Enhanced Modelling of Authenticated Key Exchange Security

Papa B. Seye and Augustin P. Sarr(✉)

Laboratoire ACCA, Université Gaston Berger de Saint-Louis, Saint Louis, Senegal
aug.sarr@gmail.com

Abstract. The security models for Authenticated Key Exchange do not consider leakages on pre-computed ephemeral data before their use in sessions. We investigate the consequences of such leakages and point out damaging consequences. As an illustration, we show the HMQV-C protocol vulnerable to a Bilateral Unknown Key Share (BUKS) and an Unilateral Unknown Key Share (UUKS) Attack, when precomputed ephemeral *public* keys are leaked. We point out some shades in the seCK model in multi-certification authorities setting. We propose an enhancement of the seCK model, which uses a liberal instantiation of the certification systems model from the ASICS framework, and allows reveal queries on precomputed ephemeral (public and private) keys. We propose a new protocol, termed eFHMQV, which in addition to provide the same efficiency as MQV, is particularly suited for implementations wherein a trusted device is used together with untrusted host machine. In such settings, the non-idle time computational effort of the device safely reduces to one digest computation, one integer multiplication, and one integer addition. The eFHMQV protocol meets our security definition, under the Random Oracle Model and the Gap Diffie-Hellman assumption.

Keywords: Unknown Key Share · seCK^{cs} · ASICS · HMQV-C · eFHMQV

1 Introduction

A large body of works on the modelling of Authenticated Key Exchange (AKE) security have been proposed since this approach was pioneered by Bellare and Rogaway [3]. The recent security models, CK [7], eCK [21], CK$_{\text{HMQV}}$ [17] and seCK [25,28] for instance, consider finely grained information leakages, including leakages on static and ephemeral private keys, session keys, and intermediate results. Working in another direction, Boyd *et al.* propose the ASICS framework [5] which provides a finely grained model of multi-certification systems and related attacks.

P.B. Seye—Supported by the CEA-MITIC of UGB.

In implementations of AKE protocols, ephemeral data are often pre-computed to boost implementations performance. The pre-computed data may then leak to an adversary. To take this into account, the recent models, such as CK [7], eCK [21], CK_{HMQV} [17] and seCK [25,28] among others, consider adversaries which may gain access to ephemeral secrets. Unfortunately, while leakages on precomputed ephemeral secrets may occur before their use in sessions, these models consider such leakages only while the keys are in use in a session (*i.e. after* the session owner knows his peer), *not before*.

The works [5,6] provide a generic framework termed ASICS, which considers not only leakages on the randomness used for ephemeral key generation, but also various attacks related to Certification Authorities (CAs) corruptions. Instantiations of the framework lead, depending on the allowed queries, to the eCK [21], the eCK^w [9], eCK-PFS [9], and to the CK_{HMQV} [17] models.

By considering an adversary which may learn the intermediate results in a session, the seCK model [25,28] aims at a better capture of information leakages. In this model, it is assumed at each party that a trusted computation area (a trusted platform module, a smart card, a hardware security module, etc.) is used together with an untrusted one (an untrusted host machine). It is assumed also that AKE implementations may differ from one party to another. Two implementations approaches are considered depending on the area wherein the ephemeral keys are computed. And, reveal queries are defined to allow an adversary to learn any information which is computed or used in the untrusted area.

Albeit the seCK model seems to provide a better capture of information leakages than the CK, eCK or ASICS models, the seCK definition considers only one honest CA and assumes that each party registers only one public key. The attacks that may occur in the multi-CA settings, wherein a party may have many certificates, and some of the CAs may be adversary controlled are not captured. Moreover, similar to the ASICS, eCK, and CK models, the seCK definition unnaturally omits leakages of ephemeral public and private keys, *before* their use in sessions. We investigate, in the multi-CA setting, the consequences of leakages on precomputed ephemeral keys. We show that even leakages on *ephemeral public keys* may have damaging consequences. As an illustration, we point out Unknown Key Share (UKS) attacks against the HMQV-C protocol [17], which was designed to provably provide explicit mutual key authentication. We propose an enhancement of the seCK model which uses a liberal instantiation of the ASICS certification systems model. Contrary to the previous models, the seCK^{cs} definition considers leakages on precomputed ephemeral public and private keys before their use in sessions, and captures various kind of UKS "related" attacks. We propose also an efficient protocol, termed eFHMQV, which is seCK^{cs}-secure under the Random Oracle model and the Gap Diffie-Hellman assumption.

This paper is organized as follows. In Sect. 2, we point out some limitations in the security models for AKE, we illustrate with UKS attacks against HMQV-C. In Sect. 3 we present the seCK^{cs} model. In Sect. 4 we propose the eFHMQV protocol.

We use the following notations. H is λ bits hash function, where λ is the security parameter, $\bar{H}$ is a $l = \lambda/2$ bits hash function. $\mathcal{G} = \langle G \rangle$ is a multiplicatively written group of prime order p, $\mathcal{G}^*$ is the set non-identity elements in $\mathcal{G}$. If n is an integer, $|n|$ denotes its bit-length and $[n]$ denotes the set $\{1, \cdots, n\}$. The symbol $\in_R$ stands for "chosen uniformly at random in". For two bit strings m_1 and $m_2, m_1||m_2$ denotes their concatenation. If $x_1, x_2, \cdots, x_k$ are objects belonging to different structures (group, bit-string, etc.) $(x_1, x_2, \cdots, x_k)$ denotes the concatenation of their representations as bit-strings.

2 Some Limitations in Existing Security Models

In this section we point out some limitations in the security models used for the analysis of Authenticated Key Exchange (AKE) protocols. We show that even leakages on pre-computed ephemeral *public* keys, may have damaging consequences. Such leakages are not considered in any of the security definitions for AKE we are aware of.

There are many arguments in favour of considering leakages on ephemeral keys (both public and private) *before* their use in sessions (*i.e.* before the peer in the session wherein the key is used is known). First, ephemeral keys pairs may be precomputed and stored in an untrusted memory; this matches, for instance, the implementation approach 1 in the seCK model [25,28] (see Fig. 1), and motivates the HMQV analysis in [17, Sect. 7]. Second, even in the seCK's implementation approach 2, wherein ephemeral keys are computed in a trusted area, there may be a limited storage space in a this area (a smart card, for instance). The ephemeral *public* keys may then be stored unencrypted[1] in the untrusted area, as when encrypted, the advantages of pre-computing may be (partially) lost, because of the time required for deciphering. It seems then realistic to consider leakages on precomputed ephemeral public keys before their use in sessions.

2.1 (Bilateral) Unknown Key Share Attacks

Key authentication is a fundamental AKE security attribute which guarantees that, besides a session owner, a session key is (possibly) known only by the peer. A key authentication is said to be *implicit* from a party $\hat{A}$ to another party $\hat{B}$, if when $\hat{B}$ completes a session with intended peer $\hat{A}$, then he has some assurance that $\hat{A}$ is the only other entity that can be in possession of the session key. *Explicit* key authentication from $\hat{A}$ to $\hat{B}$ is achieved if at the completion of the session at $\hat{B}$, he has some assurance that $\hat{A}$ is the only other entity in possession of the session key. A protocol is said to provide *mutual* key authentication (either explicit or implicit) when it provides key authentication both from $\hat{A}$ to $\hat{B}$ and from $\hat{B}$ to $\hat{A}$.

Unknown Key Share (UKS) attacks, also termed *identity misbinding* [16], seem to have been identified for the first time in [10]. Different formulations of

[1] However, digests of the public keys are stored in the tamper proof device, so that it is possible to verify that the keys were not altered.

an UKS attack can be found in the literature [4,15,16,23], although they convey essentially the same idea. The definition from [15], requires that an attacker, say $\hat{E}$, coerces two entities $\hat{A}$ and $\hat{B}$ into sharing a session key while *at least* one of them does not know that the session key is shared with the other; vulnerability to UKS attacks is then a failure in key authentication. A protocol is said to be vulnerable to an Unilateral UKS (UUKS), if an attacker can succeed in making two parties, say $\hat{A}$ and $\hat{B}$ share a session key, while *exactly* one of the parties, say $\hat{A}$ believes having shared the key with a party $\hat{C} \neq \hat{B}$. A protocol is said to be vulnerable to a BUKS attack if an attacker is able to make two entities, say $\hat{A}$ and $\hat{B}$, share a session key, while $\hat{A}$ believes having shared the key with some party $\hat{E}_1 \neq \hat{B}$ and $\hat{B}$ believes having shared the key with $\hat{E}_2 \neq \hat{A}$, the parties $\hat{E}_1$ and $\hat{E}_2$ may be different or not. BUKS attacks are then a specific case of UKS attacks (see [8] for a further discussion about UUKS and BUKS attacks).

Usually, in an (B, U)UKS attack, the attacker does not know the shared session key, he cannot then decipher or inject messages in the communications between the parties sharing the key. However, he may take advantage from the "unknown key share(s)", as shown in [4, Sect. 5.1.2] for UUKS attacks. For BUKS attacks, suppose that $\hat{A}$ is renowned chess player, $\hat{B}$ is a famous Artificial Intelligence (AI) creator, who claims having created an AI program that can win against $\hat{A}$, and the attacker $\hat{E}$ is an AI program creator who wants to take advantage from the reputations of $\hat{A}$ or $\hat{B}$. If the game parties between $\hat{A}$ and $\hat{B}$'s program are played online, using some AKE protocol Π which is vulnerable to a BUKS, $\hat{E}$ may claim having created an AI program that he expects to win against both $\hat{A}$ and the program from $\hat{B}$. Then $\hat{E}$ interferes in the session between $\hat{A}$ and $\hat{B}$ such that $\hat{A}$ (resp. $\hat{B}$) believes having shared the session key with $\hat{E}$, while it is shared with $\hat{B}$ (resp. $\hat{A}$). If $\hat{A}$ wins the game, $\hat{E}$ claims that his program won against the one from $\hat{B}$. Otherwise, he claims the converse. In any case, $\hat{E}$ takes advantage from the reputation of either $\hat{A}$ or $\hat{B}$. Such attacks may be damaging in any setting wherein the attacker can get some *credit* from a BUKS attack.

2.2 BUKS and UUKS Attacks Against HMQV-C

The HMQV protocol is a "hashed variant" of the MQV protocol [22], designed to provably overcome the "analytical shortcomings" in the MQV design [17,18]. In particular, HMQV is claimed to be provably resilient to UKS attacks. The three pass variant of HMQV, termed HMQV-C (the 'C' stands for key *confirmation*) is designed to provide, besides the HMQV security attributes, *explicit mutual key confirmation* and perfect forward secrecy. It is then a major design goal in HMQV-C that when a session key is shared between two honest parties, say $\hat{A}$ and $\hat{B}, \hat{A}$ (resp. $\hat{B}$) gets assurance that, besides himself, the session key is known only to $\hat{B}$ (resp. $\hat{A}$). Let $\hat{A}$ and $\hat{B}$ are two parties with respective static key pairs $(a, A = G^a)$ and $(b, B = G^b)$, with $A, B \in \mathcal{G}^*$. An execution of the HMQV-C protocol between them is as in Protocol 1; the execution aborts if any verification fails.

Protocol 1. The HMQV-C Protocol

(I) The initiator $\hat{A}$ does the following:
  (a) Choose $x \in_R [p-1]$ and compute $X = G^x$.
  (b) Send $(\hat{A}, \hat{B}, X)$ to $\hat{B}$.
(II) At receipt of $(\hat{A}, \hat{B}, X)$, $\hat{B}$ does the following:
  (a) Choose $y \in_R [p-1]$ and compute $Y = G^y$.
  (b) Compute $d = \bar{H}(X, \hat{B})$, $e = \bar{H}(Y, \hat{A})$, $s_B = y + eb \bmod p$, $\sigma_B = (XA^d)^{s_B}$, $K = H(\sigma_B, 1)$, and $K_m = H(\sigma_B, 0)$.
  (c) Send $\left(\hat{B}, \hat{A}, Y, \mathrm{MAC}_{K_m}(\text{“1”})\right)$ to $\hat{A}$.
(III) At receipt of $\left(\hat{B}, \hat{A}, Y, \mathrm{MAC}_{K_m}(\text{“1”})\right)$, $\hat{A}$ does the following:
  (a) Compute $d = \bar{H}(X, \hat{B})$, $e = \bar{H}(Y, \hat{A})$, $s_A = x + da \bmod p$, $\sigma_A = (YB^e)^{s_A}$, $K = H(\sigma_A, 1)$, and $K_m = H(\sigma_A, 0)$.
  (b) Validate $\mathrm{MAC}_{K_m}(\text{“1”})$.
  (c) Send $\left(\hat{A}, \hat{B}, X, \mathrm{MAC}_{K_m}(\text{“0”})\right)$ to $\hat{B}$.
(IV) At receipt of $\left(\hat{A}, \hat{B}, X, \mathrm{MAC}_{K_m}(\text{“0”})\right)$, $\hat{B}$ validates $\mathrm{MAC}_{K_m}(\text{“0”})$.
(V) The shared session key is K.

A BUKS Against HMQV-C. Suppose an attacker, with identity $\hat{E}$ (X509 Distinguished Name in [19]), which learns $\hat{A}$ and $\hat{B}$'s pre-computed ephemeral *public* keys X and Y, respectively, before their use. Proceeding as in Attack 2, $\hat{E}$ interferes such that $\hat{A}$ and $\hat{B}$ share a session key, while each of them believes having shared the key with $\hat{E}$.

Attack 2. BUKS Attack against HMQV-C

(1) Compute $d = \bar{H}(X, \hat{E})$, $X' = XA^dG$, $u = \bar{H}(X', \hat{B})$, and $E_1 = G^{-u^{-1} \bmod p}$.
(2) Register the key E_1 using the identity $\hat{E}$ to get a certificate crt_1.
(3) Compute $e = \bar{H}(Y, \hat{E})$, $Y' = YB^eG$, $v = \bar{H}(Y', \hat{A})$, and $E_2 = G^{-v^{-1} \bmod p}$.
(4) Register the key E_2 using the identity $\hat{E}$ to get a certificate crt_2.
(5) Induce $\hat{A}$ to initiate a session with peer $\hat{E}$ (using crt_2), and receive $(\hat{A}, \hat{E}, X)$ from $\hat{A}$.
(6) Initiate a session with peer $\hat{B}$ (using crt_1) by sending $(\hat{E}, \hat{B}, X')$.
(7) Receive $(\hat{B}, \hat{E}, Y, t_B = \mathrm{MAC}_{K_m}(\text{“1”}))$ from $\hat{B}$.
(8) Send $(\hat{E}, \hat{A}, Y', t_B)$ to $\hat{A}$.
(9) Receive $(\hat{A}, \hat{E}, X, t_A = \mathrm{MAC}_{K_m}(\text{“0”}))$ from $\hat{A}$.
(10) Send $(\hat{E}, \hat{B}, X', t_A)$ to $\hat{B}$.

As the attacker knows the static private keys corresponding to the keys he registers using his own identity, the registrations succeed even if a proof of knowledge of the private keys is required; he may register the keys at different CAs, in the case CAs do not register one identifier for many keys. Furthermore, the dual signature $\hat{A}$ derives is $\sigma_A = \mathrm{CDH}(XA^d, Y'E_2^v)$ wherein $d = \bar{H}(X, \hat{E})$ and $v = \bar{H}(Y', \hat{A})$. As $Y' = YB^eG$ where $e = \bar{H}(Y, \hat{E})$, and $E_2 = G^{-v^{-1}}$, we have $Y'E_2^v = YB^eG(G^{-v^{-1}})^v = YB^e$, and $\sigma_A = \mathrm{CDH}(XA^d, YB^e)$. Similarly, the session signature at $\hat{B}$ is $\sigma_B = \mathrm{CDH}(YB^e, X'E_1^u)$ where $u = \bar{H}(X', \hat{B})$. As $X' = XA^dG$, we have $X'E_1^u = XA^dG(G^{-u^{-1}})^u = XA^d$, and $\sigma_B = \mathrm{CDH}(YB^e, XA^d) = \sigma_A$. Then $\hat{A}$ and $\hat{B}$ derive the same session signature, the same session key $K = H(\sigma_A, 1) = H(\sigma_B, 1)$, and also the same MACing key $K_m = H(\sigma_A, 0) = H(\sigma_B, 0)$. Hence the MAC validations succeed in the sessions at $\hat{A}$ and $\hat{B}$, which both accept. As a consequence, $\hat{A}$ and $\hat{B}$ share the same session key ($K = H(\sigma_A, 1) = H(\sigma_B, 1)$) while each of them believes having shared the key with $\hat{E}$ (who is not in possession of the session key).

Applicability of the Attack Against other Protocols. Variants of our BUKS attack can be launched against the MQV [22], HMQV [17], SIG-DH [7], $\mathcal{P}$ [24], and DIKE [34] protocols; similar attacks are already known, from [8], against the four DHKE [29], the modified STS [4], and the alternative Oakley [4] protocols. In the HMQV instantiations under consideration for P1363 standardization (see the current P1363 draft at tinyurl.com/jolno5n), it is not mandated that the protocols be executed in the pre-specified-peer model (see [24] for a further discussion about the pre- and post-specified peer models). When these protocol are executed in the *post-specified-peer* model, *i.e.* when a session initiator discovers his peer's identity after he receives a message from him, variants of the attack can be launched *without any leakage assumption*. Without *further assumptions* the attack fails against the MQV-C and FHMQV protocols. In MQV-C, $\hat{B}$ provides to $\hat{A}$ a MAC of $(2, \hat{B}, \hat{A}, Y, X)$ and receives from him a MAC of $(3, \hat{A}, \hat{B}, X, Y)$, so when the attack is launched, although the MACing keys at $\hat{A}$ and $\hat{B}$ are the same, due to changes in the MACed data they expect, the validations fail.

An UUKS Attack Against HMQV-C. In [24], Menezes and Ustaoglu point out an UUKS against the *two-pass HMQV* in post-specified peer model. The attack can be launched if (*i*) a party can select its own identifier, and (*ii*) at key registration a proof of knowledge of the private key is *not* required. In a setting with 2^k honest parties, the attack requires roughly $2^{|p|/2-k}$ operations.

Assuming that the attacker may learn precomputed ephemeral *public* keys, we propose in Attack 3 an UUKS attack against HMQV-C. Our attack holds in the pre-specified peer model and seems to be more realistic than Menezes and Ustaoglu's attack. When Attack 3 is launched, $\hat{A}$ computes $\sigma_A = \mathrm{CDH}(XA^d, Y'E^v)$ where $d = \bar{H}(X, \hat{E})$ and $v = \bar{H}(Y', \hat{A})$. As $Y'E^v = YB^eG(G^{-v^{-1}})^v$, it follows that $\sigma_A = \mathrm{CDH}(XA^d, YB^e)$ where $e = \bar{H}(Y, \hat{A})$. The party $\hat{B}$, activated with peer $\hat{A}$, computes $\sigma_B = \mathrm{CDH}(YB^e, XA^d)$ wherein $d = \bar{H}(X, \hat{B}) = \bar{H}(X, \hat{E})$. Then $\hat{A}$ and $\hat{B}$ share the same session dual signature, making the MAC validations succeed in the sessions at both $\hat{A}$ and $\hat{B}$. So, $\hat{A}$ and $\hat{B}$ derive the same session key, while $\hat{A}$ believes having shared the key with $\hat{E}$, and $\hat{B}$ believes having shared the key with $\hat{A}$.

Similar to the attack from [24], in a setting with 2^k parties, our attack requires roughly $2^{|p|/2-k}$ operations (the computations at step 3). For $|p| = 160$ and $k = 20$, the attack requires 2^{60} operations and is not then out of reach of our computational capabilities [13,20]. Moreover, contrary to the Attack from [24], in our attack (*i*) the computations at step 3 are performed offline (after the attacker learns X), and (*ii*) the attacker knows the private key corresponding to the static key he registers. Our UUKS attack (against HMQV-C) is then more practical than the one from [24].

Attack 3. UUKS Attack against HMQV-C

```
(1)  Learn an ephemeral public key $X$ from a part, say $\hat{A}$.
(2)  Compute $\mathcal{D} = \left\{(C, \bar{H}(X, \hat{C})) : \hat{C} \text{ is an honest party}\right\}$.
(3)  Find an identifier $\hat{E}$ (which is different from honest parties identifiers) such that
     for some honest $\hat{B}$, $(\hat{B}, \bar{H}(X, \hat{E})) \in \mathcal{D}$.
(4)  Learn an ephemeral public key $Y$ at $\hat{B}$.
(5)  Compute $e = \bar{H}(Y, \hat{A})$, $Y' = YB^eG$, $v = \bar{H}(Y', \hat{A})$, and $E = G^{-v^{-1} \bmod p}$.
(6)  Register the key $E$ using the identifier $\hat{E}$.
(7)  Induce $\hat{A}$ to initiate a session with peer $\hat{E}$, and receive $(\hat{A}, \hat{E}, X)$ from $\hat{A}$.
(8)  Send $(\hat{A}, \hat{B}, X)$ to $\hat{B}$.
(9)  Intercept $\hat{B}$'s response $(\hat{B}, \hat{A}, Y, t_B = \text{MAC}_{K_m}(\text{"1"}))$.
(10) Send $(\hat{E}, \hat{A}, Y', t_B)$ to $\hat{A}$.
(11) Receive $(\hat{A}, \hat{E}, X, t_A = \text{MAC}_{K_m}(\text{"0"}))$ from $\hat{A}$.
(12) Send $(\hat{A}, \hat{B}, X, t_A)$ to $\hat{B}$.
```

2.3 About the Capture of UKS Related Attacks in Security Models

By UKS *related* attacks we refer to the attacks wherein the attacker succeeds in making non matching sessions yield unhashed secrets (session signatures) such that given one of the secrets, the other can be efficiently computed. Our attacks occur in the specific case wherein the unhashed secrets are the same.

Two weaknesses in the CK_{HMQV} model explain the co-existence of our attack and the HMQV(-C) security reduction. First, although the settings wherein ephemeral keys are pre-computed motivate the analysis in [17, Sect. 7], leakages on ephemeral keys are considered *only* while they are in use (*i.e.* after the peer in the session is known), *not* before. Then, the attacks assuming leakages on ephemeral public keys before their use are not captured. Moreover, when in addition to considering leakages on precomputed ephemeral keys, an attacker may learn some intermediate secrets (as modelled in the seCK definition [26,28]), further attacks can be launched. We stress that leakages on intermediate results is a realistic assumption. For instance, the AKE implementations in TPM2.0 are divided into two phases. In the first phase an outgoing ephemeral key is generated, using the command TPM2_EC_Ephemeral() (see [31, Sect. 19.3]). In the second phase (the relevant command is TPM2_ZGen_2Phase() [31, Sect. 14.7]) the TPM computes (using the peer's public keys) the unhashed shared secret (σ in the case of MQV). The session key is computed on the host machine (which may be infected by a malware), using the unhashed shared secret. When leakages on the unhashed shared secrets are considered, variants of our attacks can be launched against (H, C)MQV-C, even if nonces or the peers identities are included in the final digest for session key derivation.

We found no variant of our attacks against the FHMQV or SMQV protocols [25,28], as long as the CAs are honest and each party has only one certificate. However, in a multi-CA setting, where a party may have many certificates, some shades occur. We stress that considering a multi-CA setting, as modelled in the ASICS framework [5] wherein some of the CAs may be adversarially controlled, seems to be realistic. Indeed, for most browsers, only few clicks are required to add a rogue CA certificate in the trust-store (the set of CA certificates the user trusts), and it may also occur that users do not change their systems default trust-stores passwords.

For a party, say $\hat{A}$, with two certificates (with different keys), say crt_1 and crt_2, the disclosure of the private key corresponding to crt_1 should have no adverse effects in the sessions wherein $\hat{A}$ uses crt_2. And, when an attacker registers a certificate crt^* using $\hat{A}$'s identity and a static key which is different from the one corresponding to crt_2, the existence of crt^* should have no adverse effect on the sessions wherein $\hat{A}$ uses crt_2. Hence, the notion of "corruption" should be about certificates, not on parties. As a shade in the seCK model, in multi-CA settings, consider two parties $\hat{A}$ and $\hat{B}$, with respective certificates crt and crt', executing the (C, F)HMQV protocol (see [32] and [25,28] for descriptions of CMQV and FHMQV respectively), and an attacker which performs as in Attack 4.

Attack 4. Attack against (C, F)HMQV in a multi-CA setting

(a) Register $E = GA$ where A is $\hat{A}$'s static public key using $\hat{A}$'s identifier to obtain a certificate crt^*.
(b) When $\hat{A}$ initiates a session with peer $\hat{B}$ intercept his message $(\mathsf{crt}, \mathsf{crt}', X)$ and send $(\mathsf{crt}^*, \mathsf{crt}', X)$ to $\hat{B}$.
(c) Intercept $\hat{B}$'s response $(\mathsf{crt}', \mathsf{crt}^*, Y)$ and send $(\mathsf{crt}', \mathsf{crt}, Y)$ to $\hat{A}$.

The session signatures $\hat{A}$ and $\hat{B}$ derive are respectively $\sigma_A = \mathrm{CDH}(XA^d, YB^e)$ and $\sigma_B = \mathrm{CDH}(X(GA)^d, YB^e) = \sigma_A Y B^e$, where B is $\hat{B}$'s static key and d and e are the $\bar{H}$ digest values in (C, F)MQV. The sessions at $\hat{A}$ and $\hat{B}$ are non-matching and the session at $\hat{A}$ is seCK-fresh. When the attacker issues a session signature reveal query (to learn σ_B), he can compute the session key at $\hat{A}$ and succeed in a distinguishing game. An enhancement of the seCK security definition to clarify the shades and capture the consequences of leakages on precomputed ephemeral public keys is desirable. We propose such a model.

3 Enhancing the seCK Security Model

Broadly, in the seCK model [25,28], it is assumed two computation areas at each party, a trusted one (a smart card, a tamper proof device, etc.) and an untrusted one (a host machine), and that any information which is computed or used in the untrusted area can leak to an adversary. In addition, it is assumed that implementations may differ from one party to another; information leakages may then differ from one party to another. This seems to correspond to real word vulnerabilities [14,30,33]. Unfortunately, the seCK definition considers only one honest CA, and assumes that each party has only one honestly generated static key pair, and does not capture some attacks in a multi-CA setting.

In contrast, the ASICS framework considers a multi-CA setting, and captures a wide class of attacks based on adversarial key registration, including small subgroup attacks, UUKS attacks, and the attacks that may occur when a party can register many static keys. However, the ASICS model defines reveal queries only on static keys, randomness and session keys, leaving realistic leakages that may occur, through side-channel attacks for instance. As an example, in the CMQV variant, shown secure in [5,6], if an attacker learns a sufficiently large part of the ephemeral secret exponent at a part (s_A or s_B in Protocol 1), he can impersonate indefinitely the session owner to its peer [1,26].

We propose the seCK$^{\text{cs}}$ (the 'cs' stands for certification systems) to enhance the seCK model [25,28] in the following ways: (*i*) seCK$^{\text{cs}}$ provides a capture of the attacks exploiting leakages on pre-computed ephemeral public and private keys, (*ii*) it uses a liberal instantiation of the multi-CA model from [5], and (*iii*) captures various "kinds" of UKS related attacks.

3.1 The seCK$^{\text{cs}}$ Security Model

We suppose m parties $M_1, \cdots, M_m$, and an adversary $\mathcal{A}$, modelled as PPT Turing machines, sharing a securely generated set domain parameters, we denote by dp. The adversary is supposed to be in total control of the communication links between parties. We assume also n identities $\mathsf{id}_1, \cdots, \mathsf{id}_n$, with $m \leqslant n \leqslant R(\lambda)$ for some polynomial R. And, we require that different honest parties have distinct identities; we allow however a party to have many identities.

Key Generation and Certificate Registration. We assume a liberal certification authority (CA) which *accepts all the queries from the adversary*, including queries with the key and identity of an honest party. We only require that two certificates issued at distinct registrations be different, even if they have the same key and identity. In other words, we assume that each certificate has some specific information, we denote by Unique Identifier (ui), which is unique and efficiently computable. When various certificate formats are used, assuming that a CA does not issue two certificates with the same serial number, the ui can be, for instance, the quadruple (date of issuance, serial number, issuer, subject).

The adversary can direct a party, say M_i, to *generate a static key pair* trough $\mathsf{GenSKP}(M_i)$ query. This query can be issued many times at each party. When it is issued, M_i generates (using dp) a key pair (a, A) and provides $\mathcal{A}$ with A. Once A generated, $\mathcal{A}$ is allowed to direct M_i to *honestly register* A by issuing $\mathsf{HReg}(M_i, A, \mathsf{id}_k)$. When this query is issued, M_i registers A with the identity id_k to obtain a certificate. We stress that the HReg query is for *honest* key registration, so for the query to succeed, we require that no $\mathsf{HReg}(M_{i'}, A', \mathsf{id}_k)$

with $i' \neq i$ have been successfully issued before; *i.e.* that when different parties *honestly* register static keys, they use different identities.

The attacker can *maliciously* register *any* (valid or invalid) key, including honest parties static keys, together with any string of its choice (including a honest party's identity) using the $\mathsf{MReg}(Q, \mathsf{id})$ query; this query *always* succeeds. For a certificate crt, we refer to the certificate's public key, identity, and ui respectively by $\mathsf{crt.pk}, \mathsf{crt.id}$, and $\mathsf{crt.ui}$.

Sessions. A session is an instance of a protocol run at a party; $\mathcal{A}$ decides about session activations. To activate a session, say at M_i with peer $M_{i'}$, $\mathcal{A}$ issues a Create query with parameters $(\mathsf{crt}, \mathsf{crt}')$ or $(\mathsf{crt}, \mathsf{crt}', m)$, where m is a message supposed to be from $M_{i'}$, and crt and crt' are certificates belonging to M_i and $M_{i'}$ respectively. If the creation parameter is $(\mathsf{crt}, \mathsf{crt}')$, M_i is said to be the initiator ($\mathcal{I}$), otherwise he is said to be the responder ($\mathcal{R}$). At session creation, the activated party may provide $\mathcal{A}$ with an outgoing message (sid', m') where sid' is a session identifier and m' is a message to be processed in sid'. Each session is identified with a tuple $(\mathsf{crt}, \mathsf{crt}', \mathsf{out}, \mathsf{in}, \mathsf{role})$, where crt is the owner's certificate, crt' is the peer's certificate (in the owner's view), out is the list of the outgoing messages, in is the list of the incoming messages, and $\mathsf{role} \in \{\mathcal{I}, \mathcal{R}\}$ is the owner's role. For an identifier $\mathsf{sid} = (\mathsf{crt}, \mathsf{crt}'\ \mathsf{out}, \mathsf{in}, \mathsf{role})$, we refer respectively to $\mathsf{crt}, \mathsf{crt}', \mathsf{out}, \mathsf{in},$ and role by $\mathsf{sid}_{\mathsf{oc}}, \mathsf{sid}_{\mathsf{pc}}, \mathsf{sid}_{\mathsf{in}}, \mathsf{sid}_{\mathsf{out}}$, and $\mathsf{sid}_{\mathsf{role}}$. For the two pass Diffie-Hellman protocols, we refer to the incoming and outgoing ephemeral keys by $\mathsf{sid}_{\mathsf{iEPK}}$ and $\mathsf{sid}_{\mathsf{oEPK}}$ respectively. Each session has a status we denote by $\mathsf{sid}_{\mathsf{status}} \in \{\mathsf{active}, \mathsf{accepted}, \mathsf{rejected}\}$. The status is $\mathsf{accepted}$ if the session has completed, *i.e.* the session key is computed and accepted. It is $\mathsf{rejected}$ if the session has aborted, it is active if it is neither $\mathsf{accepted}$ nor $\mathsf{rejected}$. For an accepted session $\mathsf{sid}, \mathsf{sid}_{\mathsf{key}}$ denotes the derived key. The adversary can issue a $\mathsf{Sd}(\mathsf{sid}, m)$ query, where m is a message to be processed in sid. When this query is issued, the session owner is provided with m. He may update $\mathsf{sid}_{\mathsf{in}}$ to include m; he may also compute an outgoing message (sid', m') and update $\mathsf{sid}_{\mathsf{out}}$ and $\mathsf{sid}_{\mathsf{status}}$ accordingly. Two sessions sid and sid' are said to be *matching* if $\mathsf{sid}_{\mathsf{oc}} = \mathsf{sid}'_{\mathsf{pc}}, \mathsf{sid}_{\mathsf{pc}} = \mathsf{sid}'_{\mathsf{oc}}, \mathsf{sid}_{\mathsf{out}} = \mathsf{sid}'_{\mathsf{in}}, \mathsf{sid}_{\mathsf{in}} = \mathsf{sid}'_{\mathsf{out}}$, and $\mathsf{sid}_{\mathsf{role}} \neq \mathsf{sid}'_{\mathsf{role}}$.

Reveal Queries. Similar to the seCK model [25,28], we assume two computation areas at each party, a trusted and an untrusted one. We suppose that implementations may be performed differently from one party to another, and define reveal queries to allow the adversary to learn any information that is computed or used in the untrusted area. Moreover, the adversary may bypass the tamper protection mechanisms and learn the long term secrets. We assume implementations performed using one of the seCK approaches. In Approach 1, the static key is computed and used in the trusted area, and the ephemeral keys are computed in the untrusted area. This implementation approach corresponds to reveal queries as defined in the eCK and ASICS models. In Approach 2, both static and ephemeral private keys are computed and used in the trusted area, and all the other intermediate results are used in the untrusted host-machine. This approach is similar but stronger than the way AKE implementations are performed in TPM2.0. In both approaches, the session key is used in the untrusted area.

These approaches are not the only possible, and the model can be enriched with other implementation approaches, however the two approaches we consider seem to be typical in real word settings.

The adversary is allowed to direct a certificate owner, say M_i, to generate an ephemeral public key pair using a $\mathsf{GenEKP}(\mathsf{crt})$ query. When it is issued, M_i generates a key pair (x, X) and provides the attacker with X. If M_i, follows the Approach 1, $\mathcal{A}$ can issue a $\mathsf{RvEPK}(X)$ query to learn the ephemeral private key x. We stress that this query may be issued before the public key X is used in a session. At a party using Approach 2, a reveal query is defined to allow $\mathcal{A}$ to learn *any* information that is computed of used in the untrusted area. In both approaches, the adversary can learn the private key corresponding to a static public key A, by issuing $\mathsf{RvSPK}(A)$. For a completed session sid, the attacker can issue a $\mathsf{RvSesK}(\mathsf{sid})$ query to learn $\mathsf{sid}_{\mathsf{key}}$. For the protocols of the MQV family, at a party using the Approach 2, $\mathcal{A}$ can issue $\mathsf{RvSecExp}(\mathsf{sid})$ to obtain the ephemeral secret exponent in sid (s_A or s_B in HMQV-C), and a $\mathsf{RvSesSig}(\mathsf{sid})$ query to obtain the dual signature (σ_A or σ_B) (Tables 1 and 2).

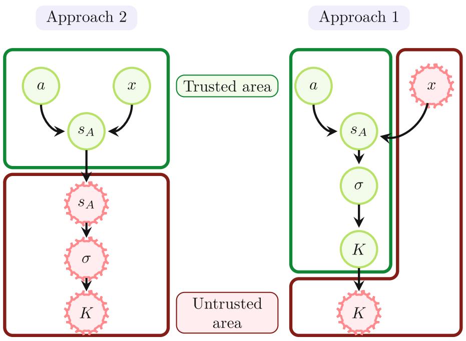

Fig. 1. (e)FHMQV implementation approaches in the seCK model [25,28]

Table 1. Summary of the queries

GenSKP, RvSPK	static key pair generation, static private key reveal query
HReg, MReg	*honest* key registration, *malicious* key registration
GenEKP, RvEPK	ephemeral key pair generation, ephemeral private key reveal query
Create, Sd	Session creation, message sending
RvSesK	session key reveal query
RvSecExp	ephemeral secret exponent reveal query (for the MQV family)
RvSesSig	session signature reveal query
Test	test session query

Table 2. Overview of the notations

dp	public domain parameters
$\mathsf{crt}, \mathsf{crt}_{x, x\in\{\mathsf{pk},\mathsf{id},\mathsf{ui}\}}$	a certificate, the public key (pk), identity (id), or unique identifier (ui) in the certificate crt
$\mathsf{sid}, \mathsf{sid}_{x, x\in\{\mathsf{oc},\mathsf{pc},\mathsf{out},\mathsf{in},\mathsf{role}\}}$	session identifier, the owner's certificate (oc), peer's certificate (pc), list of outgoing messages (out), list of incoming messages (in), or the owner's role in the session sid
$\mathsf{sid}_{x, x\in\{\mathsf{iEPK},\mathsf{oEPK}\}}$	incoming ephemeral public key (iEPK) or outgoing ephemeral public key (oEPK) in a session (for DH protocols)

Session Freshness. A completed session with identifier sid is said to be:

Locally exposed: if (a) $\mathcal{A}$ issued a $\mathsf{RvSesK}(\mathsf{sid})$ query, or (b) the session owner follows the Approach 1 and $\mathcal{A}$ issued both $\mathsf{RvSPK}(\mathsf{sid}_{\mathsf{oc}}.\mathsf{pk})$ and $\mathsf{RvEPK}(\mathsf{sid}_{\mathsf{oEPK}})$, or (c) the session owner follows the Approach 2 and $\mathcal{A}$ issued a reveal query on an intermediate result which is computed or used in the untrusted area.

Remark 1. For the protocols of the MQV family, the condition (c) is "the session owner follows the Approach 2 and $\mathcal{A}$ issued $\mathsf{RvSecExp}(\mathsf{sid})$ or $\mathsf{RvSesSig}(\mathsf{sid})$."

Exposed: if (a) it is locally exposed, or (b) its matching session exists and is locally exposed, or (c) its matching session does no exist and $(c.i)$ $\mathsf{sid}_{\mathsf{pc}}$ was maliciously registered, or $(c.ii)$ $\mathsf{sid}_{\mathsf{pc}}$ was honestly registered and $\mathcal{A}$ issued $\mathsf{RvSPK}(\mathsf{sid}_{\mathsf{pc}}.\mathsf{pk})$;

Fresh: if it is not exposed.

The security experiment is initialized with a securely generated public set of domain parameters dp for some security parameter λ. The adversary is allowed to issue all the queries defined above. At some point of the game he issues a $\mathsf{Test}(\mathsf{sid})$ query on a completed and fresh session sid. When the Test query is issued a bit $b \in_R \{0,1\}$ is chosen, and $\mathcal{A}$ is provided with $k = \begin{cases} \mathsf{sid}_{\mathsf{key}} \text{ if } b = 1 \\ k' \in_R \{0,1\}^{\lambda}, \text{otherwise.} \end{cases}$ After the Test query is issued, $\mathcal{A}$ can issue all the queries of its choice as long as sid remains fresh. Finally, he produces a bit b' and wins the game if $b = b'$.

Definition 1 (seCK$^{\text{cs}}$ security). *A protocol* Π *is said to be seCK^{cs} secure if,*

- *except with negligible probability, two sessions yield the same session key* if and only if *they are matching, and*
- *for all efficient attacker playing the above game,* $|2\Pr(b = b') - 1|$ *is negligible.*

3.2 Comparing the seCK$^{\text{cs}}$ with the seCK and ASICS Models

The seCK$^{\text{cs}}$ definition encompasses the seCK model [25,28] together with a liberal instantiation of the ASICS multi-CA setting [5,6]. The modelling of the

CAs is realistic, as illustrated with recent CA breaches [11,12]. And, as already pointed out in [5, p. 6], although we explicitly consider one CA, we implicitly capture multi-CA settings with independent CAs.

However, there are some differences between the key registration queries in the ASICS and seCK^{cs} models. The honest key registration query in the ASICS model, $\mathsf{hregister}$, takes two parameters, a public key and an identity. The parties and their implementation approaches are modelled in seCK^{cs}, so the honest key registration, HReg, is enriched to include a parameter which indicates the party registering the key. Also, we do not differentiate *malicious* key registrations depending on the validity of the static key the adversary provides, as with the $\mathsf{pkregister}$ and $\mathsf{npkregister}$ in ASICS. We assume simply that any malicious registration query succeeds (*i.e.* the MReg query always succeeds). Moreover, there are less restrictions in the seCK^{cs} freshness definition than in the ASICS instantiations from [5, Sects. 3–4]. For a session sid without a matching session, both definitions require that no $\mathsf{RvSPK}(\mathsf{sid}_{\mathsf{pc}}.\mathsf{pk})$ was successfully issued. However, while [5,6, Theorem 1] requires that $\mathsf{MReg}(\mathsf{sid}_{\mathsf{pc}}.\mathsf{pk}, \mathsf{sid}_{\mathsf{pc}}.\mathsf{id})$ was not issued, we require that $\mathsf{sid}_{\mathsf{pc}}$ was not registered by $\mathcal{A}$, meaning that sid remains fresh even if $\mathcal{A}$ issued $\mathsf{MReg}(\mathsf{sid}_{\mathsf{pc}}.\mathsf{pk}, \mathsf{sid}_{\mathsf{pc}}.\mathsf{id})$, as long as $\mathsf{sid}_{\mathsf{pc}}$ was not registered by $\mathcal{A}$. Besides, the ASICS model considers only leakages on static keys, randomness and session keys, leaving realistic leakages that may occur, on unhashed shared secrets (in AKE implementations in TPM2.0 for instance); while seCK^{cs} considers reveal queries on precomputed ephemeral keys and any information which is computed or used in the untrusted area.

The seCK^{cs} definition is strictly stronger than seCK, which is already known to be strictly stronger than the eCK model [28]. To illustrate the separation between the seCK^{cs} and seCK models, we consider the Attack 4 against (C, F)HMQV, wherein $\hat{B}$ belong to the set of parties following the second implementation approach. We recall that FHMQV and CMQV are known respectively to be secure in the seCK and ASICS models. In Attack 4, the session at $\hat{A}$ is seCK^{cs}-fresh. Given the relation between the session signatures in the sessions at $\hat{A}$ and $\hat{B}$, $\mathcal{A}$ succeeds in the seCK^{cs} distinguishing game, with probability ≈ 1, as follows: (*i*) he chooses the session at $\hat{A}$ as a test session, (*ii*) issues a $\mathsf{RvSesSig}$ on the session at $\hat{B}$ to obtain σ_B, and (*iii*) compute the session signature and the session key $\hat{A}$ derives. The attacker's success follows from its ability to make non-matching sessions yield related session signatures, such that given one of the session signatures, the other can be efficiently computed. By requiring that non-matching sessions do not yield the same session key, seCK^{cs}-security captures classical (B, U)UKS attacks. Moreover, it ensures that non-matching session do not yield related session signatures. The seCK^{cs} model captures not only "classical" UKS attacks, but also the attacks related to unknown share of unhashed session secrets.

4 The enhanced FHMQV (eFHMQV) Protocol

A main improvement in FHMQV [25,26] compared to HMQV [17] is the use of the incoming and outgoing ephemeral keys in the computation of the digest values d and e; this design choice makes FHMQV resilient to leakages on ephemeral secret exponents (s_A and s_B). We use a similar idea in the eFHMQV design. An execution of eFHMQV between two parties $\hat{A}$ and $\hat{B}$ with respective certificates crt and crt' is as in Protocol 5.

Protocol 5. The eFHMQV Protocol

(I) The initiator $\hat{A}$ does the following:
  (a) Verify that $\mathsf{crt}'.\mathsf{pk} \in \mathcal{G}^*$, choose $x \in_R [p-1]$ and compute $X = G^x$.
  (b) Send $(\mathsf{crt}, \mathsf{crt}', X)$ to $\hat{B}$.
(II) At receipt of $(\mathsf{crt}, \mathsf{crt}', X)$, $\hat{B}$ does the following:
  (a) Verify that $X \in \mathcal{G}^*$ and $\mathsf{crt}.\mathsf{pk} \in \mathcal{G}^*$, choose $y \in_R [p-1]$ and compute $Y = G^y$.
  (b) Send $(\mathsf{crt}', \mathsf{crt}, X, Y)$ to $\hat{A}$.
  (c) Compute $d = \bar{H}(X, Y, \mathsf{crt}.\mathsf{pk}, \mathsf{crt}.\mathsf{id}, \mathsf{crt}.\mathsf{ui}, \mathsf{crt}'.\mathsf{pk}, \mathsf{crt}'.\mathsf{id}, \mathsf{crt}'.\mathsf{ui})$.
  (d) Compute $e = \bar{H}(Y, X, \mathsf{crt}.\mathsf{pk}, \mathsf{crt}.\mathsf{id}, \mathsf{crt}.\mathsf{ui}, \mathsf{crt}'.\mathsf{pk}, \mathsf{crt}'.\mathsf{id}, \mathsf{crt}'.\mathsf{ui})$.
  (e) Compute $s_B = y + eb$, where $b = \log_G \mathsf{crt}'.\mathsf{pk}$, and $\sigma_B = (X(\mathsf{crt}.\mathsf{pk})^d)^{s_B}$.
  (f) Compute $K = H(\sigma_B, \mathsf{crt}.\mathsf{pk}, \mathsf{crt}.\mathsf{id}, \mathsf{crt}.\mathsf{ui}, \mathsf{crt}'.\mathsf{pk}, \mathsf{crt}'.\mathsf{id}, \mathsf{crt}'.\mathsf{ui}, X, Y)$.
(III) At receipt of $(\mathsf{crt}', \mathsf{crt}, X, Y)$, $\hat{A}$ does the following:
  (a) Verify that $Y \in \mathcal{G}^*$.
  (b) Compute $d = \bar{H}(X, Y, \mathsf{crt}.\mathsf{pk}, \mathsf{crt}.\mathsf{id}, \mathsf{crt}.\mathsf{ui}, \mathsf{crt}'.\mathsf{pk}, \mathsf{crt}'.\mathsf{id}, \mathsf{crt}'.\mathsf{ui})$.
  (c) Compute $e = \bar{H}(Y, X, \mathsf{crt}.\mathsf{pk}, \mathsf{crt}.\mathsf{id}, \mathsf{crt}.\mathsf{ui}, \mathsf{crt}'.\mathsf{pk}, \mathsf{crt}'.\mathsf{id}, \mathsf{crt}'.\mathsf{ui})$.
  (d) Compute $s_A = x + da$, where $a = \log_G \mathsf{crt}.\mathsf{pk}$, and $\sigma_A = (Y(\mathsf{crt}'.\mathsf{pk})^e)^{s_A}$.
  (e) Compute $K = H(\sigma_A, \mathsf{crt}.\mathsf{pk}, \mathsf{crt}.\mathsf{id}, \mathsf{crt}.\mathsf{ui}, \mathsf{crt}'.\mathsf{pk}, \mathsf{crt}'.\mathsf{id}, \mathsf{crt}'.\mathsf{ui}, X, Y)$.
(IV) The shared session key is K.

In an eFHMQV session with identifier $\mathsf{sid} = (\mathsf{crt}, \mathsf{crt}', X, Y, \mathcal{I})$ the digests d and e are computed as indicated in the steps (IIIb) and (IIIc). As a result, even if the step (a) of Attack 4 is modified to make $\mathcal{A}$ issues $\mathsf{MReg}(\mathsf{crt}.\mathsf{pk}, \mathsf{crt}.\mathsf{id})$, *i.e. $\mathcal{A}$ registers $\hat{A}$'s key using $\hat{A}$'s identity* to obtain crt^*, the attack fails as long as different certificates have different unique identifiers. Indeed, as $\hat{B}$ computes $d' = \bar{H}(X, Y, \mathsf{crt}^*.\mathsf{pk}, \mathsf{crt}^*.\mathsf{id}, \mathsf{crt}^*.\mathsf{ui}, \mathsf{crt}'.\mathsf{pk}, \mathsf{crt}'.\mathsf{id}, \mathsf{crt}'.\mathsf{ui})$ and $e' = \bar{H}(Y, X, \mathsf{crt}^*.\mathsf{pk}, \mathsf{crt}^*.\mathsf{id}, \mathsf{crt}^*.\mathsf{ui}, \mathsf{crt}'.\mathsf{pk}, \mathsf{crt}'.\mathsf{id}, \mathsf{crt}'.\mathsf{ui})$ and $\mathsf{crt}^*.\mathsf{ui} \neq \mathsf{crt}.\mathsf{ui}$, except with negligible probability $d' \neq d$ and $e' \neq e$. Then, even if $\mathcal{A}$ issues $\mathsf{RvSecExp}(\mathsf{crt}', \mathsf{crt}_{\mathcal{A}}, Y, X, \mathcal{R})$ in the distinguishing game and receives $s_B = y + e'b$, as $e' \neq e$, he cannot derive $\sigma_A = \mathrm{CDH}(XA^d, YB^e)$ wherein $A = \mathsf{crt}.\mathsf{pk}$, $B = \mathsf{crt}'.\mathsf{pk}$. A direct proof of this claim can be obtained using the Knowledge of Exponent Assumption [2]. However, as we show in Theorem 1, this assumption is not necessary.

An execution of eFHMQV requires at most 2.5 times a single exponentiation; this equals the efficiency of the famous MQV protocol. In addition, in Approach 2, the ephemeral public keys can be computed in idle time on a trusted device (a smart card for instance) and stored *unencrypted* in an untrusted host machine. It is only necessary that a digest of the keys be stored on the device so that alterations can be detected. When eFHMQV is implemented in this way, the non-idle time computational effort on the device reduces to one digest computation, one integer addition, and one integer multiplication. We stress that the

(C,H)MQV protocols [17,22,32] cannot achieve such a performance, as they do not confine the adverse effects of leakages on secrets exponents (s_A and s_B). And, in the seCK$^{\text{cs}}$ security definition, FHMQV and SMQV [26–28] are insecure, and cannot then provably achieve such a performance.

Theorem 1. *Under the Gap Diffie-Hellman assumption and the Random Oracle model, the eFHMQV protocol is seCK^{cs}-secure.*

The FXCR and FDCR schemes [25,28] are the main ingredients in the proof of this theorem we do not provide here (for lack of space). A detailed proof is given in the extended version of this paper.

5 Concluding Remarks

We pointed out and illustrated some limitations in existing AKE security models. We showed that even leakages on precomputed ephemeral *public* keys may have damaging consequences, we illustrated with (B, U)UKS attacks against the HMQV-C protocol. We proposed the seCK$^{\text{cs}}$ security definition which encompasses the seCK model, integrates a strong model of multi-CA settings, and considers leakages on precomputed ephemeral (public and private) keys.

We proposed the eFHMQV protocol, which is particularly suited for distributed implementation environments wherein an untrusted computer is used together with a tamper-resistant device. In such an environment, the non-idle time computational effort of the device reduces to one digest computation, one integer addition, and one integer multiplication. The eFHMQV protocol is seCK$^{\text{cs}}$-secure under the Random Oracle Model and the Gap Diffie-Hellman assumption.

In a forthcoming stage, we will be interested in Perfect Forward Secrecy in the seCK$^{\text{cs}}$ model.

References

1. Basin, D., Cremers, C.: Modeling and analyzing security in the presence of compromising adversaries. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 340–356. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15497-3_21
2. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28628-8_17
3. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994). doi:10.1007/3-540-48329-2_21
4. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer, Heidelberg (2003). doi:10.1007/978-3-662-09527-0
5. Boyd, C., Cremers, C., Feltz, M., Paterson, K.G., Poettering, B., Stebila, D.: ASICS: authenticated key exchange security incorporating certification systems. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 381–399. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40203-6_22

6. Boyd, C., Cremers, C., Feltz, M., Paterson, K.G., Poettering, B., Stebila, D.: ASICS: Authenticated key exchange security incorporating certification systems. Cryptology ePrint Archive: Report 2013/398
7. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001). doi:10.1007/3-540-44987-6_28
8. Chen, L., Tang, Q.: Bilateral unknown key-share attacks in key agreement protocols. J. Univ. Comput. Sci. **14**(3), 416–440 (2008)
9. Cremers, C., Feltz, M.: Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal. Des. Codes Crypt. **74**(1), 183–218 (2013). Springer
10. Diffie, W., Van Orschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Des. Codes Crypt. **2**(2), 107–125 (1992). Springer
11. Ducklin, P.: Serious security: Google finds fake but trusted SSL certificates for its domains, made in France. http://tinyurl.com/hrmo8pa
12. FOX IT: Black Tulip: report of the investigation into the DigiNotar Certificate Authority breach. http://preview.tinyurl.com/lj6938c
13. Güneysu, T., Pfeiffer, G., Paar, C., Schimmler, M.: Three years of evolution: cryptanalysis with COPACOBANA. In: Workshop Record of "Special-Purpose Hardware for Attacking Cryptographic Systems"–SHARCS 2009 (2009)
14. Huq, N.: PoS RAM Scraper Malware: Past, Present, and Future. A Trend Micro Research Paper (2014). http://tinyurl.com/jcwc8wz
15. Kaliski, B.S.: An unknown key-share attack on the MQV key agreement protocol. ACM Trans. Inf. Syst. Secur. (TISSEC) **4**(3), 275–288 (2001). ACM
16. Krawczyk, H.: SIGMA: the 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45146-4_24
17. Krawczyk, H.: HMQV: a hight performance secure Diffie-Hellman protocol. Cryptology ePrint Archive, Report 2005/176 (2005)
18. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005). doi:10.1007/11535218_33
19. Krawczyk, H.: HMQV in IEEE P1363. Submission to the IEEE P1363 working group. http://tinyurl.com/opjqknd
20. Kumar, S., Paar, C., Pelzl, J., Pfeiffer, G., Rupp, A., Schimmler, M.: How to break DES for € 8,980. In: International Workshop on Special-Purpose Hardware for Attacking Cryptographic Systems – SHARCS'06, Cologne, Germany, April 2006
21. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75670-5_1
22. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Des. Codes Crypt. **28**, 119–134 (2003). Springer
23. Menezes, A., Van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
24. Menezes, A., Ustaoglu, B.: Comparing the pre- and post-specified peer models for key agreement. Int. J. Appl. Crypt. **1**(3), 236–250 (2009). Inderscience
25. Sarr, A.P., Elbaz–Vincent, P.: On the security of the (F)HMQV protocol. In: Pointcheval, D., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 2016. LNCS, vol. 9646, pp. 207–224. Springer, Cham (2016). doi:10.1007/978-3-319-31517-1_11

26. Sarr, A.P., Elbaz-Vincent, P., Bajard, J.-C.: A secure and efficient authenticated Diffie–Hellman protocol. In: Martinelli, F., Preneel, B. (eds.) EuroPKI 2009. LNCS, vol. 6391, pp. 83–98. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16441-5_6
27. Sarr, A.P., Elbaz-Vincent, P., Bajard, J.C.: A secure and efficient authenticated Diffie-Hellman protocol. Cryptology ePrint Archive: Report 2009/408
28. Sarr, A.P., Elbaz-Vincent, P., Bajard, J.-C.: A new security model for authenticated key agreement. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 219–234. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15317-4_15
29. Shoup, V.: On formal models for secure key exchange. Cryptology ePrint Archive, 1999/012 (1999)
30. Trend Labs Security Intelligence Blog: RawPOS Technical Brief. http://tinyurl.com/joyazja
31. TCG: Trusted Platform Module Library Part 3: Commands, Level 00 Revision 01.38 (2016)
32. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Des. Codes Crypt. **46**(3), 329–342 (2008)
33. VISA Data Security Alert: Retail Merchants Targeted by Memory-Parsing Malware 2013. http://tinyurl.com/j3duvlg
34. Yao, A.C., Zhao, Y.: Deniable internet key exchange. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 329–348. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13708-2_20