On Continuous After-the-Fact Leakage-Resilient Key Exchange *

Mohsen Toorani
Department of Informatics
University of Bergen, Norway
mohsen.toorani@ii.uib.no

Abstract

Side-channel attacks are severe type of attack against implementation of cryptographic primitives. Leakage-resilient cryptography is a new theoretical approach to formally address the problem of side-channel attacks. Recently, the Continuous After-the-Fact Leakage (CAFL) security model has been introduced for two-party authenticated key exchange (AKE) protocols. In the CAFL model, an adversary can adaptively request arbitrary leakage of long-term secrets even after the test session is activated. It supports continuous leakage even when the adversary learns certain ephemeral secrets or session keys. The amount of leakage is limited per query, but there is no bound on the total leakage. A generic leakage-resilient key exchange protocol π has also been introduced that is formally proved to be secure in the CAFL model. In this paper, we comment on the CAFL model, and show that it does not capture its claimed security. Furthermore, we present an attack and counterproofs for the security of protocol π which invalidates the formal security proofs of protocol π in the CAFL model.

Keywords: Leakage-resilient cryptography, Cryptographic protocols, Key exchange, Security models.

1 Introduction

Security of a cryptosystem is usually proven in a formal model of computation where proofs rely on the assumption that the adversary has no information about secret keys. However, the security must hold for the actual implementation of the algorithm in the real world where the adversary might gain some information about those secrets, by observing the behavior of the algorithm. Leakage of secret parameters are not usually captured by the mathematical definition of the algorithm, but caused by its implementation. Side-channel, fault, probing, and

*A shorter version of this article will be published in [20], DOI: 10.1145/2694805.2694811.

memory attacks are different types of physical attacks to a cryptosystem [19]. Side-channel attacks are low cost, realistic, and usually considered as the most dangerous type of physical attacks. Computing devices leak information not just through input-output interaction, but through physical characteristics of computation such as power consumption [15], timing [5, 12], and electromagnetic radiation [10]. Such information leakage can break many cryptosystems in common use, and are feasible when an adversary has access to the device, as it is often the case for devices like smart-cards, TPM chips, mobile phones and laptops.

Most countermeasures to leakage attacks are ad-hoc, offer only partial remedy, and fail to capture the problem in its entirety. Recently, the notion of *Leakage Resilient Cryptography* [19] has been introduced to set the theoretical foundations to formally address the problem of side-channel attacks. A leakage-resilient cryptosystem remains secure even if arbitrary, yet bounded, information about the secret key and possibly other internal state information is leaked to an adversary [11]. As AKE protocols are one of the most important cryptographic primitives, it is important to construct leakage-resilient key exchange protocols.

Existing security models for two-party AKE protocols such as the CK [6] and the eCK [14] have considered an adversary who can fully compromise some, but not all secret keys. However, they do not capture the security with leakage of partial secret information. This motivates development of leakage-resilient key exchange security models and protocols.

Alwen et al. [4] presented a leakage-resilient PKI-based AKE protocol in the random oracle model, where they introduced the leakage-resilient security on the CK model. However, their protocol is three-pass and signature-based. If the session state in the CK model contains random coins to generate signatures, an adversary can obtain random coins for the underlying signature scheme through a `Session-State Reveal` query. The protocol will be then insecure if the underlying signature scheme is vulnerable by the random coin leakage [16]. Dodis et al. [8] proposed a framework to construct a leakage-resilient AKE protocol on the CK model. However, their protocol is three-pass, and does not capture the Key Compromise Impersonation (KCI) attack.

In an attempt to construct two-party two-pass leakage-resilient AKE protocols based on the eCK security model, three models and protocols have been proposed: The first formalization was due to Moriyama and Okamoto [16] where they proposed a two-pass protocol. However, total amount of allowed leakage is bounded in their model, and it does not provide security against continuous leakage. Furthermore, the adversary cannot obtain leakage information after the test session is activated. Alawatugoda et al. [3] tried to overcome the limitations of the Moriyama-Okamoto (MO) model by proposing the ASB model which considers both continuous and bounded leakage, and allows leakage after the test session is activated. However, their model

is just accompanied with a generic construction of a two-pass protocol for the ASB bounded leakage model. They did not propose any protocol for the ASB continuous leakage model, and left it as an open problem. In an attempt to solve the mentioned open problem, they introduced the Continuous After-the-Fact Leakage (CAFL) model [1, 2] which is a weaker variant of the ASB continuous leakage model. The CAFL is weaker in terms of the freshness condition, and aims to capture the continuous leakage, even after the test session is activated. They proposed a generic protocol π which was formally proved to be secure in the CAFL model. The protocol π is instantiated using the CCLA2-secure public-key cryptosystem of Dziembowski et al. [9], but can be instantiated using any CCLA2-secure public-key cryptosystem. They proved that protocol π can achieve the same leakage tolerance as the underlying public-key cryptosystem tolerates.

In this paper, we comment on the CAFL model, and show that it does not capture its claimed security. Furthermore, we present counterproofs for the security of protocol π in the CAFL model. This invalidates the formal security proofs in [1, 2]. We present an attack to the protocol π, and show that it is insecure in the CAFL, ABS, MO and eCK security models. The rest of this paper is organized as follows. Leakage-resilient storage and CCLA2-security are briefly introduced in Section 2 and Section 3, respectively. The CAFL model is presented and commented in Section 4. The protocol π is reviewed in Section 5. An attack and counterproofs for the security of protocol π is provided in Section 6.

2 Leakage-Resilient Storage

A leakage-resilient storage (LRS) $\Phi = (\texttt{Encode}, \texttt{Decode})$ allows to store a secret S in an encoded form such that even given leakage from the encoding, no adversary learns information about the encoded values. Protocol π is instantiated using Dziembowski et al.'s public-key encryption scheme [9] which uses the following LRS model:

The memory of the device is split into three parts L,R and C, where initially C is empty. L and R are chosen uniformly so that $\langle L, R\rangle = S$ in which $\langle .\rangle$ denotes inner product of the vectors. The leakage is modeled as a two-party protocol $\Pi = (P_L, P_R)$, executed between P_L and P_R: P_L is controlling L, and P_R holds R. C is only used to communicate information between L and R, and to store messages exchanged between them.

Let $\lambda \in \mathbb{N}$ denotes total number of bits that an adversary can learn from each $L, R \in \{0,1\}^s$. A λ-leakage game is played between an adaptive λ-limited adversary $\mathcal{M}$ and a *leakage oracle* $\Omega(L, R)$ as follows: For some $t \in \mathbb{N}$, the adversary $\mathcal{M}$ can adaptively issue a sequence $\{(f_i, x_i)\}_{i=1}^{t}$ of requests to the oracle $\Omega(L, R)$, where $x_i \in \{L, R\}$, and $f_i : \{0,1\}^s \to \{0,1\}^{\lambda_i}$. For the i^{th}

query, the oracle Ω replies with $f_i(x_i)$. Let $Out(\mathcal{M}, \Omega(L,R)) = (f_1(x_1), ..., f_t(x_t))$ denotes the output of $\mathcal{M}$ at the end of this game.

Let $\Delta(X_0; X_1) = \frac{1}{2}\sum_{x\in\mathcal{X}} |\Pr(X_0 = x) - \Pr(X_1 = x)|$ denotes the statistical distance between random variables X_0 and X_1. An LRS Φ is said to be (λ, ϵ)-secure, if for any $S, S' \in M$ and any λ-limited adversary $\mathcal{M}$, we have $\Delta(Out(\mathcal{M}, \Omega(L,R)); Out(\mathcal{M}, \Omega(L',R'))) \leq \epsilon$, where $(L,R) = \texttt{Encode}(S)$ and $(L',R') = \texttt{Encode}(S')$. The following LRS $\Phi_{\mathbb{F}}^{n,m} = (\texttt{Encode}_{\mathbb{F}}^{n,m}, \texttt{Decode}_{\mathbb{F}}^{n,m})$ is (λ, ϵ)-secure, and allows to efficiently store elements $S \in \mathbb{F}^m$ [9]:

- $\texttt{Encode}_{\mathbb{F}}^{n,m}(S)$: Randomly select $L \leftarrow \mathbb{F}^n \setminus \{(0^n)\}$, sample $R \leftarrow \mathbb{F}^{n\times m}$ such that $L.R = S$. Output (L,R).

- $\texttt{Decode}_{\mathbb{F}}^{n,m}(L,R)$: Output S.

A probabilistic protocol $(L',R') \leftarrow \texttt{Refresh}(L,R)$ has been presented in [9] that securely refreshes $(L,R) \leftarrow \texttt{Encode}(S)$, even when the adversary can continuously observe the computation from the refreshing process. `Refresh` takes as input (L,R), and outputs a fresh encoding (L',R') of S such that $\langle L,R\rangle = \langle L',R'\rangle = S$. For correctness, we require that $\texttt{Decode}(L,R) = \texttt{Decode}(L',R')$.

The computation of `Refresh` will be structured into several rounds, where in each round only (L,C) or (R,C) are touched, but never L and R at the same time. The adversary adaptively leaks a bounded amount of information from (L,C) and (R,C). Initially, P_L holds L and P_R holds R. At any point during the execution of the protocol, the adversary can interact with a leakage oracle, and learn information about the internal state of P_L and P_R. The only way in which the adversary can interact with the protocol is via the leakage oracle. At the end, the players output the refreshed encoding (L',R'), which is the new state of the protocol.

Definition 1. ***A $(\ell, \lambda, \epsilon)$-refreshing protocol****: For a LRS Φ = (*`Encode`*, *`Decode`*) with message space M, a refreshing protocol (*`Refresh`*, Φ) is $(\ell, \lambda, \epsilon)$-secure, if for every λ-limited adversary $\mathcal{M}$ and any two secrets $S, S' \in M$, we have $\Delta(\texttt{Exp}_{(\texttt{Refresh},\Phi)}(\mathcal{M}, S, \ell); \texttt{Exp}_{(\texttt{Refresh},\Phi)}(\mathcal{M}, S', \ell)) \leq \epsilon$ where $\texttt{Exp}_{(\texttt{Refresh},\Phi)}(\mathcal{M}, S, \ell)$ denotes an experiment which runs the refreshing protocol for ℓ rounds, and lets the adversary play a leakage game in each round. The experiment is defined as follows:*

1. *For a secret S, we generate the initial encoding as $(L^0, R^0) \leftarrow \texttt{Encode}(S)$.*
2. *For $i = 1$ to ℓ, run $\mathcal{M}$ against the i^{th} round of the refreshing protocol: $\mathcal{M} \leftrightarrows (\texttt{Refresh}(L^{i-1}, R^{i-1}) \rightarrow (L^i, R^i))$ in which current initial state of round i is (L^{i-1}, R^{i-1}), and the next state of P_L and P_R will be (L^i, R^i).*
3. *Return $b \in \{0,1\}$ that $\mathcal{M}$ outputs.*

The $\texttt{Refresh}_{\mathbb{F}}^{n,m}$ protocol presented in [9] is $(\ell, 0.15 \cdot n \cdot \log|\mathbb{F}| - 1, negl(n))$-refreshing protocol for the LRS $\Phi_{\mathbb{F}}^{n,m}$ where $n \in \mathbb{N}$ is a security parameter, $|\mathbb{F}| = \Omega(n)$, $m = o(n)$, ℓ is a polynomial in n, and $negl(n)$ is a negligible function.

3 CCLA2-Security

Definition 2. ***Security Against Adaptively Chosen Ciphertext After-the-fact Leakage Attacks (CCLA2-secure)****: Let $k \in \mathbb{N}$ be the security parameter. A public-key cryptosystem $PKE = (KG, Enc, Dec)$ is $\lambda(k)$-IND-CCLA2 secure if for any PPT $\lambda(k)$-limited adversary $\mathcal{M}$, the probability of winning distinguishing Game 1 is at most $1/2 + negl(k)$ [9]. $KG(1^k)$ denotes the key generation algorithm which outputs a secret key sk, and a public key pk. $Enc(.)$ is the encryption algorithm which encrypts message m using pk, and outputs ciphertext c. $Dec(.)$ is the decryption algorithm which uses the secret key for decryption.*

Game 1.

1. *Sample $b \leftarrow \{0,1\}$, and $(sk, pk) \leftarrow KG(1^k)$. Give pk to $\mathcal{M}$.*
2. *Repeat until $\mathcal{M}(1^k)$ outputs ((m_0, m_1): $\mathcal{M}(1^k) \leftrightarrows (Dec(sk, c) \rightarrow (sk', m))$, where for each decryption query c, the adversary additionally retrieves up to $\lambda(k)$ bits about the current secret state sk. For the next round, update the secret state sk to sk' ($sk \leftarrow sk'$).*
3. *The challenger computes $c^* \leftarrow Enc(pk, m_b)$, and gives it to $\mathcal{M}$.*
4. *Repeat until $\mathcal{M}(1^k)$ outputs b' : $\mathcal{M}(1^k) \leftrightarrows (Dec(sk, c) \rightarrow (sk', m))$, where for each decryption query $c \neq c^*$, the adversary additionally retrieves up to $\lambda(k)$ bits about the current secret state sk. For the next round, update the secret state sk to sk'.*
5. *If $b = b'$ then $\mathcal{M}$ wins.*

Dziembowski et al. [9] constructed a λ-IND-CCLA2-secure public-key cryptosystem in the random oracle model in which $\lambda = 0.15 \cdot n \cdot \log p - 1$, if the DDH assumption holds. Their scheme is based on the refreshing protocol presented in Section 2, and uses a simulation sound non-interactive zero knowledge (SS-NIZK) system, $(Prov, Ver)$, for proving the equivalence of discrete logarithms. A NIZK proof system is said to be simulation sound if any adversary has negligible advantage in breaking soundness (i.e., forging an accepting proof for an invalid statement), even after seeing a bounded number of proofs for (in)valid statements [17]. Dziembowski et al.'s CCLA2-secure PKE [9] is as follows:

- $\boldsymbol{KG(1^k)}$: Let $(p, \mathbb{G}) \leftarrow G(1^k)$, $g_1, g_2 \leftarrow \mathbb{G}$, $S = (x_1, x_2) \leftarrow \mathbb{Z}_p^2$, and $(L, R) \leftarrow \texttt{Encode}_{\mathbb{Z}_p}^{n,2}$. Let $sk = (L, R)$ and $pk = (p, g_1, g_2, h = g_1^{x_1} g_2^{x_2})$.

- ***Enc*(pk, m)**: Sample $r \leftarrow \mathbb{Z}_p$ uniformly at random, and compute $c = (u = g_1^r, v = g_2^r, w = h^r m)$. Run the NIZK prover $Prov(u, v, r)$ to obtain a proof π for $\log_{g_1}(u) = \log_{g_2}(v)$. Return (c, π).

- ***Dec*(sk, c)**: Input for decryption is $sk = (L, R)$ where L is given to P_L, and R is given to P_R . P_L and P_R obtain c, and parse it as (u, v, w, π). If $Ver(u, v, \pi) = reject$ then abort. Otherwise, proceed as follows:

  - P_R computes the vector $U = u^{R_1} \odot v^{R_2}$ in which $\odot$ denotes componentwise multiplication of vectors. U is sent to P_L.
  - P_L computes $V = U^{-L}$, and outputs $w \prod_i V_i$.

Leakage from the verification of the NIZK can be omitted as it only includes publicly known values. At any time, the adversary can play a λ-leakage game against $\Omega((L, U); R)$ as mentioned in Section 2.

4 The CAFL Model

In the CAFL model [1], each party U_i where $i \in [1, N_P]$, has a pair of long-term public and secret keys, (pk_{U_i}, sk_{U_i}). Each party may run multiple instances of the protocol concurrently or sequentially. The term *principal* refers to a party involved in a protocol instance. The term *session* is used for identifying a protocol instance at a principal. The notation $\prod_{U,V}^{s}$ 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. The principal which responds to the first protocol message is the *responder* of the session. A session $\prod_{U,V}^{s}$ enters an *accepted* state when it computes a session key. A session may terminate without ever entering into the accepted state. The information of whether a session has terminated with or without acceptance is public.

Definition 3. ***Partner sessions****: Two oracles $\prod_{U,V}^{s}$ and $\prod_{U',V'}^{s'}$ are said to be* partners *in the CAFL model if all the following conditions are satisfied:*

1. *$\prod_{U,V}^{s}$ and $\prod_{U',V'}^{s'}$ have computed session keys.*
2. *Sent messages from $\prod_{U,V}^{s}$ = Received messages to $\prod_{U',V'}^{s'}$.*
3. *Sent messages from $\prod_{U',V'}^{s'}$ = Received messages to $\prod_{U,V}^{s}$.*
4. *$U' = V$ and $V' = U$.*

5. *If U is the initiator, then V is the responder, or vise versa.*

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

Adversary: The adversary $\mathcal{M}$ is a probabilistic polynomial time (PPT) algorithm which controls all interaction and communication between parties. $\mathcal{M}$ initiates sessions at parties, and delivers protocol messages. $\mathcal{M}$ can create, change, delete, or reorder messages. $\mathcal{M}$ can compromise certain short-term and long-term secrets. Specially for modeling leakage, whenever a party performs an operation using its long-term key, $\mathcal{M}$ obtains some leakage information about the long-term key.

Queries: Five queries are defined in the CAFL model: `Send`, `SessionKeyReveal`, `EphemeralKeyReveal`, `Corrupt`, and `Test`. `Send` allows $\mathcal{M}$ to run the protocol. `SessionKeyReveal`, `EphemeralKeyReveal`, and `Corrupt` queries allow $\mathcal{M}$ to compromise certain session specific information from the protocol principals. The `Test` query is used to formalize the notion of semantic security of a key exchange protocol. Once the oracle $\prod_{U,V}^{s}$ has accepted a session key, $\mathcal{M}$ attempts to distinguish it from a random session key by asking the `Test` query. Definitions of queries are as follows [1]:

- `Send`$(U, V, s, m, \mathbf{f})$ query: The oracle $\prod_{U,V}^{s}$ computes the next protocol message according to the protocol specification on receipt of m, and sends it to the adversary $\mathcal{M}$, along with the leakage $\mathbf{f}(sk_U)$. $\mathcal{M}$ can also use this query to activate a new protocol instance as an initiator with blank m and $\mathbf{f}$.
- `SessionKeyReveal`(U, V, s) query: $\mathcal{M}$ is given the session key of the oracle $\prod_{U,V}^{s}$, if the oracle $\prod_{U,V}^{s}$ is in the accepted state.
- `EphemeralKeyReveal`(U, V, s) query: $\mathcal{M}$ is given the ephemeral keys of the oracle $\prod_{U,V}^{s}$.
- `Corrupt`(U) query: $\mathcal{M}$ is given the long-term secrets of the principal U. This query does not reveal any session keys or ephemeral keys to $\mathcal{M}$.
- `Test`(U, V, s) query: When $\mathcal{M}$ asks the `Test` query, the oracle $\prod_{U,V}^{s}$ first chooses a random bit $b \leftarrow \{0, 1\}$. If $b = 1$, then the actual session key is returned to $\mathcal{M}$. Otherwise, a random string chosen from the same session key space is returned to $\mathcal{M}$. This query is only allowed to be asked once across all sessions.

Definition 4. λ-*CAFL-freshness*: *Let λ be the leakage bound per occurrence. An oracle $\prod_{U,V}^{s}$ is said to be λ-CAFL-fresh if and only if:*

1. *The oracle $\prod_{U,V}^{s}$ or its partner, $\prod_{V,U}^{s'}$ (if it exists) has not been asked a* `SessionKeyReveal`.
2. *If the partner $\prod_{V,U}^{s'}$ exists, none of the following combinations have been asked:*

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

3. *If the partner* $\prod_{V,U}^{s'}$ *does not exist, none of the following combinations have been asked:*
   - `Corrupt`(V).
   - `Corrupt`(U) *and* `EphemeralKeyReveal`(U, V, s).
4. *For each* `Send`$(., U, ., ., \boldsymbol{f})$ *query, the output of* $\boldsymbol{f}$ *is at most* λ *bits.*
5. *For each* `Send`$(., V, ., ., \boldsymbol{f})$ *query, the output of* $\boldsymbol{f}$ *is at most* λ *bits.*

The Security of a key exchange protocol in the CAFL model is defined using the following security game, which is played by a PPT adversary $\mathcal{M}$ against the protocol challenger.

Game 2.

1. $\mathcal{M}$ *may ask any of* `Send`, `SessionKeyReveal`, `EphemeralKeyReveal`, *and* `Corrupt` *queries to any oracle at will.*
2. $\mathcal{M}$ *chooses a* λ*-CAFL-fresh oracle, and asks a* `Test` *query.*
3. $\mathcal{M}$ *continues asking* `Send`, `SessionKeyReveal`, `EphemeralKeyReveal`, *and* `Corrupt` *queries.* $\mathcal{M}$ *may not ask a query that violates the* λ*-CAFL-freshness of the test session.*
4. $\mathcal{M}$ *outputs the bit* $b' \leftarrow \{0, 1\}$ *which is its guess of the value* b *on the test session.* $\mathcal{M}$ *wins if* $b' = b$.

Definition 5. ***λ-CAFL-security****: A protocol* π *is said to be* λ*-CAFL-secure if there is no PPT algorithm* $\mathcal{M}$ *that can win the above game with non-negligible advantage. The advantage of an adversary* $\mathcal{M}$ *is defined as* $Adv_{\pi}^{CAFL}(\mathcal{M}) = \mid 2\Pr(Succ_{\mathcal{M}}) - 1 \mid$, *where* $Succ(\mathcal{M})$ *is the event that* $\mathcal{M}$ *wins Game 2.*

4.1 Comments on the CAFL Model

Although the ASB and CAFL models are based on the eCK model, they use different names and orders for queries than those of the eCK model. Instead of the `Reveal` query in the eCK model, they have the `SessionKeyReveal` query. Instead of the `Long-Term Key Reveal` query in the

eCK model, they have the `Corrupt` query. The `Corrupt` query was defined in the CK model but it has been removed from the eCK model as the adversary can reveal all the secret information of the party using `Long-Term Key Reveal`, `Ephemeral Key Reveal`, and `Reveal` queries. The CAFL model is a weaker variant of the ASB continuous model in terms of two aspects:

- The combination "`Corrupt`(U) and `Corrupt`(V)" has been excluded from the freshness condition for passive attacks (condition 2 of Definition 4). It means that $\mathcal{M}$ cannot have static long-term secret keys of both U and V, but just one of them. The CAFL model does not provide even the weak-Perfect Forward Secrecy (weak-PFS) that is an essential requirement for AKE protocols [13]. Then, it only allows partial weak forward secrecy upon compromising the long-term secret key of only one participant.

- The combination "`EphemeralKeyReveal`(U, V, s) and `EphemeralKeyReveal`(V, U, s')" has been excluded from freshness condition for passive attacks.

Those two combinations that are excluded from the CAFL model, are allowed in the eCK, ASB and MO models. The eCK model considers weak-PFS instead of PFS due to a belief in the security community that no two-pass AKE protocol can achieve PFS, and the best they can achieve is the weak-PFS. The belief stems from an attack presented in [13], but is disputed and considered as an incorrect belief in [7] where the eCK-PFS model is introduced.

In the CAFL model, the adversary is allowed to obtain leakage from the uncorrupted principal, in addition to allowing the adversary to corrupt one of the protocol principals. The side channel attacks and continuous leakage of long-term secrets are modeled through the leakage function $\mathbf{f}$ in the `Send` query, assuming that the leakage happens when computations take place in principals. If the `Send` query is used without $\mathbf{f}$, we will have the non-leakage version of the CAFL model which must address KCI attacks and partial weak forward secrecy.

It is claimed in [1, 2] that the CAFL model addresses "most real world attack scenarios" including active adversarial capabilities (via the `Send` query), cold-boot attacks (via the `Corrupt` query), weak random number generators (through the `EphemeralKeyReveal` query), known key attacks (via the `SessionKeyReveal` query), and malware attacks (via the `EphemeralKeyReveal` or the `Corrupt` queries). It has also been claimed that the CAFL model addresses all the attack scenarios which are addressed by the ASB model, except weak forward secrecy. A counterproof to two latest claims, i.e covering most real world attack scenarios, and covering all attacks scenarios covered by the ASB model, is the following practical attack scenario that is allowed in the eCK, ASB and MO models but not addressed by the CAFL model because of excluding the combination of "`EphemeralKeyReveal`(U, V, s) and `EphemeralKeyReveal`(V, U, s')" from the freshness condition in Definition 4: "Two honest parties execute matching sessions. Adversary

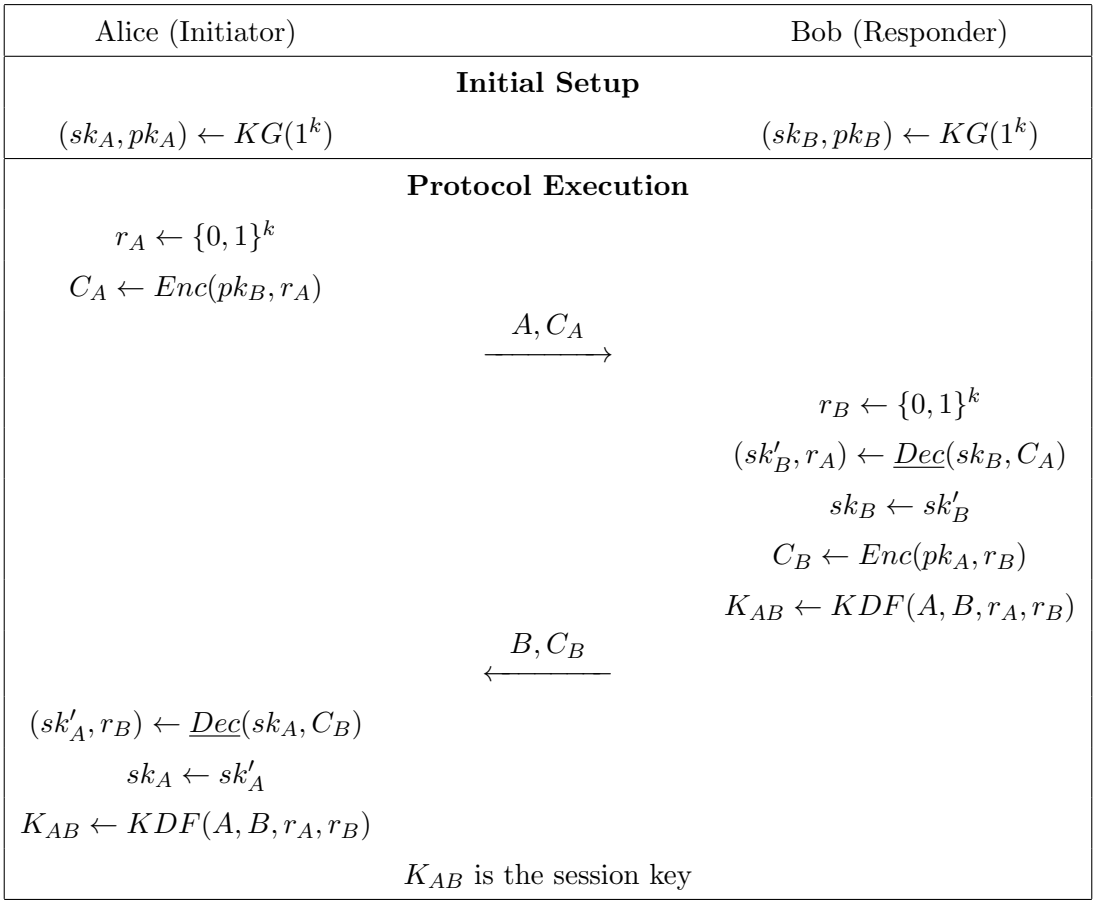

Figure 1: Protocol π [1]. Underline denotes operations to which leakage functions apply.

reveals the ephemeral secret keys of both parties, and tries to learn the session key" [14].

5 Review of the protocol

Figure 1 depicts the generic protocol π [1]. The protocol must be instantiated using a CCLA2-secure public-key encryption scheme, e.g. the scheme presented in Section 3. In Figure 1, A denotes Alice, B denotes Bob, and KDF is a secure key derivation function which generates the session key. KG, Enc, and Dec are defined in Section 3.

6 Counterproof for security of the protocol

Using the game hopping technique [18], it is proved in [1, 2] that "*The protocol π is λ-CAFL-secure, whenever the underlying public key cryptosystem PKE is CCLA2-secure, and the key derivation function KDF is secure with respect to a uniformly random key material.*" However, we prove that the protocol π is not λ-CAFL-secure. First, we prove that an ephemeral-KCI

attack is allowed in the CAFL model, and it does not violate the λ-CAFL-freshness of the test session. Then, we show that protocol π is vulnerable to the ephemeral-KCI attack. Finally, we prove that protocol π is not λ-CAFL-secure. This invalidates the security claims and proofs in [1, 2].

The ephemeral-KCI attack is a variant of the KCI attack in public key-based key exchange protocols. This attack has been considered in the eCK model, and is of practical importance. It considers consequences of weak random number selections or when the attacker can affect the random number generation at the user end, for example by some security holes in web browsers. Resilience to the ephemeral-KCI attack means that $\mathcal{M}$ that knows ephemeral secret key of A but does not know her long-term secret key, should not be able to impersonate B, and share a session key with A.

Proposition 1. *The ephemeral-KCI attack does not violate the λ-CAFL-freshness of the test session.*

Proof. It is sufficient to show that all conditions for the λ-CAFL-freshness are satisfied by an ephemeral-KCI attack. The ephemeral-KCI attack is an active attack in which an adversary reveals the ephemeral secret key of a party, and impersonates other parties to this party. It can be accomplished using `Send` (without **f**) and `EphemeralKeyReveal` queries. As it does not use `SessionKeyReveal` query, the first condition of the λ-CAFL-freshness is satisfied. As it is an active attack, for an oracle $\prod^{s}_{U,V}$, the corresponding partner $\prod^{s'}_{V,U}$ does not exist, and the second condition of the λ-CAFL-freshness is not the case. As $\mathcal{M}$ does not ask `Corrupt` queries, the third condition of the λ-CAFL-freshness is satisfied. The fourth and fifth conditions of the λ-CAFL-freshness are not the case because the ephemeral-KCI attack deals with the non-leakage variant of the CAFL model (`Send` query without **f**). □

Proposition 2. *The ephemeral-KCI attack is allowed in the CAFL model.*

Proof. $\mathcal{M}$ performs a simplified variant of Game 2. Among allowed queries in Game 2, $\mathcal{M}$ just uses the `Test`, `Send` (without **f**) and `EphemeralKeyReveal` queries. From Proposition 1, the attack does not violate the λ-CAFL-freshness of the test session. If an ephemeral-KCI attack is applicable to a protocol π, $\mathcal{M}$ can reach to a key agreement with the owner of the `EphemeralKeyReveal` query with $\Pr(Succ_{\mathcal{M}}) = 1$. According to Definition 5, we have $Adv^{CAFL}_{\pi}(\mathcal{M}) = 1$. The advantage of an adversary will be $\frac{1}{2}$ if one uses the typical formula, and defines the advantage as $\mid \Pr(Succ_{\mathcal{M}}) - \frac{1}{2} \mid$.

Here is how $\mathcal{M}$ can win Game 2 with probability 1: By a successful ephemeral-KCI attack which is a deterministic algorithm, $\mathcal{M}$ reaches to a key agreement with owner of the `EphemeralKeyReveal` query. For stage 4 of Game 2, $\mathcal{M}$ looks to output of the `Test` query. If the

output of the `Test` query is equal to the computed session key which means $b = 1$, $\mathcal{M}$ outputs the bit $b' = 1$. Otherwise, $\mathcal{M}$ outputs the bit $b' = 0$. Then, we have $b = b'$ with $\Pr(Succ_{\mathcal{M}}) = 1$. □

Proposition 3. *The protocol π is vulnerable to the ephemeral-KCI attack.*

Proof. The proof is by presenting an attack scenario in which $\mathcal{M}$ that has learned the ephemeral secret key r_A, impersonates B and shares a session key with A.

- A selects random number r_A, computes $C_A \leftarrow Enc(pk_B, r_A)$, and sends $\{A, C_A\}$ to $\mathcal{M}$.
- $\mathcal{M}$ selects random number $r_{\mathcal{M}}$, computes $C_{\mathcal{M}} \leftarrow Enc(pk_A, r_{\mathcal{M}})$ and $K_{A\mathcal{M}} \leftarrow KDF(A, B, r_A, r_{\mathcal{M}})$, and sends $\{B, C_{\mathcal{M}}\}$ to A.
- A computes $(sk'_A, r_{\mathcal{M}}) \leftarrow Dec(sk_A, C_{\mathcal{M}})$, updates the secret key sk to sk', and computes $K_{A\mathcal{M}} \leftarrow KDF(A, B, r_A, r_{\mathcal{M}})$. A and $\mathcal{M}$ reach to the same session key $K_{A\mathcal{M}}$. $\mathcal{M}$ could successfully impersonate B.

□

Theorem 1. *The protocol π is not λ-CAFL-secure.*

Proof. From Proposition 2, we know that the ephemeral-KCI attack is allowed in the CAFL model. According to Proposition 1, the ephemeral-KCI attack does not violate the λ-CAFL-freshness of the test session. From Proposition 3, we know that protocol π is vulnerable to the ephemeral-KCI attack. The protocol π is not λ-CAFL-secure because $\mathcal{M}$ wins Game 3 with probability 1 as we will always have $b' = b$.

Game 3.

1. *$\mathcal{M}$ activates a new protocol instance by* `Send`(A, B, s) *query. The oracle $\prod^s_{A,B}$ sends $\{A, C_A\}$ to $\mathcal{M}$.*
2. *$\mathcal{M}$ asks* `EphemeralKeyReveal`(A, B, s) *query, and gets r_A.*
3. *$\mathcal{M}$ selects random number $r_{\mathcal{M}}$, computes $C_{\mathcal{M}} \leftarrow Enc(pk_A, r_{\mathcal{M}})$ and $K_{A\mathcal{M}} \leftarrow KDF(A, B, r_A, r_{\mathcal{M}})$. $\mathcal{M}$ issues* `Send`$(A, B, s, \{B, C_{\mathcal{M}}\})$ *query.*
4. *$\mathcal{M}$ asks* `Test`(A, B, s) *query. The oracle $\prod^s_{A,B}$ chooses a random bit $b \leftarrow \{0, 1\}$. If $b = 1$, then the actual session key $K_{A\mathcal{M}}$ is returned to $\mathcal{M}$. Otherwise, a random string chosen from the same session keyspace is returned to $\mathcal{M}$.*
5. *$\mathcal{M}$ compares the received string with its computed session key $K_{A\mathcal{M}}$. If they are equal, $\mathcal{M}$ outputs $b' = 1$. Otherwise, $\mathcal{M}$ outputs $b' = 0$.*

□

Theorem 1 is in contradiction with security claims and proofs in [1, 2], and invalidates those security proofs. The protocol π does not provide weak-PFS. It is not secure in the CAFL, ASB, MO, and eCK security models because of its vulnerability to the ephemeral-KCI attack.

7 Conclusion

In this paper, we showed that the Continuous After-the-Fact Leakage (CAFL) security model [1,2] does not capture its claimed security. Furthermore, we provided counterproofs for the security of protocol π which was formally proved to be secure in the CAFL model [1, 2]. We showed that the ephemeral-KCI attack is allowed in the CAFL model, and any protocol that is vulnerable to such an attack cannot be secure in the CAFL model. We showed that protocol π is vulnerable to the ephemeral-KCI attack which invalidates the security proofs in [1, 2]. Protocol π is also insecure in the ASB, MO, and eCK security models.

References

[1] J. Alawatugoda, C. Boyd, and D. Stebila. Continuous after-the-fact leakage-resilient key exchange. In *Proceedings of the 19th Australasian Conference on Information Security and Privacy (ACISP'14)*, LNCS 8544, pages 258–273. Springer, July 2014.

[2] J. Alawatugoda, C. Boyd, and D. Stebila. Continuous after-the-fact leakage-resilient key exchange (full version). *IACR ePrint Archive, 2014/264*, 2014.

[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 (ASIACCS'14)*, pages 207–216, June 2014.

[4] J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *Advances in Cryptology – CRYPTO'09*, pages 36–54. Springer, 2009.

[5] B. B. Brumley and N. Tuveri. Remote timing attacks are still practical. In *ESORICS'11*, pages 355–371. Springer, 2011.

[6] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology – EUROCRYPT'01, LNCS 2045*, pages 453–474. Springer, 2001.

[7] C. Cremers and M. Feltz. Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal. In *ESORICS'12, LNCS 7459*, pages 734–751. Springer, 2012.

[8] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Efficient public-key cryptography in the presence of key leakage. In *Advances in Cryptology – ASIACRYPT'10*, pages 613–631. Springer, 2010.

[9] S. Dziembowski and S. Faust. Leakage-resilient cryptography from the inner-product extractor. In *Advances in Cryptology – ASIACRYPT'11, LNCS 7073*, pages 702–721. Springer, 2011.

[10] M. Hutter, S. Mangard, and M. Feldhofer. Power and em attacks on passive 13.56 MHz RFID devices. In *Cryptographic Hardware and Embedded Systems – CHES'07, LNCS 4727*, pages 320–333. Springer, 2007.

[11] J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In *Advances in Cryptology – ASIACRYPT'09, LNCS 5912*, pages 703–720. Springer, 2009.

[12] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology – CRYPTO'96*, pages 104–113. Springer, 1996.

[13] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *Advances in Cryptology – CRYPTO'05*, pages 546–566. Springer, 2005.

[14] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *Provable Security, LNCS 4784*, pages 1–16. Springer, 2007.

[15] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, 51(5):541–552, 2002.

[16] D. Moriyama and T. Okamoto. Leakage resilient eCK-secure key exchange protocol without random oracles. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS'11)*, pages 441–447, 2011.

[17] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553, 1999.

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

[19] F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, and E. Oswald. Leakage resilient cryptography in practice. In *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 99–134. Springer Berlin Heidelberg, 2010.

[20] M. Toorani. On continuous after-the-fact leakage-resilient key exchange. In *Proceedings of the 2nd Workshop on Cryptography and Security in Computing Systems (CS2'15)*. Amsterdam, Netherlands, January 2015.