag{9}$$

- (b) Adversary does not ask `EphemeralKeyReveal`(U^*, V^*, s^*): simulation and analysis of this case is similar to the Case 1.1, because here $\mathcal{D}$ can set the public key of the CCLA2 challenger as the public key of the protocol principal V^* and proceed with the simulation as in the Case 1.1 (only difference is that here the adversary does not corrupt the owner of the test session, as in the Case 1.1, but rest of the simulation is same as in the Case 1.1). Thus,

$$\mathrm{Adv}^{\lambda-\mathrm{CAFL}}_{\pi 1,\mathrm{Case\ 1.3.b}}(\mathcal{A}) \leq N_P^2 N_s^2 \big(\mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{PKE}}(\mathcal{D}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B})\big). \tag{10}$$

Case 2: Partner to the test session does not exist

Game 1. This game is the original game. When the `Test` query is asked, the Game 1 challenger chooses 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.

Game 2. Same as Game 1 with the following exception: before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \leftarrow \{U_1, \ldots, U_{N_P}\}$ are chosen and a random number $s^* \leftarrow \{1, \ldots, N_s\}$ is 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*. If the test session is *not* the session $\Pi^{s^*}_{U^*,V^*}$, the Game 2 challenger aborts the game.

Game 3. Same as Game 2 with the following exception: the Game 3 challenger chooses a random value $r' \xleftarrow{\$} \{0,1\}^k$.

- If the test session is on the initiator, the challenger computes the session key in the test session $K \leftarrow \mathrm{PRF}(r', U^*||C_{U^*}||V^*||C_{V^*}) \oplus \mathrm{PRF}(r_{V^*}, U^*||C_{U^*}||V^*||C_{V^*})$.
- If the test session is on the responder, the challenger computes the session key in the test session $K \leftarrow \mathrm{PRF}(r_{V^*}, V^*||C_{V^*}||U^*||C_{U^*}) \oplus \mathrm{PRF}(r', V^*||C_{V^*}||U^*||C_{U^*})$.

The session key is computed in the same way in the partner to the test session.

Game 4. Same as Game 3 with the following exception: In the `Test` query, in the target session the Game 4 challenger randomly chooses $K \xleftarrow{\$} \{0,1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the `Test` query. In sessions at V^* which has a same incoming message to V^* as in the target session, the session key is randomly chosen as $K \xleftarrow{\$} \{0,1\}^k$.

Differences between games. Game 1 is the original game. Hence,

$$\mathrm{Adv}_{\text{Game 1}}(\mathcal{A}) = \mathrm{Adv}^{\lambda-\text{CAFL}}_{\pi 1,\text{Case 2}}(\mathcal{A}). \tag{11}$$

Game 1 and Game 2. The probability of Game 2 to be halted due to incorrect choice of the test session is $1 - \frac{1}{N_P^2 N_s}$. Unless the incorrect choice happens, Game 2 is identical to Game 1. Hence,

$$\mathrm{Adv}_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s}\mathrm{Adv}_{\text{Game 1}}(\mathcal{A}). \tag{12}$$

Game 2 and Game 3. We introduce an algorithm $\mathcal{D}$ which 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 algorithm $\mathcal{D}$ uses the public key of the CCLA2 challenger as the public key of the protocol principal V^* and generates public/secret key pairs for all other 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, $r'_0, r'_1 \leftarrow \{0,1\}^k$ and passes them to the CCLA2 challenger. From the CCLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C such that $C \leftarrow \mathrm{Enc}(pk_{V^*}, r')$ where $r' = r'_0$ or $r' = r'_1$. The following describes the procedure of answering queries.

- `Send`(U, V, s, m, f) query:
  - $U = U^*$, $V = V^*$ and $s = s^*$:
    - If U^* is the initiator, $\mathcal{D}$ sends the ciphertext C to $\mathcal{A}$ as the first message of the test session. Upon receiving the second protocol message computes the session key $K \leftarrow \mathrm{PRF}(r'_1, U^* \| C_{U^*} \| V^* \| C_{V^*}) \oplus \mathrm{PRF}(r_{V^*}, U^* \| C_{U^*} \| V^* \| C_{V^*})$.
    - If U^* is the responder, upon receiving the first protocol message sends C to $\mathcal{A}$, and computes the session key $K \leftarrow \mathrm{PRF}(r'_1, V^* \| C_{V^*} \| U^* \| C_{U^*}) \oplus \mathrm{PRF}(r_{V^*}, V^* \| C_{V^*} \| U^* \| C_{U^*})$.
  - $U = U^*, V = V^*, s \neq s^*$: Executes protocol normally.
  - $U = U^*, V \neq V^*$: Executes the protocol normally.
  - $U = V^*$:
    - If this is the initiator and it is the first message, then executes the protocol normally.
    - If this is the initiator and the second protocol message, or the responder:
      - If C has come as the incoming message uses r'_1 as the decryption of the incoming message. To obtain the corresponding leakage, $\mathcal{D}$ first encrypts r'_1 using pk_{V^*}, gets that ciphertext and access the leakage oracle of CCLA2 challenger with the ciphertext of r'_1.
      - Else uses the decryption oracle to decrypt incoming messages.
  - $U, V \neq U^*$ or V^*: Executes the protocol normally.

    For all other leakage queries $f(sk_{V^*})$, $\mathcal{D}$ obtains the leakage accessing the leakage oracle, whereas for all other leakage $\mathcal{D}$ can compute the leakage by its own because $\mathcal{D}$ knows all other secret keys.

- `SessionKeyReveal`(U, V, s) query: `SessionKeyReveal` query is not allowed to the target session or its partner.
  - For sessions involving the principal V^*, and the incoming message to V^* is the same message which has come to V^* in the target session, uses r'_1 as the decryption.
  - For other sessions involving the principal V^*, $\mathcal{D}$ can decrypt the incoming messages to V^* by using the decryption oracle.
  - Otherwise, $\mathcal{D}$ can decrypt all the other incoming messages to protocol principals by its own.

  Then compute the session key using the PRF.

- `EphemeralKeyReveal`(U, V, s) query: For all `EphemeralKeyReveal` queries allowed in the freshness condition, $\mathcal{D}$ can answer correctly, because $\mathcal{D}$ has the ephemeral keys.

- `Corrupt`(U) query: Except for V^*, algorithm $\mathcal{D}$ can answer all other `Corrupt` queries. In this case we consider the situation in which the adversary is not allowed to corrupt the partner principal of the target session, so in fact, $\mathcal{D}$ can answer all `Corrupt` queries.

- `Test`(U, V, s) query: To compute the answer to the `Test`(U^*, V^*, s^*) query, the algorithm $\mathcal{D}$ uses r'_1 as the decryption of the ciphertext C and computes the session key using the r'_1 value as the ephemeral key of the principal U^*.

If r'_1 is the decryption of C coming to, V^* in the test session, the simulation constructed by $\mathcal{D}$ is identical to Game 2 whereas if r'_0 is the decryption of C, the simulation constructed by $\mathcal{D}$ is identical to Game 3. If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{D}$ can distinguish whether $C \leftarrow \mathrm{Enc}(pk_{V^*}, r'_0)$ or $C \leftarrow \mathrm{Enc}(pk_{V^*}, r'_1)$.

The algorithm $\mathcal{D}$ plays the CCLA2 game against the public-key cryptosystem PKE according to the Definition 2.10 since $\mathcal{D}$ does not ask the decryption of the challenge ciphertext C. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 2}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 3}(\mathcal{A})| \leq \mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{PKE}}(\mathcal{D}). \tag{13}$$

Game 3 and Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the session key value K is computed K is computed using the real PRF with a hidden key, or using a random function. The adversary $\mathcal{A}$ is given a K, such that it is computed using the PRF or randomly chosen from the session key space. The following describes $\mathcal{B}$'s procedure of answering queries.

- `Send`(U, V, s, m, f) query:
  - $U = U^*$, $V = V^*$ and $s = s^*$:
    - If U^* is the initiator, upon receiving the second protocol message computes the session key $K \leftarrow \mathrm{Oracle}^{\mathrm{PRF}}(U^*||C_{U^*}||V^*||C_{V^*}) \oplus \mathrm{PRF}(r_{V^*}, U^*||C_{U^*}||V^*||C_{V^*})$.
    - If U^* is the responder, upon receiving the first protocol message computes the session key $K \leftarrow \mathrm{Oracle}^{\mathrm{PRF}}(U^*||C_{U^*}||V^*||C_{V^*}) \oplus \mathrm{PRF}(r_{V^*}, U^*||C_{U^*}||V^*||C_{V^*})$.
  - $U = U^*, V = V^*, s \neq s^*$: Executes protocol normally.
  - $U = U^*, V \neq V^*$: Executes the protocol normally.
  - $U = V^*$:
    - If this is the initiator and it is the first message, then executes the protocol normally.
    - If this is the initiator and the second protocol message, or the responder:
      - If the same message that came to V^* in the test session has come as the incoming message, computes the session key using the $\mathrm{Oracle}^{\mathrm{PRF}}$.
      - Otherwise, executes the protocol normally.
  - $U, V \neq U^*$ or V^*: Executes the protocol normally.

  For all leakage queries, $\mathcal{B}$ can compute the leakage by its own, because $\mathcal{B}$ knows all the secret keys.

- `SessionKeyReveal`(U, V, s) query: `SessionKeyReveal` query is not allowed to the target session or its partner.
  - For sessions involving the principal V^*, and the incoming message to V^* is the same message which has come to V^* in the target session, $\mathcal{B}$ uses $\mathrm{Oracle}^{\mathrm{PRF}}$ to compute the session key.
  - For all other sessions, $\mathcal{B}$ computes the session key by using the PRF.

- `EphemeralKeyReveal`(U, V, s) query: $\mathcal{B}$ can answer all `EphemeralKeyReveal` queries $\mathcal{B}$ which are allowed in the freshness condition, because $\mathcal{B}$ has the ephemeral keys.

- `Corrupt`(U) query: Except for V^*, algorithm $\mathcal{B}$ can answer all other `Corrupt` queries. In this case we consider the situation in which the adversary is not allowed to corrupt the partner principal of the target session, so in fact, $\mathcal{B}$ can answer all `Corrupt` queries.

- `Test`(U, V, s) query: Answers with the K computed in `Send` query when $U = U^*$, $V = V^*$ and $s = s^*$.

If the oracle is using the real PRF with a hidden key, the simulation is identical to Game 3, whereas if the oracle is using a random function, the simulation constructed is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the PRF challenger is real or random. Hence,

$$|\mathrm{Adv}_{\text{Game }3}(\mathcal{A}) - \mathrm{Adv}_{\text{Game }4}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B}). \tag{14}$$

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,

$$\mathrm{Adv}_{\text{Game }4}(\mathcal{A}) = 0. \tag{15}$$

Combining the results above, we find,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\mathrm{CAFL}}_{\pi1,\text{Case }2}(\mathcal{A}) \leq N_P^2 N_s \big(\mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{PKE}}(\mathcal{D}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B})\big). \tag{16}$$

Combine Case 1 and Case 2

According to the analysis we can see the adversary $\mathcal{A}$'s advantage of winning against the protocol $\pi1$ challenger is

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\mathrm{CAFL}}_{\pi1}(\mathcal{A}) \leq N_P^2 N_s^2 \big(\mathrm{Adv}^{\mathrm{CCLA2}}_{\mathrm{PKE}}(\mathcal{D}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{B})\big).$$

□

B Proof of the Theorem 5.1

Proof. The proof is split into two main cases: when the partner to the test session exists, and when it does not.

Case 1: A partner session to the test session exists

In this case, the adversary is allowed to corrupt both principals or reveal ephemeral keys from both sessions. We assume that the adversary $\mathcal{A}$ can win the $\boldsymbol{\lambda} - \mathrm{w}(\cdot)$AFL-eCK challenge against the protocol π with advantage $\mathrm{Adv}^{\boldsymbol{\lambda}-\mathrm{w}(\cdot)\mathrm{AFL\text{-}eCK}}_{\pi}(\mathcal{A})$. We split this case into four sub cases as follows:

1. Adversary corrupts both the owner and partner principals to the test session.
2. Adversary corrupts neither owner or nor partner principal to the test session.
3. Adversary corrupts the owner to the test session, but does not corrupt the partner to the test session.
4. Adversary corrupts the partner to the test session, but does not corrupt the owner to the test session.

Case 1.1: Adversary corrupts both the owner and partner principals to the test session

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

$$\mathrm{Adv}_{\text{Game }1}(\mathcal{A}) = \mathrm{Adv}^{\boldsymbol{\lambda}-\mathrm{w}(\cdot)\mathrm{AFL\text{-}eCK}}_{\pi,\text{Case }1.1}(\mathcal{A}). \tag{17}$$

Game 2. Abort the simulation if there exists two sessions outputting the same ephemeral public keys (Same $X = g^x$ values). Since the ephemeral keys are coming from $\mathbb{Z}_q^*$, the total number of ephemeral keys are q. Total number of session in the simulation is ${N_P}^2{N_s}^2$, because N_P is the number of protocol principals and each protocol principal owns N_s number of sessions. Hence,

$$|\mathrm{Adv}_{\text{Game }1}(\mathcal{A}) - \mathrm{Adv}_{\text{Game }2}(\mathcal{A})| \leq \frac{{N_P}^2{N_s}^2}{q}. \tag{18}$$

Game 3. Before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \xleftarrow{\$} \{U_1, ..., U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \xleftarrow{\$} \{1, ...N_s\}$ 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 3 challenger aborts the game. Unless the incorrect choice happens, Game 3 is identical to Game 2. Hence,

$$\mathrm{Adv}_{\mathrm{Game\ 3}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2}\mathrm{Adv}_{\mathrm{Game\ 2}}(\mathcal{A}). \tag{19}$$

Game 4. Game 4 challenger randomly chooses $z \xleftarrow{\$} \mathbb{Z}_q^*$ and computes the session key of the target session and its partner session, using the KDF and the PRF as $ms \leftarrow \mathrm{KDF}(g^z, \perp, k, \perp)$ and $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$, when U^* is the initiator, or $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$, when U^* is the responder.

We construct an algorithm $\mathcal{C}$ against the DDH challenge, using the adversary $\mathcal{A}$. The DDH challenger sends values $(X = g^x, Y = g^y, Z = g^z)$ such that either $z = xy$ or $z \xleftarrow{\$} \mathbb{Z}_q^*$, as the inputs to the algorithm $\mathcal{C}$. $\mathcal{C}$ uses the value X as the ephemeral public key of U^* and Y as the ephemeral public key of V^* in the test session, and computes the session key using Z as the input to the KDF in the session key derivation process.

If $\mathcal{C}$'s input is a Diffie-Hellman triple, the simulation constructed by $\mathcal{C}$ is identical to Game 3, otherwise it is identical to Game 4. If $\mathcal{A}$ can distinguish whether $g^z = g^{xy}$ or not, then $\mathcal{C}$ can answer the DDH challenge. Note that `EphemeralKeyReveal`(U^*, V^*, s^*) or `EphemeralKeyReveal`(V^*, U^*, t^*) is prohibited since the adversary corrupts both the owner and the partner to the test session. $\mathcal{C}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game\ 3}}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game\ 4}}(\mathcal{A})| \leq \mathrm{Adv}^{\mathrm{DDH}}_{q,g}(\mathcal{C}). \tag{20}$$

Game 5. The Game 5 challenger randomly chooses $ms \xleftarrow{\$} \{0,1\}^k$ and computes the session key of the target session and its partner session, using the PRF as $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$, when U^* is the initiator or $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$, when U^* is the responder.

We construct an algorithm $\mathcal{B}$ against a KDF challenger, using the adversary $\mathcal{A}$. The KDF challenger sends a ms value which is either generated using the KDF or randomly chosen. $\mathcal{B}$ uses the received ms value to compute the session key of the target session using the PRF.

If ms is computed using the KDF, simulation constructed by $\mathcal{B}$ is identical to Game 4, otherwise it is identical to Game 5. If $\mathcal{A}$ can distinguish the difference between Game 4 and Game 5, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the ms value is computed using KDF or randomly chosen. $\mathcal{B}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game\ 4}}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game\ 5}}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}). \tag{21}$$

Game 6. The Game 6 challenger randomly chooses $K \xleftarrow{\$} \{0,1\}^k$ as session key of the target session and its partner session.

We construct an algorithm $\mathcal{J}$ against an Oracle$^{\mathrm{PRF}}$, using the adversary $\mathcal{A}$. The Oracle$^{\mathrm{PRF}}$ sends a K value which is either generated using the PRF with a hidden key, or a random function. $\mathcal{J}$ uses the received K as the session key of the target session.

If K is generated using the PRF with a hidden key, simulation constructed by $\mathcal{J}$ is identical to Game 5, otherwise it is identical to Game 6. If $\mathcal{A}$ can distinguish the difference between Game 5 and Game 6, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{J}$, which is used to distinguish whether the Oracle$^{\mathrm{PRF}}$ is real or a random function. $\mathcal{J}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game\ 5}}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game\ 6}}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}). \tag{22}$$

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,

$$\mathrm{Adv}_{\mathrm{Game\ 6}}(\mathcal{A}) = 0. \tag{23}$$

We find,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\mathrm{w}(\cdot)\mathrm{AFL\text{-}eCK}}_{\pi,\mathrm{Case\ 1.1}}(\mathcal{A}) \leq N_P^2 N_s^2 \Big[\big(\mathrm{Adv}^{\mathrm{DDH}}_{q,g}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) \big) + \frac{1}{q} \Big].$$

Case 1.2: Adversary corrupts neither owner or nor partner principal to the test session

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

$$\mathrm{Adv}_{\mathrm{Game\ 1}}(\mathcal{A}) = \mathrm{Adv}^{\boldsymbol{\lambda}-\mathrm{w}(\cdot)\mathrm{AFL\text{-}eCK}}_{\pi,\mathrm{Case\ 1.2}}(\mathcal{A}). \tag{24}$$

Game 2. Abort the simulation if there exists two sessions outputting the same ephemeral public keys (Same $X = g^x$ values). Since the ephemeral keys are coming from $\mathbb{Z}_q^*$, the total number of ephemeral keys are q. Total number of session in the simulation is ${N_P}^2{N_s}^2$, because N_P is the number of protocol principals and each protocol principal owns N_s number of sessions. Hence,

$$|\mathrm{Adv}_{\mathrm{Game\ 1}}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game\ 2}}(\mathcal{A})| \leq \frac{{N_P}^2{N_s}^2}{q}. \tag{25}$$

Game 3. Before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \xleftarrow{\$} \{U_1, ..., U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \xleftarrow{\$} \{1, ...N_s\}$ 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 3 challenger aborts the game. Unless the incorrect choice happens, Game 3 is identical to Game 2. Hence,

$$\mathrm{Adv}_{\mathrm{Game\ 3}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2}\mathrm{Adv}_{\mathrm{Game\ 2}}(\mathcal{A}). \tag{26}$$

Game 4. Game 4 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r_{U^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and computes the ephemeral key $r_{U^*} \xleftarrow{\$} \mathrm{Enc}(p_{U^*}, \widetilde{r_{U^*}})$, in the target session.

We introduce an algorithm $\mathcal{F}$ which is constructed using the adversary $\mathcal{A}$, against the ϵ-pair-generation indistinguishability challenger (ϵ-PG). $\mathcal{F}$ receives a pair $(r_{U^*}, \widetilde{r_{U^*}})$ such that $\widetilde{r_{U^*}} = \mathrm{Dec}(s_{U^*}, r_{U^*})$. $\mathcal{F}$ uses r_{U^*} as the ephemeral key of U^* and $\widetilde{r_{U^*}}$ as the pseudo-ephemeral key of U^* in the target session.

If a random ephemeral key $r_{U^*} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the pseudo-ephemeral value $\widetilde{r_{U^*}} \leftarrow \mathrm{Dec}(s_{U^*}, r_{U^*})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 3. Otherwise if a random pseudo-ephemeral value $\widetilde{r_{U^*}} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the ephemeral key $r_{U^*} \xleftarrow{\$} \mathrm{Enc}(p_{U^*}, \widetilde{r_{U^*}})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{F}$ can distinguish whether a message/ciphertext pair (m, c) belongs to the distribution D_1 or D_2 (ϵ-pair-generation indistinguishability challenge). $\mathcal{F}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game\ 3}}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game\ 4}}(\mathcal{A})| \leq \epsilon. \tag{27}$$

Game 5. Game 5 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r'_{U^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and uses it as the pseudo ephemeral value of U^* in the target session.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$, against the CPLA2 challenger. The algorithm $\mathcal{D}$ uses the public-key of the CPLA2 challenger as the public key of the protocol principal U^* and generates public/secret key pairs for all other protocol principals. $\mathcal{D}$ generates signing/verification key pairs for every protocol principal. $\mathcal{D}$ picks two random strings, $r_0, r_1 \xleftarrow{\$} \{0,1\}^k$ and passes them to the CPLA2 challenger. From the CPLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C_1 such that $C_1 \xleftarrow{\$} \text{Enc}(p_{U^*}, r_\theta)$ where $r_\theta = r_0$ or $r_\theta = r_1$. The following describes the procedure of answering queries:

- Send$(U, V, s, m, \mathbf{f})$ query: When $U = U^*$, $V = V^*$ and $s = s^*$, $\mathcal{D}$ takes r_1 as $\widetilde{r'_{U^*}}$, computes $g^{\widetilde{r'_{U^*}}}$ and computes its signature using the signing key sk_{U^*}. Then $\mathcal{D}$ creates the protocol message and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(s_{U^*})$, where the leakage $\mathbf{f}(s_{U^*})$ is obtained by accessing the leakage oracle of the CPLA2 challenger.

  For all other Send queries, $\mathcal{D}$ can execute the protocol normally, because $\mathcal{D}$ has all the public keys and can compute protocol messages accordingly. Except U^* $\mathcal{D}$ can compute the leakage by its own, and for U^* $\mathcal{D}$ accesses the leakage oracle to obtain the leakage.

- SessionKeyReveal(U, V, s) query: $\mathcal{D}$ will abort if SessionKeyReveal(U^*, V^*, s^*) or SessionKeyReveal(V^*, U^*, t^*) query is asked. $\mathcal{D}$ can easily compute the answers using the corresponding psuedo-ephemeral keys for other SessionKeyReveal queries.

- EphemeralKeyReveal(U, V, s) query: For the EphemeralKeyReveal(U^*, V^*, s^*) query, $\mathcal{D}$ uses C_1 as the answer. For all other EphemeralKeyReveal queries $\mathcal{D}$ will answer with the corresponding ephemeral-key which is computed by encrypting a pseudo-ephemeral value with the secret key of the corresponding principal.

- Corrupt(U) query: Except for U^* and V^*, algorithm $\mathcal{D}$ can answer all other Corrupt queries. In this case we consider the situation in which the adversary corrupts neither owner or nor partner principal to the test session, so these exceptions will not occur.

- Test(U, s) query: The algorithm $\mathcal{D}$ will abort the game if the adversary issues a Test query other than Test(U^*, s^*). To compute the answer to the Test(U^*, s^*) query, the algorithm $\mathcal{D}$ computes:

  - If U^* is the initiator, computes $ms \leftarrow \text{KDF}(X_{V^*}^{\widetilde{r'_{U^*}}}, \perp, k, \perp)$, $K \leftarrow \text{PRF}(ms, U^* || X_{U^*} || \sigma_{U^*} || V^* || X_{V^*} || \sigma_{V^*})$ where $\widetilde{r'_{U^*}} = r_1$.
  - If U^* is the responder, computes $ms \leftarrow \text{KDF}(X_{V^*}^{\widetilde{r'_{U^*}}}, \perp, k, \perp)$, $K \leftarrow \text{PRF}(ms, V^* || X_{V^*} || \sigma_{V^*} || U^* || X_{U^*} || \sigma_{U^*})$.

  Then using K answers the Test query.

If $\theta = 1$, then r_1 is the decryption of C_1 and the simulation constructed by $\mathcal{D}$ is identical to Game 4 whereas if $\theta = 0$, then r_0 is the decryption of C_1 and the simulation constructed by $\mathcal{D}$ is identical to Game 5. If $\mathcal{A}$ can distinguish the difference between Game 4 and Game 5, then $\mathcal{D}$ can be used against a CPLA2 challenger. Hence,

$$|\text{Adv}_{\text{Game 4}}(\mathcal{A}) - \text{Adv}_{\text{Game 5}}(\mathcal{A})| \leq \text{Adv}_{\text{PKE}}^{\text{CPLA2}}(\mathcal{D}). \tag{28}$$

Game 6. Game 6 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and computes the ephemeral key $r_{V^*} \xleftarrow{\$} \text{Enc}(p_{V^*}, \widetilde{r_{V^*}})$ in the partner to the target session.

We introduce an algorithm $\mathcal{F}$ which is constructed using the adversary $\mathcal{A}$, against the ϵ-pair-generation indistinguishability challenger (ϵ-PG). $\mathcal{F}$ receives a pair $(r_{V^*}, \widetilde{r_{V^*}})$ such that $\widetilde{r_{V^*}} = \text{Dec}(s_{V^*}, r_{V^*})$. $\mathcal{F}$ uses r_{V^*} as the ephemeral key of V^* and $\widetilde{r_{V^*}}$ as the pseudo-ephemeral key of V^*.

If a random ephemeral key $r_{V^*} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the pseudo-ephemeral value $\widetilde{r_{V^*}} \leftarrow \text{Dec}(s_{V^*}, r_{V^*})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 5. Otherwise if a random pseudo-ephemeral value $\widetilde{r_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the ephemeral key $r_{V^*} \xleftarrow{\$} \text{Enc}(p_{V^*}, \widetilde{r_{V^*}})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 6. If $\mathcal{A}$ can distinguish the difference between Game 5 and Game 6, then $\mathcal{F}$ can distinguish whether a message/ciphertext pair (m, c) belongs to the distribution D_1 or D_2 (ϵ-pair-generation indistinguishability challenge). $\mathcal{F}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 5}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 6}(\mathcal{A})| \leq \epsilon. \tag{29}$$

Game 7. Game 7 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r'_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and uses it as the pseudo ephemeral value of V^* in the partner to the target session.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$, against the CPLA2 challenger. The algorithm $\mathcal{D}$ uses the public-key of the CPLA2 challenger as the public key of the protocol principal V^* and generates public/secret key pairs for all other protocol principals. $\mathcal{D}$ generates signing/verification key pairs for every protocol principal. $\mathcal{D}$ picks two random strings, $r'_0, r'_1 \xleftarrow{\$} \{0,1\}^k$ and passes them to the CPLA2 challenger. From the CPLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C_2 such that $C_2 \xleftarrow{\$} \mathrm{Enc}(p_{V^*}, r'_\theta)$ where $r'_\theta = r'_0$ or $r'_\theta = r'_1$. The following describes the procedure of answering queries:

- `Send`$(U, V, s, m, \mathbf{f})$ query: When $U = V^*$, $V = U^*$ and $s = t^*$, $\mathcal{D}$ takes r'_1 as $\widetilde{r'_{V^*}}$, computes $g^{\widetilde{r'_{V^*}}}$ and computes its signature using the signing key sk_{V^*}. Then $\mathcal{D}$ creates the protocol message and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(s_{V^*})$, where the leakage $\mathbf{f}(s_{V^*})$ is obtained by accessing the leakage oracle of the CPLA2 challenger.

  For all other `Send` queries, $\mathcal{D}$ can execute the protocol normally, because $\mathcal{D}$ has all the public keys and can compute protocol messages accordingly. Except V^* $\mathcal{D}$ can compute the leakage by its own, and for V^* $\mathcal{D}$ accesses the leakage oracle to obtain the leakage.

- `SessionKeyReveal`(U, V, s) query: $\mathcal{D}$ will abort if `SessionKeyReveal`(U^*, V^*, s^*) or `SessionKeyReveal`(V^*, U^*, t^*) query is asked. $\mathcal{D}$ can easily compute the answers using the corresponding psuedo-ephemeral keys for other `SessionKeyReveal` queries.

- `EphemeralKeyReveal`(U, V, s) query: For the `EphemeralKeyReveal`(V^*, U^*, t^*) query, $\mathcal{D}$ uses C_2 as the answer. For all other `EphemeralKeyReveal` queries $\mathcal{D}$ will answer with the corresponding ephemeral-key which is computed by encrypting a pseudo-ephemeral value with the secret key of the corresponding principal.

- `Corrupt`(U) query: Except for U^* and V^*, algorithm $\mathcal{D}$ can answer all other `Corrupt` queries. In this case we consider the situation in which the adversary corrupts neither owner or nor partner principal to the test session, so these exceptions will not occur.

- `Test`(U, s) query: The algorithm $\mathcal{D}$ will abort the game if the adversary issues a `Test` query other than `Test`(U^*, s^*). To compute the answer to the `Test`(U^*, s^*) query, the algorithm $\mathcal{D}$ computes:

  - If U^* is the initiator, computes $ms \leftarrow \mathrm{KDF}(X_{U^*}^{\widetilde{r'_{V^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$.
  - If U^* is the responder, $ms \leftarrow \mathrm{KDF}(X_{U^*}^{\widetilde{r'_{V^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$.

  Then using K answers the `Test` query.

If $\theta = 1$, then r'_1 is the decryption of C_2 and the simulation constructed by $\mathcal{D}$ is identical to Game 6 whereas if $\theta = 0$, then r'_0 is the decryption of C_2 and the simulation constructed by $\mathcal{D}$ is identical to Game 7. If $\mathcal{A}$ can distinguish the difference between Game 6 and Game 7, then $\mathcal{D}$ can be used against a CPLA2 challenger. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 6}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 7}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D}). \tag{30}$$

Game 8. Game 8 challenger randomly chooses $z \xleftarrow{\$} \mathbb{Z}_q^*$ and computes the session key of the target session and its partner session, using the KDF and the PRF as $ms \leftarrow \mathrm{KDF}(g^z, \perp, k, \perp)$ and $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$, when U^* is the initiator or $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$, when U^* is the responder.

We construct an algorithm $\mathcal{C}$ against the DDH challenge, using the adversary $\mathcal{A}$. The DDH challenger sends values $(X = g^x, Y = g^y, Z = g^z)$ such that either $z = xy$ or $z \xleftarrow{\$} \mathbb{Z}_q^*$, as the inputs to the algorithm

$\mathcal{C}$. $\mathcal{C}$ uses the value X as the ephemeral public key of U^* and Y as the ephemeral public key of V^* in the test session, and computes the session key using Z as the input to the KDF in the session key derivation process.

If $\mathcal{C}$'s input is a Diffie-Hellman triple, the simulation constructed by $\mathcal{C}$ is identical to Game 7, otherwise it is identical to Game 8. If $\mathcal{A}$ can distinguish whether $g^z = g^{xy}$ or not, then $\mathcal{C}$ can answer the DDH challenge. $\mathcal{C}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. In this case `EphemeralKeyReveal` query is not allowed to the target session and its partner. Hence,

$$|\mathrm{Adv}_{\text{Game }7}(\mathcal{A}) - \mathrm{Adv}_{\text{Game }8}(\mathcal{A})| \leq \mathrm{Adv}_{q,g}^{\mathrm{DDH}}(\mathcal{C}). \tag{31}$$

Game 9. The Game 9 challenger randomly chooses $ms \xleftarrow{\$} \{0,1\}^k$ and computes the session key of the target session and its partner session, using the PRF as $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$, when U^* is the initiator or $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$, when U^* is the responder.

We construct an algorithm $\mathcal{B}$ against a KDF challenger, using the adversary $\mathcal{A}$. The KDF challenger sends a ms value which is either generated using the KDF or randomly chosen. $\mathcal{B}$ uses the received ms value to compute the session key of the target session using the PRF.

If ms is computed using the KDF, simulation constructed by $\mathcal{B}$ is identical to Game 8, otherwise it is identical to Game 9. If $\mathcal{A}$ can distinguish the difference between Game 8 and Game 9, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the ms value is computed using KDF or randomly chosen. $\mathcal{B}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\text{Game }8}(\mathcal{A}) - \mathrm{Adv}_{\text{Game }9}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}). \tag{32}$$

Game 10. The Game 10 challenger randomly chooses $K \xleftarrow{\$} \{0,1\}^k$ as session key of the target session and its partner session.

We construct an algorithm $\mathcal{J}$ against an Oracle$^{\mathrm{PRF}}$, using the adversary $\mathcal{A}$. The Oracle$^{\mathrm{PRF}}$ sends a K value which is either generated using the PRF with a hidden key, or a random function. $\mathcal{J}$ uses the received K as the session key of the target session.

If K is generated using the PRF with a hidden key, simulation constructed by $\mathcal{J}$ is identical to Game 9, otherwise it is identical to Game 10. If $\mathcal{A}$ can distinguish the difference between Game 9 and Game 10, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{J}$, which is used to distinguish whether the Oracle$^{\mathrm{PRF}}$ is real or a random function. $\mathcal{J}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\text{Game }9}(\mathcal{A}) - \mathrm{Adv}_{\text{Game }10}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}). \tag{33}$$

Semantic security of the session key in Game 10. 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 10. Hence,

$$\mathrm{Adv}_{\text{Game }10}(\mathcal{A}) = 0. \tag{34}$$

We find,

$$\mathrm{Adv}_{\pi,\text{Case 1.2}}^{\boldsymbol{\lambda}-\mathrm{w}(\cdot)\text{AFL-eCK}}(\mathcal{A}) \leq N_P^2 N_s^2 \Big[\big(\mathrm{Adv}_{q,g}^{\mathrm{DDH}}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) + 2\mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D}) + 2\epsilon\big) + \frac{1}{q}\Big].$$

Case 1.3: Adversary corrupts the partner, but not the owner to the test session

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

$$\mathrm{Adv}_{\text{Game }1}(\mathcal{A}) = \mathrm{Adv}_{\pi,\text{Case 1.3}}^{\boldsymbol{\lambda}-\mathrm{w}(\cdot)\text{AFL-eCK}}(\mathcal{A}). \tag{35}$$

Game 2. Abort the simulation if there exists two sessions outputting the same ephemeral public keys (Same $X = g^x$ values). Since the ephemeral keys are coming from $\mathbb{Z}_q^*$, the total number of ephemeral keys are q. Total number of session in the simulation is ${N_P}^2{N_s}^2$, because N_P is the number of protocol principals and each protocol principal owns N_s number of sessions. Hence,

$$|\mathrm{Adv}_{\mathrm{Game\ 1}}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game\ 2}}(\mathcal{A})| \leq \frac{{N_P}^2{N_s}^2}{q}. \tag{36}$$

Game 3. Before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \xleftarrow{\$} \{U_1, ..., U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \xleftarrow{\$} \{1, ...N_s\}$ 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 3 challenger aborts the game. Unless the incorrect choice happens, Game 3 is identical to Game 2. Hence,

$$\mathrm{Adv}_{\mathrm{Game\ 3}}(\mathcal{A}) = \frac{1}{{N_P}^2N_s^2}\mathrm{Adv}_{\mathrm{Game\ 2}}(\mathcal{A}). \tag{37}$$

Game 4. Game 4 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r_{U^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and computes the ephemeral key $r_{U^*} \xleftarrow{\$} \mathrm{Enc}(p_{U^*}, \widetilde{r_{U^*}})$ in the target session.

We introduce an algorithm $\mathcal{F}$ which is constructed using the adversary $\mathcal{A}$, against the ϵ-pair-generation indistinguishability challenger (ϵ-PG). $\mathcal{F}$ receives a pair $(r_{U^*}, \widetilde{r_{U^*}})$ such that $\widetilde{r_{U^*}} = \mathrm{Dec}(s_{U^*}, r_{U^*})$. $\mathcal{F}$ uses r_{U^*} as the ephemeral key of U^* and $\widetilde{r_{U^*}}$ as the pseudo-ephemeral key of U^* in the target session.

If a random ephemeral key $r_{U^*} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the pseudo-ephemeral value $\widetilde{r_{U^*}} \leftarrow \mathrm{Dec}(s_{U^*}, r_{U^*})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 3. Otherwise if a random pseudo-ephemeral value $\widetilde{r_{U^*}} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the ephemeral key $r_{U^*} \xleftarrow{\$} \mathrm{Enc}(p_{U^*}, \widetilde{r_{U^*}})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{F}$ can distinguish whether a message/ciphertext pair (m, c) belongs to the distribution D_1 or D_2 (ϵ-pair-generation indistinguishability challenge). $\mathcal{F}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game\ 3}}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game\ 4}}(\mathcal{A})| \leq \epsilon. \tag{38}$$

Game 5. Game 5 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r'_{U^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and uses it as the pseudo ephemeral value of U^* in the target session.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$, against the CPLA2 challenger. The algorithm $\mathcal{D}$ uses the public-key of the CPLA2 challenger as the public key of the protocol principal U^* and generates public/secret key pairs for all other protocol principals. $\mathcal{D}$ generates signing/verification key pairs for every protocol principal. $\mathcal{D}$ picks two random strings, $r_0, r_1 \xleftarrow{\$} \{0,1\}^k$ and passes them to the CPLA2 challenger. From the CPLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C_1 such that $C_1 \xleftarrow{\$} \mathrm{Enc}(p_{U^*}, r_\theta)$ where $r_\theta = r_0$ or $r_\theta = r_1$. The following describes the procedure of answering queries:

- $\mathtt{Send}(U, V, s, m, \mathbf{f})$ query: When $U = U^*$, $V = V^*$ and $s = s^*$, $\mathcal{D}$ takes r_1 as $\widetilde{r'_{U^*}}$, computes $g^{\widetilde{r'_{U^*}}}$ and computes its signature using the signing key sk_{U^*}. Then $\mathcal{D}$ creates the protocol message and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(s_{U^*})$, where the leakage $\mathbf{f}(s_{U^*})$ is obtained by accessing the leakage oracle of the CPLA2 challenger.

  For all other `Send` queries, $\mathcal{D}$ can execute the protocol normally, because $\mathcal{D}$ has all the public keys and can compute protocol messages accordingly. Except U^* $\mathcal{D}$ can compute the leakage by its own, and for U^* $\mathcal{D}$ accesses the leakage oracle to obtain the leakage.

- $\mathtt{SessionKeyReveal}(U, V, s)$ query: $\mathcal{D}$ will abort if $\mathtt{SessionKeyReveal}(U^*, V^*, s^*)$ or $\mathtt{SessionKeyReveal}(V^*, U^*, t^*)$ query is asked. $\mathcal{D}$ can easily compute the answers using the corresponding psuedo-ephemeral keys for other `SessionKeyReveal` queries.

- `EphemeralKeyReveal`(U, V, s) query: For the `EphemeralKeyReveal`(U^*, V^*, s^*) query, $\mathcal{D}$ uses C_1 as the answer. For all other `EphemeralKeyReveal` queries $\mathcal{D}$ will answer with the corresponding ephemeral-key which is computed by encrypting a pseudo-ephemeral value with the secret key of the corresponding principal.
- `Corrupt`(U) query: Except for U^*, algorithm $\mathcal{D}$ can answer all other `Corrupt` queries. In this case we consider the situation in which the adversary does not corrupt the partner to the test session, so these exceptions will not occur.
- `Test`(U, s) query: The algorithm $\mathcal{D}$ will abort the game if the adversary issues a `Test` query other than `Test`(U^*, s^*). To compute the answer to the `Test`(U^*, s^*) query, the algorithm $\mathcal{D}$ computes:
  - If U^* is the initiator, computes $ms \leftarrow \mathrm{KDF}(X_{V^*}^{\widetilde{r'_{U^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^* ||X_{V^*}||\sigma_{V^*})$.
  - If U^* is the responder, computes $ms \leftarrow \mathrm{KDF}(X_{V^*}^{\widetilde{r'_{U^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*} ||U^*||X_{U^*}||\sigma_{U^*})$.

  Then using K answers the `Test` query.

If $\theta = 1$, then r_1 is the decryption of C_1 and the simulation constructed by $\mathcal{D}$ is identical to Game 4 whereas if $\theta = 0$, then r_0 is the decryption of C_1 and the simulation constructed by $\mathcal{D}$ is identical to Game 5. If $\mathcal{A}$ can distinguish the difference between Game 4 and Game 5, then $\mathcal{D}$ can be used against a CPLA2 challenger. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 4}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 5}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D}). \tag{39}$$

Game 6. Game 6 challenger randomly chooses $z \xleftarrow{\$} \mathbb{Z}_q^*$ and computes the session key of the target session and its partner session, using the KDF and the PRF as $ms \leftarrow \mathrm{KDF}(g^z, \perp, k, \perp)$ and $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$, when U^* is the initiator, or $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$, when U^* is the responder.

We construct an algorithm $\mathcal{C}$ against the DDH challenge, using the adversary $\mathcal{A}$. The DDH challenger sends values $(X = g^x, Y = g^y, Z = g^z)$ such that either $z = xy$ or $z \xleftarrow{\$} \mathbb{Z}_q^*$, as the inputs to the algorithm $\mathcal{C}$. $\mathcal{C}$ uses the value X as the ephemeral public key of U^* and Y as the ephemeral public key of V^* in the test session, and computes the session key using Z as the input to the KDF in the session key derivation process.

If $\mathcal{C}$'s input is a Diffie-Hellman triple, the simulation constructed by $\mathcal{C}$ is identical to Game 5, otherwise it is identical to Game 6. If $\mathcal{A}$ can distinguish whether $g^z = g^{xy}$ or not, then $\mathcal{C}$ can answer the DDH challenge. $\mathcal{C}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 5}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 6}(\mathcal{A})| \leq \mathrm{Adv}_{q,g}^{\mathrm{DDH}}(\mathcal{C}). \tag{40}$$

Game 7. The Game 7 challenger randomly chooses $ms \xleftarrow{\$} \{0,1\}^k$ and computes the session key of the target session and its partner session, using the PRF as $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$, when U^* is the initiator or $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$, when U^* is the responder.

We construct an algorithm $\mathcal{B}$ against a KDF challenger, using the adversary $\mathcal{A}$. The KDF challenger sends a ms value which is either generated using the KDF or randomly chosen. $\mathcal{B}$ uses the received ms value to compute the session key of the target session using the PRF.

If ms is computed using the KDF, simulation constructed by $\mathcal{B}$ is identical to Game 6, otherwise it is identical to Game 7. If $\mathcal{A}$ can distinguish the difference between Game 6 and Game 7, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the ms value is computed using KDF or randomly chosen. $\mathcal{B}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 6}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 7}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}). \tag{41}$$

Game 8. The Game 8 challenger randomly chooses $K \xleftarrow{\$} \{0,1\}^k$ as session key of the target session and its partner session.

We construct an algorithm $\mathcal{J}$ against an Oracle$^{\text{PRF}}$, using the adversary $\mathcal{A}$. The Oracle$^{\text{PRF}}$ sends a K value which is either generated using the PRF with a hidden key, or a random function. $\mathcal{J}$ uses the received K as the session key of the target session.

If K is generated using the PRF with a hidden key, simulation constructed by $\mathcal{J}$ is identical to Game 7, otherwise it is identical to Game 8. If $\mathcal{A}$ can distinguish the difference between Game 7 and Game 8, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{J}$, which is used to distinguish whether the Oracle$^{\text{PRF}}$ is real or a random function. $\mathcal{J}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\text{Adv}_{\text{Game 7}}(\mathcal{A}) - \text{Adv}_{\text{Game 8}}(\mathcal{A})| \leq \text{Adv}_{\text{PRF}}(\mathcal{J}). \tag{42}$$

Semantic security of the session key in Game 8. 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 8. Hence,

$$\text{Adv}_{\text{Game 8}}(\mathcal{A}) = 0. \tag{43}$$

We find,

$$\text{Adv}^{\boldsymbol{\lambda}\text{-w}(\cdot)\text{AFL-eCK}}_{\pi,\text{Case 1.3}}(\mathcal{A}) \leq N_P^2 N_s^2 \Big[\big(\text{Adv}^{\text{DDH}}_{q,g}(\mathcal{C}) + \text{Adv}_{\text{KDF}}(\mathcal{B}) + \text{Adv}_{\text{PRF}}(\mathcal{J}) + \text{Adv}^{\text{CPLA2}}_{\text{PKE}}(\mathcal{D}) + \epsilon\big) + \frac{1}{q}\Big].$$

Case 1.4: Adversary corrupts the owner, but not the partner to the test session

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

$$\text{Adv}_{\text{Game 1}}(\mathcal{A}) = \text{Adv}^{\boldsymbol{\lambda}\text{-w}(\cdot)\text{AFL-eCK}}_{\pi,\text{Case 1.4}}(\mathcal{A}). \tag{44}$$

Game 2. Abort the simulation if there exists two sessions outputting the same ephemeral public keys (Same $X = g^x$ values). Since the ephemeral keys are coming from $\mathbb{Z}_q^*$, the total number of ephemeral keys are q. Total number of session in the simulation is ${N_P}^2{N_s}^2$, because N_P is the number of protocol principals and each protocol principal owns N_s number of sessions. Hence,

$$|\text{Adv}_{\text{Game 1}}(\mathcal{A}) - \text{Adv}_{\text{Game 2}}(\mathcal{A})| \leq \frac{{N_P}^2{N_s}^2}{q}. \tag{45}$$

Game 3. Before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \xleftarrow{\$} \{U_1, ..., U_{N_P}\}$ are chosen and two random numbers $s^*, t^* \xleftarrow{\$} \{1, ...N_s\}$ 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 3 challenger aborts the game. Unless the incorrect choice happens, Game 3 is identical to Game 2. Hence,

$$\text{Adv}_{\text{Game 3}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2}\text{Adv}_{\text{Game 2}}(\mathcal{A}). \tag{46}$$

Game 4. Game 4 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and computes the ephemeral key $r_{V^*} \xleftarrow{\$} \text{Enc}(p_{V^*}, \widetilde{r_{V^*}})$, in the partner to the target session.

We introduce an algorithm $\mathcal{F}$ which is constructed using the adversary $\mathcal{A}$, against the ϵ-pair-generation indistinguishability challenger (ϵ-PG). $\mathcal{F}$ receives a pair $(r_{V^*}, \widetilde{r_{V^*}})$ such that $\widetilde{r_{V^*}} = \text{Dec}(s_{V^*}, r_{V^*})$. $\mathcal{F}$ uses r_{V^*} as the ephemeral key of V^* and $\widetilde{r_{V^*}}$ as the pseudo-ephemeral key of V^* in the partner to the target session.

If a random ephemeral key $r_{V^*} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the pseudo-ephemeral value $\widetilde{r_{V^*}} \leftarrow \mathrm{Dec}(s_{V^*}, r_{V^*})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 3. Otherwise if a random pseudo-ephemeral value $\widetilde{r_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the ephemeral key $r_{V^*} \xleftarrow{\$} \mathrm{Enc}(p_{V^*}, \widetilde{r_{V^*}})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{F}$ can distinguish whether a message/ciphertext pair (m, c) belongs to the distribution D_1 or D_2 (ϵ-pair-generation indistinguishability challenge). $\mathcal{F}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 3}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 4}(\mathcal{A})| \le \epsilon. \tag{47}$$

Game 5. Game 5 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r'_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and uses it as the pseudo ephemeral value of V^* in the partner to the target session.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$, against the CPLA2 challenger. The algorithm $\mathcal{D}$ uses the public-key of the CPLA2 challenger as the public key of the protocol principal V^* and generates public/secret key pairs for all other protocol principals. $\mathcal{D}$ generates signing/verification key pairs for every protocol principal. $\mathcal{D}$ picks two random strings, $r'_0, r'_1 \xleftarrow{\$} \{0,1\}^k$ and passes them to the CPLA2 challenger. From the CPLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C_2 such that $C_2 \xleftarrow{\$} \mathrm{Enc}(p_{V^*}, r'_\theta)$ where $r'_\theta = r'_0$ or $r'_\theta = r'_1$. The following describes the procedure of answering queries:

- `Send`$(U, V, s, m, \mathbf{f})$ query: When $U = V^*$, $V = U^*$ and $s = t^*$, $\mathcal{D}$ takes r'_1 as $\widetilde{r'_{V^*}}$, computes $g^{\widetilde{r'_{V^*}}}$ and computes its signature using the signing key sk_{V^*}. Then $\mathcal{D}$ creates the protocol message and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(s_{V^*})$, where the leakage $\mathbf{f}(s_{V^*})$ is obtained by accessing the leakage oracle of the CPLA2 challenger.

  For all other `Send` queries, $\mathcal{D}$ can execute the protocol normally, because $\mathcal{D}$ has all the public keys and can compute protocol messages accordingly. Except V^* $\mathcal{D}$ can compute the leakage by its own, and for V^* $\mathcal{D}$ accesses the leakage oracle to obtain the leakage.

- `SessionKeyReveal`(U, V, s) query: $\mathcal{D}$ will abort if `SessionKeyReveal`(U^*, V^*, s^*) or `SessionKeyReveal`(V^*, U^*, t^*) query is asked. $\mathcal{D}$ can easily compute the answers using the corresponding psuedo-ephemeral keys for other `SessionKeyReveal` queries.

- `EphemeralKeyReveal`(U, V, s) query: For the `EphemeralKeyReveal`(V^*, U^*, t^*) query, $\mathcal{D}$ uses C_2 as the answer. For all other `EphemeralKeyReveal` queries $\mathcal{D}$ will answer with the corresponding ephemeral-key which is computed by encrypting a pseudo-ephemeral value with the secret key of the corresponding principal.

- `Corrupt`(U) query: Except for U^* and V^*, algorithm $\mathcal{D}$ can answer all other `Corrupt` queries. In this case we consider the situation in which the adversary does not corrupt the partner principal to the test session, so these exceptions will not occur.

- `Test`(U, s) query: The algorithm $\mathcal{D}$ will abort the game if the adversary issues a `Test` query other than `Test`(U^*, s^*). To compute the answer to the `Test`(U^*, s^*) query, the algorithm $\mathcal{D}$ computes:

  - If U^* is the initiator, $ms \leftarrow \mathrm{KDF}(X_{U^*}^{\widetilde{r'_{V^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$ where $\widetilde{r'_{U^*}} = r_1$..
  - If U^* is the responder computes $ms \leftarrow \mathrm{KDF}(X_{U^*}^{\widetilde{r'_{V^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$.

  Then using K answers the `Test` query.

If $\theta = 1$, then r'_1 is the decryption of C_2 and the simulation constructed by $\mathcal{D}$ is identical to Game 4 whereas if $\theta = 0$, then r'_0 is the decryption of C_2 and the simulation constructed by $\mathcal{D}$ is identical to Game 5. If $\mathcal{A}$ can distinguish the difference between Game 4 and Game 5, then $\mathcal{D}$ can be used against a CPLA2 challenger. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 4}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 5}(\mathcal{A})| \le \mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D}). \tag{48}$$

Game 6. Game 6 challenger randomly chooses $z \xleftarrow{\$} \mathbb{Z}_q^*$ and computes the session key of the target session and its partner session, using the KDF and the PRF as $ms \leftarrow \mathrm{KDF}(g^z, \perp, k, \perp)$ and $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$, when U^* is the initiator, or $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$, when U^* is the responder.

We construct an algorithm $\mathcal{C}$ against the DDH challenge, using the adversary $\mathcal{A}$. The DDH challenger sends values $(X = g^x, Y = g^y, Z = g^z)$ such that either $z = xy$ or $z \xleftarrow{\$} \mathbb{Z}_q^*$, as the inputs to the algorithm $\mathcal{C}$. $\mathcal{C}$ uses the value X as the ephemeral public key of U^* and Y as the ephemeral public key of V^* in the test session, and computes the session key using Z as the input to the KDF in the session key derivation process.

If $\mathcal{C}$'s input is a Diffie-Hellman triple, the simulation constructed by $\mathcal{C}$ is identical to Game 5, otherwise it is identical to Game 6. If $\mathcal{A}$ can distinguish whether $g^z = g^{xy}$ or not, then $\mathcal{C}$ can answer the DDH challenge. $\mathcal{C}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 5}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 6}(\mathcal{A})| \leq \mathrm{Adv}_{q,g}^{\mathrm{DDH}}(\mathcal{C}). \tag{49}$$

Game 7. The Game 7 challenger randomly chooses $ms \xleftarrow{\$} \{0,1\}^k$ and computes the session key of the target session and its partner session, using the PRF as $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$, when U^* is the initiator or $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$, when U^* is the responder.

We construct an algorithm $\mathcal{B}$ against a KDF challenger, using the adversary $\mathcal{A}$. The KDF challenger sends a ms value which is either generated using the KDF or randomly chosen. $\mathcal{B}$ uses the received ms value to compute the session key of the target session using the PRF.

If ms is computed using the KDF, simulation constructed by $\mathcal{B}$ is identical to Game 6, otherwise it is identical to Game 7. If $\mathcal{A}$ can distinguish the difference between Game 6 and Game 7, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the ms value is computed using KDF or randomly chosen. $\mathcal{B}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 6}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 7}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}). \tag{50}$$

Game 8. The Game 8 challenger randomly chooses $K \xleftarrow{\$} \{0,1\}^k$ as session key of the target session and its partner session.

We construct an algorithm $\mathcal{J}$ against an $\mathrm{Oracle}^{\mathrm{PRF}}$, using the adversary $\mathcal{A}$. The $\mathrm{Oracle}^{\mathrm{PRF}}$ sends a K value which is either generated using the PRF with a hidden key, or a random function. $\mathcal{J}$ uses the received K as the session key of the target session.

If K is generated using the PRF with a hidden key, simulation constructed by $\mathcal{J}$ is identical to Game 7, otherwise it is identical to Game 8. If $\mathcal{A}$ can distinguish the difference between Game 7 and Game 8, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{J}$, which is used to distinguish whether the $\mathrm{Oracle}^{\mathrm{PRF}}$ is real or a random function. $\mathcal{J}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game}\ 7}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game}\ 8}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}). \tag{51}$$

Semantic security of the session key in Game 8. Since the session key K of $\Pi_{U^*,V^*}^{s^*}$ is chosen randomly and independently from all other values, $\mathcal{A}$ does not have any advantage in Game 8. Hence,

$$\mathrm{Adv}_{\mathrm{Game}\ 8}(\mathcal{A}) = 0. \tag{52}$$

We find,

$$\mathrm{Adv}_{\pi,\mathrm{Case}\ 1.4}^{\boldsymbol{\lambda}\mathrm{-w}(\cdot)\mathrm{AFL\text{-}eCK}}(\mathcal{A}) \leq N_P^2 N_s^2 \Big[\big(\mathrm{Adv}_{q,g}^{\mathrm{DDH}}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) + \mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D}) + \epsilon\big) + \frac{1}{q}\Big].$$

Case 2: A partner session to the test session does not exist

When the partner session does not exist, the owner of the test session shares the session key with the active adversary. In this situation adversary is not allowed to corrupt the intended partner principal to the test session. We split this case into two sub cases as follows:

1. Test session is at the responder.
2. Test session is at the initiator.

Case 2.1: Test session is at the responder Let V^* be the initiator and U^* be the responder. Let X_{U^*} be the ephemeral public key of U^*, and X_{V^*} be the ephemeral public key of V^*, in the target session. In this case there are three sub cases, which address the three different situations occur when the challenger interacts with the adversary. We will analyze the adversaries advantage in winning the w$(\cdot)$AFL-eCK challenge in following three different cases.

- (a) There is no session at peer V^* with X_{V^*}: Here the adversary tries to compute the protocol message from V^* to U^*, by its own.
- (b) There exists a session at V^* with X_{V^*} and X_{U^*} (but σ_{U^*} computed by U^* is different from the σ_{U^*} received to V^* with the protocol message, in the target session): For instance the adversary corrupts U^*, re-sign the protocol message from U^* to V^*, executes the protocol and makes a non-matching session at V^*. Then, tries to reveal the session key of that non-matching session at V^*, and win the game.
- (c) There exists a session at V^* with X_{V^*} and $X'_{U^*} \neq X_{U^*}$: Here the adversary, changes the message from U^* to V^*, such that there is no matching session at V^*.

Case 2.1.a: There is no session at V^* with X_{V^*}

Assume that the adversary $\mathcal{A}$ asks a `Send` query to some fresh session, such that it accepts, but the signature used in the query is not generated by a legitimate party.

Game 1. This game is the original game. Hence,

$$\mathrm{Adv}_{\text{Game 1}}(\mathcal{A}) = \mathrm{Adv}_{\pi,\text{Case 2.1.a}}^{\boldsymbol{\lambda}-\text{w}(\cdot)\text{AFL-eCK}}(\mathcal{A}). \tag{53}$$

Game 2. Before $\mathcal{A}$ begins, the Game 2 challenger guesses the identity, V^*, of the partner principal to the test session and if the guess is incorrect it aborts the game. The probability of Game 2 to be aborted due to incorrect guess of the partner principal to the test session is $1 - \frac{1}{N_P}$. Unless the incorrect guess happens, Game 2 is identical to Game1. Hence,

$$\mathrm{Adv}_{\text{Game 2}}(\mathcal{A}) = \frac{1}{N_P}\mathrm{Adv}_{\text{Game 1}}(\mathcal{A}). \tag{54}$$

The algorithm $\mathcal{E}$ sets the verification key of the signature scheme challenger to the principal V^*. The owner principal accepts the message coming from the intended partner, because the owner computes $\mathrm{Vfy}(vk_{V^*}, X_{V^*}, \sigma_{V^*})$ is "true". But the principal V^* is not corrupted and the message X_{V^*} is not signed by the principal V^*, because there is no partner to the test session. Hence,

$$\mathrm{Adv}_{\text{Game 2}}(\mathcal{A}) = \mathrm{Adv}_{\text{SIG}}^{\text{UFCMLA}}(\mathcal{E}). \tag{55}$$

We find,

$$\mathrm{Adv}_{\pi,\text{Case 2.1.a}}^{\boldsymbol{\lambda}-\text{w}(\cdot)\text{AFL-eCK}}(\mathcal{A}) = N_P \mathrm{Adv}_{\text{SIG}}^{\text{UFCMLA}}(\mathcal{E}).$$

Case 2.1.b: There exists a session at V^* with X_{V^*} and X_{U^*} (but σ_{V^*} computed by V^* is different from the σ_{V^*} received to U^* with the protocol message)

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

$$\mathrm{Adv}_{\text{Game 1}}(\mathcal{A}) = \mathrm{Adv}_{\pi,\text{Case 2.1.b}}^{\boldsymbol{\lambda}-\text{w}(\cdot)\text{AFL-eCK}}(\mathcal{A}). \tag{56}$$

Game 2. Abort the simulation if there exists two sessions outputting the same ephemeral public keys (Same $X = g^x$ values). Since the ephemeral keys are coming from $\mathbb{Z}_q^*$, the total number of ephemeral keys are q. Total number of session in the simulation is ${N_P}^2 {N_s}^2$, because N_P is the number of protocol principals and each protocol principal owns N_s number of sessions. Hence,

$$|\mathrm{Adv}_{\mathrm{Game\ 1}}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game\ 2}}(\mathcal{A})| \leq \frac{{N_P}^2 {N_s}^2}{q}. \tag{57}$$

Game 3. Before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \xleftarrow{\$} \{U_1, ..., U_{N_P}\}$ are chosen as the owner and the peer, and two random numbers $s^*, t^* \xleftarrow{\$} \{1, ...N_s\}$ 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 almost partner session to the target session. If the test session is not the session $\Pi^{s^*}_{U^*,V^*}$, the Game 3 challenger aborts the game. Unless the incorrect choice happens, Game 3 is identical to Game 2. Hence,

$$\mathrm{Adv}_{\mathrm{Game\ 3}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} \mathrm{Adv}_{\mathrm{Game\ 2}}(\mathcal{A}). \tag{58}$$

The almost partner session is the session, which communicates with the target session but due to adversaries interaction it does not preserve the partnering conditions.

Game 4. Game 4 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r_{U^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and computes the ephemeral key $r_{U^*} \xleftarrow{\$} \mathrm{Enc}(p_{U^*}, \widetilde{r_{U^*}})$ in the target session.

We introduce an algorithm $\mathcal{F}$ which is constructed using the adversary $\mathcal{A}$, against the ϵ-pair-generation indistinguishability challenger (ϵ-PG). $\mathcal{F}$ receives a pair $(r_{U^*}, \widetilde{r_{U^*}})$ such that $\widetilde{r_{U^*}} = \mathrm{Dec}(s_{U^*}, r_{U^*})$. $\mathcal{F}$ uses r_{U^*} as the ephemeral key of U^* and $\widetilde{r_{U^*}}$ as the pseudo-ephemeral key of U^* in the target session.

If a random ephemeral key $r_{U^*} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the pseudo-ephemeral value $\widetilde{r_{U^*}} \leftarrow \mathrm{Dec}(s_{U^*}, r_{U^*})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 3. Otherwise if a random pseudo-ephemeral value $\widetilde{r_{U^*}} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the ephemeral key $r_{U^*} \xleftarrow{\$} \mathrm{Enc}(p_{U^*}, \widetilde{r_{U^*}})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{F}$ can distinguish whether a message/ciphertext pair (m, c) belongs to the distribution D_1 or D_2 (ϵ-pair-generation indistinguishability challenge). $\mathcal{F}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game\ 3}}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game\ 4}}(\mathcal{A})| \leq \epsilon. \tag{59}$$

Game 5. Game 5 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r'_{U^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and uses it as the pseudo ephemeral value of U^* in the target session.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$, against the CPLA2 challenger. The algorithm $\mathcal{D}$ uses the public-key of the CPLA2 challenger as the public key of the protocol principal U^* and generates public/secret key pairs for all other protocol principals. $\mathcal{D}$ generates signing/verification key pairs for every protocol principal. $\mathcal{D}$ picks two random strings, $r_0, r_1 \xleftarrow{\$} \{0,1\}^k$ and passes them to the CPLA2 challenger. From the CPLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C_1 such that $C_1 \xleftarrow{\$} \mathrm{Enc}(p_{U^*}, r_\theta)$ where $r_\theta = r_0$ or $r_\theta = r_1$. The following describes the procedure of answering queries:

- $\mathtt{Send}(U, V, s, m, \mathbf{f})$ query: When $U = U^*$, $V = V^*$ and $s = s^*$, $\mathcal{D}$ takes r_1 as $\widetilde{r'_{U^*}}$, computes $g^{\widetilde{r'_{U^*}}}$ and computes its signature using the signing key sk_{U^*}. Then $\mathcal{D}$ creates the protocol message and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(s_{U^*})$, where the leakage $\mathbf{f}(s_{U^*})$ is obtained by accessing the leakage oracle of the CPLA2 challenger.

  For all other `Send` queries, $\mathcal{D}$ can execute the protocol normally, because $\mathcal{D}$ has all the public keys and can compute protocol messages accordingly. Except U^* $\mathcal{D}$ can compute the leakage by its own, and for U^* $\mathcal{D}$ accesses the leakage oracle to obtain the leakage.

- `SessionKeyReveal`(U, V, s) query: $\mathcal{D}$ will abort if `SessionKeyReveal`(U^*, V^*, s^*) or `SessionKeyReveal`(V^*, U^*, t^*) query is asked. $\mathcal{D}$ can easily compute the answers using the corresponding psuedo-ephemeral keys for other `SessionKeyReveal` queries.

- `EphemeralKeyReveal`(U, V, s) query: For the `EphemeralKeyReveal`(U^*, V^*, s^*) query, $\mathcal{D}$ uses C_1 as the answer. For all other `EphemeralKeyReveal` queries $\mathcal{D}$ will answer with the corresponding ephemeral-key which is computed by encrypting a pseudo-ephemeral value with the secret key of the corresponding principal.

- `Corrupt`(U) query: Algorithm $\mathcal{D}$ can answer all the `Corrupt` queries allowed in the freshness condition.

- `Test`(U, s) query: The algorithm $\mathcal{D}$ will abort the game if the adversary issues a `Test` query other than `Test`(U^*, s^*). To compute the answer to the `Test`(U^*, s^*) query, the algorithm $\mathcal{D}$ computes:

  - If U^* is the initiator, computes $ms \leftarrow \mathrm{KDF}(X_{V^*}^{\widetilde{r'_{U^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^* ||X_{V^*}||\sigma_{V^*})$ where $\widetilde{r'_{U^*}} = r_1$, when U^* is the initiator.

  - If U^* is the responder, computes $ms \leftarrow \mathrm{KDF}(X_{V^*}^{\widetilde{r'_{U^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*} ||U^*||X_{U^*}||\sigma_{U^*})$.

  Then using K answers the `Test` query.

If $\theta = 1$, then r_1 is the decryption of C_1 and the simulation constructed by $\mathcal{D}$ is identical to Game 4 whereas if $\theta = 0$, then r_0 is the decryption of C_1 and the simulation constructed by $\mathcal{D}$ is identical to Game 5. If $\mathcal{A}$ can distinguish the difference between Game 4 and Game 5, then $\mathcal{D}$ can be used against a CPLA2 challenger. Hence,

$$|\mathrm{Adv}_{\mathrm{Game\ 4}}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game\ 5}}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D}). \tag{60}$$

Game 6. Game 6 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and computes the ephemeral key $r_{V^*} \xleftarrow{\$} \mathrm{Enc}(p_{V^*}, \widetilde{r_{V^*}})$ in the almost partner session.

We introduce an algorithm $\mathcal{F}$ which is constructed using the adversary $\mathcal{A}$, against the ϵ-pair-generation indistinguishability challenger (ϵ-PG). $\mathcal{F}$ receives a pair $(r_{V^*}, \widetilde{r_{V^*}})$ such that $\widetilde{r_{V^*}} = \mathrm{Dec}(s_{V^*}, r_{V^*})$. $\mathcal{F}$ uses r_{V^*} as the ephemeral key of V^* and $\widetilde{r_{V^*}}$ as the pseudo-ephemeral key of V^* in the almost partner session.

If a random ephemeral key $r_{V^*} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the pseudo-ephemeral value $\widetilde{r_{V^*}} \leftarrow \mathrm{Dec}(s_{V^*}, r_{V^*})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 5. Otherwise if a random pseudo-ephemeral value $\widetilde{r_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the ephemeral key $r_{V^*} \xleftarrow{\$} \mathrm{Enc}(p_{V^*}, \widetilde{r_{V^*}})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 6. If $\mathcal{A}$ can distinguish the difference between Game 5 and Game 6, then $\mathcal{F}$ can distinguish whether a message/ciphertext pair (m, c) belongs to the distribution D_1 or D_2 (ϵ-pair-generation indistinguishability challenge). $\mathcal{F}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\mathrm{Game\ 5}}(\mathcal{A}) - \mathrm{Adv}_{\mathrm{Game\ 6}}(\mathcal{A})| \leq \epsilon. \tag{61}$$

Game 7. Game 7 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r'_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and uses it as the pseudo ephemeral value of V^* in the almost partner session.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$, against the CPLA2 challenger. The algorithm $\mathcal{D}$ uses the public-key of the CPLA2 challenger as the public key of the protocol principal V^* and generates public/secret key pairs for all other protocol principals. $\mathcal{D}$ generates signing/verification key pairs for every protocol principal. $\mathcal{D}$ picks two random strings, $r'_0, r'_1 \xleftarrow{\$} \{0,1\}^k$ and passes them to the CPLA2 challenger. From the CPLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C_2 such that $C_2 \xleftarrow{\$} \mathrm{Enc}(p_{V^*}, r'_\theta)$ where $r'_\theta = r'_0$ or $r'_\theta = r'_1$. The following describes the procedure of answering queries:

- Send$(U, V, s, m, \mathbf{f})$ query: When $U = V^*$, $V = U^*$ and $s = t^*$, $\mathcal{D}$ takes r'_1 as $\widetilde{r'_{V^*}}$, computes $g^{\widetilde{r'_{V^*}}}$ and computes its signature using the signing key sk_{V^*}. Then $\mathcal{D}$ creates the protocol message and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(s_{V^*})$, where the leakage $\mathbf{f}(s_{V^*})$ is obtained by accessing the leakage oracle of the CPLA2 challenger.

  For all other Send queries, $\mathcal{D}$ can execute the protocol normally, because $\mathcal{D}$ has all the public keys and can compute protocol messages accordingly. Except V^* $\mathcal{D}$ can compute the leakage by its own, and for V^* $\mathcal{D}$ accesses the leakage oracle to obtain the leakage.

- SessionKeyReveal(U, V, s) query: $\mathcal{D}$ will abort if SessionKeyReveal(U^*, V^*, s^*) or SessionKeyReveal(V^*, U^*, t^*) query is asked. $\mathcal{D}$ can easily compute the answers using the corresponding psuedo-ephemeral keys for other SessionKeyReveal queries.

- EphemeralKeyReveal(U, V, s) query: For the EphemeralKeyReveal(V^*, U^*, t^*) query, $\mathcal{D}$ uses C_2 as the answer. For all other EphemeralKeyReveal queries $\mathcal{D}$ will answer with the corresponding ephemeral-key which is computed by encrypting a pseudo-ephemeral value with the secret key of the corresponding principal.

- Corrupt(U) query: Algorithm $\mathcal{D}$ can answer all the Corrupt queries allowed in the freshness condition.

- Test(U, s) query: The algorithm $\mathcal{D}$ will abort the game if the adversary issues a Test query other than Test(U^*, s^*). To compute the answer to the Test(U^*, s^*) query, the algorithm $\mathcal{D}$ computes:

  - If U^* is the initiator, computes $ms \leftarrow \mathrm{KDF}(X_{U^*}^{\widetilde{r'_{V^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^* ||X_{V^*}||\sigma_{V^*})$ where $\widetilde{r'_{U^*}} = r_1$.
  - If U^* is the responder, computes $ms \leftarrow \mathrm{KDF}(X_{U^*}^{\widetilde{r'_{V^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*} ||U^*||X_{U^*}||\sigma_{U^*})$.

  Then using K answers the Test query.

If $\theta = 1$, then r'_1 is the decryption of C_2 and the simulation constructed by $\mathcal{D}$ is identical to Game 6 whereas if $\theta = 0$, then r'_0 is the decryption of C_2 and the simulation constructed by $\mathcal{D}$ is identical to Game 7. If $\mathcal{A}$ can distinguish the difference between Game 6 and Game 7, then $\mathcal{D}$ can be used against a CPLA2 challenger. Hence,

$$|\mathrm{Adv}_{\text{Game 6}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 7}}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D}). \tag{62}$$

Game 8. Game 8 challenger randomly chooses $z \xleftarrow{\$} \mathbb{Z}_q^*$ and computes the session key of the target session, using the KDF and the PRF as $ms \leftarrow \mathrm{KDF}(g^z, \perp, k, \perp)$ and $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^* ||X_{V^*}||\sigma_{V^*})$, when U^* is the initiator, or $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$, when U^* is the responder.

We construct an algorithm $\mathcal{C}$ against the DDH challenge, using the adversary $\mathcal{A}$. The DDH challenger sends values $(X = g^x, Y = g^y, Z = g^z)$ such that either $z = xy$ or $z \xleftarrow{\$} \mathbb{Z}_q^*$, as the inputs to the algorithm $\mathcal{C}$. $\mathcal{C}$ uses the value X as the ephemeral public key of U^* and Y as the ephemeral public key of V^* in the test session, and computes the session key using Z as the input to the KDF in the session key derivation process. In this game, for the almost partner session, the value Z is used as the Diffie-Hellman shared secret.

If $\mathcal{C}$'s input is a Diffie-Hellman triple, the simulation constructed by $\mathcal{C}$ is identical to Game 7, otherwise it is identical to Game 8. If $\mathcal{A}$ can distinguish whether $g^z = g^{xy}$ or not, then $\mathcal{C}$ can answer the DDH challenge. $\mathcal{C}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\text{Game 7}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 8}}(\mathcal{A})| \leq \mathrm{Adv}_{q,g}^{\mathrm{DDH}}(\mathcal{C}). \tag{63}$$

Game 9. The Game 9 challenger randomly chooses $ms \xleftarrow{\$} \{0,1\}^k$ and computes the session key of the target session, using the PRF as $K \leftarrow \text{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$, when U^* is the initiator or $K \leftarrow \text{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$, when U^* is the responder.

We construct an algorithm $\mathcal{B}$ against a KDF challenger, using the adversary $\mathcal{A}$. The KDF challenger sends a ms value which is either generated using the KDF or randomly chosen. $\mathcal{B}$ uses the received ms value to compute the session key of the target session using the PRF. In this game, for the almost partner session, the value ms is used as the shared value derived using the KDF.

If ms is computed using the KDF, simulation constructed by $\mathcal{B}$ is identical to Game 8, otherwise it is identical to Game 9. If $\mathcal{A}$ can distinguish the difference between Game 8 and Game 9, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{B}$, which is used to distinguish whether the ms value is computed using KDF or randomly chosen. $\mathcal{B}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\text{Adv}_{\text{Game 8}}(\mathcal{A}) - \text{Adv}_{\text{Game 9}}(\mathcal{A})| \leq \text{Adv}_{\text{KDF}}(\mathcal{B}). \tag{64}$$

Game 10. The Game 10 challenger randomly chooses $K \xleftarrow{\$} \{0,1\}^k$ as session key of the target session.

We construct an algorithm $\mathcal{J}$ against an Oracle$^{\text{PRF}}$, using the adversary $\mathcal{A}$. The Oracle$^{\text{PRF}}$ sends a K value which is either generated using the PRF with a hidden key, or a random function. $\mathcal{J}$ uses the received K as the session key of the target session.

If K is generated using the PRF with a hidden key, simulation constructed by $\mathcal{J}$ is identical to Game 9, otherwise it is identical to Game 10. If $\mathcal{A}$ can distinguish the difference between Game 9 and Game 10, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{J}$, which is used to distinguish whether the Oracle$^{\text{PRF}}$ is real or a random function. $\mathcal{J}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\text{Adv}_{\text{Game 9}}(\mathcal{A}) - \text{Adv}_{\text{Game 10}}(\mathcal{A})| \leq \text{Adv}_{\text{PRF}}(\mathcal{J}). \tag{65}$$

Semantic security of the session key in Game 10. 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 10. Hence,

$$\text{Adv}_{\text{Game 10}}(\mathcal{A}) = 0. \tag{66}$$

We find,

$$\text{Adv}^{\boldsymbol{\lambda}-\text{w}(\cdot)\text{AFL-eCK}}_{\pi,\text{Case 2.1.b}}(\mathcal{A}) \leq N_P^2 N_s^2 \Big[\big(\text{Adv}^{\text{DDH}}_{q,g}(\mathcal{C}) + \text{Adv}_{\text{KDF}}(\mathcal{B}) + \text{Adv}_{\text{PRF}}(\mathcal{J}) \\ +2\text{Adv}^{\text{CPLA2}}_{\text{PKE}}(\mathcal{D}) + 2\epsilon\big) + \frac{1}{q}\Big].$$

Case 2.1.c: There exists a session at V^* with X_{V^*} and $X'_{U^*} \neq X_{U^*}$

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

$$\text{Adv}_{\text{Game 1}}(\mathcal{A}) = \text{Adv}^{\boldsymbol{\lambda}-\text{w}(\cdot)\text{AFL-eCK}}_{\pi}(\mathcal{A}). \tag{67}$$

Game 2. Abort the simulation if there exists two sessions outputting the same ephemeral public keys (Same $X = g^x$ values). Since the ephemeral keys are coming from $\mathbb{Z}_q^*$, the total number of ephemeral keys are q. Total number of session in the simulation is ${N_P}^2{N_s}^2$, because N_P is the number of protocol principals and each protocol principal owns N_s number of sessions. Hence,

$$|\text{Adv}_{\text{Game 1}}(\mathcal{A}) - \text{Adv}_{\text{Game 2}}(\mathcal{A})| \leq \frac{{N_P}^2{N_s}^2}{q}. \tag{68}$$

Game 3. Before $\mathcal{A}$ begins, two distinct random principals $U^*, V^* \overset{\$}{\leftarrow} \{U_1, ..., U_{N_P}\}$ are chosen as the owner and the peer, and two random numbers $s^*, t^* \overset{\$}{\leftarrow} \{1, ...N_s\}$ 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 almost partner session to the target session. If the test session is not the session $\Pi^{s^*}_{U^*,V^*}$, the Game 3 challenger aborts the game. Unless the incorrect choice happens, Game 3 is identical to Game 2. Hence,

$$\mathrm{Adv}_{\text{Game 3}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} \mathrm{Adv}_{\text{Game 2}}(\mathcal{A}). \tag{69}$$

Game 4. Game 4 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r_{U^*}} \overset{\$}{\leftarrow} \mathbb{Z}_q^*$, and computes the ephemeral key $r_{U^*} \overset{\$}{\leftarrow} \mathrm{Enc}(p_{U^*}, \widetilde{r_{U^*}})$ in the target session.

We introduce an algorithm $\mathcal{F}$ which is constructed using the adversary $\mathcal{A}$, against the ϵ-pair-generation indistinguishability challenger (ϵ-PG). $\mathcal{F}$ receives a pair $(r_{U^*}, \widetilde{r_{U^*}})$ such that $\widetilde{r_{U^*}} = \mathrm{Dec}(s_{U^*}, r_{U^*})$. $\mathcal{F}$ uses r_{U^*} as the ephemeral key of U^* and $\widetilde{r_{U^*}}$ as the pseudo-ephemeral key of U^* in the target session.

If a random ephemeral key $r_{U^*} \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ is chosen first and the pseudo-ephemeral value $\widetilde{r_{U^*}} \leftarrow \mathrm{Dec}(s_{U^*}, r_{U^*})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 3. Otherwise if a random pseudo-ephemeral value $\widetilde{r_{U^*}} \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ is chosen first and the ephemeral key $r_{U^*} \overset{\$}{\leftarrow} \mathrm{Enc}(p_{U^*}, \widetilde{r_{U^*}})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 4. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{F}$ can distinguish whether a message/ciphertext pair (m, c) belongs to the distribution D_1 or D_2 (ϵ-pair-generation indistinguishability challenge). $\mathcal{F}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\text{Game 3}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 4}}(\mathcal{A})| \leq \epsilon. \tag{70}$$

Game 5. Game 5 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r'_{U^*}} \overset{\$}{\leftarrow} \mathbb{Z}_q^*$, and uses it as the pseudo ephemeral value of U^* in the target session.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$, against the CPLA2 challenger. The algorithm $\mathcal{D}$ uses the public-key of the CPLA2 challenger as the public key of the protocol principal U^* and generates public/secret key pairs for all other protocol principals. $\mathcal{D}$ generates signing/verification key pairs for every protocol principal. $\mathcal{D}$ picks two random strings, $r_0, r_1 \overset{\$}{\leftarrow} \{0,1\}^k$ and passes them to the CPLA2 challenger. From the CPLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C_1 such that $C_1 \overset{\$}{\leftarrow} \mathrm{Enc}(p_{U^*}, r_\theta)$ where $r_\theta = r_0$ or $r_\theta = r_1$. The following describes the procedure of answering queries:

- `Send`$(U, V, s, m, \mathbf{f})$ query: When $U = U^*$, $V = V^*$ and $s = s^*$, $\mathcal{D}$ takes r_1 as $\widetilde{r'_{U^*}}$, computes $g^{\widetilde{r'_{U^*}}}$ and computes its signature using the signing key sk_{U^*}. Then $\mathcal{D}$ creates the protocol message and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(s_{U^*})$, where the leakage $\mathbf{f}(s_{U^*})$ is obtained by accessing the leakage oracle of the CPLA2 challenger.

  For all other `Send` queries, $\mathcal{D}$ can execute the protocol normally, because $\mathcal{D}$ has all the public keys and can compute protocol messages accordingly. Except U^* $\mathcal{D}$ can compute the leakage by its own, and for U^* $\mathcal{D}$ accesses the leakage oracle to obtain the leakage.

- `SessionKeyReveal`(U, V, s) query: $\mathcal{D}$ will abort if `SessionKeyReveal`(U^*, V^*, s^*) or `SessionKeyReveal`(V^*, U^*, t^*) query is asked. $\mathcal{D}$ can easily compute the answers using the corresponding psuedo-ephemeral keys for other `SessionKeyReveal` queries.

- `EphemeralKeyReveal`(U, V, s) query: For the `EphemeralKeyReveal`(U^*, V^*, s^*) query, $\mathcal{D}$ uses C_1 as the answer. For all other `EphemeralKeyReveal` queries $\mathcal{D}$ will answer with the corresponding ephemeral-key which is computed by encrypting a pseudo-ephemeral value with the secret key of the corresponding principal.

- `Corrupt`(U) query: Algorithm $\mathcal{D}$ can answer all the `Corrupt` queries allowed in the freshness condition.

- $\texttt{Test}(U, s)$ query: The algorithm $\mathcal{D}$ will abort the game if the adversary issues a $\texttt{Test}$ query other than $\texttt{Test}(U^*, s^*)$. To compute the answer to the $\texttt{Test}(U^*, s^*)$ query, the algorithm $\mathcal{D}$ computes:
  - If U^* is the initiator, $ms \leftarrow \text{KDF}(X_{V^*}^{\widetilde{r'_{U^*}}}, \perp, k, \perp)$, $K \leftarrow \text{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$ where $\widetilde{r'_{U^*}} = r_1$.
  - If U^* is the responder computes $ms \leftarrow \text{KDF}(X_{V^*}^{\widetilde{r'_{U^*}}}, \perp, k, \perp)$, $K \leftarrow \text{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^* ||X_{U^*}||\sigma_{U^*})$.

  Then using K answers the $\texttt{Test}$ query.

If $\theta = 1$, then r_1 is the decryption of C_1 and the simulation constructed by $\mathcal{D}$ is identical to Game 4 whereas if $\theta = 0$, then r_0 is the decryption of C_1 and the simulation constructed by $\mathcal{D}$ is identical to Game 5. If $\mathcal{A}$ can distinguish the difference between Game 4 and Game 5, then $\mathcal{D}$ can be used against a CPLA2 challenger. Hence,

$$|\text{Adv}_{\text{Game } 4}(\mathcal{A}) - \text{Adv}_{\text{Game } 5}(\mathcal{A})| \leq \text{Adv}_{\text{PKE}}^{\text{CPLA2}}(\mathcal{D}). \tag{71}$$

Game 6. Game 6 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and computes the ephemeral key $r_{V^*} \xleftarrow{\$} \text{Enc}(p_{V^*}, \widetilde{r_{V^*}})$ in the almost partner session.

We introduce an algorithm $\mathcal{F}$ which is constructed using the adversary $\mathcal{A}$, against the ϵ-pair-generation indistinguishability challenger (ϵ-PG). $\mathcal{F}$ receives a pair $(r_{V^*}, \widetilde{r_{V^*}})$ such that $\widetilde{r_{V^*}} = \text{Dec}(s_{V^*}, r_{V^*})$. $\mathcal{F}$ uses r_{V^*} as the ephemeral key of V^* and $\widetilde{r_{V^*}}$ as the pseudo-ephemeral key of V^* in the almost partner session.

If a random ephemeral key $r_{V^*} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the pseudo-ephemeral value $\widetilde{r_{V^*}} \leftarrow \text{Dec}(s_{V^*}, r_{V^*})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 5. Otherwise if a random pseudo-ephemeral value $\widetilde{r_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$ is chosen first and the ephemeral key $r_{V^*} \xleftarrow{\$} \text{Enc}(p_{V^*}, \widetilde{r_{V^*}})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 6. If $\mathcal{A}$ can distinguish the difference between Game 5 and Game 6, then $\mathcal{F}$ can distinguish whether a message/ciphertext pair (m, c) belongs to the distribution D_1 or D_2 (ϵ-pair-generation indistinguishability challenge). $\mathcal{F}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\text{Adv}_{\text{Game } 5}(\mathcal{A}) - \text{Adv}_{\text{Game } 6}(\mathcal{A})| \leq \epsilon. \tag{72}$$

Game 7. Game 7 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r'_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$, and uses it as the pseudo ephemeral value of V^* in the almost partner session.

We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$, against the CPLA2 challenger. The algorithm $\mathcal{D}$ uses the public-key of the CPLA2 challenger as the public key of the protocol principal V^* and generates public/secret key pairs for all other protocol principals. $\mathcal{D}$ generates signing/verification key pairs for every protocol principal. $\mathcal{D}$ picks two random strings, $r'_0, r'_1 \xleftarrow{\$} \{0,1\}^k$ and passes them to the CPLA2 challenger. From the CPLA2 challenger, $\mathcal{D}$ receives a challenge ciphertext C_2 such that $C_2 \xleftarrow{\$} \text{Enc}(p_{V^*}, r'_\theta)$ where $r'_\theta = r'_0$ or $r'_\theta = r'_1$. The following describes the procedure of answering queries:

- $\texttt{Send}(U, V, s, m, \mathbf{f})$ query: When $U = V^*$, $V = U^*$ and $s = t^*$, $\mathcal{D}$ takes r'_1 as $\widetilde{r'_{V^*}}$, computes $g^{\widetilde{r'_{V^*}}}$ and computes its signature using the signing key sk_{V^*}. Then $\mathcal{D}$ creates the protocol message and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(s_{V^*})$, where the leakage $\mathbf{f}(s_{V^*})$ is obtained by accessing the leakage oracle of the CPLA2 challenger.

  For all other $\texttt{Send}$ queries, $\mathcal{D}$ can execute the protocol normally, because $\mathcal{D}$ has all the public keys and can compute protocol messages accordingly. Except V^* $\mathcal{D}$ can compute the leakage by its own, and for V^* $\mathcal{D}$ accesses the leakage oracle to obtain the leakage.

- $\texttt{SessionKeyReveal}(U, V, s)$ query: $\mathcal{D}$ will abort if $\texttt{SessionKeyReveal}(U^*, V^*, s^*)$ or $\texttt{SessionKeyReveal}(V^*, U^*, t^*)$ query is asked. $\mathcal{D}$ can easily compute the answers using the corresponding psuedo-ephemeral keys for other $\texttt{SessionKeyReveal}$ queries.

- `EphemeralKeyReveal`(U, V, s) query: For the `EphemeralKeyReveal`(V^*, U^*, t^*) query, $\mathcal{D}$ uses C_2 as the answer. For all other `EphemeralKeyReveal` queries $\mathcal{D}$ will answer with the corresponding ephemeral-key which is computed by encrypting a pseudo-ephemeral value with the secret key of the corresponding principal.
- `Corrupt`(U) query: Algorithm $\mathcal{D}$ can answer all the `Corrupt` queries allowed in the freshness condition.
- `Test`(U, s) query: The algorithm $\mathcal{D}$ will abort the game if the adversary issues a `Test` query other than `Test`(U^*, s^*). To compute the answer to the `Test`(U^*, s^*) query, the algorithm $\mathcal{D}$ computes:
  - If U^* is the initiator, $ms \leftarrow \mathrm{KDF}(X_{U^*}^{\widetilde{r'_{V^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$ where $\widetilde{r'_{U^*}} = r_1$.
  - If U^* is the responder computes $ms \leftarrow \mathrm{KDF}(X_{U^*}^{\widetilde{r'_{V^*}}}, \perp, k, \perp)$, $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$.

  Then using K answers the `Test` query.

If $\theta = 1$, then r'_1 is the decryption of C_2 and the simulation constructed by $\mathcal{D}$ is identical to Game 6 whereas if $\theta = 0$, then r'_0 is the decryption of C_2 and the simulation constructed by $\mathcal{D}$ is identical to Game 7. If $\mathcal{A}$ can distinguish the difference between Game 6 and Game 7, then $\mathcal{D}$ can be used against a CPLA2 challenger. Hence,

$$|\mathrm{Adv}_{\text{Game 6}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 7}}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{CPLA2}}(\mathcal{D}). \tag{73}$$

Game 8. Game 8 challenger randomly chooses $ms \xleftarrow{\$} \{0,1\}^k$ and computes the session key of the target session, using the PRF as $K \leftarrow \mathrm{PRF}(ms, U^*||X_{U^*}||\sigma_{U^*}||V^*||X_{V^*}||\sigma_{V^*})$, when U^* is the initiator or $K \leftarrow \mathrm{PRF}(ms, V^*||X_{V^*}||\sigma_{V^*}||U^*||X_{U^*}||\sigma_{U^*})$, when U^* is the responder.

We construct an algorithm $\mathcal{R}$ against the ODH challenge, using the adversary $\mathcal{A}$. The ODH challenge being for group $\mathbb{G}$ with prime order q, generator g and function $\mathrm{KDF}(\cdot, \perp, k, \cdot)$. The ODH challenger sends values $(X = g^x, Y = g^y, Z)$ such that either $Z \leftarrow \mathrm{KDF}(g^{xy}, \perp, k, \perp)$ or $Z \xleftarrow{\$} \{0,1\}^k$, as the inputs to the algorithm $\mathcal{R}$. $\mathcal{R}$ uses the value X as the ephemeral public key of U^* and Y as the ephemeral public key of V^* in the test session, and computes the session key using Z as the ms value, which is the input to the PRF, in the session key derivation process. In this game, for the almost partner session (a session at V^* with X_{V^*} and $X'_{U^*} \neq X_{U^*}$), the ODH oracle $\mathcal{O}^{\mathrm{ODH}}$ is used to compute the ms value. For all the other honest sessions, simulator knows the ephemeral keys and can compute the shared Diffie-Hellman value and compute the session key normally.

If $\mathcal{R}$'s input $Z = \mathrm{KDF}(g^{xy}, \perp, k, \perp)$, the simulation constructed by $\mathcal{R}$ is identical to Game 7, otherwise it is identical to Game 8. Hence,

$$|\mathrm{Adv}_{\text{Game 7}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 8}}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{KDF}(\cdot,\perp,k,\perp),q,g}^{\mathrm{ODH}}(\mathcal{R}). \tag{74}$$

Game 9. The Game 9 challenger randomly chooses $K \xleftarrow{\$} \{0,1\}^k$ as session key of the target session.

We construct an algorithm $\mathcal{J}$ against an Oracle$^{\mathrm{PRF}}$, using the adversary $\mathcal{A}$. The Oracle$^{\mathrm{PRF}}$ sends a K value which is either generated using the PRF with a hidden key, or a random function. $\mathcal{J}$ uses the received K as the session key of the target session.

If K is generated using the PRF with a hidden key, simulation constructed by $\mathcal{J}$ is identical to Game 8, otherwise it is identical to Game 9. If $\mathcal{A}$ can distinguish the difference between Game 8 and Game 9, then $\mathcal{A}$ can be used as a subroutine of an algorithm $\mathcal{J}$, which is used to distinguish whether the Oracle$^{\mathrm{PRF}}$ is real or a random function. $\mathcal{J}$ can answer all the adversarial queries allowed in this case, because it has all the long-term and ephemeral secret keys of the allowed queries. Hence,

$$|\mathrm{Adv}_{\text{Game 8}}(\mathcal{A}) - \mathrm{Adv}_{\text{Game 9}}(\mathcal{A})| \leq \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}). \tag{75}$$

Semantic security of the session key in Game 9. 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 9. Hence,

$$\mathrm{Adv}_{\text{Game } 9}(\mathcal{A}) = 0. \tag{76}$$

We find,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\mathrm{w}(\cdot)\text{AFL-eCK}}_{\pi,\text{Case 2.1.c}}(\mathcal{A}) \leq N_P^2 N_s^2 \Big[\big(\mathrm{Adv}^{\mathrm{ODH}}_{q,g}(\mathcal{R}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) \\ +2\mathrm{Adv}^{\mathrm{CPLA2}}_{\mathrm{PKE}}(\mathcal{D}) + 2\epsilon\big) + \frac{1}{q}\Big].$$

Therefore, in Case 2.1,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\mathrm{w}(\cdot)\text{AFL-eCK}}_{\pi,\text{Case 2.1}}(\mathcal{A}) \leq \max \Big[N_P^2 N_s^2 \big[\big(\mathrm{Adv}^{\mathrm{DDH}}_{q,g}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) \\ +2\mathrm{Adv}^{\mathrm{CPLA2}}_{\mathrm{PKE}}(\mathcal{D}) + 2\epsilon\big) + \frac{1}{q}\big], N_P^2 N_s^2 \big[\big(\mathrm{Adv}^{\mathrm{ODH}}_{q,g}(\mathcal{R}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) \\ +2\mathrm{Adv}^{\mathrm{CPLA2}}_{\mathrm{PKE}}(\mathcal{D}) + 2\epsilon\big) + \frac{1}{q}\big], N_P \mathrm{Adv}^{\mathrm{UFCMLA}}_{\mathrm{SIG}}(\mathcal{E})\Big].$$

Case 2.2: Test session is at the initiator Let U^* be the initiator and V^* be the responder. Let X_{U^*} be the ephemeral public key of U^*, and X_{V^*} be the ephemeral public key of V^*, in the target session. In this case there are three sub cases, which address the three different situations occur when the challenger interacts with the adversary, which are same as to the Case 2.1.

- (a) There is no session at V^* with X_{V^*}.
- (b) There exists a session at V^* with X_{U^*} and X_{V^*} (but σ_{U^*} computed by U^* is different from the σ_{U^*} received to V^* with the protocol message, in the target session).
- (c) There exists a session at V^* with X_{V^*} and $X'_{U^*} \neq X_{U^*}$.

This is almost same as the Case 2.1. The difference is that in this case the initiator is the owner of the test session. Same as to the Case 2.1, here we obtain,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\mathrm{w}(\cdot)\text{AFL-eCK}}_{\pi,\text{Case 2.2}}(\mathcal{A}) \leq \max \Big[N_P^2 N_s^2 \big[\big(\mathrm{Adv}^{\mathrm{DDH}}_{q,g}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) \\ +2\mathrm{Adv}^{\mathrm{CPLA2}}_{\mathrm{PKE}}(\mathcal{D}) + 2\epsilon\big) + \frac{1}{q}\big], N_P^2 N_s^2 \big[\big(\mathrm{Adv}^{\mathrm{ODH}}_{q,g}(\mathcal{R}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) \\ +2\mathrm{Adv}^{\mathrm{CPLA2}}_{\mathrm{PKE}}(\mathcal{D}) + 2\epsilon\big) + \frac{1}{q}\big], N_P \mathrm{Adv}^{\mathrm{UFCMLA}}_{\mathrm{SIG}}(\mathcal{E})\Big].$$

Combining Case 1 and Case 2

According to the analysis we can obtain,

$$\mathrm{Adv}^{\boldsymbol{\lambda}-\mathrm{w}(\cdot)\text{AFL-eCK}}_{\pi}(\mathcal{A}) \leq \max \Big[N_P^2 N_s^2 \big[\big(\mathrm{Adv}^{\mathrm{DDH}}_{q,g}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J})\big) + \frac{1}{q}\big], \\ N_P^2 N_s^2 \big[\big(\mathrm{Adv}^{\mathrm{DDH}}_{q,g}(\mathcal{C}) + \mathrm{Adv}_{\mathrm{KDF}}(\mathcal{B}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) + 2\mathrm{Adv}^{\mathrm{CPLA2}}_{\mathrm{PKE}}(\mathcal{D}) + 2\epsilon\big) + \frac{1}{q}\big], \\ N_P^2 N_s^2 \big[\big(\mathrm{Adv}^{\mathrm{ODH}}_{q,g}(\mathcal{R}) + \mathrm{Adv}_{\mathrm{PRF}}(\mathcal{J}) + 2\mathrm{Adv}^{\mathrm{CPLA2}}_{\mathrm{PKE}}(\mathcal{D}) + 2\epsilon\big) + \frac{1}{q}\big], N_P \mathrm{Adv}^{\mathrm{UFCMLA}}_{\mathrm{SIG}}(\mathcal{E})\Big].$$

□