# Review on Leakage Resilient Key Exchange Security Models

Clement Chan Zheng Wei[1], Chuah Chai Wen[2], Janaka Alawatugoda[3]

[1,2]Information Security Interest Group(ISIG)
[1,2]Faculty of Computer Science and Information Technology, University Tun Hussein Onn Malaysia, Malaysia
[3]Department of Computer Engineering, Faculty of Engineering, University of Peradeniya, Peradeniya, Sri Lanka

***Abstract*:** In leakage resilient cryptography, leakage resilient key exchange protocols are constructed to defend against leakage attacks. Then, the key exchange protocol is proved with leakage resilient security model to determine whether its security proof can provide the security properties it claimed or to find out any unexamined flaw during protocol building. It is an interesting work to review the meaningful security properties provided by these security models. This work review how a leakage resilient security model for a key exchange protocol has been evolved over years according to the increasing security requirement which covers a different range of attacks. The relationship on how an adversary capability in the leakage resilient security model can be related to real-world attack scenarios is studied. The analysis work for each leakage resilient security model here enables a better knowledge on how an adversary query addresses different leakage attacks setting, thereby understand the motive of design for a cryptographic primitive in the security model.

***Keywords*:** Leakage-resilient cryptography, Key exchange protocol, Security models, Leakage attacks, Side-channel attacks, Leakage resilience

## 1. Introduction

Authenticated Key Agreement (AKA) protocols are well-known cryptography primitives. Generally, key exchange(KE) mechanism is executed by two parties to enable the generation of a common shared secret key to assuring a secure communication while authenticating each other. The shared secret key is crucial as the core of establishing a secure channel between communicating parties in protecting information transmission over the communication networks. The secure channel means with exists of any active or passive malicious third party, the communication between two parties are still being protected.

Canetti-Krawczyk (CK model) [10] and extended Canetti-Krawczyk (eCK model) [18] are two main standard of key exchange(KE) protocol security models that address different capabilities of polynomial time adversary. The standard security models are modelled to ensure that with the exist of polynomial time adversary, the security of the communicate between two parties which follows the protocol is preserved. Cryptography community researchers continuously proposed stronger and more efficient KE protocol to address stronger power polynomial time adversary. This has directly affected the development of stronger security models to prove KE protocols are secure.

To date, a cryptographic scheme is based on Kerckhoffs's principle where the cryptographic ciphers are public, but the cryptographic keys, internal state computations and session randomness information are kept opaque to adversary. Leakage-resilient cryptography aims to construct a cryptographic scheme that proven to be secured even leakage attacks happen. The example of leakages are long term secret key leakage, ephemeral secret keys leakage, information of parameters leakage involved in computations, cache and memory information leakage and also side channel information leakages.

The definitions of two security models with its security properties (1) CK and (2) eCK are reviewed. As stronger adversarial power surface in the real world, CK model failed to address possible attacks, this leads to the development of eCK model which cover a wider range of adversary attacks. However, side channel attacks exist to further impose stronger leakage attacks resulting in the need for stronger design extension on eCK security model. Literature [8-9, 16, 19, 25] study several examples of side-channel attacks. Side channel attacks which can be exploited to sabotage proven cryptographic schemes and protocols including secure key exchange protocol in leakage susceptible security models.

Therefore, this work here is to examine six existing proposed leakage-resilient security models (1) leakage resilient in CK security model (LR-CK) by Yevgeniy Dodis et al. [13], (2) leakage resilient authenticated key exchange protocol in eCK security model (LR-eCK) by Daisuke Moriyama and Tatsuaki Okamoto [20], (3) bounded after-the-fact leakage eCK model (BAFL-eCK) [3] and (4) continuous after-the-fact leakage eCK model (CAFL-eCK) [4] which are instantiated from after-the-fact leakage eCK model ((•)AFL-eCK) by Alawatugoda [3], (5) continual leakage eCK model naming (GCL-eCK) game by Jui-Di Wu et al. [23], (6) challenge-dependent leakage resilient eCK (CLR-eCK) by Rongmao Chen et al. [11].

Different proposed security models and protocols have their respective underlying cryptographic preliminaries and security properties. It is an interesting work to formally review the preliminaries definitions used to construct the key exchange protocol security models. Hence, this information may use to propose more stronger security notions for KE protocols.

The paper by Cremers [12] is referred for the detailed analysis of security model CK and eCK. Noted that both CK and eCK security models work based on indistinguishability game-based approach, where the adversary is needed to identify between the computed real session key and an independent randomly chosen random string. The notion of indistinguishability of session keys requires the attacker cannot distinguish the shared secret key from a random string. Security model is built to determine whether the security proof of a key exchange protocol can provide the security properties it claimed to have or any security flaw its missed out during protocol building, if an attacker success (win the security game), either the security proof is wrong or the attack is outside the security model.

In this paper, six different protocols [3, 4, 11, 13, 20, 23] which address meaningful security properties based on their defined base security model had been reviewed. The differences between security properties respective to their proposed leakage setting are addressed and also discuss the addressed adversary capabilities by these protocols. This work study how a security model for key exchange protocol has been evolved over years according to the increasing security requirement thus review the relationship between proposed key exchange protocol security models and real-world scenarios.

The review works are proceeded as below, section 2 discusses the concept of indistinguishability-based security models for KE protocols and describes the CK and eCK models in brief details. In section 3, different proposed KE protocols based on their base security models and cryptographic preliminaries are discussed. In section 4, table are tabulated to show a comparison of leakage setting between existing KE protocols and shows a comparison in term of adversary capabilities between existing KE protocols.

## 2. Security Models for Key Exchange Protocols

Section 2 provides an overview of security models for KE protocol discussing CK and eCK in brief details.

Before addressing specific security model for key exchange protocols, a formal basic notion is established to define the general understanding about security model. In general, the main interests of a security model are (1) intended communication party having a fresh matching session and must compute the same session key, and (2) the session key of the test session must be indistinguishable from a random string with same length where the adversary may distinguish the session key from the random string with negligible probability. To address the aim in which intended communication parties are indeed session partner, a notion define as matching session or partnering session is assumed. The matching session is defining as a form of correctness for KE protocol where the matching session must always compute the same session key. Matching session is assumed as protocol correctness because there exists a certain query that allowed to reveal information about session-keys based on session matching relations. Another notion defines as session freshness is to ensure a matching session stay unique, hence addressing replay attacks on KE protocol. Different security models have different matching session prerequisite and will be discussed in detail in section 2.2 and 2.3.

For adversary, there exist two types of adversary which are passive adversary and active adversary. In passive adversary (passive session), the adversary does not have access to modify or change protocol messages between the communication of two parties. In active adversary (active session), the adversary has the power to forge the protocol messages between the communication of two parties. The adversary for both types then uses the internal state information it acquires according to their capabilities to aid in the distinguishability game and guess the real session key from a random string. Both types of adversary have their differences in term of query power respectively to its security models and will also be discussed in detail in section 2.2 and 2.3.

### 2.1 Security Properties

#### *A.* Unknown Key Share Security

Unknown Key Share(UKS) security mentioned that a party $\boldsymbol{A}$ in a protocol execution should never wrongly believed it is sharing a session key with its intended party $\boldsymbol{B}$ but instead it is actually sharing a session key with another third party $\boldsymbol{C}$. This security property was first defined by Blake-Wilson et al. [7], it is an attack on authenticated key agreement with key confirmation protocol. This property is also related to 'Identity-Misbinding Attack'.

#### *B.* Known Key Security

For Known Key Security, any session key computed on a session should never have any bit of relation with other session keys computed on the other sessions. Thereby, if occurred a compromisation of a session that leaks any information regarding of the session keys can never allow the adversary to learn the session key from another session. Each security of different session is maintained even when other sessions are compromised.

#### *C.* Forward Secrecy

Forward Secrecy security is achieved only if an adversary who obtains the secret keys(long-terms) of a party or its partner of a session is unable to compute the session keys of past sessions between these two parties. The compromise of the long-term secret keys should never compromise the past session keys, even if a party is corrupted, nothing can be learned about the unexposed past sessions within the party [22].

#### *D.* Key Compromise Impersonation Resilience

Key Compromise Impersonation(KCI) is introduced by Blake-Wilson, Johnson and Menezes [6]. Key compromise impersonation attacks allow an adversary who had successfully compromised the long-term secret key of a party to impersonate the compromised party identity to other party or server, and also to impersonate any other party or server to the compromised party. Key compromise impersonation resilience is where an adversary who knows the secret key of a party should never able to impersonate other honest parties identity to the compromised party.

#### *E.* Ephemeral Key Leakage Resilient

An ephemeral key of a key exchange protocol is the key that is generated every time for each protocol execution or can also be denoted as session per randomness. Ephemeral key leakage addresses attacks which involved an adversary who able to reveals the internal state information of a party such that revealing the ephemeral secret key(randomness used in a specified session). Ephemeral key leakage vulnerabilities examples are weak random number generators, ephemeral secrets leakage specific malware attacks, that compromise the randomness factor of a session.

### 2.2 Canetti-Krawczyk (CK) Model

In CK model [10], session identifier for protocol participant in each session is a requirement to activate any matching session. More specifically, the input of a KE protocol for party $\boldsymbol{P_A}$ hold a general form of ($\boldsymbol{P_A}$, $\boldsymbol{P_B}$,$\boldsymbol{s}$, $\boldsymbol{role}$), in which $\boldsymbol{P_B}$ is the identity of another party, where $\boldsymbol{s}$ is the session id, $\boldsymbol{role}$ can be represented as either *initiator* or *responder* of a protocol execution. A complete matching session between $\boldsymbol{P_A}$ and $\boldsymbol{P_B}$ have a matching input in the form of ($\boldsymbol{P_A}$, $\boldsymbol{P_B}$, $\boldsymbol{s}$, $\boldsymbol{initiator}$) and ($\boldsymbol{P_B}$, $\boldsymbol{P_A}$, $\boldsymbol{s}$, $\boldsymbol{responder}$). The session id, $\boldsymbol{s}$ in two

different key exchange sessions must not be identical even at the same protocol participant.

There are three notable reveal queries defined as Session Key Reveal, Session State Reveal and Corrupt query:

- Session Key Reveal aims to reveal the session key of a completed session.
- Session State Reveal aims to reveal the internal state information (ephemeral parameters) of a session. CK model as general does not particularize the specific contents of the internal state information of a session, but instead requires the KE protocol to specify the internal information explicitly. The only requirement is that the internal state information cannot contain long-term secret of the protocol participants.
- Corrupt query aims to reveal complete internal state information of protocol participants (ability to overwrite any value of choices for the long-term secret key of the corrupted protocol participant).

CK security model restricts a subset of possible sequences of queries in order to maintain the freshness of a session. A term 'locally exposed' is mention by Ran Canetti [10] to define a session is no longer fresh. If a session or its matching session is locally exposed, then the session is compromised. To announce the session is locally exposed in CK model, the adversary must perform one of the following queries:

- Session State Reveal query on a session
- Session Key Reveal query on a session
- Corrupt a protocol participant before the session expired

To specify the KE protocol to be secure in CK model, the requirements that must have are:

1. Two uncorrupted parties have fresh CK-matching session and output the same session key.
2. Exist no probabilistic polynomial time (PPT) adversary that can win the indistinguishability game approach with non-negligible probability.

There exist few security properties addressed in CK model which related to real-world adversary scenarios. Session Key Reveal query in CK model is to address "Known Key Security". Known Key Security stated that any session key should never have any relation with other session keys of the other sessions. Thus, knowing any information regarding of a certain session key can never allow the adversary to learn the session key from another session. The Corrupt query in CK model addresses "Unknown Key Share" where party $\boldsymbol{A}$ believes it is sharing a session key with party $\boldsymbol{B}$ but instead it actually shares a session key with party $\boldsymbol{C}$. That is, if a session key is shared between parties $\boldsymbol{A}$ and $\boldsymbol{B}$, the corruption of another party $\boldsymbol{C}$, should never expose any information of the session key shared between $\boldsymbol{A}$ and $\boldsymbol{B}$. Similarly, in corrupt query, no matter of whether an adversary had actively interacted with the session protocol, the adversary can acquire the long-term secret key of protocol participants, thus CK security model address "Perfect Forward Secrecy". Whereas, in CK model before a session is expired, the adversary is never authorized to study the long-term secret key of the protocol participants, hence CK model did not address "Key Compromise Impersonation".

### 2.3 Extended Canetti-Krawczyk (eCK) Model

In eCK model [18], session identifier is defined to consist the identities of two protocol participants and the information they exchange. The session identifier is recognized in a general form of $\boldsymbol{sid}$=($\boldsymbol{role}, \boldsymbol{ID}, \boldsymbol{ID}^*, \boldsymbol{comm_1}, \ldots, \boldsymbol{comm_n}$), where ID denotes the identity of protocol participant executing the session meanwhile ID* denotes the identity of the other protocol participant, $\boldsymbol{role}$ can be represented as either *initiator* or *responder* of a protocol execution and $\boldsymbol{comm_n}$ denote the $\boldsymbol{n}$-th communication between the protocol participants. $\boldsymbol{sid}$ denotes as KE session completed by party $\boldsymbol{A}$ with its partner party $\boldsymbol{B}$, $\boldsymbol{sid}^*$ denotes the matching session to $\boldsymbol{sid}$ which executed by party $\boldsymbol{B}$.

There are three notable reveal queries defined as Session Key Reveal, Long Term Key Reveal and Ephemeral Key Reveal query:

- Session Key Reveal aims to reveal the session key of a completed session.
- Long Term Key Reveal aims to disclose the long-term secret key (static key) of the protocol participant.
- Ephemeral Key Reveal aims to reveal the ephemeral secret key of a session.

In eCK security model, there exist a subset of possible queries sequences to keep the session fresh. If a session $\boldsymbol{sid}$ or its matching session $\boldsymbol{sid}^*$ in eCK is not clean then the session is considered exposed. A session in eCK is no longer clean (analogous to the term "locally exposed" in CK model) if and only if the adversary performed such condition:

- Session Key Reveal query on a session $\boldsymbol{sid}$ or its matching session $\boldsymbol{sid}^*$(if exist)
- Matching session $\boldsymbol{sid}^*$ does not exist and adversary run either Long Term Key Reveal on party $\boldsymbol{B}$ or both Long Term Key Reveal and Ephemeral Key Reveal on party $\boldsymbol{A}$
- Matching session $\boldsymbol{sid}^*$ exist and adversary run either both Long Term Key Reveal and Ephemeral Key Reveal on party $\boldsymbol{A}$ or both Long Term Key Reveal and Ephemeral Key Reveal on party $\boldsymbol{B}$
- Party $\boldsymbol{A}$ or party $\boldsymbol{B}$ is corrupted where both long term key and ephemeral key is reveal(adversary-controlled)

To specify a KE protocol to be secure in eCK model, the requirements that must have are:

1. Two uncorrupted parties have fresh eCK-matching session and output the same session key.
2. Exist no probabilistic polynomial time(PPT) adversary that can win the indistinguishability game approach with non-negligible probability.

There exist few security properties addressed in eCK model which related to real-world adversary scenarios. Ephemeral Key Reveal query in eCK model allows an adversary to reveal the internal state of a protocol participant such as revealing the ephemeral secret key (randomness used in a specified session) provided the long term secret key is not leaked. This query can be related to real-world malware attacks where it covers the malware attack [15] which steals ephemeral keys, such as hardware modules which the long term secret key is stored separately from the ephemeral key. Long Term Key Reveal in eCK model addresses "Unknown Key Share" and "Key Compromise Impersonation" because this query let the adversary to reveal the long-term secret key of protocol participants before a session is expired. Whereas, after the session is activated, the long-term secret key of both protocol participants is able to be learnt by the adversary only if the adversary does not actively interfere the protocol session, so eCK security model only address "Weak Forward Secrecy". Session Key Reveal query in eCK model addresses "Known Key Security" where it covers attack which attaches by

knowing the past session keys. Known Key Security requires that any session key should never have any relation with other session keys of the other sessions. Thereby, the leakage of past session keys will never allow attacker to get any knowledge about other session keys of another session.

### 2.4 Differences In Term Of Security Properties

CK model does not include "Key Compromise Impersonation" properties because an adversary is not allowed to obtain the long-term secret key of the protocol participants before a session is expired. eCK model only includes "Weak-Perfect Forward Secrecy" because the long-term secret key of both protocol participants is able to be learnt by the adversary if and only if the adversary has not actively interacted with the session. One primary difference between CK model and eCK model is that CK model does not address the concept of ephemeral secret keys leakage meanwhile eCK model address the leakage of ephemeral secret keys. Ephemeral key leakages address the weak random number generators attacks in a cryptosystem where attacker can obtain or determine the randomness generated correctly with advantageous probability. These differences are shown in Table 1.

**Table 1.** Differences In Term of Security Properties of CK and eCK model

| Security Properties | Canetti-Krawczyk (CK) model [10] | Extended Canetti-Krawczyk (eCK) model [18] |
|---|---|---|
| **Unknown Key Share** | Allow | Allow |
| **Known Key Security** | Allow | Allow |
| **Forward Secrecy (FS)** | Perfect-FS | Weak-PFS |
| **Key Compromise Impersonation** | Not Allow | Allow |
| **Ephemeral Key Leakage** | Not Allow | Allow |

The differences between Weak-PFS and Perfect-FS is that weak perfect forward secrecy is a weaker security notion such that the security of established session keys is guaranteed when the protocol participants long term secret key are compromised if and only if the adversary did not actively interfere with the session execution. Full Perfect-FS require the adversary did actively interfere with the session and the established session keys can still remain secure. This security notion was introduced in [17] by Hugo Krawczyk.

## 3. Security Models For Key Exchange Protocols In Addressing Leakage

Section 3 shows a detail review of security models for key exchange protocols according to their respective cryptographic preliminaries and leakage setting. This section also discusses the strength of adversary power relative to the real-world capabilities.

### 3.1 LR-CK Model

LR-CK security model [13] proposed by Yevgeniy Dodis et al. [13] show a construction of authenticated key exchange protocol with Relative Leakage Model setting. Relative Leakage Model formalized by Adi Akavia et at. [1] implicitly assumes, there exists a leakage attack that discloses a portion of secret key regardless of the secret key size. LR-CK was designed to resist cache side-channel attacks also referred to as "memory attack". Memory attacks is an attack based on inside information obtained from the observation of cryptographic implementation in a computer system. Relating to the real world scenarios, even an adversary only learns a small amount of information about the secret key via memory attack, it still gives the adversary an extra advantage on breaking the cryptosystem without violating any underlying cryptographic primitives. LR-CK proposed a concept on a setting where protocol can only be assumed secure if and only if the adversary learns a bounded size of secret key respective to leakage parameter defined in the system.

Yevgeniy Dodis et al. [13] constructs an authenticating Diffie-Hellman (DH) key exchange based on leakage-resilient signature scheme. Their first construction follows result from Joël Alwen et al. [5] a protocol "eSig-DH" authenticated with a leakage-resilient signature scheme where a protocol principal authenticates to his protocol partner by signing the message he received from him. The next construction is another protocol "Enc-DH" based on leakage resilient CCA-secure PKE scheme (Leakage Resilient Chosen Ciphertext Attack-secure Public Key Encryption Scheme). The second construction is a modification on DH key exchange where it requires both protocol principal to authenticate to each other by accurately decrypt the DH ephemeral public key encrypted under the long-term public key. This idea provides authentication via public key encryption, only the protocol principal with the correct secret key is able to accurately decrypt the ciphertext encrypted under the corresponding public key.

LR-CK model does not allow leakage query during the execution of the challenge session, and both of it construction contain 3-rounds of authenticated key exchange protocol. LR-CK is an extension of CK security model, hence it possesses security properties of CK model.

### 3.2 LR-eCK Model

Daisuke Moriyama and Tatsuaki Okamoto [20] proposed a leakage resilient model for authenticated key exchange protocol named LR-eCK. This model also formalized by the notion of Relative Leakage Model by Adi Akavia et at. [1] and is an extension of eCK security model by Brian LaMacchia et al. [18]. In their paper, they stated that NAXOS trick in [18] is not necessary to be implemented for archiving eCK security. NAXOS trick 'hides' the exponent of ephemeral public key $\boldsymbol{X}$ by computing $\boldsymbol{X} = \boldsymbol{g}^{\tilde{\boldsymbol{x}}}$, where $\tilde{\boldsymbol{x}}$ is the hashing of ephemeral private key $\boldsymbol{x}$ and static private key $\boldsymbol{a}$, i.e., $\tilde{\boldsymbol{x}} = \boldsymbol{H}(\boldsymbol{x}, \boldsymbol{a})$, additionally the hashed value $\tilde{\boldsymbol{x}}$ is not stored but computed every time when required. There exists an attack which is power analysis side channel attack that allows an adversary to observe the exponent of the ephemeral public key. Hence, this technique is not secure against in real-world scenarios like power analysis side channel attack. To overcome this weakness, they proposed a model called LR-eCK without NAXOS trick.

This LR-eCK construction without NAXOS trick is modified based on the Okamoto protocol by Tatsuaki Okamoto [21] where the differences between the two are the component used in the static public key and static private key. The protocol execution is done in a 2-rounds authenticated key exchange protocol. The base security model of LR-eCK is based from eCK, so this model follows the eCK security properties by addressing the leakage of either the ephemeral secret key or long-term secret key leakage but not both together. Also, LR-eCK is modelled under challenge-independent leakage setting which does not consider leakage query during the challenge session.

LR-eCK applies pair-wise independent pseudo-random function family, collision resistant hash function family and under the Decision Diffie-Hellman (DDH) assumption without random oracle. The queries available to be executed by the adversary is the same in eCK security model but with one extra LR-eCK specific query. The specific queries available in LR-eCK model which is, *EstablishParty* which allows the adversary to register a static public key on behalf of a protocol party. The party is denoted as adversary-controlled.

### 3.3 (•)AFL-eCK Model

After-the-fact leakage eCK model ((•)AFL-eCK) by Alawatugoda [3] can be instantiated into Bounded-after-the-fact leakage-eCK model (BAFL-eCK) and Continuous-after-the-fact leakage-eCK model (CAFL-eCK). In both of the instantiated model of (•)AFL-eCK model, the differences can be seen on both the freshness condition of the protocol session which varies according to the leakage function permit. Both instantiated share the same similarities where the partnering notion and adversary queries are identical. BAFL-eCK bounds the overall amount of leakage information of ephemeral secret keys or long-term secret key for the whole protocol execution. Meanwhile, the total amount of leakage information of ephemeral secret keys or long-term secret key allowed in CAFL-eCK is unbounded but the amount of disclosure in each protocol execution occurrence is fixed.

#### 3.3.1 BAFL-eCK Model

Bounded-after-the-fact leakage-eCK model (BAFL-eCK) by Alawatugoda [3] is modelled to capture long-term secret key leakage taking place after the activation of test session. It allows an adversary to submit queries after the session key is established but with a split-state model restriction. In the split state model, the secret state of a cryptosystem is partitioned to parts and the adversary can obtain leakage of its choice on every split part independently but not global leakage from the entire secret key. Execution of cryptographic primitives in split-state mode means that the execution is split into a sequential series of stages and every of these stages used a segment of the secret key. A leakage function is then to stimulate in each occurrence of split stages.

There exists a query in BAFL-eCK which embedded the leakage function within the query:

- Send query that allows the adversary to run the protocol by instructing the protocol principal to execute the protocol messages by following the protocol specification, then sends it back to the adversary together with the leakage function.

By issuing Send query which having a leakage function embedded into it, the adversary can get information about the long-term secret key in bounded-form. Specifically, the total amount of long-term secret key which can be learnt by the adversary is bounded within a leakage parameter (leakage parameter varies between different primitives) for each split of secret key. One notable mention of differences within queries in this model is Corrupt query and Send query. Corrupt query here grants the adversary to study the entire protocol principal long-term secret key where Send query provides the adversary to study bounded leakage amount of long-term secret key (using the leakage function embedded within the query).

BAFL-eCK secure KE protocol applies CPLA2-secure public key cryptosystems (adaptively chosen plaintext leakage attack) with special property which is pair generation indistinguishability (any randomly chosen ciphertext should be decrypted without rejection). Then the protocol message is signed and authenticated by unforgeable against chosen message leakage secure signature scheme (UFCMLA) to prevent attacker simply replace original protocol messages. Alawatugoda [2] builds a BAFL-eCK secure key exchange protocol, the EphemeralKeyReveal query in the proposed BAFL-eCK does not allow the adversary to obtain the randomness used during the signing process of the session. This is because leakage-resilient signature schemes should not allow the full leakage of randomness used. A modification is made to cater use of available leakage-resilient signature scheme in protocol instantiation had weakened the security model to **w**BAFL-eCK to fit the assumption where EphemeralKeyReveal query cannot disclose the randomness used to compute the signature. This important arrangement is missed out in [3] by Alawatugoda. Yang et al. [24] also mentions in their paper the fact that where a mismatching sessions can compute the same session keys if full leakage on randomness in signing signature is allowed.

Relating to real-world scenarios, BAFL-eCK is modelled to address cold boot attacks. Cold boot attack is a type of side channel attack in which attackers can retrieve the ephemeral secret keys or the long-term secret key of protocol principal through physical access to the cryptosystem and force a cold reboot. The base model of BAFL-eCK is eCK model, so this model exhibits the security properties of an eCK security model which address ephemeral secret key or long-term key leakage possible attacks. Corrupt query which lets the adversary to obtain the long-term secret key from protocol principal address malware attack that leaks or steal the long-term secret key.

#### 3.3.2 CAFL-eCK Model [4]

Continuous-after-the-fact leakage-eCK model (CAFL-eCK) by Alawatugoda [4], is similar to what they proposed in BAFL-eCK but with a specific difference which the adversary can obtain continuous leakage of each split of secret key with the restriction where the leakage amount per occurrence is bounded by a leakage parameter(leakage parameter varies between different primitives). The restriction is enforced to bound the amount of leakage per occurrence.

The first concrete and secure continual leakage key exchange protocol is proposed by Alawatugoda et al. [4] that employ a key refreshing technique called inner-product extractor method by Dziembowski and Faust [14]. The key exchange protocol is proved under CAFL-eCK security model. This model consists of a leakage resilient split-state storage that split elements into two parts using a randomized encoding technique. Such encodings then can be refreshed in a leakage resilient way so that the newly generated part can be used again, thereby achieve a continuous leakage resilient primitive setting. Each of the cryptographic keys is split into sequential part and used separately in different stages.

There exist also a CAFL-eCK query which embedded a leakage function within the query:

- Send query that allows the adversary to run the protocol by instructing the protocol principal to execute the protocol messages by following the protocol specification, then sends it back to the adversary together with the leakage function.

Similarly, with BAFL-eCK, the CAFL-eCK let the adversary submit queries after the session key is established but with a split-state model restriction. In real-world scenario, CAFL-eCK security model can be implemented as a framework for a key exchange protocol that addresses continuous leakage side-channel attacks such as EM radiation and power analysis. These leakage attacks leak internal secret information continuously once any computation takes places on the cryptosystems.

### 3.4 GCL-eCK Model [23]

Jui-Di Wu et al. [23] proposed an efficient leakage resilient AKA protocol in Continual Leakage eCK model is proposed. This security model allows an overall boundless leakage setting in continual leakage eCK mode. This model is an improvement of [4] from Alawatugoda et al. which employed a key refreshing technique [14] that adopts an inner-product extractor method to update the long-term secret keys after each protocol execution. In GCL-eCK model, Jui-Di Wu et al. employ a different technique called multiplicative blinding to replace the time-inefficient inner product extractor method.

This security model is established in a split-state restriction. GCL-eCK allows the adversary to continuously disclose a determinate amount of leakage for every protocol session split while possessing an overall boundless amount of leakage for the whole cryptosystem lifecycle. To achieve the overall boundless leakage setting during the whole cryptosystem lifecycle, a cryptographic scheme usually consists of different computation rounds whereby each long-term secret key involved will be refreshed for every computation round. The long term secret key leakage of each round is independent with each other and the leakage information of every computation is fixed to a portion of amount.

There exist a specific GCL-eCK query to address overall unbound leakage setting:

- Leak query that allows adversary to acquire fractional leakage information about the ephemeral secret keys and long term secret keys involved in both the key-refreshing and key-agreement phase.

GCL-eCK model address the real-world attacks scenarios which relate to continual leakage setting. This model implicitly addresses any kinds of side channel attack that continuously leaks detail of the parameters involved in a cryptosystem computation. Leak query in GCL-eCK model is defined exactly to tackle continuous fractional leakage attacks of ephemeral secret keys and long term secret keys. Each continuous ongoing executions of cryptosystem computation have a probability to reveal even a small amount of secret information, and this model is established to provide security in such continual leakage setting.

### 3.5 CLR-eCK Model [11]

Rongmao Chen et al. proposed an authenticated key exchange security model name challenge dependent leakage resilient eCK (CLR-eCK) model. Their security model is modelled to capture both leakages on long-term secret key and ephemeral secret key together. To be specific, CLR-eCK allows one (long-term/ephemeral) secret key to be partially leaked and the other (ephemeral/long-term) secret key to be completely leaked. This model is a split-state-free model that allow leakage queries to be run before, during and after the challenge session, hence achieve the notion of challenge dependent leakage. In real world scenario, there may exist an attacker which able to disclose one (ephemeral/long-term) secret key and obtain partial disclosure on the other (ephemeral/long-term) secret key. In their proposed model, they also consider leakage setting which is Relative Leakage Model by Adi Akavia et at. [1]. The framework can be considered as an alternative version of authenticated key exchange model proposed by Tatsuaki Okamato et al. [20].

Cryptographic preliminaries such as pseudorandom function, pair-wise independent pseudo-random function family and strong randomness extractors are applied in the construction of the general framework for CLR-eCK, then they present a general framework that practically instantiated under DDH assumption without random oracle or more specifically Rongmao et.al. employ an (extended) smooth projective hash function (SPHF) based on DDH assumption. CLR-eCK is an extension of eCK security model hence it possesses the security properties of eCK model and also obtains more leakage information than eCK model which allows the adversary to submit leakage queries during the activation of challenge session.

To capture the notion named Challenge-Dependent Leakage there are 2 specific queries available only in CLR-eCK model which are:

- LongTermKeyLeakage that allows an adversary to ask for arbitrary leakage function of the long-term secret key of the party with the condition of before it obtains the ephemeral secret key(which is by issuing Ephemeral Key Reveal).
- EphemeralKeyLeakage that allows an adversary to ask for arbitrary leakage function of the ephemeral secret key of the session with the condition of before it obtains the long-term secret key(which is by issuing Long Term Key Reveal).

Additionally, the adversary is allowed to adaptively choose the leakage functions (LongTermKeyLeakage or EphemeralKeyLeakage) during and even after the challenge phase as long as the restriction hold.

## 4. Comparison According To Leakage Setting

Table 2 summarizes the comparison of security properties mentioned of each security model reviewed in Section 3. All security models in table 2 are formalized with Relative Leakage Model by Adi Akavia et at. [1] which implicitly bring up a concept stating leakage attack that reveals a portion of secret key regardless of the secret key size.

CAFL-eCK model in AFK-eCK is a continuous leakage variant which possesses overall boundless leakage in protocol execution but with a fixed amount for each occurrence. Such leakage setting is interesting where a leakage-resilient storage cryptographic primitives are used to instantiate a generic protocol for continuous leakage variant in the proposed AFK-eCK. In CAFL-eCK, the model employs a key refreshing technique called inner-product extractor method. This technique used an encoding scheme to split key into parts then refresh it to be reused again. As in GCL-eCK model, this model had successfully provided a generic protocol for continual leakage setting in eCK model using a splitting storage idea call multiplicative blinding technique for key refreshing. The idea of this technique is to split the secret key into partition of different parts while the leakage of each partition is independent to each other. Then, the secret key will

be refreshed once the key is involved in a session key execution thereby achieving overall unbound setting. Multiplicative blinding key refreshing technique provides a better computation efficiency than the time-consuming inner-product extractor method.

(•)AFL-eCK model, GCL-eCK model and CLR-eCK model achieve a notion proposed by Rongmao Chen [11], Challenge-Dependent leakage where the adversary can submit queries especially during and after the challenge session is activated.

Every proposed security model in Table 2 above tries to address as many possible attacks including side-channel attacks. This shows researchers in above literature are giving significant attention to side-channel attacks such as EM radiation, cache-attack, power analysis, cold boot attacks and memory attacks. They study intensively and modelled a leakage setting to prevent the leakage of information from side channel attacks. In GCL-eCK, the model tackles a continual leakage setting with eCK base model. For CLR-eCK, they study a more extensive leakage area where both the leakage of ephemeral secret keys and long-term secret key. This shows the proposed security model in Table 2 is an extension to cover a wider range of attacks compared with the base models (CK or eCK).

Table 3 compare the adversary capabilities in term of query submission for LR-CK, LR-eCK, BAFL-eCK, CAFL-eCK, GCL-eCK and CLR-eCK security models. The adversary is modelled as a PPT adversary that able to activate and controls every communications message between the protocol participants. The adversary is given the power to schedules the activations of parties and control the destination or request of message delivery during a session. In (•)AFL-eCK, the activation query is embedded with a leakage function to get leakage information. Such capabilities can be related as an attacker with full control of communication link such as man-in-the-middle attack.

All of the queries submission in Table 3 must follow a rule where the adversary is restricted to only submit a subset of possible queries sequences to keep the session fresh. This restriction exists in both base model CK and eCK, whereby the proposed security models in Table 3 are an extension of these 2 base models, hence lies the necessity to comply with the session freshness rule.

Adversary is also allowed to reveal the session keys for a protocol execution in above-proposed security models. This capability is connected with real-world interleaving attacks where the leakage of secret keys from one session can help to get another key in other sessions. The adversary is able to schedules session keys at will.

The capability of learning the long-term secret keys of the protocol participants by corrupting the protocol participants is mentioned in security models in Table 3. The difference is the naming of the query used by difference proposed security model varies. In LR-CK model, the adversary is not allowed to submit the query and learn the long-term secret key before the session is completed because this is the restriction set based on the base model(CK).

Table 3 also presents an adversary capability query of the learning of session states or ephemeral secret keys of the target session or its partner session in a protocol execution. In LR-CK model, the adversary can receive the all internal state information (excluding long-term secret) of the session and is named as Session-State Reveal Query. Meanwhile, in LR-eCK, (•)AFL-eCK, GCL-eCK and CLR-eCK, instead of allowing the adversary to completely reveal the internal state information of the protocol like CK model, these eCK-based models define a new query naming Ephemeral Key Reveal to specifically disclose only ephemeral secret key in a specified session.

In LR-eCK and CLR-eCK models, both security models had explicitly stated Establish Party query which provides the adversary with the power to establish a public key for protocol participant. Adversary allows to register a long-term public key for a protocol participant and the party will be denoted as dishonest party or adversary-controlled party.

## 5. Conclusions

In this paper, an overview on base model CK and eCK for key exchange protocol is shown. For each model, there are different matching session prerequisite and session freshness to be abided for achieving the desired security properties. Each base model also provides different adversary query power related to key exchange security properties that capable to address maximum possible known attacks.

Six protocols had been reviewed in this paper where all of these six are either an extension of CK base model or eCK base model. Each of these protocol extensions claims to propose distinct security notion to capture different leakage setting, and also the adversarial capabilities are modelled as strong as possible to cover a wider range of leakage attacks. However, there is no single strongest model to suit every leakage scenario but only the most appropriate model to be deployed to target specific domain type of leakage setting. Such protocols may become insecure if the adversary is able to perform leakage attacks outside the scope of the security model provide. This work aims to give a better understanding of which leakage security model is the most suitable to be used to tackle the desired leakage setting.

## 6. Acknowledgement

This work is supported and funded by the Tier H082, Research Management Centre (RMC), of Universiti Tun Hussein Onn Malaysia (UTHM). Thank RongMao Chen for his helpful comments during the work of this paper.

## References

[1] A. Akavia, S. Goldwasser, and V. Vaikuntanathan, "Simultaneous hardcore bits and cryptography against memory attacks," in *Theory of Cryptography Conference*, pp. 474–495, 2009.

[2] J. Alawatugoda, "On the leakage-resilient key exchange," *Journal of Mathematical Cryptology*, Vol. 11, No. 4, pp. 215–269, 2017.

[3] J. Alawatugoda, D. Stebila, and C. Boyd, "Modelling after-the-fact leakage for key exchange," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, pp. 207–216, 2014.

[4] J. Alawatugoda, D. Stebila, and C. Boyd, "Continuous after-the-fact leakage-resilient eCK-secure key exchange," in *IMA international conference on cryptography and coding*, Vol. 9496, pp. 277–294, 2015.

[5] J. Alwen, Y. Dodis, and D. Wichs, "Leakage-resilient public-key cryptography in the bounded-retrieval model," in *Advances in Cryptology-CRYPTO 2009*, pp. 36–54, 2009.

[6] S. Blake-Wilson, D. Johnson, and A. Menezes, "Key agreement protocols and their security analysis," in *IMA international conference on cryptography and coding*, pp. 30–45, 1997.

[7] S. Blake-Wilson and A. Menezes, "Unknown key-share attacks on the station-to-station (STS) protocol," in *International Workshop on Public Key Cryptography*, pp. 154-170, 1999.
[8] D. Boneh, R. A. Demillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 37–51, 1997.
[9] D. Brumley and D. Boneh, "Remote timing attacks are practical," *Computer Networks*, Vol. 48, No. 5, pp. 701–716, 2005.
[10] R. Canetti and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 453–474, 2001.
[11] R. Chen, Y. Mu, G. Yang, W. Susilo, and F. Guo, "Strongly leakage-resilient authenticated key exchange," in *Cryptographers' Track at the RSA Conference*, pp. 19–36, 2016.
[12] C. Cremers, "Examining indistinguishability-based security models for key exchange protocols," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pp. 80-91, 2011.
[13] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs, "Efficient public-key cryptography in the presence of key leakage," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 613–631, 2010.
[14] S. Dziembowski and S. Faust, "Leakage-resilient cryptography from the inner-product extractor," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 702–721, 2011.
[15] K. Kasslin, "Kernel malware: The attack from within," *Computer Security Journal*, Vol. 23, No. 1, pp. 43–59, 2007.
[16] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *Journal of Cryptographic Engineering*, Vol. 1, No. 1, pp. 5–27, 2011.
[17] H. Krawczyk, "HMQV: A High-Performance Secure Diffie-Hellman Protocol," in *Annual International Cryptology Conference*, pp. 546–566, 2005.
[18] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger Security of Authenticated Key Exchange," in *International Conference on Provable Security*, pp. 1–16, 2007.
[19] J. Longo, E. De Mulder, D. Page, and M. Tunst All, "SoC it to EM: Electromagnetic side-channel attacks on a complex system-on-chip," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 620–640, 2015.
[20] D. Moriyama and T. Okamoto, "An eCK-Secure Authenticated Key Exchange Protocol without Random Oracles," in *International Conference on Provable Security*, pp. 154–167, 2009.
[21] T. Okamoto, "Authenticated Key Exchange and Key Encapsulation Without Random Oracles," in *Cryptology ePrint Archive*, Vol. 2007, pp. 473-484, 2007.
[22] Thomas Wu, "The Secure Remote Password Protocol," *Proceedings of the Symposium Network and Distributed System Security*, Vol 98, pp. 97-111, 1998.
[23] J. Di Wu, Y. M. Tseng, and S. S. Huang, "Efficient Leakage-Resilient Authenticated Key Agreement Protocol in the Continual Leakage eCK Model," *IEEE Access*, Vol. 6, pp. 17130–17142, 2018.
[24] Z. Yang and S. Li, "On security analysis of an after-the-fact leakage resilient key exchange protocol," in *Information Processing Letter*, Vol. 116, o. 1, pp. 33–40, 2016.
[25] R. Aouinatou, M. Belkasmi, and M. Askali, "A dynamic study with side channel against an identification based encryption," in *International Journal of Communication Networks and Information Security*, Vol. 7, No. 1, 2015.

**Table 2.** Comparison of security properties according leakage setting

| **Security model setting** | **LR-CK** [13] | **LR-eCK** [20] | **(•)AFL-eCK** [3] | | **GCL-eCK** [23] | **CLR-eCK** [11] |
|---|---|---|---|---|---|---|
| | | | **BAFL-eCK** | **CAFL-eCK** | | |
| **Base Models** | CK model | eCK model | eCK model | eCK model | eCK model | eCK model |
| **Leakage Models** | Relative Leakage | Relative Leakage | Relative Leakage | Relative Leakage | Relative Leakage | Relative Leakage |
| **Split-State Model** | No | No | Yes | Yes | Yes | No |
| **Query Submission Phase** | Before the challenge phase | Before the challenge phase | Before, during and after the challenge phase | Before, during and after the challenge phase | Before, during and after the challenge phase | Before, during and after the challenge phase |
| **Amount of Leakage** | Bounded | Bounded | Bounded | Unbounded | Unbounded | Bounded |
| **Proposed Key Exchange Protocol** | Enc-DH [13] | MO Protocol [20] | Protocol $\pi$ (under wBAFL-eCK)[1] [2] | Protocol P2 [2] | LR-AKA [23] | CLR-eCK secure AKE [11] |
| **Protocol Instantiated Cryptographic Primitives** | Leakage Resilient Signature Scheme, Leakage Resilient CCA-secure Encryption | DDH assumption, Pseudo-Random Function, Pairwise-Independent PRF, Collision Resistant Hash Function | CPLA2-Secure Public Key Cryptosystems, UFCMLA-Secure Signature Schemes, DDH assumption, Key Derivation Function | Diffie-Hellman Problems, Leakage-Resilient Storage | Bilinear group pairing, Generic bilinear group model, Discrete logarithm hardness assumption, Entropy | Randomness Extractor, Pseud-Random Function, Pairwise-Independent PRF, Smooth Projective Hash Function, |
| **Side Channel Attack Resilient** | Cache-side channel attack, Memory Attacks | Power Analysis | Cold-Boot Attack, Malware Attacks | Continuous leakage on Power Analysis, EM Radiation | Continuous leakage information of parameters involved in the computations | Attack involving leakage on both Ephemeral and Long-term secret keys. |

**Table 3.** Comparison of adversary capable query between security models

| **Adversary Capable Query** | **LR-CK** [13] | **LR-eCK** [20] | **(•)AFL-eCK** [3] | | **GCL-eCK** [23] | **CLR-eCK** [11] |
|---|---|---|---|---|---|---|
| | | | **BAFL-eCK** | **CAFL-eCK** | | |
| Adversary allows to activate and controls every communications between the protocol participants. | Adversary allowed to control the scheduling of protocol and initiation of protocol and protocol message delivery. | **Send Query** : Adversary activates the protocol party with protocol messages, controlling the activation of sessions. | Send Query embedded with leakage function | Send Query embedded with leakage function | **Send Query** : Obtain the protocol messages and sends the corresponding results to run the protocol according to message. | **Send Query** : Send protocol message to session initiator on behalf on session partner and obtain the response. |
| Adversary allows to reveal session keys | **Session-Key Query** : Attacker allows to receive the session key generated by the session. | **Session Key Reveal Query** : Query the session key of the completed session. | **Session Key Reveal Query** : Session key of completed session is given to adversary. | **Session Key Reveal Query** : Session key of completed session is given to adversary. | **Reveal Query** : Adversary can obtain the session key of an activated session. | **Session Key Reveal Query** : Query the session key of the completed session. |
| Obtain the long-term secret keys of the protocol participants by corrupting the protocol participants | Not allowed to learn the long-term secret key before the session is completed. | **Static Key Reveal Query** : Adversary obtain the static private key of the protocol principal. | **Corrupt Query** : Long-term secret key of protocol participants is given to adversary. (Does not reveal ephemeral secret) | **Corrupt Query** : Long-term secret key of protocol participants is given to adversary. (Does not reveal ephemeral secret) | **Corrupt Query** : Adversary allows to obtain the private key of the protocol principal. | **Long Term Key Reveal** : Adversary allows to learn the long-term secret key of honest protocol principal. |
| Learning session states or ephemeral secret keys of target session or its partner session | **Session-State Reveal Query** : Attacker receives all internal state (excluding long-term secret) of the session. | **Ephemeral Key Reveal** : Query the ephemeral private key of the session. | **Ephemeral Key Reveal** : Ephemeral secret key of the session is given to adversary. | **Ephemeral Key Reveal** : Ephemeral secret key of the session is given to adversary. | **Ephemeral-secret-leakage** : Adversary issue this query to obtain the ephemeral secret key of the session. | **Ephemeral Key Reveal** : Query the ephemeral secret key of the session. |
| Establishing public key for protocol participant | - | **Establish Party Query** : Adversary allows to register a long-term public key for a protocol participant. | - | - | - | **Establish Party Query** : Adversary allows to register a long-term public key for a protocol participant. |

[1] Modification is made to allow the use of available leakage-resilient signature scheme in protocol instantiation had weakened the security model to (**wBAFL-eCK)** to fit the assumption where EphemeralKeyReveal query cannot disclose the randomness used to compute the signature.