Modelling After-the-fact Leakage for Key Exchange
(full version)

Janaka Alawatugoda[1] Douglas Stebila[1,2] Colin Boyd[3]

[1] *School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, Australia*
[2] *Mathematical Sciences School, Queensland University of Technology, Brisbane, Australia*
janaka.alawatugoda@qut.edu.au, stebila@qut.edu.au
[3] *Department of Telematics, Norwegian University of Science and Technology, Trondheim, Norway*
colin.boyd@item.ntnu.no

Abstract

Security models for two-party authenticated key exchange (AKE) protocols have developed over time to prove the security of AKE protocols even when the adversary learns certain secret values. In this work, we address more granular leakage: *partial leakage* of long-term secrets of protocol principals, even after the session key is established. We introduce a generic key exchange security model, which can be instantiated allowing bounded or continuous leakage, even when the adversary learns certain ephemeral secrets or session keys. Our model is the strongest known partial-leakage-based security model for key exchange protocols. We propose a generic construction of a two-pass leakage-resilient key exchange protocol that is secure in the proposed model, by introducing a new concept: the leakage-resilient NAXOS trick. We identify a special property for public-key cryptosystems: *pair generation indistinguishability*, and show how to obtain the leakage-resilient NAXOS trick from a pair generation indistinguishable leakage-resilient public-key cryptosystem.

Keywords: key exchange protocols, public-key, side-channel attacks, security models, leakage-resilient, after-the-fact, NAXOS

This is the full version of a paper published in the *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security (ASIACCS) 2014* [2], in Kyoto, Japan, June 4-6, 2014, and hosted by National Institute of Information and Communications Technology (NICT), Kyushu University, and Ritsumeikan University, Japan.

Contents

1 Introduction

In order to capture leakage (side-channel) attacks in cryptography, the notion of leakage resilience has been developed. Examples of information which may leak and so allow side-channel attacks include timing information [5, 9, 24], electromagnetic radiation [20], and power consumption [28]. Leakage may reveal partial information about the secret parameters which have been used for computations in cryptosystems. To abstractly model leakage attacks, cryptographers have proposed the notion of *leakage-resilient* cryptography [1, 3, 8, 17, 18, 22, 21, 27], where the information that leaks is not fixed, but instead chosen adversarially. As key exchange protocols are among the most widely used cryptographic protocols, it is important to analyze their leakage resilience.

Earlier key exchange security models, such as the Bellare–Rogaway [4], Canetti–Krawczyk [10], and extended Canetti–Krawczyk (eCK) [26] models, aim to capture security against an adversary who can fully compromise some, but not all, secret values. This is not a very granular form of leakage, and thus is not suitable for modelling side-channel attacks enabled by partial leakage of secret keys. This motivates the development of leakage-resilient key exchange security models [3, 13, 30, 33].

Previous key exchange models including leakage have been limited in one or more ways. In most of the proposed models [3, 13, 30] the total amount of leakage allowed is bounded, which is troublesome because a protocol may reveal a limited amount of leakage each time it runs, and hence reveal "continuous" leakage. In addition, the adversary cannot obtain any leakage information after the session key is established for the session which the adversary targets for its attack. This is problematic because it does not address the security of one session, even if some leakage happens in subsequent sessions. A recent paper [33] uses a different leakage model with allows auxiliary input [14] but this cannot be directly compared with other leakage models. Although this model allows the adversary to make leakage queries on the complete secret, the values returned to the adversary are limited to those which are hard to invert and therefore are of limited use to the adversary. Our aim is to accommodate all the common leakage resistance models. We do not consider the auxiliary input model here.

In this paper, we construct 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. 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. We emphasize that the leakage functions are arbitrary polynomial time functions. Thus, our approach allows *after-the-fact leakage* in bounded or continuous leakage model. We now review the various approaches to modelling leakage, and then describe our contributions.

1.1 Leakage Models

Inspired by "cold boot" attacks, Akavia, Goldwasser and Vaikuntanathan [1] constructed a general framework to model memory attacks for public-key cryptosystems. With the knowledge of the public key, the adversary can choose an efficiently computable arbitrary leakage function, f and send it to the leakage oracle. The leakage oracle returns $f(sk)$ to the adversary where sk is the secret key. The only restriction here is that the sum of output length of all the leakage functions that an adversary can obtain is bounded by some parameter λ which is smaller than the size of sk.

In the work of Micali and Reyzin [29], a general framework was introduced to model the leakage that occurs during computation with secret parameters. This framework relies on the assumption that *only computation leaks information*. They mentioned that leakage only occurs from the secret memory portions which are actively involved in a computation. The adversary is allowed to obtain leakage from many computations, therefore the overall leakage amount is *unbounded* and in particular it can be larger than the size of the secret-key. Zvika et al. [8] proposed a continual memory leakage model in which it is not assumed that the information is only leaked from the secret memory portions involved in computations. Instead it is assumed that leakage happens from the entire secret memory but the amount of leakage is bounded per occurrence. This model allows the adversary to obtain arbitrarily large amounts of leakage information.

Security Model	Monolithic Leakage Queries			Partial Leakage Queries	
	`Session Key`	`Long-term Key`	`Ephemeral Key`	Leakage Model	After-the-fact
eCK [26]	Yes	Yes	Yes	None	No
MO [30]	Yes	Yes	Yes	Bounded	No
BAFL-eCK (Section 3.3)	Yes	Yes	Yes	Bounded	Yes
CAFL-eCK (Section 3.4)	Yes	Yes	Yes	Continuous	Yes

Table 1: Key exchange security models with reveal queries and leakage allowed

1.2 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 to distinguish the real plaintext corresponding to a particular ciphertext from a random 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 [4, 10, 26]. 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.

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 [31], 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 et al. [19] introduced after-the-fact leakage-resilient semantic security (CPLA2) on public-key cryptosystems. In their security experiment, the adversary is not allowed to access the decryption oracle. Dziembowski et al. [16] 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.

Recall that in key exchange security models, the challenge to the adversary is to distinguish the real session key of a chosen session from a random session key. In the previous leakage-resilient key exchange security models [30, 3], the adversary is not allowed to obtain leakage after the session key is established, because if the adversary gets the ephemeral secret key of the owner by ephemeral key reveal query, the adversary can encode the specification of the key derivation function with the ephemeral secret key and other public keys into the leakage function. This reveals some information about the session key allowing the adversary to successfully answer the challenge. In the literature there are no key exchange protocols or security models available that allow leakage after the session key is established.

1.3 Our Contribution

We propose a *generic eCK-style security model*, which additionally allows partial leakage of long-term secret keys of protocol principals, even after the session key is established. We choose to build on the eCK model because it is a widely used security model capturing a wide variety of possible attacks such as key compromise impersonation attacks, weak forward secrecy, and unknown key share attacks. Our generic model can be instantiated in two different ways. The instantiations of the generic model differ in the extent to which the adversary can obtain the leakage of long-term keys in a *bounded* amount or obtain *continuous* leakage such that a protocol execution may reveal a small amount of leakage each time, and still expect security. Thus, we begin by presenting the *after-the-fact leakage eCK model* ((·)AFL-eCK model), where leakage is modelled using the output of a *tuple leakage function* $\mathbf{f}$ such that $\mathbf{f} = (f_1, f_2, \ldots, f_n)$. Introducing a tuple leakage function allows us to address the leakage of different protocol constructions, such as protocols constructed

Protocol	Initiator Cost	Responder Cost	Security Model	Partial Leakage Resilience	
				Leakage Model	After-the-fact
NAXOS [26]	4 **Exp**	4 **Exp**	eCK	None	None
MO [30]	8 **Exp**	8 **Exp**	MO	Bounded	No
π	12 **Exp**	12 **Exp**	BAFL-eCK	Bounded	Yes

Table 2: Security and efficiency comparison of key exchange protocols

using stateful cryptographic primitives, or using more than one cryptographic primitive, or both. Table 1 shows a clear picture of our contribution.

We introduce a new property for public-key cryptosystems, which states that any randomly chosen ciphertext should be decrypted without rejection. The new property is named *pair generation indistinguishability* (PG-IND) and is defined in Definition 4.1. We demonstrate the use of PG-IND property in *leakage-resilient NAXOS trick* computation, which we introduce as the key idea of constructing $(\cdot)$AFL-eCK-secure key exchange protocols. We then construct a *generic protocol* π which can be proven secure in the generic $(\cdot)$AFL-eCK model. The protocol π is a key-agreement-style protocol which relies on two primitives: (1) a public-key cryptosystem that is PG-IND and after-the-fact leakage-resilient semantically secure (CPLA2); and (2) an unforgeable signature scheme against chosen message leakage attacks (UFCMLA). Whenever both of these two primitives are proven secure in either the bounded leakage model or the continuous leakage model, the instantiation of the protocol π is secure in the BAFL-eCK model or CAFL-eCK model respectively.

In Table 2, we compare an instantiation of the proposed generic protocol, π, with the NAXOS [26] and the Moriyama-Okamoto (MO) [30] protocols. The protocol π is instantiated using the CPLA2-secure, 0-PG-IND public-key cryptosystem of Halevi et al. [19], and UFCMLA-secure signature scheme of Katz et al. [21], in the bounded leakage model. The instantiation of the generic protocol π is BAFL-eCK-secure, and provides significant leakage resilience for practically achievable computation costs. We could not instantiate a CAFL-eCK-secure protocol π, because we do not have a CPLA2-secure, PG-IND public-key cryptosystem in the continuous leakage model. Halevi et al.[19] emphasized that adjusting their scheme to be CPLA2-secure in the continuous leakage model is an open problem. There are UFCMLA-secure signature schemes in the continuous leakage model [27, 13, 8]. So once CPLA2-secure, PG-IND public-key cryptosystem is available in the continuous leakage model, CAFL-eCK-secure π can be instantiated. Thus, the generic protocol π is a viable framework for $(\cdot)$AFL-eCK-secure protocols, and can achieve the leakage tolerance which the underlying schemes provide in the bounded or the continuous leakage model.

2 Preliminaries

We review the formal security definitions we will use to construct the generic protocol π.

2.1 CPLA2-Secure Public-Key Cryptosystems

We review the definition of CPLA2 security in the split-states-model, where the secret key s is split into arbitrarily n parts such that $s = (s_1, \ldots, s_n)$. The tuple leakage function $\mathbf{f} = (f_1, \ldots, f_n)$ is an adversary chosen efficiently computable adaptive tuple leakage function, which consists of n arbitrary leakage functions. Each leakage function f_i leaks $f_i(s_i)$ from each s_i split of the secret key individually. Following we consider bounded leakage from each split.

Definition 2.1. *(After-the-fact Leakage-resilient Semantic Security (CPLA2)).* 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 $\boldsymbol{\lambda}_{pre} = (\lambda_{pre_1}, \ldots, \lambda_{pre_n})$ is the leakage bound vector before the challenge ciphertext is issued, and $\boldsymbol{\lambda}_{post} = (\lambda_{post_1}, \ldots, \lambda_{post_n})$ is the leakage bound vector after the challenge ciphertext is issued. Let $\mathrm{PKE} = (\mathrm{KG}, \mathrm{Enc}, \mathrm{Dec})$ be a public-key cryptosystem, we define $Adv^{\mathrm{CPLA2}}_{\mathrm{PKE}}(\mathcal{D})$ 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. $(m_0, m_1) \leftarrow \mathcal{D}^{\text{Leak}(\mathbf{f})}(p) || m_0| = |m_1|$, for $i = 1, \ldots, n$, Leak($\mathbf{f}$) returns $f_i(s_i)$ if $\Sigma|f_i(s_i)| \leq \lambda_{pre_i}$.
3. $b \xleftarrow{\$} \{0, 1\}$.
4. $C \xleftarrow{\$} \text{Enc}(pk, m_b)$.
5. $b' \leftarrow \mathcal{D}^{\text{Leak}(\mathbf{f})}(p, C)$ for $i = 1$ to $i = n$, Leak($\mathbf{f}$) returns $f_i(s_i)$ if $\Sigma|f_i(s_i)| \leq \lambda_{post_i}$.
6. $\mathcal{D}$ wins if $b' = b$.

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

Halevi et al. [19] constructed a generic CPLA2-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 et al. [31] with decryption cost of 4 **Exp**onentiations.

2.2 UFCMLA-Secure Signature Schemes

We review the definition of UFCMLA security in the split-states-model, where the signing key sk is split into arbitrarily n parts such that $sk = (sk_1, \ldots, sk_n)$. The tuple leakage function $\mathbf{f} = (f_1, \ldots, f_n)$ is an adversary chosen efficiently computable adaptive tuple leakage function, which consists of n arbitrary number of leakage functions. Each leakage function f_i leaks $f_i(sk_i)$ from each sk_i split of the secret key individually. Following we consider bounded leakage from each split.

Definition 2.2. *(Unforgeability Against Chosen Message Leakage Attacks (UFCMLA)).* Let $k \in \mathbb{N}$ be the security parameter and $\boldsymbol{\lambda}$ be a vector of n elements. Let SIG = (KG, Sign, Vfy) be a signature scheme, we define $Adv^{\text{UFCMLA}}_{\text{SIG}}(\mathcal{E})$ as the advantage of any PPT adversary $\mathcal{E}$, winning the following game:

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

$\mathcal{O}^{\text{UFCMLA}}(m, \mathbf{f})$

- $\sigma \xleftarrow{\$} (sk, m)$
- for $i = 1, \ldots, n$, $\gamma_i \leftarrow f_i(sk_i)$
- If $\Sigma|\gamma_i| \leq \lambda_i$, then append γ_i to γ
- Return (σ, γ)

SIG is UFCMLA if $Adv^{\text{UFCMLA}}_{\text{SIG}}(\mathcal{E})$ is negligible in k.

Katz et al. [21] constructed an UFCMLA-secure signature scheme in bounded leakage model in which $n = 1$. It contains signing and verification operations based on NIZK proofs, where signature can be generated with cost of 2 **Exp**onentiations, and verified with cost of 4 **Exp**onentiations (with a simple NIZK proof).

Remark 1. Both Definition 2.1 and 2.2 can be easily adjusted into their continuous leakage versions.

2.3 Decision Diffie-Hellman (DDH) Problem

The decision Diffie-Hellman (DDH) problem is a computational hardness assumption based on discrete logarithms in a cyclic group [6]. Consider a cyclic group $\mathbb{G}$ of prime order q, with a generator g. For $a, b, c \in \mathbb{Z}_q$, the DDH problem is to distinguish the triple (g^a, g^b, g^{ab}) from the triple (g^a, g^b, g^c).

2.4 Key Derivation Functions

We review the definitions of key derivation functions proposed by Krawczyk [25].

Definition 2.3. *(Key Derivation Function).* 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 k and two additional arguments, a salt value r defined over a set of possible salt values and a context variable c, both which are optional i.e., can be set to a null. The KDF output is a string of k bits.

Definition 2.4. *(Source of Key Material).* A source of keying material Σ is a two-valued (σ, κ) probability distribution generated by an efficient probabilistic algorithm.

Definition 2.5. *(Security of key derivation function with respect to a source of key material).* Let KDF be a key derivation function, we define $Adv_{\mathrm{KDF}}(\mathcal{B})$ as the advantage of any PPT adversary $\mathcal{E}$, winning the following game:

1. $(\sigma, \kappa) \leftarrow \Sigma$. (Both the probability distribution as well as the generating algorithm have been referred by 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 k 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 $\mathrm{KDF}(\sigma, r, k, c)$ else $\mathcal{B}$ is given a random string of k bits.
6. $\mathcal{B}$ outputs a bit $b' \leftarrow \{0,1\}$. $\mathcal{B}$ wins if $b' = b$.

KDF is secure with respect to the source of key material, if $Adv_{\mathrm{KDF}}(\mathcal{B})$ is negligible in k.

3 The Generic After-the-fact Leakage-eCK ($(\cdot)$AFL-eCK) Model

The generic after-the-fact leakage eCK ($(\cdot)$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 $(\cdot)$AFL-eCK model the partnering definition and the adversarial powers are same. The freshness conditions differ by means of the leakage allowed. So we can define the partnering and adversarial powers in the generic $(\cdot)$AFL-eCK model and define the freshness separately in each BAFL-eCK and CAFL-eCK models.

Let $\mathcal{U} = \{U_1, ..., U_{N_P}\}$ be a set of n parties. We use the term *principal* to identify a party involved in a protocol instance. 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}). The term *session* is used to identify a protocol instance at a principal. Each principal may have multiple sessions and they may run concurrently. The oracle $\Pi^s_{U,V}$ 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.

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

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

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

3.1 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", we have modeled the leakage 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".

We introduce a tuple of n adaptively chosen efficiently computable leakage functions $\mathbf{f} = (f_1, f_2, \ldots, f_n)$; the size n of the tuple is *protocol-specific*. A key exchange protocol may use more than one cryptographic primitive and each primitive uses a distinct secret key. Hence, we need to address the leakage of secret keys from each of those primitives. Otherwise, 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 use one part of the secret key.

3.2 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 oracles which represent protocol instances. The following query allows the adversary $\mathcal{A}$ to run the protocol.

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

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

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

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

- `Test`(U, s) query: When $\mathcal{A}$ asks the `Test` query, the challenger first chooses a random bit $b \xleftarrow{\$} \{0, 1\}$ 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 λ-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 bounded amount of 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 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. Further, we emphasize that the technique of sending a tuple leakage function $\mathbf{f}$ with the `Send` query can be applied to any of the existing key exchange security models to obtain their leakage versions.

3.3 The 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_i 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 $\sum |f_i(sk_i)| \leq \lambda_i$.

Definition 3.2. *($\boldsymbol{\lambda}$-BAFL-eCK-freshness).* Let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ be a vector of n elements (same size as $\mathbf{f}$ in `Send` query). An oracle $\Pi^s_{U,V}$ is said to be $\boldsymbol{\lambda}$-BAFL-eCK-fresh if and only if:

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

3.4 The 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, 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_i 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 $f_i(sk_i) \leq \lambda_i$, 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_i outputs leakage bits of the i^{th} split of the secret key, then for leakage resilience of the stateful cryptographic primitive we need that $f_i(sk_i) \leq \lambda_i$, per leakage occurrence.

Definition 3.3. *($\boldsymbol{\lambda}$-CAFL-eCK-freshness).* Let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ be a vector of n elements (same size as $\mathbf{f}$ in `Send` query). An oracle $\Pi^s_{U,V}$ is said to be $\boldsymbol{\lambda}$-CAFL-eCK-fresh if and only if: Conditions (1)-(4) of Definition 3.2 hold, and

5. For each `Send`$(\cdot, U, \cdot, \cdot, \mathbf{f})$ query, size of the output of $f_i(sk_{U\,i}) \leq \lambda_i$.
6. For each `Send`$(\cdot, V, \cdot, \cdot, \mathbf{f})$ queries, size of the output of $f_i(sk_{V\,i}) \leq \lambda_i$.

3.5 Security Game and Security Definition

We introduce the security game of the generic $(\cdot)$AFL-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.

Definition 3.4. *($(\cdot)$AFL-eCK security game).* Security of a key exchange protocol in the generic $(\cdot)$AFL-eCK model is defined using the following security game, which is played by PPT adversary $\mathcal{A}$ against the protocol challenger.

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

$Succ_{\mathcal{A}}$ is the event that the adversary $\mathcal{A}$ wins the security game in Definition 3.4.

Definition 3.5. *((·)AFL-eCK-security).* A protocol π is said to be (·)AFL-eCK-secure if there is no PPT algorithm $\mathcal{A}$ that can win the (·)AFL-eCK security game with non-negligible advantage. The advantage of an adversary $\mathcal{A}$ is defined as $Adv_{\pi}^{(\cdot)\text{AFL-eCK}}(\mathcal{A}) = |2\Pr(\text{Succ}_{\mathcal{A}}) - 1|$.

3.6 Practical Interpretation of Security of the Generic (·)AFL-eCK Model

We review the relationship between the generic (·)AFL-eCK model and real world attack scenarios.

Active adversarial capabilities: `Send` queries address the powers of an active adversary who can control the message flow over the network.

Side-channel attacks: Leakage functions are embedded with the `Send` query. Thus, a wide variety of side-channel attacks based on **leakage of long-term secrets** are addressed, assuming that the leakage happens when computations take place in protocol principals. BAFL-eCK model addresses the situation where the adversary is only allowed to obtain a bounded amount of total leakage. CAFL-eCK model addresses a stronger situation, where the adversary is allowed to obtain continuous leakage, but the amount of leakage per invocation is bounded.

Cold-boot attacks: `Corrupt` queries address situations which reveal the long-term secret keys of protocol principals like in 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.

Weak random number generators: After knowing a previous set of randomly generated ephemeral values the adversary may be able to identify the statistical pattern of the random number generator and hence correctly guess the next value with a high probability. `EphemeralKeyReveal` query addresses situations where the adversary can get the ephemeral secrets.

Known key attacks: `SessionKeyReveal` query covers the attacks which can be mounted by knowing past session keys.

Key compromise impersonation attacks: In the generic (·)AFL-eCK model, freshness conditions allow the adversary to corrupt the owner of the test session before the activation of the test session. Hence, the generic (·)AFL-eCK model security protects against the key compromise impersonation attacks.

Weak forward secrecy: In the generic (·)AFL-eCK model, freshness conditions allow the adversary to corrupt both of the protocol principals, after the test session is activated. Hence, the generic (·)AFL-eCK model addresses weak forward secrecy.

eCK security: The generic (·)AFL-eCK model is a leakage-resilient version of the eCK model [26], hence, the generic (·)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, (·)AFL-eCK model, and then propose two instantiations (BAFL-eCK model and CAFL-eCK model) is to offer more flexibility to construct leakage-resilient key exchange protocols. The generic (·)AFL-eCK model gives a reasonable security framework for

key exchange protocols capturing 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 captures all the properties mentioned in Section 3.6 in the bounded leakage model, we use the BAFL-eCK model as the security framework, whereas if we need security in the continuous leakage model, we use the CAFL-eCK model as the security framework.

4 Constructing (·)AFL-eCK-secure Key Exchange Protocols

We investigate how to construct (·)AFL-eCK-secure key exchange protocols. The motivation of LaMacchia et al. [26] 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": a "psuedo" ephemeral key $\widetilde{esk}$ is computed as the hash of the long-term key lsk and the actual ephemeral key esk: $\widetilde{esk} \leftarrow H(esk, lsk)$. 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 $\widetilde{esk} = H(esk, lsk)$ 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.

4.1 Leakage-Resilient NAXOS Trick

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, we need some kind of *leakage-resilient NAXOS trick*. 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. We replace the hash function H 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 can be achieved by using the *decryption* function of a CPLA2-secure public-key cryptosystem [19]. 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.

4.2 Pair Generation Indistinguishability

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. We formally define the special property as *pair generation indistinguishability*.

Definition 4.1. *(Pair Generation Indistinguishability).* Let PKE = (KeyGen, Enc, Dec) be a public-key cryptosystem. For $(p, s) \xleftarrow{\$} \text{KeyGen}(1^k)$, let $D_1^{(p,s)}$, $D_2^{(p,s)}$ be two distributions such that $D_1^{(p,s)} = \{(m, c) : m \xleftarrow{\$} M, c \xleftarrow{\$} \text{Enc}(p, m)\}$ and $D_2^{(p,s)} = \{(m, c) : c \xleftarrow{\$} \text{C}, m \leftarrow \text{Dec}(s, c)\}$ 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* (ϵ-PG-IND) if for all $(p, s) \xleftarrow{\$} \text{KeyGen}(1^k)$, $\text{SD}(D_1^{(p,s)}, D_2^{(p,s)}) \leq \epsilon$.

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

Now we show a 0-PG-IND public-key cryptosystem available in the literature. Naor et al. [31] described the framework of a hash proof system [11] as a key-encapsulation mechanism using the notion of Kiltz et al. [23]. Let K be the symmetric key space, C be the 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 et al. encrypts an arbitrary message $m \xleftarrow{\$} \text{M}$ as $(\Psi, c, seed)$, where $c \xleftarrow{\$} \text{C}$ with the corresponding witness ω, $seed \xleftarrow{\$} \{0,1\}^t$ is a random seed and $\Psi = Ext(Pub(p, c, \omega), seed) \oplus m$. $Ext : \text{K} \times \{0,1\}^t \rightarrow \{0,1\}^m$ is a public average-case strong extractor function [15], p is the public key and Pub is the deterministic public evaluation function of the underlying key-encapsulation mechanism. So whenever a random $(\Psi, c, seed)$ is sampled, the decryption, $m \leftarrow \Psi \oplus Ext(Priv_s(c), seed)$ corresponds to a random $m \in \text{M}$. $Priv$ is a private evaluation algorithm of the underlying key-encapsulation mechanism, which is parametrized by the secret key s. Thus, the leakage-resilient public-key cryptosystem of Naor et al. is 0-PG-IND. The generic CPLA2-secure public-key cryptosystem of Halevi et al. [19] can be instantiated using the leakage-resilient public-key cryptosystem of Naor et al. Hence, the instantiation of the generic CPLA2-secure public-key cryptosystem of Halevi et al. is also 0-PG-IND.

Remark 3. In Table 3, let $\widehat{\text{C}}$ be the ciphertext space: in a setting like Naor et al. [31] the random r. values are not just chosen from the C but from $\widehat{\text{C}}$ which gives random r. in the form $(\Psi, c, seed)$.

4.3 Authenticating Protocol Messages

In this section we review how to provide authentication to the 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, we need to provide authenticity to the protocol messages. There are UFCMLA-secure signature schemes available in the literature [21, 18, 27, 8], which we can use 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.

4.4 Protocol π

In Table 3, we show the construction of protocol π. KeyGen, Enc and Dec are the key generation, encryption and decryption algorithms of the underlying CPLA2-secure, ϵ-PG-IND-public-key cryptosystem PKE respectively. KG, Sign and Vfy are the key generation, signature generation and signature verification algorithms of the underlying leakage-resilient signature scheme SIG respectively. KDF is a secure key derivation function which generates the session key of length k. The protocol π is a Diffie-Hellman-type [12] key agreement protocol where $\mathbb{G}$ is a group of prime order q and generator g. We require that q is the size of the message space M. After exchanging the public values both principals compute a Diffie-Hellman-type shared secret value, and then compute the session key using the key derivation function KDF, with inputs identities of the two principals and the Diffie-Hellman-type shared secret. The computations which leak information are underlined.

4.5 Security Proof of the Protocol π

We prove the security of the generic protocol π in the $(\cdot)$AFL-eCK model. If the underlying primitives are secure in bounded or continuous leakage model, the protocol π is BAFL-eCK-secure or CAFL-eCK-secure respectively.

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" $\{$
$\widetilde{r_A} \leftarrow \underline{\mathrm{Dec}}(s_A, r_A)$		$r_B \xleftarrow{\$} \widehat{\mathrm{C}}$
$X_A \leftarrow g^{\widetilde{r_A}}$		$\widetilde{r_B} \leftarrow \underline{\mathrm{Dec}}(s_B, r_B)$
$\sigma_A \xleftarrow{\$} \underline{\mathrm{Sign}}(sk_A, (A, B, X_A))$	$\xrightarrow{A,B,X_A,\sigma_A}$	$X_B \leftarrow g^{\widetilde{r_B}}$
	$\xleftarrow{B,A,X_B,\sigma_B}$	$\sigma_B \xleftarrow{\$} \underline{\mathrm{Sign}}(sk_B, (B, A, X_B))$
If $\mathrm{Vfy}(vk_B, (B, A, X_B), \sigma_B) =$ "true" $\{$		
$\widetilde{r_A} \leftarrow \underline{\mathrm{Dec}}(s_A, r_A)$		
$K_{AB} \leftarrow \mathrm{KDF}(A, B, X_B^{\widetilde{r_A}})$		$K_{AB} \leftarrow \mathrm{KDF}(A, B, X_A^{\widetilde{r_B}})$
$\}$		$\}$

Table 3: Protocol π. Underline denotes operations to which leakage functions apply.

Theorem 4.1. *Let $\mathcal{A}$ be any PPT adversary against the protocol π. Then the advantage of $\mathcal{A}$ against* $(\cdot)$AFL-eCK*-security of protocol π, $Adv_\pi^{(\cdot)\text{AFL-eCK}}$ is:*

$$Adv_\pi^{(\cdot)\text{AFL-eCK}}(\mathcal{A}) \leq \max\Big[N_P Adv_{\text{SIG}}^{\text{UFCMLA}}(\mathcal{E}),\quad N_P^2 {N_s}^2\big(\epsilon_{pg} + 2Adv_{\text{PKE}}^{\text{CPLA2}}(\mathcal{D}) + Adv_{q,g}^{\text{DDH}}(\mathcal{C}) + Adv_{\text{KDF}}(\mathcal{B})\big)\Big].$$

$\mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}$ are efficient algorithms constructed using the adversary $\mathcal{A}$, against the underlying key derivation function, KDF*, DDH problem, public-key cryptosystem,* PKE *and the signature scheme,* SIG*, respectively, where* PKE *is* ϵ-PG-*IND.*

In order to formally prove the $(\cdot)$AFL-eCK-security of the protocol π, we use the game hopping technique [32]: define a sequence of games and relate the adversary's advantage of distinguishing each game from the previous game to the advantage of breaking one of the underlying cryptographic primitive. The proof structure is similar to Boyd et al. [7]. The proof of Theorem 4.1 is available in Appendix A. The proof is split into two main cases: when the partner to the test session exists, and when it does not.

5 Conclusion

We have proposed a *generic security model* and protocol that improves the amount and type of secret leakage that can be tolerated in authenticated key exchange protocols. Our generic model allows the adversary to fully compromise a variety of long-term and short-term ephemeral values, as well as obtain partial, adaptive, *either bounded or continuous*, after-the-fact leakage of long-term secret keys. Previous key exchange security models either do not consider partial leakage at all (BR, CK, eCK) or allow only leakage before the test session is queried and none after. Our model captures a wide variety of practical attack scenarios, including side channel attacks. Further, the model gives flexibility to choose either bounded or continuous leakage setting according to the necessity and available underlying primitives, and it provides a security framework to construct after-the-fact leakage-resilient key exchange protocols. We have given a generic protocol, secure in our generic model, that relies on a CPLA2-secure ϵ-PG-IND-public-key cryptosystem and an UFCMLA-secure signature scheme. Using such schemes from the literature, our protocol can be instantiated without much more cost than previous schemes which tolerate leakage *only before the test session is activated.*

Instantiating a CAFL-eCK-secure protocol is an open problem at this stage. We cannot use existing after-the-fact leakage-resilient public-key cryptosystems in the continuous leakage model for the leakage-resilient NAXOS computation, because they do not satisfy pair generation indistinguishability. If we can construct a CPLA2 or CCLA2-secure public-key cryptosystem which is ϵ-PG-IND, in the continuous leakage model, then we can construct a CAFL-eCK-secure protocol, and achieve leakage tolerance provided by the underlying public-key cryptosystem and the signature scheme.

References

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

[2] J. Alawatugoda, D. Stebila, and C. Boyd. Modelling after-the-fact leakage for key exchange. In *ASIACCS*, 2014.

[3] J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

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

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

[6] D. Boneh. The decision Diffie-Hellman problem. In *Algorithmic Number Theory Symposium*, pages 48–63, 1998.

[7] C. Boyd, Y. Cliff, J. M. G. Nieto, and K. G. Paterson. One-round key exchange in the standard model. *International Journal of Advanced Computer Technology*, pages 181–199, 2009.

[8] 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.

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

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

[11] 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.

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

[13] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT*, pages 613–631, 2010.

[14] Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.

[15] 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.

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

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

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

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

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

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

[22] E. Kiltz and K. Pietrzak. Leakage resilient elgamal encryption. In *ASIACRYPT*, pages 595–612, 2010.

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

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

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

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

[27] 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.

[28] 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.

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

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

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

[32] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, Report 2004/332, 2004.

[33] G. Yang, Y. Mu, W. Susilo, and D. S. Wong. Leakage resilient authenticated key exchange secure in the auxiliary input model. In *ISPEC*, pages 204–217, 2013.

A Security Proof

Proof. The proof of Theorem 4.1 is split into two main cases: when the partner to the test session exists, and when it does not.

A.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 oracles. We assume that the adversary $\mathcal{A}$ can win the $(\cdot)$AFL-eCK challenge against the protocol π with non-negligible advantage $Adv_{\pi}^{(\cdot)\text{AFL-eCK}}(\mathcal{A})$. 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.

A.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.
Game 2: Same as Game 1 with the following exception: 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 oracle $\Pi^{s^*}_{U^*,V^*}$ is chosen as the target session and the oracle $\Pi^{t^*}_{V^*,U^*}$ is chosen as the partner to the target session. If the test session is not the oracle $\Pi^{s^*}_{U^*,V^*}$ or partner to the oracle 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 randomly chooses $z \xleftarrow{\$} \mathbb{Z}^*_q$ and computes $K_{U^*V^*} \leftarrow \mathrm{KDF}(U^*, V^*, g^z)$. When the adversary asks the $\mathtt{Test}(U^*, V^*, s^*)$ query, Game 3 challenger will answer with K_{U^*,V^*}. Further, since a partner session to the test session exists, when the adversary asks $\mathtt{Test}(V^*, U^*, t^*)$ query, Game 3 challenger will answer with K_{U^*,V^*}.
Game 4: Same as Game 3 with the following exception: the Game 4 challenger randomly chooses $K \xleftarrow{\$} \{0,1\}^k$ and sends it to the adversary $\mathcal{A}$ as the answer to the $\mathtt{Test}(U^*, V^*, s^*)$ query or $\mathtt{Test}(V^*, U^*, t^*)$ query.

Differences between games: In this section the adversary's advantage of distinguishing each game from the previous game is investigated. $Succ_{\text{Game x}}(\mathcal{A})$ denotes the event that the adversary $\mathcal{A}$ wins Game x, $Adv_{\text{Game x}}(\mathcal{A})$ denotes the advantage of the adversary $\mathcal{A}$ of winning Game x. Game 1 is the original game. Hence,

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

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,

$$Adv_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} Adv_{\text{Game 1}}(\mathcal{A}). \tag{2}$$

Game 2 and Game 3: We construct an algorithm $\mathcal{C}$ against the DDH challenge, using the adversary $\mathcal{A}$. The DDH challenger sends values (g^x, g^y, g^z) such that either $z = xy$ or $z \xleftarrow{\$} \mathbb{Z}^*_q$, as the inputs to the algorithm $\mathcal{C}$. If $\mathcal{C}$'s input is a Diffie-Hellman triple, simulation constructed by $\mathcal{C}$ is identical to Game 2, otherwise it is identical to Game 3. If $\mathcal{A}$ can distinguish whether $g^z = g^{xy}$ or not, then $\mathcal{C}$ can answer the DDH challenge. Note that $\mathtt{EphemeralKeyReveal}(U^*, V^*, s^*)$ or $\mathtt{EphemeralKeyReveal}(V^*, U^*, t^*)$ is prohibited since the adversary is allowed to corrupt both the owner and the partner to the test session. Hence,

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

Game 3 and Game 4: We construct an algorithm $\mathcal{B}$ against the security of the key derivation function KDF, using the adversary $\mathcal{A}$. $\mathcal{B}$ receives K such that K is computed using the KDF or randomly chosen from the session key space. If K is computed using the KDF, simulation constructed by $\mathcal{B}$ is identical to Game 3, otherwise it is identical to Game 4. If $\mathcal{A}$ can distinguish between Game 3 and Game 4, then $\mathcal{B}$ can distinguish whether the value K is computed using KDF or randomly chosen, and answer the security challenge on key derivation function in Definition 2.5. Hence,

$$|Adv_{\text{Game 3}}(\mathcal{A}) - Adv_{\text{Game 4}}(\mathcal{A})| \leq Adv_{\text{KDF}}(\mathcal{B}). \tag{4}$$

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,

$$Adv_{\text{Game 4}}(\mathcal{A}) = 0 \tag{5}$$

Using equations (1)–(5) we find,

$$Adv^{(\cdot)\text{AFL-eCK}}_{\pi}(\mathcal{A}) \leq N_P^2 {N_s}^2 \big(Adv^{\text{DDH}}_{q,g}(\mathcal{C}) + Adv_{\text{KDF}}(\mathcal{B})\big).$$

A.1.2 Adversary corrupts neither owner or nor partner principal to the test session.

Game 1:Same as Game 1 in Case A.1.1.
Game 2: Same as Game 2 in Case A.1.1.
Game 3: Same as Game 2 with the following exception: the Game 3 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r_{U^*}} \xleftarrow{\$} \mathbb{Z}_q^*$ and computes the ephemeral key $r_{U^*} \xleftarrow{\$} \text{Enc}(p_{U^*}, \widetilde{r_{U^*}})$. Then computes $K_{U^*V^*} \leftarrow \text{KDF}(U^*, V^*, X_{V^*}^{\widetilde{r_{U^*}}})$. When the adversary asks the $\texttt{Test}(U^*, V^*, s^*)$, Game 3 challenger will answer with K_{U^*,V^*}. Further, since a partner session to the test session exists, when the adversary asks $\texttt{Test}(V^*, U^*, t^*)$, Game 3 challenger will answer with K_{U^*,V^*}.
Game 4: Same as Game 3 with the following exception: the Game 4 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r'_{U^*}} \xleftarrow{\$} \mathbb{Z}_q^*$. Then computes $K_{U^*V^*} \leftarrow \text{KDF}(U^*, V^*, X_{V^*}^{\widetilde{r'_{U^*}}})$ and sends it to the adversary $\mathcal{A}$ as the answer to the $\texttt{Test}(U^*, V^*, s^*)$ query or $\texttt{Test}(V^*, U^*, t^*)$ query.
Game 5: Same as Game 4 with the following exception: the Game 5 challenger randomly chooses a pseudo-ephemeral value $\widetilde{r'_{V^*}} \xleftarrow{\$} \mathbb{Z}_q^*$. Then computes $K_{U^*V^*} \leftarrow \text{KDF}(U^*, V^*, X_{U^*}^{\widetilde{r'_{V^*}}})$ and sends it to the adversary $\mathcal{A}$ as the answer to the $\texttt{Test}(U^*, V^*, s^*)$ query or $\texttt{Test}(V^*, U^*, t^*)$ query.
Game 6: Same as Game 3 in Case A.1.1.
Game 7: Same as Game 4 in Case A.1.1.

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

$$Adv_{\text{Game 1}}(\mathcal{A}) = Adv_{\pi}^{(\cdot)\text{AFL-eCK}}(\mathcal{A}). \tag{6}$$

Game 1 and Game 2: Same as Game 1 and Game 2 in Case A.1.1.

$$Adv_{\text{Game 2}}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} Adv_{\text{Game 1}}(\mathcal{A}). \tag{7}$$

Game 2 and Game 3: We introduce an algorithm $\mathcal{F}$ which is constructed using the adversary $\mathcal{A}$. If $\mathcal{A}$ can distinguish the difference between Game 2 and Game 3, then $\mathcal{F}$ can distinguish whether a message/cphertext pair (m, c) belongs to the distribution D_1 or D_2. $\mathcal{F}$ receives a pair $(r_{U^*}, \widetilde{r_{U^*}})$ such that $\widetilde{r_{U^*}} = \text{Dec}(s_{U^*}, r_{U^*})$. $\mathcal{F}$ uses r_{U^*} as the ephemeral key of U^* and $\widetilde{r_{U^*}}$ as the pseudo-ephemeral key of U^*. If a random ephemeral key $r_{U^*} \xleftarrow{\$} \mathbb{Z}_p^*$ is chosen first and the pseudo-ephemeral value $\widetilde{r_{U^*}} \leftarrow \text{Dec}(s_{U^*}, r_{U^*})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 2. Otherwise if a random pseudo-ephemeral value $\widetilde{r_{U^*}} \xleftarrow{\$} \mathbb{Z}_p^*$ is chosen first and the ephemeral key $r_{U^*} \xleftarrow{\$} \text{Enc}(p_{U^*}, \widetilde{r_{U^*}})$ is computed, then the simulation constructed by $\mathcal{F}$ is identical to Game 3. If $\mathcal{A}$ can distinguish whether a message/cphertext pair (m, c) belongs to the distribution D_1 or D_2. Hence,

$$|Adv_{\text{Game 2}}(\mathcal{A}) - Adv_{\text{Game 3}}(\mathcal{A})| \leq \epsilon. \tag{8}$$

Game 3 and Game 4: We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$. If $\mathcal{A}$ can distinguish the difference between Game 3 and Game 4, then $\mathcal{D}$ can be used against a 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 ciphertexts C_1 such that $C_1 \xleftarrow{\$} \text{Enc}(pk_{U^*}, r_\theta)$ where $r_\theta = r_0$ or $r_\theta = r_1$. The following describes the procedure of answering queries. (assuming that U^* is the initiator, otherwise session key is computed like $\text{KDF}(V^*, U^*, \cdot)$)

- $\texttt{Send}(U, V, s, m, \mathbf{f})$ query: When $U = U^*$, $V = V^*$ and $s = t^*$, $\mathcal{D}$ takes r_1 as $\widetilde{r'_{U^*}}$, computes $g^{\widetilde{r'_{U^*}}}$ and computes its signature. Then $\mathcal{D}$ creates the protocol message and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(sk_{U^*})$. For all other $\texttt{Send}$ queries, $\mathcal{D}$ randomly picks a pseudo-ephemeral value $\widetilde{r_U} \xleftarrow{\$} \{0,1\}^k$, computes $r_U \xleftarrow{\$} \text{Enc}(pk_V, \widetilde{r_U})$ as the ephemeral key, and computes $g^{\widetilde{r_U}} \pmod n$. Then computes the corresponding signature, creates the protocol message and sends it to $\mathcal{A}$ with the leakage $\mathbf{f}(sk_U)$ which obtained from the leakage oracle.

- $\texttt{SessionKeyReveal}(U, V, s)$ query: $\mathcal{D}$ will abort the game if $\texttt{SessionKeyReveal}(U^*, V^*, s^*)$ query or $\texttt{SessionKeyReveal}(V^*, U^*, t^*)$ query is asked. For all other $\texttt{SessionKeyReveal}$ queries $\mathcal{D}$ can compute the answer using the corresponding psuedo-ephemeral keys and answer the queries.
- $\texttt{EphemeralKeyReveal}(U, V, s)$ query: $\mathcal{D}$ uses C_1 as the ephemeral key for $\texttt{EphemeralKeyReveal}(U^*, V^*, s^*)$. For all other $\texttt{EphemeralKeyReveal}$ queries $\mathcal{D}$ will answer with the corresponding ephemeral-key which is computed by encrypting pseudo-ephemeral value with the secret key of corresponding principal.
- $\texttt{Corrupt}(U)$ query: Except for U^* and V^*, algorithm $\mathcal{D}$ can answer all other $\texttt{Corrupt}$ queries. In this case we consider the situation in which the adversary corrupts neither owner or nor partner principal to the test session.
- $\texttt{Test}(U, s)$ 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 $\mathrm{KDF}(U^*, V^*, X_{V^*}^{\widetilde{r'_{U^*}}})$ where $\widetilde{r'_{U^*}} = r_1$, and answers to 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 3 whereas if $\theta = 0$, then r_0 is the decryption of C_1 and the simulation constructed by $\mathcal{D}$ is identical to Game 4. Hence,

$$|Adv_{\text{Game }3}(\mathcal{A}) - Adv_{\text{Game }4}(\mathcal{A})| \leq Adv_{\text{PKE}}^{\text{CPLA2}}(\mathcal{D}). \tag{9}$$

Game 4 and Game 5: We introduce an algorithm $\mathcal{D}$ which is constructed using the adversary $\mathcal{A}$. If $\mathcal{A}$ can distinguish the difference between Game 4 and Game 5, then $\mathcal{D}$ can be used against a 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 ciphertexts C_2 such that $C_2 \xleftarrow{\$} \mathrm{Enc}(pk_{V^*}, r'_\theta)$ where $r'_\theta = r'_0$ or $r'_\theta = r'_1$. $\mathcal{D}$ uses C_2 as the ephemeral key for $\texttt{EphemeralKeyReveal}(V^*, U^*, t^*)$. Answering the queries is similar to the Game 3 and Game 4, now we use the public key of the CPLA2 challenger as the public key of the protocol principal V^*. 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. Hence,

$$|Adv_{\text{Game }4}(\mathcal{A}) - Adv_{\text{Game }5}(\mathcal{A})| \leq Adv_{\text{PKE}}^{\text{CPLA2}}(\mathcal{D}). \tag{10}$$

Game 5 and Game 6: Same as Game 2 and Game 3 in Case A.1.1.

$$|Adv_{\text{Game }5}(\mathcal{A}) - Adv_{\text{Game }6}(\mathcal{A})| \leq Adv_{q,g}^{\text{DDH}}(\mathcal{C}). \tag{11}$$

Game 6 and Game 7: Same as Game 3 and Game 4 in Case A.1.1.

$$|Adv_{\text{Game }6}(\mathcal{A}) - Adv_{\text{Game }7}(\mathcal{A})| \leq Adv_{\text{KDF}}(\mathcal{B}). \tag{12}$$

Semantic security of the session key in Game 7: Same as the semantic security in Game 4 in Case A.1.1.

$$Adv_{\text{Game }7}(\mathcal{A}) = 0 \tag{13}$$

Using equations (6)–(13) we find,

$$Adv_{\pi}^{(\cdot)\text{AFL-eCK}}(\mathcal{A}) \leq {N_P}^2 {N_s}^2 \big(\epsilon + 2Adv_{\text{PKE}}^{\text{CPLA2}}(\mathcal{D}) + Adv_{q,g}^{\text{DDH}}(\mathcal{C}) + Adv_{\text{KDF}}(\mathcal{B})\big).$$

A.1.3 Adversary corrupts the partner, but not the owner to the test session.

Game 1: Same as Game 1 in Case A.1.1.
Game 2: Same as Game 2 in Case A.1.1.
Game 3: Same as Game 3 in Case A.1.2.
Game 4: Same as Game 4 in Case A.1.2.

Game 5: Same as Game 3 in Case A.1.1.
Game 6: Same as Game 4 in Case A.1.1.

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

$$Adv_{\text{Game }1}(\mathcal{A}) = Adv_{\pi}^{(\cdot)\text{AFL-eCK}}(\mathcal{A}). \tag{14}$$

Game 1 and Game 2: Same as Game 1 and Game 2 in Case A.1.1.

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

Game 2 and Game 3: Same as Game 2 and Game 3 in Case A.1.2.

$$|Adv_{\text{Game }2}(\mathcal{A}) - Adv_{\text{Game }3}(\mathcal{A})| \leq \epsilon. \tag{16}$$

Game 3 and Game 4: Same as Game 3 and Game 4 in Case A.1.2.

$$|Adv_{\text{Game }3}(\mathcal{A}) - Adv_{\text{Game }4}(\mathcal{A})| \leq Adv_{\text{PKE}}^{\text{CPLA2}}(\mathcal{D}). \tag{17}$$

Game 4 and Game 5: Same as Game 2 and Game 3 in Case A.1.1.

$$|Adv_{\text{Game }4}(\mathcal{A}) - Adv_{\text{Game }5}(\mathcal{A})| \leq Adv_{q,g}^{\text{DDH}}(\mathcal{C}). \tag{18}$$

Game 5 and Game 6: Same as Game 3 and Game 4 in Case A.1.1.

$$|Adv_{\text{Game }5}(\mathcal{A}) - Adv_{\text{Game }6}(\mathcal{A})| \leq Adv_{\text{KDF}}(\mathcal{B}). \tag{19}$$

Semantic security of the session key in Game 6: Same as the semantic security in Game 4 in Case A.1.1.

$$Adv_{\text{Game }6}(\mathcal{A}) = 0 \tag{20}$$

Using equations (14)–(20) we find,

$$Adv_{\pi}^{(\cdot)\text{AFL-eCK}}(\mathcal{A}) \leq N_P^2 {N_s}^2 \big(\epsilon + Adv_{\text{PKE}}^{\text{CPLA2}}(\mathcal{D}) + Adv_{q,g}^{\text{DDH}}(\mathcal{C}) + Adv_{\text{KDF}}(\mathcal{B})\big).$$

A.1.4 Adversary corrupts the owner, but not the partner to the test session.

Game 1: Same as Game 1 in Case A.1.1.
Game 2: Same as Game 2 in Case A.1.1.
Game 3: Same as Game 3 in Case A.1.2.
Game 4: Same as Game 5 in Case A.1.2.
Game 5: Same as Game 3 in Case A.1.1.
Game 6: Same as Game 4 in Case A.1.1.

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

$$Adv_{\text{Game }1}(\mathcal{A}) = Adv_{\pi}^{(\cdot)\text{AFL-eCK}}(\mathcal{A}). \tag{21}$$

Game 1 and Game 2: Same as Game 1 and Game 2 in Case A.1.1.

$$Adv_{\text{Game }2}(\mathcal{A}) = \frac{1}{{N_P}^2 N_s^2} Adv_{\text{Game }1}(\mathcal{A}). \tag{22}$$

Game 2 and Game 3: Same as Game 2 and Game 3 in Case A.1.2.

$$|Adv_{\text{Game }2}(\mathcal{A}) - Adv_{\text{Game }3}(\mathcal{A})| \leq \epsilon. \tag{23}$$

Game 3 and Game 4: Same as Game 4 and Game 5 in Case A.1.2.

$$|Adv_{\text{Game }3}(\mathcal{A}) - Adv_{\text{Game }4}(\mathcal{A})| \leq Adv_{\text{PKE}}^{\text{CPLA2}}(\mathcal{D}). \tag{24}$$

Game 4 and Game 5: Same as Game 2 and Game 3 in Case A.1.1.

$$|Adv_{\text{Game }4}(\mathcal{A}) - Adv_{\text{Game }5}(\mathcal{A})| \leq Adv^{\text{DDH}}_{q,g}(\mathcal{C}). \tag{25}$$

Game 5 and Game 6: Same as Game 3 and Game 4 in Case A.1.1.

$$|Adv_{\text{Game }5}(\mathcal{A}) - Adv_{\text{Game }6}(\mathcal{A})| \leq Adv_{\text{KDF}}(\mathcal{B}). \tag{26}$$

Semantic security of the session key in Game 6: Same as the semantic security in Game 4 in Case A.1.1.

$$Adv_{\text{Game }6}(\mathcal{A}) = 0 \tag{27}$$

Using equations (21)–(27) we find,

$$Adv^{(\cdot)\text{AFL-eCK}}_{\pi}(\mathcal{A}) \leq N_P^2 {N_s}^2 \big(\epsilon + Adv^{\text{CPLA2}}_{\text{PKE}}(\mathcal{D}) + Adv^{\text{DDH}}_{q,g}(\mathcal{C}) + Adv_{\text{KDF}}(\mathcal{B})\big).$$

From case A.1 we get,

$$Adv^{(\cdot)\text{AFL-eCK}}_{\pi}(\mathcal{A}) \leq N_P^2 {N_s}^2 \big(\epsilon_{pg} + 2Adv^{\text{CPLA2}}_{\text{PKE}}(\mathcal{D}) + Adv^{\text{DDH}}_{q,g}(\mathcal{C}) + Adv_{\text{KDF}}(\mathcal{B})\big).$$

A.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. Assume that the adversary $\mathcal{A}$ asks a `Send` query to some fresh oracle, 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. 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, the Game 2 challenger guesses the identity, V^*, of the partner principal to the test session and if the guess in incorrect it aborts the game.

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

$$Adv_{\text{Game }1}(\mathcal{A}) = Adv^{(\cdot)\text{AFL-eCK}}_{\pi}(\mathcal{A}). \tag{28}$$

Game 1 and Game 2: The probability of Game 2 to be aborted due to incorrect guess of the partner principal to the test session is $1 - \frac{1}{N_P^2}$. Unless the incorrect guess happens, Game 2 is identical to Game1. Hence,

$$Adv_{\text{Game }2}(\mathcal{A}) = \frac{1}{N_P} Adv_{\text{Game }1}(\mathcal{A}). \tag{29}$$

The owner principal accepts the message coming from the intended partner, because the owner computes $\text{Vfy}(vk_{V^*}, X_{V^*}, \sigma_{V^*})$ is "true". But the principal V^* is not corrupted and the message X_{V^*} is not signed by the principal V^*, because of no partner. Hence,

$$Adv_{\text{Game }2}(\mathcal{A}) = Adv^{\text{UFCMLA}}_{\text{SIG}}(\mathcal{E}). \tag{30}$$

Using equations (28)–(30) we find,

$$Adv^{(\cdot)\text{AFL-eCK}}_{\pi}(\mathcal{A}) = N_P Adv^{\text{UFCMLA}}_{\text{SIG}}(\mathcal{E}).$$

From case A.2 we get,

$$Adv^{(\cdot)\text{AFL-eCK}}_{\pi}(\mathcal{A}) \leq N_P Adv^{\text{UFCMLA}}_{\text{SIG}}(\mathcal{E}).$$

Combine Case A.1 and A.2 to obtain the relationship in Theorem 4.1.

$$Adv^{(\cdot)\text{AFL-eCK}}_{\pi}(\mathcal{A}) \leq \max\Big[N_P Adv^{\text{UFCMLA}}_{\text{SIG}}(\mathcal{E}), \quad N_P^2 {N_s}^2 \big(\epsilon_{pg} + 2Adv^{\text{CPLA2}}_{\text{PKE}}(\mathcal{D}) + Adv^{\text{DDH}}_{q,g}(\mathcal{C}) + Adv_{\text{KDF}}(\mathcal{B})\big)\Big].$$

□