On the Security of the (F)HMQV Protocol

Augustin Sarr, Philippe Elbaz-Vincent

On the Security of the (F)HMQV Protocol

Augustin P. Sarr[1⋆] and Philippe Elbaz–Vincent[2⋆⋆]

[1] Laboratoire ACCA, Université Gaston Berger de Saint–Louis
[2] Institut Fourier – CNRS, Université Grenoble Alpes

Abstract. The HMQV protocol is under consideration for IEEE P1363 standardization. We provide a complementary analysis of the HMQV protocol. Namely, we point a Key Compromise Impersonation (KCI) attack showing that the two and three pass HMQV protocols cannot achieve their security goals. Next, we revisit the FHMQV building blocks, design and security arguments; we clarify the security and efficiency separation between HMQV and FHMQV, showing the advantages of FHMQV over HMQV.

Keywords: Authenticated Key Exchange, FHMQV, HMQV, KCI Attack, Security Model.

1 Introduction

Designing authenticated key agreement protocols is a notoriously subtle task. Bellare and Rogaway proposed a new approach for the analysis of key agreement protocols [3], which was later refined in many security models, including, and among others the CK [6], eCK [20] and seCK [29] models.

The HMQV protocol [16], inspired by the famous MQV [21] protocol, was shown secure in a variant of the CK model, termed here $\mathrm{CK}_{\mathrm{HMQV}}$. HMQV was designed to resist a variety of attacks and was shown to provably achieve its security attributes. Among others, Krawczyk was able to show that HMQV remains secure even if public keys are not tested to be of correct order ($\mathcal{G}$–tests). As the computational cost of these tests may be significant, avoiding them may induce a significant efficiency improvement. With this efficiency improvement, HMQV was proposed for standardization in P1363 [18]. The HMQV P1363 submission states that the tests to ensure ephemeral keys to be of correct order "are required only in settings where ephemeral exponents are more vulnerable to attack than long–term secrets. In all other cases, i.e., where ephemeral and long–term secrets are equally protected, HMQV can safely skip these tests, thus providing superior performance especially when the cofactor is large" [18, p. 2].

In [23,24], some attacks against HMQV are proposed to recover the victim's static private key; the attacks can be launched in the case the static and ephemeral keys are not tested to be of correct order. Even if the attack against

⋆ Partially supported by the CEA–MITIC
⋆⋆ Partially supported by the LabEx PERSYVAL–Lab (ANR–11–LABX–0025–01)

the one–pass HMQV is realistic, the attacks proposed against the two and three pass variants seem less realistic, as the attacker needs to learn some ephemeral private keys from the target victim. The work [25] delves further in the effects of omitting public key validation in HMQV, and some new attacks are presented in the cases static public keys only or ephemeral public keys only are tested to be of correct order; however the attacks are proposed in groups which are not used in practice. In [7], Chalkias, *et al.* explore KCI against the One Pass (H)MQV protocols and show that these protocols are vulnerable to KCI attacks. In [27,28] Sarr *et al.* explore the consequences of secret exponent leakage in a HMQV session. They show that (partial) leakage on ephemeral secret exponents lead to impersonation and man–in–the–middle attacks. Basing on theses findings they propose the FHMQV protocol they show to confine the effects of such leakages.

In this paper, we investigate the effects of omitting ephemeral key validation in the HMQV protocol. We show that the (two and three pass) HMQV protocol(s) are vulnerable to KCI, unless further restrictions are considered in the underlying group. Namely, in the case the group keys are supposed to belong is a subgroup of a DSA group $GF(q)$, with $(q-1)$ divisible by a sufficiently "large" integer, without $\mathcal{G}$–tests, HMQV is vulnerable to a KCI; our attack invalidates the HMQV resistance to KCI, stated in [16, Theorem 18 and Lemma 21]. A main feature of the KCI attack we present is that it requires the entity to be impersonated to omit ephemeral key validation *only once*.

Besides, we re–examine the FHMQV building blocks, showing that contrary to what is suggested in [22] changing the interaction order has no effect on the building blocks security. We clarify also the separation between FHMQV and HMQV, showing the security and efficiency improvements in FHMQV.

This paper is organized as follows. In §2, we revisit the HMQV protocol, pointing a KCI attack. In §3 we revisit the FXCR scheme, showing that its security is not dependent to interaction order. The FDCR scheme is revised in §4. In §5 we clarify the separation between FHMQV and HMQV.

The following notations are used in this paper H is λ bit hash function, where λ is the security parameter; $\bar{H}$ is a $l = \lambda/2$ bits hash function. $\mathcal{G}$ is a multiplicatively written group, $\mathcal{G}^*$ is the set of non–identity elements in $\mathcal{G}$. If n is an integer, $|n|$ denotes its bit–length; we refer to the length of a list $\mathcal{L}$ by $|\mathcal{L}|$. 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 the concatenation of m_1 and m_2. 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 a representation such that each element can be univocally parsed.

2 Key Compromise Impersonation against HMQV

A protocol is said to be vulnerable to KCI impersonation, if an attacker who learns the long term secret of a party, say $\hat{A}$, is able to impersonate another party, say $\hat{B}$, to $\hat{A}$. When a protocol is vulnerable to such attacks, a static key leakage may lead to harms that go far beyond the sole ability to impersonate the static key's owner. For instance, in the case $\hat{A}$ is a bank client and $\hat{B}$ a bank server,

the attacker may impersonate the server to the client to collect more sensitive information (such as a credit card number or a security code, for instance). As another example, $\hat{B}$ may be a trusted software update server, in this case, the attacker may impersonate the server to $\hat{A}$ to make him/her install a malicious software (spyware, worm, virus, etc.), and gain much more sensitive information, such as passwords, credit card numbers, etc. KCI resilience is then an important security attribute, particularly for protocols intended to be standardized, such as HMQV.

In this section, we present a KCI against the three–pass HMQV protocol. About prime–order tests, we show that without these tests HMQV is vulnerable to KCI, unless further restrictions are specified about the underlying group. This shows also that the HMQV KCI resilience stated in [16, Theorem 18 on p. 40 and Lemma 21 on p. 41] does not hold.

Let q be a prime, p a prime dividing $(q-1)$, and G an element of $GF(q)$ of order p. Let $\hat{A}$ and $\hat{B}$ be two parties with respective static key pairs $(a, A = G^a)$, $(b, B = G^b)$ with $a, b \in \{1, \cdots, p-1\}$. An execution of the three–pass HMQV between $\hat{A}$ and $\hat{B}$ is as in Protocol 1; if any verification fails the execution aborts.

Protocol 1 Three Pass HMQV Key Exchange

I) The initiator $\hat{A}$ does the following:
   a) Choose $x \in_R \{1, \cdots, p-1\}$ 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) Verify that $X \in GF(q) \setminus \{0, 1\}$.
   b) Choose $y \in_R \{1, \cdots, p-1\}$ and compute $Y = G^y$.
   c) Compute $d = \bar{H}(X, \hat{B})$ and $e = \bar{H}(Y, \hat{A})$.
   d) Compute $s_B = y + eb \bmod p$, $\sigma_B = (XA^d)^{s_B}$, $K = H(\sigma_B, 1)$ and $K_m = H(\sigma_B, 0)$.
   e) Send $\big(\hat{B}, \hat{A}, Y, MAC_{K_m}(\text{“1”})\big)$ to $\hat{A}$.

III) At receipt of $\big(\hat{B}, \hat{A}, Y, MAC_{K_m}(\text{“1”})\big)$, $\hat{A}$ does the following:
   a) Verify that $Y \in GF(q) \setminus \{0, 1\}$.
   b) Compute $d = \bar{H}(X, \hat{B})$ and $e = \bar{H}(Y, \hat{A})$.
   c) Compute $s_A = x + da \bmod p$, $\sigma_A = (YB^e)^{s_A}$, $K = H(\sigma_B, 1)$ and $K_m = H(\sigma_B, 0)$.
   d) Validate $MAC_{K_m}(\text{“1”})$.
   e) Send $\big(\hat{A}, \hat{B}, X, Y, MAC_{K_m}(\text{“0”})\big)$ to $\hat{B}$.

IV) At receipt of $\big(\hat{A}, \hat{B}, X, Y, MAC_{K_m}(\text{“0”})\big)$, $\hat{B}$ validates $MAC_{K_m}(\text{“0”})$.

V) The shared session key is K.

The HMQV protocol is shown secure in a variant of the CK model, the CK_{HMQV} model [16] (see [8] for a comparison between the CK, CK_{HMQV}, and eCK models).

Suppose that q and p are primes such that $p \,|\, (q-1)$. Let G' be a primitive element in $GF(q)$; the element $G = G'^{(q-1)/p}$ has order p, and generates a group $\mathcal{G}$ of order p. For concreteness, suppose in addition that $(|q|, |p|) \in \{(1024, 160), (2048, 224), (3072, 256)\}$. The complexity of the Number Field Sieve

for prime field discrete logarithm[3] [13,30,31] is

$$L_q[1/3,\ \sqrt[3]{64/9}+o(1)] \approx \exp\left(\left(\sqrt[3]{64/9}+o(1)\right)(\ln(q))^{1/3}(\ln\ln(q))^{2/3}\right).$$

Hence, omitting the term $o(1)$ we have, $L_q[1/3,\ \sqrt[3]{64/9}] \geqslant 2^{87}$ when $|q| = 1024$ and $L_q[1/3,\ \sqrt[3]{64/9}] \geqslant 2^{117}$when $|q| = 2048$ and $L_q[1/3,\ \sqrt[3]{64/9}] \geqslant 2^{139}$ when $|q| = 3072$. So, we have $L_q[1/3,\ \sqrt[3]{64/9}] \geqslant p^{1/2}$; the complexity of the DLP on $\mathcal{G}$ reduces then to that of the generic attacks.

Let $\lambda = |p|$, $t = \lambda/3$ and suppose in addition, that $2^t|(q-1)$. Primes satisfying these conditions can be efficiently found using the following process[4]: (*i*) choose a prime p such that $|p| = \lambda$, and (*ii*) set $\alpha = 2^t \cdot p$; then (*iii*) try to find an integer s with bit–length $(|q| - |\alpha|)$, such that $q = s \cdot \alpha + 1$ is prime. By the theorem of Dirichlet on primes in arithmetic progression [11], we know that an infinity of primes in the form $s \cdot \alpha + 1$ exist; moreover the interval $\left[2^{|q|}, 1.048 \cdot 2^{|q|}\right]$ contains *at least* one prime from the progression [10]. Hence the interval $\left[2^{|q|}, 2^{|q|+1}\right]$ contains *at least* 14 of such primes (which is very pessimistic).

An example of such primes for $(|q|, |p|) = (3072, 256)$ is

$$\begin{aligned} q_{3072} &= 2^{91} \cdot 3^7 \cdot 5^7 \cdot 11^8 \cdot 17^3 \cdot 37^6 \cdot 67^2 \cdot 131^4 \cdot 257^4 \cdot 521^4 \cdot 1031^5 \cdot 2053 \cdot \\ &\quad 4099^7 \cdot 8209^4 \cdot 16411^5 \cdot 32771^4 \cdot 65537^4 \cdot 131101 \cdot 262147^5 \cdot 524309^7 \cdot \\ &\quad 1048583^5 \cdot 2097169^5 \cdot 4194319^2 \cdot 8388617 \cdot 16777259^4 \cdot 33554467^6 \cdot \\ &\quad 67108879^2 \cdot 134217757^5 \cdot 268435459^3 \cdot 536870923^8 \cdot 1073741827^5 \cdot \\ &\quad 2147483659^6 \cdot 4294967311^5 \cdot 8589934609^4 \cdot 17179869209^7 \cdot p_{256} + 1, \text{ with} \\ p_{256} &= 57896044618658097711785492504343953926634992332820282019728792003956564820063. \end{aligned}$$

Following the KCI scenario considered in [16, pp. 40–42], suppose that $\hat{A}$ and $\hat{B}$ are two honest parties, and $\mathcal{A}$ an attacker which knows $\hat{A}$'s static key a, and aims to impersonate $\hat{B}$ to $\hat{A}$. Suppose that $\hat{B}$ chooses his/her ephemeral keys in $\mathcal{G}^*$ as prescribed.

The attacker can proceed as follows: (*i*) using an invalid ephemeral public key, he/she learns the ephemeral secret exponent s_B at $\hat{B}$ in a three pass HMQV[5] session, as described, in Attack 2, and (*ii*) using s_B, the attacker impersonates indefinitely $\hat{B}$ to $\hat{A}$.

Attack 2 Online stage of an Ephemeral Secret Exponent Recovering

1) Compute $X = G'^{(q-1)/2^t}$.
2) Send $(\hat{A}, \hat{B}, X)$ to $\hat{B}$ to initiate a three–pass HMQV session.
3) Intercept $\hat{B}$'s response to $\hat{A}$, $(\hat{B}, \hat{A}, Y, \mathtt{tag}_{\hat{B}} = \mathrm{MAC}_{K_m}(\text{“1”}))$ and halt.

In a three–pass HMQV session, the key used at the responder for MACing is $K_m = H(\sigma, 0)$ with $\sigma = \left(XA^d\right)^{s_B}$ wherein $s_B = y + eb \mod p$, with $d = \bar{H}(X, \hat{B})$

[3] This is to date the best sieving algorithm for discrete logarithm over a prime field.
[4] It takes few seconds on a i7–4790K to find such primes.
[5] To launch this phase in the two–pass HMQV, the attacker has simply to wait, for instance, that $\hat{B}$ uses the key to authenticate some value he/she knows.

and $e = \bar{H}(Y, \hat{A})$. As

$$\sigma = \left(XA^d\right)^{s_B} = X^{s_B}\left(YB^e\right)^{da},$$

the attacker computes $\sigma_0 = \left(YB^e\right)^{da}$ and $\sigma_{1_i} = X^i$ and $K'_i = H(\sigma_{1_i}\sigma_0, 0)$, for $i = 1, 2, 3, \cdots, 2^t$ until $\mathrm{MAC}_{K'_i}(\text{“1”}) = \mathsf{tag}_{\hat{B}}$. By this exhaustive search, the attacker finds the t least significant bits of s_B. Then, using the relation

$$\sigma_0 = \left(YB^e\right)^{da} = \left(A^d\right)^{s_B},$$

the attacker recovers the remaining bits of s_B (recall that $t = \lambda/3$) using $\mathcal{O}\left(2^{(\lambda-t)/2}\right) = \mathcal{O}\left(2^t\right)$ operations [12, §B]. And then, the whole offline stage requires $\mathcal{O}\left(2^t\right)$ operations. The rough computational cost of the attack for different values of λ, in the case $t = \lambda/3$, are given hereunder.

Value of λ	**Rough computational cost**
160	2^{54}
224	2^{75}
256	2^{86}

As a concrete example, for $\lambda = 224$ (recall that $\lambda = |p|$) we have $t = 75$, then recovering s_B requires roughly 2^{75} operations, which is far from the 2^{112} operations required for the discrete logarithm problem, and not out of reach of our computational capabilities [14,19].

From a knowledge of s_B and the ephemeral public key Y generated by $\hat{B}$, the attacker can indefinitely impersonate $\hat{B}$ to $\hat{A}$, in both the two and three pass HMQV variants [27]. We stress that the attacker cannot recover $\hat{B}$'s static private key from s_B; this shows that for any primes p and q such that p divides $(q-1)$, 2^t divides $(q-1)$ and $\max\{2^{(|p|-t)/2}, 2^t\}$ operations are not out of reach, omitting ephemeral key validation *only once* is sufficient for an effective KCI attack. As *the attacker never learns an ephemeral private key*, this invalidates the claim that public key validation is required in the HMQV protocol “only in settings where ephemeral exponents are more vulnerable to attack than long–term secrets” [18]. Also, the “minimal requirement for a secure key–exchange ... that the attacker, not knowing the private key of a party $\hat{B}$, should not be able to impersonate $\hat{B}$” [16, p. 18] is not achieved.

About the factorization of $q - 1$. We presented our attack in the case a sufficiently large power of 2 divides $(q-1)$, however the attack can be launched as long as $(q-1)$ divisible by a “sufficiently large” integer. We stress that in real word settings, to avoid “sieving” attacks [26], q is chosen to be much larger than p; for instance the NIST recommends [1] the following pairs for $(|q|, |p|)$: (1024,160), (2048,224), and (3072,256). Hence for real word domain parameters, it is likely that q has a factor of bit–length $\approx |p|/3$. If M is a divisor of $q-1$ with bit–length $\lambda/3$ (recall that $\lambda = |p|$), the element $X = G'^{(q-1)/M}$ has order M,

and can be used as outgoing ephemeral key in the online stage (Attack 2). By exhaustive search, $\beta_1 = s_B \mod M$ can be found using M operations. Then, as $s_B = y + eb \mod p = \beta_2 M + \beta_1 \mod p$, with $|\beta_2| \leqslant 2\lambda/3$, from the relation $\sigma_0 = (YB^e)^{da} = (A^d)^{s_B} = (A^d)^{\beta_2 M + \beta_1}$, one obtains $\sigma_0 A^{-d\beta_1} = (A^{dM})^{\beta_2}$; the remaining part β_2 can then be recovered using $\mathcal{O}(2^{\lambda/3})$ operations [12]. The attack can then be launched for any divisor M of $q-1$ of bit–length t such that $\mathcal{O}(\max\{2^{(\lambda-t)/2}, 2^t\})$ operations are not out of reach.

2.1 On the HMQV Security Reduction

The KCI attack is totally well grounded in the $\mathrm{CK}_{\mathrm{HMQV}}$ model; a natural question is then how can it co–exist with the HMQV security reduction.

The attack is rooted in the interpretation of the XCR security reduction in the analysis of the DCR scheme. In fact, a DCR signature is an XCR signature by $\hat{B}$ (resp. $\hat{A}$) with the challenge XA^d (resp. YB^e). As the DCR reduction uses the XCR reduction [16, pp. 20–25], wherein challenges are supposed to belong to $\mathcal{G}^*$, it becomes a requirement that both XA^d and YB^e belong to $\mathcal{G}^*$. Hence, when KCI is considered, namely, when a is known to the attacker, the security reduction leads to $\mathrm{CDH}(X, B)$. Unfortunately, when $X \notin \mathcal{G}^*$, there is no guarantee that computing $\mathrm{CDH}(X, B)$ is hard. As the core of the HMQV protocol is the DCR scheme, it then becomes also a requirement that ephemeral keys be tested for membership in $\mathcal{G}^*$. This point was missed in the analysis of the HMQV protocol and explains the co–existence of the attack and the security reduction in [16].

We stress that contrary to the DCR and XCR schemes, the FDCR signature of $\hat{A}$ and $\hat{B}$ on messages m_1, m_2 and challenges X and Y is not a FXCR signature of $\hat{A}$ (resp. $\hat{B}$) on the message m_2 (resp. m_1) and challenge YB^e (resp. XA^d) [28]. Also, the attack does not apply to protocols that mandate ephemeral key validation, such as MQV [21] and FHMQV [28].

Nonetheless, in the case of MQV, when ephemeral keys are not validated, the attack can be launched. In this case, as the ephemeral secret exponents $s_A = x + (\tilde{X} \mod 2^l)a$ and $s_B = y + (\tilde{Y} \mod 2^l)b$, where $\tilde{X}$ is the integer representation of X, are not tied to the peer's identity, the attacker can not only impersonate $\hat{B}$ to $\hat{A}$, but to *any party*. Moreover, there is no need for the attacker to learn an honest party's static key, the attacker can *use his/her own static key together with an invalid ephemeral key.*

In the case of FHMQV, which is resilient to ephemeral secret exponent leakage, we do not know how the attack can be launched. However, if ephemeral keys are not validated and ephemeral private key leakage is considered at the victim $\hat{B}$, the attacker can disclose the victim's static private key, in both MQV, HMQV and FHMQV.

3 FXCR Security in the Reversed Interaction Order

In this section, we revisit the FXCR scheme [27], clarifying its advantages over the XCR scheme. We show also that the recent critics from [22] about the FXCR

security reduction (which is the main ingredient in FHMQV security arguments) are erroneous. As already reported in [27], even if the ephemeral keys are tested for membership in $\mathcal{G}^*$, the HMQV protocol is sensitive to partial leakage of the ephemeral exponents s_A and s_B. This observation lead to the design of the FXCR scheme.

Definition 1 (FXCR Scheme). *Let $\hat{B}$ be an entity with public key $B \in \mathcal{G}^*$. $\hat{B}$'s signature on a challenge X provided by a verifier $\hat{A}$ together with a message m is $FSig_{\hat{B}} = (Y, X^{y+\bar{H}(Y,X,m)b})$. The verifier accepts a pair (Y, σ) as a valid signature if $\left(YB^{\bar{H}(Y,X,m)}\right)^x = \sigma$.*

From [27], it is shown that no efficient attacker, even if given the secret exponent s_B at each signature generation can forge a valid FXCR signature, unless with negligible probability. The authors of [22][6] consider a reversed interaction order between the signer and the verifier and claim that the FXCR security reduction is flawed, as the simulation becomes invalid in this case. Namely, if the challenge is provided to the signer after it generates Y, the security reduction does not hold.

Strictly speaking changing the interaction order defines another signature scheme; and the security reduction may become inapplicable for the new scheme. Furthermore, even if the interaction order is changed, all the security attributes claimed in [27] about the FXCR scheme remain valid; and contrary to what is suggested in [22], no additional Gap DH assumption is required. We still denote the variant of the signature scheme obtained by changing the interaction order by FXCR and consider a signer $\hat{B}$ and a verifier $\hat{A}$ interacting as described in Figure 3.

Figure 3 FXCR Interactions for Signature Generation

1) At signature request with a message m, $\hat{B}$ generates $Y \in \mathcal{G}^*$ and provides $\hat{A}$ with (m, Y, B).
2) $\hat{A}$ chooses $x \in_R \{1, \cdots, p-1\}$ and provides $\hat{B}$ with $(m, X = G^x, Y, B)$.
3) $\hat{B}$ verifies that $X \in \mathcal{G}^*$. If the verification succeeds, it provides $\hat{A}$ with $(m, X, Y, B, \sigma, s_B)$ wherein $\sigma = X^{s_B}$ and $s_B = y + \bar{H}(Y, X, m)b$.
4) The verifier accepts $\hat{B}$'s signature as valid if $\left(YB^{\bar{H}(Y,X,m)}\right)^x = \sigma$.

We stress that the verifier is provided also with the secret exponent s_B; this models total secret exponent leakage in each signature generation.

Definition 2 (FXCR Security). *The FXCR scheme is said to be secure, if no efficient attacker can succeed in the game in Figure 4 with non–negligible probability.*

[6] Their abstract starts with "HMQV is one of the most efficient (provably secure) authenticated key–exchange protocols based on public–key cryptography, and is widely standardized." To date, we are not aware of any standardization body which has already adopted the HMQV protocol.

Figure 4 The FXCR Security Game

1) The attacker $\mathcal{A}$ is given a public key B, a challenge X_0, together with a signing oracle as described in Figure 3, and also a hashing oracle.
2) The attacker halts with output $(0,0,0,0,0)$ to indicate a failure, or a quintuple $(m_0, X_0, Y_0, B, \sigma_0)$ such that
   a) (Y_0, σ_0) is a valid signature with respect to the public key B and message–challenge pair (m_0, X_0), and
   b) (Y_0, σ_0) is a fresh signature, *i. e.*, (Y_0, σ_0) was never generated by $\hat{B}$ on signature request on (m_0, X_0).

Notice that contrary to [16, §4.1], which requires that "the pair (Y_0, m_0) did not appear in any of the responses of $\hat{B}$", we use the minimal requirement that (Y_0, σ_0) was not generated by the signer on the message–challenge pair (m_0, X_0).

Theorem 1. *Under the Computational Diffie–Hellman (CDH) assumption in $\mathcal{G}$ and the Random Oracle (RO) model, the FXCR scheme is secure in the sense of Definition 2.*

Proof. As the attacker is supposed to be polynomial in $|p|$, let P be a polynomial and $T = P(|p|)$ an upper bound on the number of digest queries on messages with format (Y, Z, m) with $Y, Z \in \mathcal{G}^*$, the attacker issues after it receives (m, Y, B) from the signing oracle (step 1 of Figure 3) and before it provides the signing oracle with its challenge (m, X, Y, B) (step 2 of Figure 3). Also, we suppose that the number of signature queries the attacker issues is upper bounded by $L = Q(|p|)$ for some polynomial Q.

To lighten the presentation, we suppose that the attacker behaves as follows. First, for each signature generation, the attacker issues exactly $T = P(|p|)$ digest queries on messages with format (Y, Z, m), with $Z \in \mathcal{G}^*$, after he/she receives (m, Y, B) from the signing oracle, and before he/she provides the signing oracle with the challenge (m, X, Y, B). The attacker may discard digest values with no interest. Second, among the digest queries the attacker issues, one query is on (Y, X, m), where X is the challenge to be submitted to the signing oracle. Third, the attacker never submits to the hashing oracle the same message twice (the attacker can keep track of his/her previous digest queries).

We stress that the attacker we consider remains polynomial in $|p|$ and from any efficient attacker $\mathcal{A}'$ one can derive an efficient attacker $\mathcal{A}$ which behaves as described and succeeds with the same probability than $\mathcal{A}'$. The attacker $\mathcal{A}$ behaves exactly as $\mathcal{A}'$ except that for signature generation, after he/she receives (m, Y, B) from the signing oracle and before he/she provides $\hat{B}$ with the challenge X, he/she ensures that T digest queries on messages with format (Y, Z, m), including one query on (Y, X, m), are issued. $\mathcal{A}$ ignores the digest values $\mathcal{A}'$ does not issue; he/she remains polynomial and has the same success probability than $\mathcal{A}'$.

The attacker's interactions with the signing and hashing oracles are summarized in Figure 5; without loss of generality, we omit digest queries of other kinds the attacker may issue between consecutive steps of Figure 5.

Figure 5 Modified queries for Signature Generation

1) For the j–the signature query, activate the signing oracle with a message m_j to obtain (m_j, Y_j, B).
2) Generate a challenge $X_j \in \mathcal{G}^*$ and issue T digest queries on messages with format $(Y_j, Z_{j,i}, m_j)_{i\in\{1,\cdots,T\}}$ with one of the $Z_{j,i}$'s being the challenge X_j to be submitted to the signing oracle.
3) Provide the signing oracle with (m_j, X_j, Y_j, B).
4) And receive the signature $(m_j, X_j, Y_j, B, \sigma_j, s_{j,B})$ from the signing oracle.

Let $\Pr(\mathrm{Succ}_{\mathcal{A}})$ denote the probability that $\mathcal{A}$ succeeds in the FXCR security game, and $\mathbf{V} = \{1, \cdots, T\}^L$ be the set of L–uples of elements of $\{1, \cdots, T\}$. We denote by V the random variable that takes values in $\mathbf{V}$ and describes the digest queries at step 2 of Figure 5 wherein the attacker provides the hashing oracle with the message (Y_j, X_j, m_j) (X_j being the challenge to be submitted to the signing oracle). Namely, for $v = (v_1, \cdots, v_L) \in \mathbf{V}$, we denote by $\Pr(V = v)$ the probability that for all $j \in \{1, \cdots, L\}$, the v_j–th digest query at step 2 in the j–th signature generation, Z_{j,v_j} equals the challenge X_j; *i. e.* $\Pr(V = v)$ denotes the probability that the attacker provides the signing oracle with challenge Z_{1,v_1} in the first signature query, and Z_{2,v_2} as a challenge in the second signature query, and so forth. For $v \in \mathbf{V}$, we say that v is *possible* if $\Pr(V = v)$ is non–zero and denote by $\mathrm{Poss}(\mathbf{V})$ the subset of $\mathbf{V}$ consisting of possible v's.
The probability of success of the adversary $\mathcal{A}$ is

$$\begin{aligned}
\Pr(\mathrm{Succ}_{\mathcal{A}}) &= \sum_{v\in\mathrm{Poss}(\mathbf{V})} \Pr(\mathrm{Succ}_{\mathcal{A}} \cap V = v) \\
&= \sum_{v\in\mathrm{Poss}(\mathbf{V})} \Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v) \Pr(V = v) \\
&\leqslant \sum_{v\in\mathrm{Poss}(\mathbf{V})} \left(\max_{v\in\mathrm{Poss}(\mathbf{V})} \Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v) \right) \Pr(V = v) \\
&\leqslant \max_{v\in\mathrm{Poss}(\mathbf{V})} \Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v). \qquad (1)
\end{aligned}$$

It then suffices to show that for all $v \in \mathrm{Poss}(\mathbf{V})$, $\Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v)$ is negligible. Suppose there is v such that $\Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v)$ is non–negligible. Using $\mathcal{A}$, we show the existence of an efficient CDH solver $\mathcal{S}$ which succeeds with non–negligible probability. The solver works as in Figure 6.
The simulator is efficient, moreover it provides a consistent simulation; and under the RO model, this simulated environment is indistinguishable from a real one. The probability that the attacker provides a valid forgery without issuing $\bar{H}(Y_0, X_0, m_0)$ is 2^{-l}. Hence, in this simulation, the attacker succeeds with non–negligible probability. From the General Forking Lemma [2], the probability the attacker succeeds in the simulation and in the repeat experiment is

$$\Pr(\mathrm{Succ}_2) \geqslant \Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v) \left(\frac{\Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v)}{q} - 2^{-l} \right),$$

Figure 6 A CDH solver $\mathcal{S}$ from $\mathcal{A}$

Run of $\mathcal{A}$:

1) When $\mathcal{S}$ is activated with a message m_j, it does as follows:
   a) Choose $s_{j,B} \in_R \{1, \cdots, p-1\}$, $e_j \in_R \{0,1\}^l$, set $Y_j = G^{s_{j,B}} B^{e_j^{-1}}$.
   b) Create an empty list $\mathcal{L}_{e_j, Y_j, s_{j,B}, m_j}$.
   c) Provide the attacker with (m_j, Y_j, B).
2) At digest query on a message which does not have format (Y, Z, m), the simulator $\mathcal{S}$ responds with $e \in_R \{0,1\}^l$.
3) At digest query on a message with format (Y, Z, m), $\mathcal{S}$ does as follows:
   a) If $\mathcal{A}$ was provided with (m_j, Y_j, B) (Step 2 of Figure 3) such that $m_j = m$ and $Y_j = Y$ and if $|\mathcal{L}_{e_j, Y_j, m_j}| = v_j - 1$, $\mathcal{S}$ provides the attacker with e_j, and appends Z to $\mathcal{L}_{e_j, Y_j, s_{j,B}, m_j}$.
   b) Otherwise, $\mathcal{S}$ responds with $e \in_R \{0,1\}^l$, and if $m_j = m$ and $Y_j = Y$, it appends Z to $\mathcal{L}_{e_j, Y_j, m_j}$.
4) When $\mathcal{A}$ provides $\mathcal{S}$ with (m_j, X_j, Y_j, B), $\mathcal{S}$ responds with $(m_j, X_j, Y_j, B, G^{s_{j,B}}, s_{j,B})$. Notice that this is consistent with the digest simulation at steps 2 and 3.
5) At $\mathcal{A}$'s halt, $\mathcal{S}$ verifies that $\mathcal{A}$'s output is different from $(0,0,0,0,0)$ and satisfies the following conditions; if not $\mathcal{S}$ aborts.
   – $Y_0 \in \mathcal{G}^*$ and $\bar{H}(Y_0, X_0, m_0)$ was issued from $\bar{H}$.
   – The signature (Y_0, σ_0) was not returned by $\hat{B}$ on query (m_0, X_0).

Repeat: $\mathcal{S}$ executes a new run of $\mathcal{A}$, using the same input and coins; and answering to all digest queries before $\bar{H}(Y_0, X_0, m_0)$ with the same values as in the previous run. The new query of $\bar{H}(Y_0, X_0, m_0)$ and subsequent queries to $\bar{H}$ are answered with new random values.

Output: If $\mathcal{A}$ outputs a second signature $(m_0, X_0, Y_0, B, \sigma_0')$ satisfying conditions of step 5, with a hash value $\bar{H}(Y_0, X_0, m_0)_2 = e_0' \neq e_0 = \bar{H}(Y_0, X_0, m_0)_1$ then $\mathcal{S}$ outputs $\left(\sigma_0/\sigma_0'\right)^{(e_0 - e'_0)^{-1}}$ as a guess for $\mathrm{CDH}(B, X_0)$.

where q is the number of digest queries the attacker issues, which is non–negligible, unless $\Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v)$ is negligible. Moreover, if the repeat experiment succeeds, the digest values e_0 and e_0' are different with probability $1 - 2^{-l}$, and then the computation $\left(\sigma_0/\sigma_0'\right)^{(e_0 - e_0')^{-1}}$, leads to $\mathrm{CDH}(X_0, B)$ with non–negligible probability, contradicting then the CDH assumption. Hence, under the RO model and the CDH assumption, for all $v \in \mathbf{V}$, $\Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v)$ is negligible; using (1), we conclude that $\Pr(\mathrm{Succ}_{\mathcal{A}})$ is negligible. □

This shows that the FXCR CDH–based security reduction holds not only in what the authors of [22] calls a "regular interaction order", but also if the interaction order is reversed.

4 FDCR Security in the Reversed Interaction Order

As for the FXCR scheme, we show here that the security of the FDCR scheme totally holds, in the reversed interaction order, wherein the signer provides his/her ephemeral public key before receiving a challenge from the verifier.

Definition 3 (FDCR Scheme). *Let $\hat{A}$ and $\hat{B}$ be entities with respective public keys A and B in $\mathcal{G}^*$. The FDCR signature of $\hat{A}$ and $\hat{B}$ on challenge–message pairs (X, m_1) and (Y, m_2) provided respectively by $\hat{A}$ and $\hat{B}$, with $X, Y \in \mathcal{G}^*$ is*

$$FDSig_{\hat{A},\hat{B}}(m_1, m_2, X, Y) = (XA^d)^{y+eb} = (YB^e)^{x+da},$$

where $d = \bar{H}(X, Y, m_1, m_2)$ and $e = \bar{H}(Y, X, m_1, m_2)$.

To show the FDCR security in the reversed interaction order, we consider a verifier interacting with a signer $\hat{B}$ as described in Figure 7, and the game in Figure 8.

Figure 7 FDCR interactions for Signature Generation

1) The verifier $\hat{A}$ provides $\hat{B}$ with (m_1, A).
2) The signer $\hat{B}$ responds with (m_1, m_2, Y, A, B), with $Y \in \mathcal{G}^*$.
3) The verifier chooses $x \in \{1, \cdots, p-1\}$ and provides $\hat{B}$ with $(m_1, m_2, X = G^x, Y, A, B)$.
4) The signer verifies that $X \in \mathcal{G}^*$, and provides $\hat{A}$ with $(m_1, m_2, X, Y, A, B, \sigma)$ wherein $\sigma = (XA^d)^{y+eb}$ with $d = \bar{H}(X, Y, m_1, m_2)$ and $e = \bar{H}(Y, X, m_1, m_2)$.
5) The verifier accepts $\hat{B}$'s signature as valid if $(YB^e)^{x+da} = \sigma$.

Figure 8 The FDCR Security Game

1) The attacker $\mathcal{A}$ is given a key pair (A, a) and a message–challenge pair (X_0, m_{1_0}); $\mathcal{A}$ is also given access to a hashing oracle, and is allowed to interact with a signing oracle as described in Figure 7.
2) The attacker halts with output $(0, 0, 0, 0, 0, 0, 0)$ to indicate a failure, or a septuple $(m_{1_0}, m_{2_0}, X_0, Y_0, A, B, \sigma_0)$ such that
   a) σ_0 is a valid FDCR signature on messages m_{1_0}, m_{2_0} and challenges X_0, Y_0 with respect to the public keys A and B.
   b) σ_0 is a fresh, *i. e.*, it was not generated as a signature on message–challenge pairs (m'_1, X_0), (m'_2, Y_0) such that $m'_1 || m'_2 = m_{1_0} || m_{2_0}$.

Definition 4 (FDCR Security). *The FDCR scheme is said to be secure if no efficient attacker can succeed in the game in Figure 8 with non–negligible probability.*

Theorem 2. *Under the RO model and the CDH assumption, the FDCR scheme is secure in the sense of Definition 4.*

Proof. To lighten the presentation, as in Theorem 1, we suppose that the attacker issues $L = Q(|p|)$ signature queries and $T = P(|p|)$ digest queries on messages with format (Z_1, Z_2, m_1, m_2), with $Z_1, Z_2 \in \mathcal{G}$, between the steps 2 and 3 of Figure 7. We suppose also that the attacker issues a digest query on (Y, X, m_1, m_2), before providing its challenge X to the signer; and also that he/she does not issue the same digest query twice. We stress that the attacker remains polynomial, and may discard the digest values of no interest. We summarize the queries for a signature generation in Figure 9.

Figure 9 Attacker's queries for FDCR Signature Generation

1) For the j–the signature query, $\mathcal{A}$ activates the signing oracle with (m_{1_j}, A).
2) The signer provides the attacker with $(m_{1_j}, m_{2_j} Y_j, A, B)$ with $Y_j \in \mathcal{G}^*$.
3) $\mathcal{A}$ generates $X_j \in \mathcal{G}^*$ and issues T digest queries on messages with format $(Y_j, Z_{j,i}, m_{1_j}, m_{2,j})_{i \in \{1,\cdots,T\}}$ with one query on $(Y_j, X_j, m_{1_j}, m_{2,j})$.
4) The attacker provides the signing oracle with $(m_{1_j}, m_{2_j}, X_j, Y_j, A, B)$.
5) And receives $(m_{1_j}, m_{2_j}, X_j, Y_j, A, B, \sigma_j)$ from the signing oracle.

We still denote $\{1, \cdots, T\}^L$ by $\mathbf{V}$, for $v = (v_1, \cdots, v_L) \in \mathbf{V}$, we denote by $\Pr(V = v)$ the probability that for all $j \in \{1, \cdots, L\}$, for the j–th signature generation, the attacker issues a digest query on $(Y_j, X_j, m_{1_j}, m_{2,j})$ at the v_j–th digest query in step 2. The notations $\mathrm{Poss}(\mathbf{V})$ from the proof of Theorem 1 is used again. Conditioning on V, we still obtain

$$\Pr(\mathrm{Succ}_{\mathcal{A}}) \leqslant \max_{v \in \mathrm{Poss}(\mathbf{V})} \Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v).$$

Suppose that there is $v \in \mathrm{Poss}(\mathbf{V})$ such that $\Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v)$ is non–negligible. Using $\mathcal{A}$, we build an efficient FXCR forger $\mathcal{S}$ such that $\Pr(\mathrm{Succ}_{\mathcal{S}} \mid V = v)$ is non–negligible. The forger $\mathcal{S}$ works as described in Figure 10.

Under the random oracle model, the simulation in Figure 10 is perfect, except with negligible probability; a deviation occurs when the same message–challenge pair (m_{2_j}, Y_j) is chosen twice in two signature queries on the same pair (m_{1_j}, X_j). As the simulator chooses its challenges uniformly at random in $\mathcal{G}^*$, this occurs with probability $L/(p-1)$ which is negligible. Also, the probability the attacker provides a valid forgery without issuing $\bar{H}(X_0, Y_0, m_{1_0}, m_{2_0})$ and $\bar{H}(Y_0, X_0, m_{1_0}, m_{2_0})$ is smaller than 2^{-l}, which is negligible. Hence, if $\mathcal{A}$ succeeds with non–negligible probability in a real environment, it succeeds also with non–negligible probability under this simulation. Furthermore $\mathcal{S}$ succeeds with probability

$$\Pr(\mathrm{Succ}_{\mathcal{S}} \mid V = v) \geqslant \Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v) - \frac{L}{(p-1)} - 2^{-l},$$

which is non–negligible if $\Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v)$ is non–negligible. As already shown in Theorem 1, this is impossible under the RO model and the CDH assumption. Hence, for all $v \in \mathrm{Poss}(\mathbf{V})$, $\Pr(\mathrm{Succ}_{\mathcal{A}} \mid V = v)$ is negligible, and then $\Pr(\mathrm{Succ}_{\mathcal{A}})$ is negligible. □

Figure 10 A FXCR Forger $\mathcal{S}$ from $\mathcal{A}$

Run of $\mathcal{A}$:

1) When $\mathcal{S}$ is activated with (m_{1_j}, A), it does the following:
   a) Choose $s_{j,B} \in_R \{1, \cdots, p-1\}$, $e_j \in_R \{0,1\}^l$, $m_{2_j} \in \{0,1\}^{F(|p|)}$ for some positive polynomial F, set $Y_j = G^{s_{j,B}} B^{e_j^{-1}}$.
   b) Create an empty list $\mathcal{L}_{e_j, Y_j, s_{j,B}, m_{1_j}, m_{2_j}}$.
   c) Provides the attacker with $(m_{1_j}, m_{2_j}, Y_j, B)$.
2) At $\mathcal{A}$'s digest query on a message which does not have format (Y, Z, m_1, m_2), the simulator $\mathcal{S}$ responds with $e \in_R \{0,1\}^l$.
3) At digest query on messages with format (Y, Z, m_1, m_2), $\mathcal{S}$ does as follows:
   a) If it provided the attacker with $(m_{1_j}, m_{2_j}, Y_j, A, B)$ such that $m_{1_j}||m_{2_j} = m_1||m_2$, $Y = Y_j$ and if $|\mathcal{L}_{e_j, Y_j, s_{j,B}, m_{1_j}, m_{2_j}}| = v_j - 1$, it provides the attacker with e_j and appends Z to $\mathcal{L}_{e_j, Y_j, s_{j,B}, m_{1_j}, m_{2_j}}$.
   b) Otherwise, it responds with $e \in_R \{0,1\}^l$, and if $m_{1_j}||m_{2_j} = m_1||m_2$ and $Y_j = Y$ then it appends Z to $\mathcal{L}_{e_j, Y_j, s_{j,B}, m_{1_j}, m_{2_j}}$.
4) When $\mathcal{A}$ provides $(m_{1_j}, m_{2_j}, X_j, Y_j, A, B)$, if no value is already assigned to $d = \hat{H}(X_j, Y_j, m_{1_j}, m_{1_j})$ $\mathcal{S}$ chooses $d \in_R \{0,1\}^l$, and responds with $(m_{1_j}, m_{2_j}, X_j, Y_j, A, B, (X_j A^d)^{s_{j,B}})$.
5) At $\mathcal{A}$'s halt with a non–null output $(m_{1_0}, m_{2_0}, X_0, Y_0, A, B, \sigma_0)$ $\mathcal{S}$ verifies that the following conditions are satisfied; if not it aborts.
   – $Y_0 \in \mathcal{G}^*$ and $d_0 = \bar{H}(Y_0, X_0, m_{1_0}, m_{2_0})$ and $e_0 = \bar{H}(X_0, Y_0, m_{1_0}, m_{2_0})$ were issued from the hashing oracle.
   – $\mathcal{S}$ never issued a signature $(m_1', m_2', X_0, Y_0, A, B, \sigma_0)$ such that $m_1'||m_2' = m_{1_0}||m_{2_0}$.

Output: If all the conditions at step 5 are satisfied, $\mathcal{S}$ outputs $\sigma_0(Y_0 B^{e_0})^{-d_0 a} = (Y_0 B^{e_0})^{x_0}$ as a FXCR forgery on $m_{1_0}||m_{2_0}$.

This shows that all the FDCR security attributes remain intact in the interaction order considered in [22].

5 Separation between FHMQV and HMQV

Security Separation

The sensitivity of the HMQV protocol to partial leakages on intermediate exponents s_A and s_B [27], exploited again with KCI attack in §2, motivated the FHMQV design which is resilient to such leakages. FHMQV was shown secure in a mixture of two security definitions (termed ck and eck in [27]), which was latter refined into the seCK model [29]. In the CK_{FHMQV} model (ck model in [27]), it is assumed that at all parties the ephemeral keys are as protected as the static ones. This assumption matches some common implementations; such as (EC)DSA signature generation (where a leakage of an ephemeral private key leads to a disclosure of the signer's static private key). However, it does not seem reasonable to assume implementations performed in the same way at all

parties, and then consider the same leakages at all parties. We point out that the CK, $\mathrm{CK}_{\mathrm{HMQV}}$, eCK and $\mathrm{CK}_{\mathrm{FHMQV}}$ models define the information that can be leaked in the same way for all parties. While in real word settings, implementations may be different, depending on environments specificities (presence of a desktop computer (DC) only, of a power limited smart–card and a DC, etc.). This observation is one of the motivations of seCK model and corresponds to real–world vulnerabilities [15,32,33,34,35].

Broadly, in the seCK model, it is assumed for DH protocols that at each party, implementation is performed using one of the two following approaches[7].

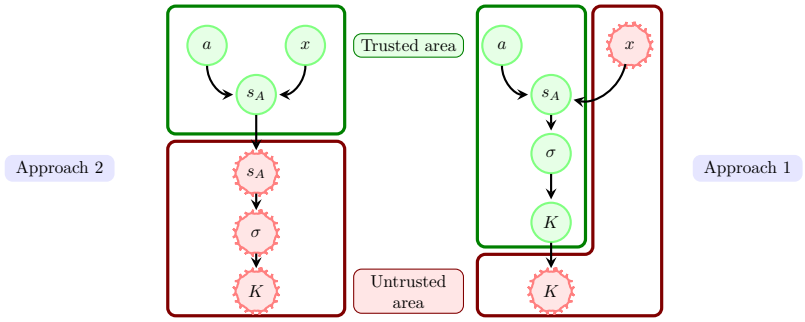

Fig. 1. (F)HMQV Implementation Approaches in the seCK Model

Approach 1. It is assumed that the ephemeral keys are generated in an untrusted area, and the session keys are used also in this area. All the other intermediate results are computed in the trusted area. At a party using this approach, the attacker is allowed the following queries:
- EphemeralKeyReveal(session) to learn a session's ephemeral private key;
- SessionKeyReveal(session) to learn a session key;
- $\mathsf{Corrupt}_{\mathsf{SC}}$(party) to model an attacker which bypasses the tamper protection mechanisms and learns the party's static key;
- EstablishParty(party, key) to register a static key on behalf of the party; a party against which this query is not issued is said to be *honest*.

Approach 2. In this approach, it is supposed that both the static and ephemeral keys are computed and used in the trusted area, and all other computations are performed in the untrusted area. So, the attacker is provided with
- SessionKeyReveal(session) and EstablishParty(party, key) queries, and
- a reveal query to learn any intermediate result that is computed or used in the untrusted area.

Matching sessions. A session at a party $\hat{P}_i$ is identified with a quintuple $(\hat{P}_i, \hat{P}_j, \mathsf{out}, \mathsf{in}, \mathsf{role})$ wherein $\hat{P}_j$ is the peer, out is the list of the messages sent to the peer, in is the list of the messages received, and role is $\hat{P}_i$'s role, initiator $\mathcal{I}$ or responder $\mathcal{R}$. Two sessions $(\hat{P}_i, \hat{P}_j, \mathsf{out}, \mathsf{in}, \mathsf{role})$ and $(\hat{P}'_i, \hat{P}'_j, \mathsf{out'}, \mathsf{in'}, \mathsf{role'})$

[7] These implementation approaches are not the only possible, however they seem to be common enough in real word to be considered in the model.

are said to be matching if $\hat{P}_i = \hat{P}'_j$, $\hat{P}_j = \hat{P}'_i$, $\mathsf{out} = \mathsf{in}'$, $\mathsf{in} = \mathsf{out}'$, and $\mathsf{role} \neq \mathsf{role}'$.

Session freshness. A session at an honest party following the implementation approach 1 is said to be *locally exposed* if it were issued a SessionKeyReveal query, or if it were issued an EphemeralKeyReveal query and its owner were issued a $\mathsf{Corrupt}_{\mathsf{SC}}$ query. At an honest party following the second approach, a session is said to be locally exposed, if it were issued a SessionKeyReveal query or an intermediate result query. A session is said to be *exposed* if it is locally exposed or if its matching session (if any) is locally exposed. A non–exposed session is said to be *seCK–fresh*.

seCK Security. A protocol is said to be secure if (*i*) when two honest parties complete matching session, then they both derive the same session key; and (*ii*) an efficient attacker in total control of communication links cannot distinguish a fresh session key from a random value chosen uniformly from the distribution of session keys with probability significantly greater than 1/2.

As already reported in [29], seCK security implies eCK security[8]; seCK security is also strictly stronger than $\mathrm{CK}_{\mathrm{FHMQV}}$ security . The seCK model and the $\mathrm{CK}_{\mathrm{HMQV}}$ security models are formally incomparable, as the seCK model considers only role–asymmetric protocols while the $\mathrm{CK}_{\mathrm{HMQV}}$ model considers only role–symmetric protocols [8][9]. Nevertheless, as shown in [29], there are attacks which are captured in the seCK model but not captured in the eCK and $\mathrm{CK}_{\mathrm{HMQV}}$ models. While any real word attack that is captured in the $\mathrm{CK}_{\mathrm{HMQV}}$ and eCK models is also captured in the seCK model.

We stress that even when $\mathcal{G}$–tests are performed, HMQV is insecure in the seCK model, for two reasons. First, HMQV is known to be vulnerable to a KCI impersonation attack when leakages on the shared secret σ are considered [16, pp. 17–18]. Second, in the case of a ("sufficient" partial) leakage on ephemeral secret exponents s_A or s_B in a session, an attacker can indefinitely impersonate the session owner; the HMQV protocol cannot then, meet a security definition which allows total leakages on both the shared secret σ and the ephemeral secret exponents s_A and s_B.

Theorem 3. *Under the RO model and the Gap Diffie–Hellman Assumption in $\mathcal{G}$, the FHMQV protocol is seCK–secure.*

Although we already analyzed the main ingredients of the proof of Theorem 3 (the FXCR and FDCR schemes), for lack of space, we do not provide the proof here. We defer the security reduction to the extended version of this paper.

[8] There is no dynamic key registration query in the eCK model [20]; the adversary is only allowed to select dishonest parties before starting its game. Dynamic key registration permits the adversary to select the parties it sets as dishonest *after* having seen their behaviour; this is an advantage for the adversary, and does not affect the comparability between the seCK and the eCK models.

[9] Given the work [8], the *Claim 1* from [22] about the formal incomparability between $\mathrm{CK}_{\mathrm{FHMQV}}$ and the $\mathrm{CK}_{\mathrm{HMQV}}$ models is trivial.

Protocol 11 The FHMQV Key Exchange

I) The initiator $\hat{A}$ does the following:
  a) Choose $x \in_R \{1, \cdots p-1\}$ 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) Verify that $X \in \mathcal{G}^*$.
  b) Choose $y \in_R \{1, \cdots p-1\}$ and compute $Y = G^y$.
  c) Send $(\hat{B}, \hat{A}, Y)$ to $\hat{A}$.
  d) Compute $d = \bar{H}(X, Y, \hat{A}, \hat{B}), e = \bar{H}(Y, X, \hat{A}, \hat{B})$ and $s_B = y + eb \bmod p$.
  e) Compute $\sigma_B = (XA^d)^{s_B}$ and $K = H(\sigma_B, \hat{A}, \hat{B}, X, Y)$.

III) At receipt of $(\hat{B}, \hat{A}, Y)$, $\hat{A}$ does the following:
  a) Verify that $Y \in \mathcal{G}^*$.
  b) Compute $d = \bar{H}(X, Y, \hat{A}, \hat{B}), e = \bar{H}(Y, X, \hat{A}, \hat{B})$ and $s_A = x + da \bmod p$.
  c) Compute $\sigma_A = (YB^e)^{s_A}$ and $K = H(\sigma_A, \hat{A}, \hat{B}, X, Y)$.

IV) The shared session key is K.

Efficiency Separation

Without a proper validation of ephemeral keys, the HMQV protocol cannot achieve its security goals. When ephemeral keys are validated in HMQV, the FHMQV protocol is as efficient as HMQV in the implementation approach 1. Moreover, for FHMQV, in approach 2, if ephemeral keys are computed in idle–time, only one digest computation, one modular integer addition and one modular integer multiplication has to be performed in the trusted area in non–idle-time; *no exponentiation is performed in the trusted area (usually a smart–card or a hardware security module) in non–idle time.* As neither HMQV, nor MQV can confine the effects of a secret exponent (s_A or s_B) leakage to the leaked session, none of these protocols can achieve such a performance.

6 Concluding remarks

We revisited the FXCR and the FDCR signature schemes which are the building blocks of the FHMQV protocol, clarifying their strengths, independence to interaction order, and security advantages compared to the XCR and DCR schemes. We clarified also both the security and efficiency separation between HMQV and FHMQV, showing that even if ephemeral keys are validated in HMQV, the FHMQV protocol is strictly stronger than HMQV both in security and efficiency. In settings wherein a trusted device is used to store static and ephemeral keys, a FHMQV implementation can achieve performances which cannot be achieved by MQV or HMQV.

We pointed out a Key Compromise Impersonation attack against HMQV. Namely we showed that omitting ephemeral key validation *only once* is sufficient for a Key Compromise Impersonation. Besides, we revisited the motivations of the seCK model, showing that it is formally stronger than the eCK model, and from a real word perspective, stronger than the $\mathrm{CK}_{\mathrm{HMQV}}$ model.

In a future work we will be interested in generalizing the compiler from [9] to security models allowing dynamic key registration and intermediate results leakage in the multiple CAs setting [4,5].

References

1. Barker E., Barker W., Burr W., Polk W., and Smid M.: NIST Special Publication 800–57 Recommendation for Key Management – Part 1: General (Revision 3), July 2012 (see also the draft of Revision 4 at http://tinyurl.com/qdluuqj)
2. Bellare M., Neven G.: Multi–Signatures in the Plain Public–Key Model and a General Forking Lemma. In Proc. of the 13th ACM conference on Computer and communications security, pp. 390–399, ACM, 2006.
3. Bellare M., Rogaway P.: Entity Authentication and Key Distribution. In Proc. of Crypto 93, LNCS, Vol. 773, pp. 232–249, Springer–Verlag, 1993.
4. Boyd C., Cremers C., Feltz M., Paterson K. G., Poettering B., Stebila, D.: ASICS: Authenticated key exchange security incorporating certification systems. In Computer Security—ESORICS 2013, pp. 381–399. Springer Berlin Heidelberg, 2013.
5. 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.
6. Canetti R., Krawczyk H.: Analysis of Key–Exchange Protocols and Their Use for Building Secure Channels. In Proc. of Eurocrypt 01, LNCS, Vol. 2045, pp. 453–474, Springer–Verlag, 2001.
7. Chalkias, K., Baldimtsi, F., Hristu–Varsakelis, D., Stephanides, G.: Two types of key–compromise impersonation attacks against one–pass key establishment protocols. In e-Business and Telecommunications, pp. 227–238, Springer Berlin Heidelberg, 2009.
8. Cremers C.: Examining Indistinguishability–based Security Models for Key Exchange Protocols: the case of CK, CK–HMQV, and eCK. In Proc. 6th ACM Symposium on Information, Computer and Communications Security, pp. 80–91, ACM, 2011.
9. Cremers C., Feltz M.: Beyond eCK: Perfect Forward Secrecy Under Actor Compromise and Ephemeral–key Reveal. Designs, Codes and Cryptography Vol. 74, Issue 1, pp. 183–218, Springer, 2013.
10. Cullinan, J., Hajir, F.: Primes of prescribed congruence class in short intervals. Integers, Vol. 12, A56, De Gruyter, 2012.
11. Ellison W., Ellison F.: Prime Numbers, Wiley and Hermann editions, 1985.
12. Gopalakrishnan K., Thériault N., Yao C. Z.: Solving Discrete Logarithms from Partial Knowledge of the Key. LNCS, Vol. 4859, pp. 224–237, Springer, 2007.
13. Gordon D. M.: Discrete logarithms in GF(P) using the number field sieve. SIAM Journal on Discrete Mathematics, Vol. 6(1), pp. 124–138, SIAM, 1993.
14. 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'09. 2009.
15. Huq N.: PoS RAM Scraper Malware: Past, Present, and Future, A Trend Micro Research Paper, 2014. http://tinyurl.com/jcwc8wz
16. Krawczyk H.: HMQV: A Hight Performance Secure Diffie–Hellman Protocol. Cryptology ePrint Archive, Report 2005/176, 2005.

17. Krawczyk H.: HMQV: A Hight Performance Secure Diffie–Hellman Protocol. Advances in Cryptology – Crypto 2005, LNCS, Vol. 3621, pp. 546–566, Springer, 2005.
18. Krawczyk H.: HMQV in IEEE P1363. Submission to the IEEE P1363 working group, July 2006. Available at http://tinyurl.com/opjqknd.
19. Kumar S., Paar C., Pelzl J., Pfeiffer G., Rupp A., and 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.
20. LaMacchia B., Lauter K., Mityagin A.: Stronger Security of Authenticated Key Exchange. In Proc. of ProvSec 2007, LNCS, Vol. 4784, pp. 1–16, Springer, 2007.
21. Law L., Menezes A., Qu M., Solinas J., Vanstone S.: An efficient Protocol for Authenticated Key Agreement. Designs, Codes and Cryptography, Vol. 28, pp. 119–134, Springer, 2003.
22. Liu S. Sakurai K., Weng J., Zhang F., Zhao Y.: Security Model and Analysis of FHMQV, Revisited. In Proc. of Information Security and Cryptology – Inscrypt 2013, LNCS, Vol. 8567, pp. 255–269, Springer, 2014.
23. Menezes A.: Another Look at HMQV. Journal of Mathematical Cryptology, Vol. 1, Issue 1, pp. 47–64, De Gruyter, 2007.
24. Menezes A.: Another Look at HMQV. Cryptology ePrint Archive: Report 2005/205.
25. Menezes A., Ustaoglu B.: On the Importance of Public–Key Validation in the MQV and HMQV Key Agreement Protocols. In Proc. of Indocrypt 2006, LNCS, Vol. 4329, pp. 133-147, Springer, 2006.
26. Odlyzko A. M.: Discrete logarithms in finite fields and their cryptographic significance. Advances In Cryptology, LNCS, Vol. 209, pp. 224-314, Springer, 2000.
27. Sarr A. P., Elbaz–Vincent Ph., Bajard J. C.: A Secure and Efficient Authenticated Diffie–Hellman Protocol. In Proc. of Public Key Infrastructures, Services and Applications – EuroPKI 2009, LNCS, Vol. 6391, pp. 83-98, Springer, 2010.
28. Sarr A. P., Elbaz–Vincent Ph., Bajard J. C.: A Secure and Efficient Authenticated Diffie–Hellman Protocol. Cryptology ePrint Archive: Report 2009/408.
29. Sarr A. P., Elbaz–Vincent Ph., Bajard J. C.: A New Security Model for Authenticated Key Agreement. In Proc. of the 7th International Conference on Security and Cryptography for Networks — SCN 2010, LNCS, Vol. 6280, pp. 219–234, Springer–Verlag, 2010.
30. Schirokauer O.: Using number fields to compute logarithms in finite fields. Mathematics of Computations, Vol. 69(231), pp. 1267–1283, AMS, 2000.
31. Thomé E.: Théorie algorithmique des nombres et applications à la cryptanalyse de primitives cryptographiques. Habilitation to conduct research. Université de Lorraine, 2012. 218 pp. https://hal.inria.fr/tel-00765982
32. Trend Labs Security Intelligence Blog: RawPOS Technical Brief, April 2015. http://tinyurl.com/joyazja
33. VISA Data Security Alert: Debugging Software Memory–Parsing Vulnerability, 2008. http://tinyurl.com/joyazja
34. VISA Data Security Alert: Targeted Hospitality Sector Vulnerabilities, 2009. http://tinyurl.com/nnpsl3a
35. VISA Data Security Alert: Retail Merchants Targeted by Memory–Parsing Malware, 2013. http://tinyurl.com/j3duvlg