A Secure and Efficient Authenticated Diffie–Hellman Protocol

Augustin P. Sarr[1], Philippe Elbaz–Vincent[2], and Jean–Claude Bajard[3]

[1]Netheos R&D
[2]Institut Fourier – CNRS, Université Grenoble 1
[1,3]LIRMM – CNRS, Université Montpellier 2
[1]I3M – CNRS, Université Montpellier 2

August 26, 2009

Abstract. The Exponential Challenge Response (XRC) and Dual Exponential Challenge Response (DCR) signature schemes are the building blocks of the HMQV protocol. We propose a complementary analysis of these schemes; on the basis of this analysis we show how impersonation and man in the middle attacks can be mounted against the HMQV protocol, when some session specific information leakages happen. We define the Full Exponential Challenge Response (FXRC) and Full Dual Exponential Challenge Response (FDCR) signature schemes; using these schemes we propose the Fully Hashed MQV protocol (with security arguments), which preserves the remarkable performance of the (H)MQV protocol and resists the attacks we present.

1 Introduction

Implicitly authenticated key exchange protocols have gained wide acceptance; in addition to providing implicit authentication, these protocols are usually more efficient than the explicitly authenticated ones. The HMQV protocol [11], inspired by the famous MQV protocol [13, 1, 2, 9, 10, 19, 8], was proposed with security arguments in the Canetti–Krawczyk model [5]. HMQV was designed in accord with the principle that "a good security system is not one that denies the possibility of failures but rather one designed to confine the adverse effects of such failures to the possible minimum" [11]. Session secret leakages may happen; in that case the exposed session may be compromised, but this should have no effect on the security of any other unexposed session.

In this paper, we propose a complementary analysis of the Exponential Challenge Response (XCR) and Dual Exponential Challenge Response (DCR) signature schemes. On the basis of this analysis we show how impersonation and man in the middle attacks can be done against HMQV when some session specific information leakages happen. We propose the Full Exponential Challenge Response (FXRC) and Full Dual Exponential Challenge Response (FDCR) signature schemes. With these schemes we define the Fully Hashed MQV protocol (with security arguments), which resists the attacks we present and preserves the remarkable performance of the (H)MQV protocol.

This paper is organized as follows. In section 2 we analyze some aspects of the XCR and DCR signatures schemes; we show how session specific information leakages can be used for impersonation and man in the middle attacks against (H)MQV. In section 3 we define a Canetti–Krawczyk type security model [5, 12] for (H)MQV type protocols. In section 4, we propose the FXCR and FDCR signature schemes; and using these schemes we propose the FHMQV protocol. Section 5 deals with security arguments for the FHMQV protocol; in section 6 we present the FHMQV–C protocol (the 'C' stands for *key confirmation*) , which provides additional security attributes, namely key confirmation and perfect forward secrecy. We conclude in section 7.

The following notations are used in this paper: $\mathcal{G}$ is a multiplicatively written cyclic group of prime order q generated by G, $|q|$ is the bit length of q. The identity element in $\mathcal{G}$ is denoted

$\bar{1}$, and $\mathcal{G}^*$ is the set of non–identity elements in $\mathcal{G}$; all public keys are supposed to belong to $\mathcal{G}^*$. For a group element $X \in \mathcal{G}$, the lowercase x denotes the discrete logarithm of X in base G. The identity of an entity with public key A (and private key a) is denoted $\hat{A}$ ($\hat{A}$ is supposed to contain A, or sufficient information to learn A); For two identities $\hat{A} \neq \hat{B}$, we suppose that there is no substring of $\hat{A}$ which equals $\hat{B}$, and conversely. H is a λ–bit hash function where λ is the length of the desired session key, and $\bar{H}$ is a l–bit hash function where $l = (\lfloor \log_2 q \rfloor + 1)/2$ (see [11] for a discussion on the value of l); the concatenation of n strings $s_1, \ldots, s_n$ is denoted $(s_1, \ldots, s_n)$. The symbol "$\in_R$" stands for "chosen uniformly at random in." The Computational Diffie–Hellman (CDH) assumption is supposed to hold in $\mathcal{G}$, i.e. given $U = G^u$ and $V = G^v$ with $U, V \in_R \mathcal{G}^*$, computing $CDH(U, V) = G^{uv}$ is infeasible.

2 Complementary Analysis of the HMQV design

We show in this section how session specific information leakages, can be used for impersonation and man in the middle attacks against HMQV. In our description of HMQV, the ephemeral public keys are tested for membership in $\mathcal{G}^*$; while public key validation is voluntarily omitted in [11], the HMQV protocol is known to be insecure if public keys are not correctly validated [17, 16].

2.1 Exploiting Secret Leakage in the XCR and DCR Signature Schemes

Definition 1 (Exponential Challenge–Response signature [11])**.** Let $\hat{B}$ be an entity with valid public key $B \in \mathcal{G}^*$, and $\hat{A}$ a verifier. $\hat{B}$'s signature on message m and challenge X provided by $\hat{A}$ ($X = G^x$, $x \in_R \mathbb{Z}_q^*$ is chosen and kept secret by $\hat{A}$) is $Sig_{\hat{B}}(m, X) = (Y, X^{s_B})$, where $Y = G^y$, $y \in_R \mathbb{Z}_q^*$ is chosen by $\hat{B}$, and $s_B = y + \bar{H}(Y, m)b$. The verifier $\hat{A}$ accepts a pair (Y, σ_B) provided by $\hat{B}$ as a valid signature if $Y \in \mathcal{G}^*$ and $(YB^e)^x = \sigma_B$, where $e = \bar{H}(Y, m)$.

In this scheme, the information s_B "allows" an attacker to generate valid signatures. Indeed, given the s_B, "corresponding" to some message m and some Y, one can generate a valid signature for any message–challenge pair (m, X_1) (X_1 is a new challenge and the message is unchanged). In the (H)MQV[1] protocol, the identity of the entity $\hat{B}$ stands for $\hat{A}$'s message to $\hat{B}$, and thus does not change from one session (between $\hat{A}$ and $\hat{B}$) to another; hence this can be exploited when s_B leakage happens. This is the basic idea we formalize and use hereafter for impersonation and man in the middle attacks.

Proposition 1. *Let $\hat{B}$ be an entity, with public key $B \in \mathcal{G}^*$, signing a message–challenge pair (m, X). If we learn the β most significant bits of s_B, then we can generate valid signatures with respect to $\hat{B}$'s public key, on any message–challenge pair (m, X_1) (the message is unchanged); this requires $\mathcal{O}(2^{\frac{|q|-\beta}{2}})$ time complexity and $\mathcal{O}(2^{\frac{|q|-\beta}{2}})$ space complexity.*

Proof. From Shank's baby step/giant step lemma [24], given $\sigma_B = X^{s_B}$, $\sigma_B, X \in \langle G \rangle$, and the β most significant bits of s_B, we can compute s_B in $\mathcal{O}(2^{\frac{|q|-\beta}{2}})$ time complexity and $\mathcal{O}(2^{\frac{|q|-\beta}{2}})$ space complexity. Hence given a message–challenge (m, X_1), we replay Y (since Y is chosen by the signer) and output $(Y, X_1^{s_B})$ as signature with respect to $\hat{B}$'s public key. The output signature is valid, thus Proposition 1 holds. □

Shanks method is deterministic, but requires a large storage; using the Pollard's Kangaroo method [23, 24] one can obtain s_B with negligible storage, in probabilistic run time $\mathcal{O}(2^{\frac{|q|-\beta}{2}})$.

Definition 2 (Dual XCR signature [11])**.** Let $\hat{A}$ and $\hat{B}$ be two entities with valid public keys $A, B \in \mathcal{G}$; and m_1, m_2 two messages. The Dual XCR (DCR) signature of $\hat{A}$ and $\hat{B}$ on m_1, m_2 is

$$DSig_{\hat{A},\hat{B}}(m_1, m_2, X, Y) = G^{(x+da)(y+eb)},$$

where $X = G^x \in_R \mathcal{G}^*$ is chosen by $\hat{A}$ (resp. $Y = G^y \in_R \mathcal{G}^*$ is chosen by $\hat{B}$), $d = \bar{H}(X, m_1)$, and $e = \bar{H}(Y, m_2)$.

[1] When regarded through XCR schemes, the XCR variant corresponding to MQV does not use the message in the computation of s_B ($s_B = y + \bar{Y}b$, with $\bar{Y} = 2^l + (\dot{Y} \bmod 2^l)$ where $\dot{Y}$ is the integer representation of Y), and thus can be analyzed as if it takes a constant message.

Once $\hat{A}$ and $\hat{B}$ have exchanged their respective message–challenge pairs (m_1, X) and (m_2, Y), they can both compute the same DCR signature $\sigma_A = (YB^e)^{x+da} = (XA^d)^{y+eb} = \sigma_B$. Notice that the DCR signature of $\hat{A}$ and $\hat{B}$ on messages m_1, m_2 is an XCR of $\hat{A}$ on message m_1 and challenge YB^e.

Proposition 2. *Let $\hat{A}$ and $\hat{B}$ be two entities, with public keys $A, B \in \mathcal{G}^*$, signing the messages m_1, m_2, with challenges X, Y. If we learn the β most significant bits of $s_A = x + da$ ($d = \bar{H}(X, m_1)$), then for any message m_2', and any challenge Y' from $\hat{B}$, we can compute a valid DCR of $\hat{A}$ and $\hat{B}$ on messages m_1, m_2' and challenges X, Y'; this requires $\mathcal{O}(2^{\frac{|q|-\beta}{2}})$ time complexity and $\mathcal{O}(2^{\frac{|q|-\beta}{2}})$ space complexity.*

Proof. Since, the DCR signature of $\hat{A}$ and $\hat{B}$ on (m_1, m_2'), is also a XCR signature of $\hat{A}$ on challenge $Y'B^e$ and message m_1, Proposition 1 implies the result. □

To meet the two–and–half exponentiations per party performance in the DCR scheme, s_A (and s_B) have to be computed an the exponentiation $(YB^e)^{s_A}$ has to be done, and then s_A (or s_B) leakage may happen independently of the ephemeral private keys.

2.2 Exploiting Session Specific Secret Leakages in the HMQV Protocol

A HMQV key exchange between two entities $\hat{A}$ and $\hat{B}$ is done as in Protocol 1; if any verification fails, the execution aborts.

Protocol 1 HMQV key exchange

I) $\hat{A}$ chooses $x \in_R \mathbb{Z}_q^*$, computes $X = G^x$ and sends $(\hat{A}, \hat{B}, X)$ to $\hat{B}$.
II) $\hat{B}$ does the following things:
  a) Verify that $X \in \mathcal{G}^*$.
  b) Choose $y \in_R \mathbb{Z}_q^*$ and compute $Y = G^y$.
  c) Send $(\hat{B}, \hat{A}, Y)$ to $\hat{A}$.
  d) Compute $d = \bar{H}(X, \hat{B})$, $e = \bar{H}(Y, \hat{A})$, $s_B = (y + eb) \bmod q$, $\sigma_B = (XA^d)^{s_B}$, and $K = H(\sigma_B)$.
III) $\hat{A}$ does the following things:
  a) Verify that $Y \in \mathcal{G}^*$.
  b) Compute $d = \bar{H}(X, \hat{B})$, $e = \bar{H}(Y, \hat{A})$, $s_A = (x + da) \bmod q$, $\sigma_A = (YB^e)^{s_A}$, and $K = H(\sigma_A)$.
IV) The shared session key is K.

Roughly speaking, the secret shared between $\hat{A}$ and $\hat{B}$ is a DCR signature with messages fixed to $\hat{A}$ and $\hat{B}$. In [11], the XCR scheme is presented as a new variant of the following Schnorr's identification scheme: *(i)* $\hat{B}$ chooses $y \in_R \mathbb{Z}_q^*$ and sends $Y = G^y$ to $\hat{A}$; *(ii)* $\hat{A}$ chooses $e \in_R \mathbb{Z}_q^*$ and sends e to $\hat{B}$; *(iii)* $\hat{B}$ computes $s = y + eb$ and sends s to $\hat{A}$; the verifier $\hat{A}$ accepts s as a valid signature if $Y \in \mathcal{G}^*$ and $G^s = YB^e$. There is however a subtlety: in Schnorr's scheme the random element e, used by $\hat{B}$ when computing s, is *always* provided by the verifier $\hat{A}$; while in the XCR and DCR schemes, when $\hat{A}$'s message m_1 is fixed (to $\hat{B}$ as in all sessions between $\hat{A}$ and $\hat{B}$) the value of e, used when computing s_B, depends only on the ephemeral key Y provided by (the signer) $\hat{B}$. This is precisely what makes replay attacks possible against the XCR and DCR schemes, and the HMQV protocol, when s_A or s_B leakage happens.

2.2.1 Impersonation Attack using Session Specific Secret Leakage

We show here how s_A (or s_B) leakage can be used for impersonation attack[2]. The following definition gives a broader view of the points needed for impersonation attack; these points are recalled to make the analysis reading easier.

[2]This attack is also reported in [3] (Appendix C); but we submitted (on February 9th, 2009) at WCC 2009 (http://wcc2009.org/) a paper (paper #1569187679) which describes this attack, therefore before [3] was published at http://eprint.iacr.org/2009/079 (on February 12th, 2009).

Definition 3 (Point for impersonation attack, i–point)**.** Let $\hat{A}$ and $\hat{B}$ be two entities with respective public keys A, $B \in \mathcal{G}^*$. A group element $R \in \mathcal{G}^*$ is said to be a HMQV i–point for $\hat{A}$ to $\hat{B}$ if there exists some $k \in \mathbb{Z}_q^*$ such that $R = G^k A^{-\bar{H}(R,\hat{B})}$; k is said to be the decomposition.

Proposition 3. *Let $\mathcal{G} = \langle G \rangle$ be a group with prime order q, $\hat{A}$ and $\hat{B}$ two entities with respective public keys A, $B \in \mathcal{G}^*$. There exists at least $q - (2^l + 1)$ HMQV i–points for $\hat{A}$ to $\hat{B}$.*

Proof. Let $\overline{\mathcal{G}}$ be the image of $\mathcal{G}$ through $(R \xrightarrow{\bar{H}} \bar{R} = \bar{H}(R, \hat{B}))$. Since $\bar{H}$ is a l–bit hash function, the cardinal of $\overline{\mathcal{G}}$ is $\leqslant 2^l$. For every $\bar{Y} \in \overline{\mathcal{G}}$ there is at most one element $R_0 \in \mathcal{G}$ such that $\bar{R}_0 = \bar{Y}$ and $R_0 A^{\bar{R}_0} = \bar{1}$; since the existence of another element $R_0' \in \mathcal{G}$, which satisfies $\bar{R}_0' = \bar{Y}$ and $R_0' A^{\bar{R}_0'} = \bar{1}$, would imply $R_0 A^{\bar{Y}} = R_0' A^{\bar{Y}}$ i.e. $R_0' = R_0$.

Let $\mathcal{R}_0$ be the set of such R_0 points. The cardinal of $\mathcal{R}_0$ is at most 2^l. Every element $R \in \mathcal{G}^* \setminus \mathcal{R}_0$ is a HMQV i–point for $\hat{A}$ to $\hat{B}$. Indeed for a such element R, $RA^{\bar{R}} \neq \bar{1}$ and since R and A are in $\mathcal{G}^*$, there exists some $k \in \mathbb{Z}_q^*$ such that $RA^{\bar{R}} = G^k$, or equivalently $R = G^k A^{-\bar{R}}$. Hence there is at least $q - (2^l + 1)$ HMQV i–points for $\hat{A}$ to $\hat{B}$. □

The following proposition links the decomposition of an i–point to impersonation attack. Notice that the important aspect is knowing the decomposition of an i–point.

Proposition 4. *Let $\hat{A}$ and $\hat{B}$ be two entities with respective public keys A, $B \in \mathcal{G}^*$. Given a HMQV i–point for $\hat{A}$ to $\hat{B}$ X' and its decomposition k, we can impersonate $\hat{A}$ to $\hat{B}$ with no more computations than needed by a HMQV execution.*

Algorithm 2 HMQV impersonation of $\hat{A}$ to $\hat{B}$

Require: A HMQV i–point for $\hat{A}$ to $\hat{B}$ X' and its decomposition k.
1) Send $(\hat{A}, \hat{B}, X')$ to $\hat{B}$.
2) Intercept $(\hat{B}, \hat{A}, Y)$ and do the following things:
   a) Verify that $Y \in \mathcal{G}^*$.
   b) Compute $\sigma_A = \left(YB^e\right)^k$ where $e = \bar{H}(Y, \hat{A})$, and $K = H(\sigma_A)$.
3) Use K as session key with $\hat{B}$.

Proof of Proposition 4. Consider Algorithm 2. This algorithm does not need more computations than a HMQV execution. Since X' is a valid public key, $\hat{B}$'s verification at step IIa of Protocol 1 does not fail. Hence $\hat{B}$ sends $(\hat{B}, \hat{A}, Y)$ at step IIc of Protocol 1. The value of σ_B that $\hat{B}$ uses at step IId of Protocol 1 is

$$\sigma_B = \left(X' A^{H(X',\hat{B})}\right)^{s_B} = \left(G^k A^{-H(X',\hat{B})} A^{H(X',\hat{B})}\right)^{s_B} = \left(G^{s_B}\right)^k = \left(YB^e\right)^k, \text{ where } e = \bar{H}(Y, \hat{A}).$$

This is the value we compute at step 2b in Algorithm 2. Thus the session key we obtain at 2b) is that $\hat{B}$ obtains at step IId of HMQV. □

A naive approach for decomposed i–point for $\hat{A}$ to $\hat{B}$ search consists in choosing $u \in \{0,1\}^l$ and computing the 2^l points $R_{ku} = G^k A^{-u}$, for $k = 1, \ldots, 2^l$. If the hash function $\bar{H}$ is supposed random $Pr(\bar{H}(R_{ku}, \hat{B}) = u) = 1/2^l$. The number of successes $(R_{ku} : \bar{H}(R_{ku}, \hat{B}) = u)$ in these computations is a binomial random variable with parameters $(2^l, 1/2^l)$; hence these computations lead to a decomposed i–point with a probability of success $Prs = 1 - (1 - 1/2^l)^{2^l} \approx 1 - e^{-1} \approx 0.63 > 1/2$ for l sufficiently large.

In the following proposition, we link partial s_A (resp. s_B) leakage to impersonation attack against $\hat{A}$ (resp. $\hat{B}$).

Proposition 5. *Let $\hat{A}$ be an entity executing the HMQV protocol with some peer $\hat{B}$. If we learn the β most significant bits of s_A, defined at step IIIb of Protocol 1, then: (a) we can recover the session key shared between $\hat{A}$ and $\hat{B}$; (b) we can indefinitely impersonate $\hat{A}$ to $\hat{B}$; this requires $\mathcal{O}(2^{\frac{|q|-\beta}{2}})$ time complexity and $\mathcal{O}(2^{\frac{|q|-\beta}{2}})$ space complexity.*

Proof. The claim a) is immediate from Shank's baby step/giant step lemma and the definition of the session key. If we know s_A, then we know a HMQV i–point for $\hat{A}$ to $\hat{B}$ and its decomposition ($XA^{\bar{H}(X,\hat{B})} = G^{s_A}$ i.e. $X = G^{s_A}A^{-\bar{H}(X,\hat{B})}$); hence Proposition 4 implies the claim b). □

Remark 1. *(i)* For the the MQV(–C) protocol, if we (partly) learn s_A for some session between $\hat{A}$ and $\hat{B}$, we can not only impersonate $\hat{A}$ to $\hat{B}$ (as in HMQV(–C)), but also to *any* other entity. A weaker form of this attack was proposed in [14]. *(ii)* To meet the two–and–half exponentiations per party performance, which partly makes the attractiveness of the (H)MQV protocol, s_A has to be computed, and the exponentiation $(YB^e)^{s_A}$ has to be done, and then s_A leakage may happen (through side channel attacks for instance), independently of the ephemeral private key x. As shown hereafter, s_A leakage does not probably imply $\hat{A}$'s static private key discloser; while it is not difficult to see that both s_A and x leakages on the *same* session imply $\hat{A}$'s static key discloser. *(iii)* The leakage of consecutive middle part bits of s_A is not discussed, but with tools from [7], the analysis we propose applies in this case with minor modifications.

Proposition 6. *Any algorithm $\mathcal{A}$ which given s_A, X, A and $\hat{B}$, finds $\hat{A}$'s ephemeral private key x or the static one a in $C_{\mathcal{A}}$ time complexity, allows to solve two instances of the DLP in $\mathcal{G}^*$ in $C_{\mathcal{A}} + C_{DLP}$ time complexity, where C_{DLP} is the complexity of solving one instance of the DLP.*

Proof. Suppose an algorithm $\mathcal{A}$ which given s_A, X, A, and $\hat{B}$ finds the ephemeral private key x or the static one a; and let $X', A' \in_R \mathcal{G}^*$, and $S' = X'A'^{\bar{H}(X',\hat{B})}$. We suppose $S' \neq \bar{1}$ (otherwise, solving the DLP for X' suffices to find $\log_G A'$). We find $\log_G S'$ within C_{DLP} time complexity, and use the algorithm $\mathcal{A}$ to find $\log_G X'$ (or $\log_G A'$); and the remaining logarithm is found from the relation relation $\log_G S' = \log_G X' + \bar{H}(X', \hat{B}) \log_G A'$. □

Hence s_A leakage implies (but is not equivalent to) *session key reveal*, and does probably imply neither *static key reveal* nor *ephemeral key reveal*; while it is not difficult to see that *both s_A leakage and ephemeral key reveal* on the *same* session imply the session owner's static key discloser.

2.2.2 Man in the Middle Attack using Session Specific Secret Leakages

We show here how a man in the middle attack can be mounted, using session specific information leakages. If in addition to s_A, we learn s_B in a HMQV execution between $\hat{A}$ and $\hat{B}$, we can mount a man in the middle attack as in Algorithm 3. To make the description easier, we denote $s_A^{(l)}$ and $s_B^{(l)}$ the s_A and s_B that we learn in previous runs of HMQV, $X^{(l)}$ and $Y^{(l)}$ are $\hat{A}$ and $\hat{B}$'s outgoing ephemeral public keys in the sessions in which s_A and s_B leakages happened respectively. Notice that it is not required that $s_A^{(l)}$ and $s_B^{(l)}$ (partial) leakages happen in matching sessions.

Algorithm 3 Man in the middle attack

1) Send ($\hat{A}$, $\hat{B}$, $X^{(l)}$) to $\hat{B}$.
2) Intercept $\hat{B}$'s response to $\hat{A}$ ($\hat{B}$, $\hat{A}$, Y) and send ($\hat{B}$, $\hat{A}$, $Y^{(l)}$) to $\hat{A}$.
3) Intercept $\hat{A}$'s response to $\hat{B}$, ($\hat{A}$, $\hat{B}$, X).
4) Compute $K_A = H((XA^{d_A})^{s_B^{(l)}})$, where $d_A = H(X, \hat{B})$.
5) Compute $K_B = H((YB^{e_B})^{s_A^{(l)}})$, where $e_B = H(Y, \hat{A})$.
6) Use the key K_B to communicate with $\hat{B}$ on behalf of $\hat{A}$.
7) Use the key K_A to communicate with $\hat{A}$ on behalf of $\hat{B}$.

Roughly speaking, in Algorithm 3, we simultaneously impersonate $\hat{A}$ to $\hat{B}$, and $\hat{B}$ to $\hat{A}$. In $\hat{B}$'s belief, $\hat{A}$ initiates a session with him, with $\hat{A}$'s ephemeral public key being $X^{(l)}$; and in $\hat{A}$'s believe, $\hat{B}$ initiates a session with him, with $\hat{B}$'s ephemeral public key being $Y^{(l)}$. Hence the session key that $\hat{A}$ derives is $K_A = H((Y^{(l)}B^{e_A})^{x+d_Aa}) = H((XA^{d_A})^{s_B^{(l)}})$, where $e_A = H(Y^{(l)}, \hat{A})$ and $d_A = H(X, \hat{B})$. This is the K_A we compute in step 4. Similarly, the session key that $\hat{B}$ derives is $K_B = H((YB^{e_B})^{s_A^{(l)}})$ where $e_B = H(Y, \hat{A})$. In Algorithm 3 communications are

initiated by the attacker, but the attack remains possible when communications are initiated by $\hat{A}$ (or $\hat{B}$).

3 Security Model

We define a security model, inspired by the (extended) Canetti–Krawczyk model [5, 12], for (H)MQV type protocols. We aim to a better capture of session specific information leakages. While both s_A and x (resp. s_B and y) leakages on the same session imply the session owner's private key disclosure, resistance to s_A leakage is a desirable security attribute. We propose a security model which takes into account this possible information leakage.

Rationale of the model. Security arguments validity depends on allowed information leakages. The extended Canetti–Krawczyk (eCK) model [12] does not consider separately *session state reveal*; the *ephemeral key* of a session is required to contain all session specific information. When this requirement is fully satisfied, it becomes difficult to simulate consistently information leakages; in practice, the *ephemeral key* is not always defined to contain *all* session specific information (for instance in [26] the *ephemeral key* does not contain the *ephemeral Diffie–Hellman exponent*).

The derivation of a session key generally involves some intermediate results, on which leakages may happen; these intermediate results cannot always be computed, given only the session's ephemeral private key. Hence leakages on these intermediate results are not necessarily captured through *ephemeral key reveal*. In [6], it is shown that the NAXOS protocol [12] (proven secure in the eCK model) cannot meet its security goals if *session state reveal* is allowed; this suggests that *ephemeral key reveal* (as used in [12]) is not sufficient to capture session specific information leakages. One can see also that in the CMQV protocol [26] (proven secure in the eCK model), the leakage of s_A in a session between $\hat{A}$ and $\hat{B}$, allows an attacker to impersonate indefinitely $\hat{A}$ to $\hat{B}$.

In the Canetti–Krawczyk (CK) model [5], session state reveal is allowed. However, it is not always clear, which information in a session can be revealed. Moreover, the ephemeral information that can be available in a session depend on the reached step in the session's tree of computations. Hence to capture precisely the leakage of ephemeral information, computed in a session, one has to consider the session's tree of computations. Elsewhere, there are some attacks which are captured in the eCK model, but not in the CK model [12] (key compromise impersonation attack for instance).

Since security is affected by allowed information leakages; it becomes important to define precisely information leakages In the model we propose, the eCK model is completed with a precise definition of session state reveal; we aim to capture both session state reveal and ephemeral key reveal. It is however difficult to simulate simultaneously and consistently, both ephemeral key reveal and session state(s) reveal. This is the reason why the model we propose follows two stages. In the first, the leakages of the intermediate results in the tree of computations are analyzed; and in the second the leakage of ephemeral private keys is analyzed.

Session. We suppose $n \leqslant \mathcal{P}(|q|)$ (for some polynomial $\mathcal{P}$) parties $\hat{P}_{i,i=1,\ldots,n}$ modeled as probabilistic polynomial time machines, and a certification authority (CA) trusted by all parties. All public keys are supposed to belong to $\mathcal{G}^*$, this corresponds to the fact that the CA is (only) required to verify that public keys are valid ones. Each party has a static public key together with a certificate binding his identity to his public key.

A session is an instance of the protocol run at a party. A session at $\hat{A}$ (with peer $\hat{B}$) can be created with parameter $(\hat{A}, \hat{B})$ or $(\hat{B}, \hat{A}, Y)$; $\hat{A}$ is viewed as initiator if the creation (or activation) parameter is $(\hat{A}, \hat{B})$, otherwise as responder. At session activation, a session state is created to contain the information specific to that session. Each session is identified with a quadruple $(\hat{A}, \hat{B}, X, \star)$, where $\hat{A}$ is the session holder, $\hat{B}$ is the peer, X is the outgoing message, and $\star$ is the incoming message Y if it exists, otherwise a special symbol meaning that an incoming message is not received yet; in that case when $\hat{A}$ receives the ephemeral public key Y, the session identifier is updated to $(\hat{A}, \hat{B}, X, Y)$. Two sessions $(\hat{B}, \hat{A}, Y, X)$ and $(\hat{A}, \hat{B}, X, Y)$ are said to be matching. Notice that the session matching $(\hat{B}, \hat{A}, Y, X)$ can be any session $(\hat{A}, \hat{B}, X, \star)$; since

X and Y are chosen uniformly at random in $\mathcal{G}^*$, a session cannot have (except with negligible probability) more than one matching session.

Since each protocol is singular about (session internal) computations that lead to the session key, session specific information leakage analyze has to consider the results of these computations. Hence, (regarding) the tree of (session internal) computations that lead to the session key may help in identifying session specific information leakages that can happen in practice.

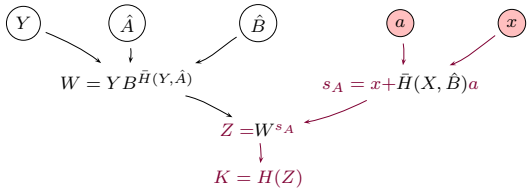

Figure 1: Tree of computations in a HMQV session.

Adversary. The adversary (denoted $\mathcal{A}$) is a probabilistic polynomial time machine. It is a common assumption that an adversary is able to eavesdrop, modify, delete any message sent in a cryptographic protocol, or inject its own messages. This is captured through the assumption that outgoing messages are submitted to $\mathcal{A}$ for delivery ($\mathcal{A}$ decides about messages delivery); $\mathcal{A}$ is also supposed to control session activations at each party $\hat{P}_i$ via the *Send*$(\hat{P}_i, \hat{P}_j)$ and *Send*$(\hat{P}_j, \hat{P}_i, Y)$ queries. To capture information leakages that can happen, the adversary is allowed queries from one of the following sets. At the beginning of its run, the adversary adopts one of the sets of queries; it can then perform queries from the selected set (and only those queries).

In **Set 1**, the following queries are allowed.

- *StaticKeyReveal(party)* to obtain the static private key of a party.
- *SessionKeyReveal(session)* to obtain the derived key in a session.
- *SecretExponentReveal(session)* to obtain the exponent $s = x + da$ (or $y + eb$).
- *SecretGroupElementReveal(session)* to obtain the secret $Z = (YB^e)^{x+da}$.
- *EstablishParty(party)* to register a static public key on behalf of a party; from there, the party is supposed totally controlled by $\mathcal{A}$. A party against whom this query is not issued is said *honest*.

We allow a reveal query on any intermediate node which computation requires a secret information. The YB^e and $\bar{H}$ evaluations are not considered, since these computations require no secret information (recall that messages are submitted to $\mathcal{A}$ for delivery, public keys are then known to $\mathcal{A}$); the node da is not considered, since no secret is required to compute d, and a is already considered. (The ephemeral private key x is considered in Set 2; we cannot simulate simultaneously both s_A reveal and ephemeral private key reveal.)

In **Set 2**, the allowed queries are the following; definitions remain unchanged for queries belonging also to Set 1.

- *StaticKeyReveal(party).*
- *SessionKeyReveal(session).*
- *EphemeralKeyReveal(session)* to obtain the ephemeral private key used by the session owner.
- *EstablishParty(party).*

One can notice that the *SecretExponentReveal* query is stronger than *SecretGroupElementReveal*, which is stronger than the *SessionKeyReveal* query; *SecretExponentReveal* and *EphemeralKeyReveal* queries on the *same* session imply the session owner's static private key discloser.

Remark 2. We implicitly assume that the considered protocol has a tree of computations "matching" that of the (H)MQV protocol; otherwise some queries (*SecretExponentReveal* for instance) may become meaningless. Nevertheless, the analysis paradigm is applicable to other protocols. For that, the Set 2 remains unchanged (and if there exists nodes which computations require only the ephemeral DH exponent as secret information, the ephemeral key is defined to contain the information computed in these nodes); in Set 1, the *StaticKeyReveal, SessionKeyReveal,* and *EstablishParty* queries remain considered, and for any node which computation requires a secret information, and which does not require only the ephemeral DH exponent as secret information, a corresponding reveal query is allowed, and when the query is about an

information which is already considered, the node is skipped. Although there may be many trees of computations, it seems reasonable to consider only the most efficient one(s).

To make clear the motivation for the finely grained definition of information reveals, consider the the Station–to–station protocol [8] (chap. 4, pp. 194, Algorithm 4.50). In this protocol, the initiator's signature (step 3.6) is not computed unless the responder provides a valid signature; while the shared secret Z is computed once the peer's ephemeral public key is validated (step 3.2). Hence an ephemeral key reveal query can be done on the initiator's session, once the incoming ephemeral key is validated; but one may legitimately require that reveal queries on information used in the initiator's signature computation cannot be done, before a valid (peer's) signature is provided to the initiator. When the reveal query on *session state* (or *ephemeral key*) is defined to provide all ephemeral information (as in the CK or eCK models), such (legitimate) differentiations cannot be done. The finely grained definition of information leakages addresses such timing issues.

Definition 4. Let *sid* be the identifier of a completed session at an honest party $\hat{A}$, with some honest peer $\hat{B}$, and sid^* the matching session's identifier.

- The session *sid* is said to be *ck–fresh*, if none of the following conditions hold:
  a) $\mathcal{A}$ issues a *SecretExponentReveal* query on *sid* or sid^* (if sid^* exists);
  b) $\mathcal{A}$ issues a *SecretGroupElementReveal* query on *sid* or sid^* (if sid^* exists);
  c) $\mathcal{A}$ issues a *SessionKeyReveal* query on *sid* or sid^* (if sid^* exists);
  d) sid^* does not exist and $\mathcal{A}$ makes a *StaticKeyReveal* query on $\hat{B}$.
- And *sid* is said to be *eck–fresh*, if none of the following conditions hold:
  a) $\mathcal{A}$ issues a *SessionKeyReveal* query on *sid* or sid^* (if sid^* exists);
  b) $\mathcal{A}$ issues a *StaticKeyReveal* query on $\hat{A}$ and an *EphemeralKeyReveal* query on *sid*;
  c) sid^* exists and $\mathcal{A}$ makes a *StaticKeyReveal* query on $\hat{B}$ and an *EphemeralKeyReveal* query on sid^*;
  d) sid^* does not exist and $\mathcal{A}$ makes a *StaticKeyReveal* query on $\hat{B}$.

With these definitions of freshness, we define a secure Diffie–Hellman protocol.

Definition 5. Let Π be a Diffie–Hellman key agreement protocol, such that if two honest parties complete matching sessions, then they both compute the same session key.

- The protocol Π is said to be *ck–secure* in $\mathcal{G}$, if no polynomially bounded adversary (performing queries from the Set 1) can distinguish a ck–fresh session key, from a random value chosen under the distribution of session keys, with probability (taken over the random coins of the adversary and the choice of static and ephemeral public keys in $\mathcal{G}$) significantly greater than 1/2.
- Π is said to be *eck–secure* in $\mathcal{G}$, if no polynomially bounded adversary (performing queries from the Set 1) can distinguish an eck–fresh session key, from a random value chosen under the distribution of session keys, with probability significantly greater than 1/2.
- And Π is said to be *secure* in $\mathcal{G}$, if it is both *ck–secure* and *eck–secure* in $\mathcal{G}$.

4 A New Authenticated Diffie–Hellman Protocol

In this section, we define the Full Exponential Challenge Response (FXCR) and Full Dual exponential Challenge Response (FDCR) schemes, which confine to the minimum the consequences of s_A leakage; and we provide security arguments for these schemes. With these schemes, we define the Fully Hashed MQV (FHMQV) protocol, which preserves the performance of the (H)MQV protocol, in addition to resistance to the attacks we present in section 2.

4.1 Full Exponential Challenge Response Signature scheme

Definition 6 (FXCR signature scheme)**.** Let $\hat{B}$ be an entity with public key $B \in \mathcal{G}^*$, and $\hat{A}$ a verifier. $\hat{B}$'s signature on message m and challenge X provided by $\hat{A}$ ($X = G^x$, $x \in_R \mathbb{Z}_q^*$ is chosen and kept secret by $\hat{A}$) is $FSig_{\hat{B}}(m, X) = (Y, X^{s_B})$, where $Y = G^y$, $y \in_R \mathbb{Z}_q^*$ is chosen by $\hat{B}$, and $s_B = y + \bar{H}(Y, X, m)b$; the verifier $\hat{A}$ accepts the pair (Y, σ_B) as a valid signature if $Y \in \mathcal{G}^*$ and $(YB^{\bar{H}(Y,X,m)})^x = \sigma_B$.

The FXCR scheme delivers all the security attributes of the XCR scheme; in addition the "replay attack" we present in section 2 does not hold anymore. Indeed, suppose an attacker who has learned ${s_B}^{(l)} = y^{(l)} + \bar{H}(Y^{(l)}, X^{(l)}, m)b$. When he is provided with a new challenge X (chosen at random) and the same message m, except with negligible probability $X \neq X^{(l)}$ (and $\bar{H}(Y^{(l)}, X^{(l)}, m) \neq \bar{H}(Y^{(l)}, X, m)$). Hence, to replay $Y^{(l)}$ on the message–challenge pair (m, X), the attacker has to find $s_B = y^{(l)} + \bar{H}(Y^{(l)}, X, m)b$; it is not difficult to see that if he can compute s_B from ${s_B}^{(l)}$, then he can find b from s_B. Proposition 6 shows that this cannot probably happen.

Definition 7 (FXCR signature scheme security)**.** The FXCR scheme is said to be secure in $\mathcal{G}$, if given a public key B, a challenge X_0 ($B, X_0 \in_R \mathcal{G}^*$), and hashing and signing oracles, no probabilistic polynomial time attacker, performing queries adaptively, can output with non negligible success probability a triple (m_0, Y_0, σ_0) such that:

- (Y_0, σ_0) is a valid signature with respect to the public key B, and the message–challenge pair (m_0, X_0);
- (Y_0, σ_0) was not obtained from the signing oracle with a query on (m_0, X_0) (freshness).

Using the "oracle replay" technique [21, 22] (and following the approach of [11]), we show that the FXCR scheme is secure in the sense of definition 7. Recall that a function $\mathcal{F}$ with parameter ζ is said to be negligible, if for every polynomial $\mathcal{P}$, and every sufficiently large ζ, $\mathcal{F}(\zeta) < (|\mathcal{P}(\zeta)|)^{-1}$; otherwise $\mathcal{F}$ is said non–negligible.

Proposition 7. *Under the CDH assumption in $\mathcal{G}$ and the RO model, the FXCR signature scheme is secure in the sense of definition 7.*

Proof. Suppose a probabilistic polynomial time attacker $\mathcal{A}$, which given $B, X_0 \in_R \mathcal{G}^*$ succeeds with non–negligible probability in forging a fresh and valid signature, with respect to the public key B and challenge X_0. Let Q_h and Q_s be respectively the number of queries that $\mathcal{A}$ asks to the hashing and signing oracles. Using $\mathcal{A}$ we build a polynomial time CDH solver $\mathcal{S}$ which succeeds with non–negligible probability. The solver $\mathcal{S}$ provides $\mathcal{A}$ with random coins, and simulates the digest and signature queries. The interactions between $\mathcal{S}$ and $\mathcal{A}$ are detailed in Figure 4.

Figure 4 CDH solver from $\mathcal{A}$

Run of $\mathcal{A}$:

a) At $\mathcal{A}$'s digest query on (Y, X, m), $\mathcal{S}$ responds as follows:
- if a value is already assigned to $\bar{H}(Y, X, m)$, $\mathcal{S}$ returns the value of $\bar{H}(Y, X, m)$;
- otherwise $\mathcal{S}$ responds with $e \in_R \{0,1\}^l$, and sets $\bar{H}(Y, X, m) = e$.

b) At $\mathcal{A}$'s signature query on (m, X), $\mathcal{S}$ responds as follows:
- $\mathcal{S}$ chooses $s_B \in_R \mathbb{Z}_q^*$, $e \in_R \{0,1\}^l$, sets $Y = G^{s_B} B^{-e}$ and $\bar{H}(Y, X, m) = e$. If $\bar{H}(Y, X, m)$ was previously defined, $\mathcal{S}$ aborts;
- $\mathcal{S}$ responds with (Y, X^{s_B}, s_B) (notice that the forger is given s_B in addition to X^{s_B}).

c) At $\mathcal{A}$'s halt, $\mathcal{S}$ verifies that $\mathcal{A}$'s output $(Y_0, X_0, m_0, \sigma_0)$ (if any) satisfies the following conditions. If one of these conditions is not satisfied $\mathcal{S}$ aborts.
- $Y_0 \in \mathcal{G}^*$ and $\bar{H}(Y_0, X_0, m_0)$ was queried 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 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 on $(Y_0, X_0, m_0, \sigma_0)$ satisfying conditions of step c, with a hash value $\bar{H}(Y_0, X_0, m_0)_2 = e_0^{(2)} \neq e_0^{(1)} = \bar{H}(Y_0, X_0, m_0)_1$, then $\mathcal{S}$ outputs $(\sigma_0^{(1)} / \sigma_0^{(2)})^{(e_0^{(1)} - e_0^{(2)})^{-1}}$ as a guess for $CDH(B, X_0)$.

Under the RO model, the distribution of simulated signatures is indistinguishable from the distribution of real signatures generated by $\hat{B}$, except the deviation that happens when $\bar{H}(Y, X, m)$ was queried before. Since the number of queries to the oracles is less than $(Q_h + Q_s)$, and Y is

chosen uniformly at random in $\mathcal{G}$, this deviation happens with probability less than $(Q_h+Q_s)/q$, which is negligible. Hence this simulation is perfect, except with negligible probability. Moreover the probability of outputting a valid forgery without querying $\bar{H}(Y_0, X_0, m_0)$ is 2^{-l}. Thus under this simulation, $\mathcal{A}$ outputs with non–negligible probability a valid and fresh forgery $(Y_0, X_0, m_0, \sigma_0^{(1)})$; we denote $\bar{H}(Y_0, X_0, m_0)$ by $e_0^{(1)}$.

From the forking lemma [21, 22], the repeat experiment outputs with non–negligible probability a valid and fresh signature $(Y_0, X_0, m_0, \sigma_0^{(2)})$ with a digest $e_0^{(2)}$, which with probability $1 - 2^{-l}$, is different from $e_0^{(1)}$. Then the computation

$$\left(\frac{\sigma_0^{(1)}}{\sigma_0^{(2)}}\right)^{\left(e_0^{(1)}-e_0^{(2)}\right)^{-1}} = \left(\frac{\left(Y_0 B^{e_0^{(1)}}\right)^{x_0}}{\left(Y_0 B^{e_0^{(2)}}\right)^{x_0}}\right)^{\left(e_0^{(1)}-e_0^{(2)}\right)^{-1}} = B^{x_0}$$

gives $CDH(B, X_0)$. Recall that such a polynomial CDH solver, succeeding with non–negligible probability, can be transformed into an efficient CDH solver [15]. □

NOTE: The protocol that follows naturally from the FXCR signature scheme is the following; however because of lack of space, the analysis of this protocol will be done in a forthcoming paper.

A) $\hat{A}$ *chooses* $x \in_R \mathbb{Z}_q^*$, *and sends* $(\hat{A}, \hat{B}, X = G^x)$ *to* $\hat{B}$.

B) $\hat{B}$ *chooses* $y \in_R \mathbb{Z}_q^*$, *and sends* $(\hat{B}, \hat{A}, Y = G^y)$ *to* $\hat{A}$.

C) $\hat{A}$ *(resp.* $\hat{B}$*) verifies that* Y *(resp.* X*)* $\in \mathcal{G}^*$, *and computes the shared session key* $K_A = H((YB^e)^a, B^{x+da}, \hat{A}, \hat{B}, X, Y) = H(A^{y+eb}, (XA^d)^b, \hat{A}, \hat{B}, X, Y) = K_B$, *where* $d = \bar{H}(X, Y, \hat{A}, \hat{B})$ *and* $e = \bar{H}(Y, X, \hat{A}, \hat{B})$.

4.2 Full Dual Exponential Challenge Response Signature scheme

Definition 8 (FDCR signature scheme)**.** Let $\hat{A}$ and $\hat{B}$ be two entities with public keys $A, B \in \mathcal{G}^*$, and m_1, m_2 two messages. The FDCR signature of $\hat{A}$ and $\hat{B}$ on messages m_1, m_2 is

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

where $X = G^x \in_R \mathcal{G}^*$ is chosen by $\hat{A}$ (resp. $Y = G^y \in_R \mathcal{G}^*$ is chosen by $\hat{B}$), $d = \bar{H}(X, Y, m_1, m_2)$, and $e = \bar{H}(Y, X, m_1, m_2)$.

In the FDCR scheme, as in the DCR scheme, once $\hat{A}$ and $\hat{B}$ have provided their respective message–challenge pairs, they can both compute the same signature; however 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, Y, is not a FXCR signature of $\hat{A}$ on m_1 and YB^e.

Definition 9 (Security of the FDCR scheme)**.** Let $A = G^a, B, X_0 \in_R \mathcal{G}^*$ $(A \neq B)$. The FDCR scheme is said to be secure in $\mathcal{G}$, if given a, A, B, X_0, and a message m_{1_0}, and hashing and signing oracles, no probabilistic polynomial time attacker (performing queries adaptively), can output with non negligible success probability a triple (m_{2_0}, Y_0, σ_0) such that:

- $(m_{1_0}, m_{2_0}, X_0, Y_0, \sigma_0)$ is a valid FDCR signature with respect to the public keys A, B.
- (Y_0, σ_0) was not obtained from the signing oracle with a query on (m_1', X') such that $X_0 = X'$ and $(m_1', m_2') = (m_{1_0}, m_{2_0})$, where (m_{1_0}, m_{2_0}) denotes the concatenation of m_{1_0} and m_{2_0}, m_2' is the message returned at signature query on (m_1', X') (freshness).

Remark 3. Since we suppose that if $\hat{A} \neq \hat{A}'$, no substring of $\hat{A}$ equals $\hat{A}'$ (and conversely), if $\hat{A} \neq \hat{A}'$ or $\hat{B} \neq \hat{B}'$ then $(\hat{A}, \hat{B})$ cannot equals $(\hat{A}', \hat{B}')$.

Proposition 8. *Under the CDH assumption in* $\mathcal{G}$ *and the RO model, the FDCR signature scheme is secure in the sense of definition 9.*

Proof. Suppose an attacker $\mathcal{A}$, which given a, A, B, X_0, m_{1_0} ($A \neq B$) outputs with non–negligible success probability a valid and fresh signature forgery (m_{2_0}, Y_0, σ_0). Using $\mathcal{A}$ we build a polynomial time machine $\mathcal{S}$ which succeeds with non–negligible probability in forging a FXCR signature. The forger $\mathcal{S}$ provides $\mathcal{A}$ with random coins, a, A, B, X_0, m_{1_0}, and simulates the role of $\hat{B}$.

i) At $\mathcal{A}$'s digest query on (X, Y, m_1, m_2), $\mathcal{S}$ responds as follows:
- if a value is already assigned to $\bar{H}(X, Y, m_1, m_2)$, $\mathcal{S}$ returns the value of $\bar{H}(X, Y, m_1, m_2)$;
- otherwise $\mathcal{S}$ responds with $d \in_R \{0,1\}^l$, and sets $\bar{H}(X, Y, m_1, m_2) = d$.

ii) At signature query on (m_1, X), $\mathcal{S}$ responds as follows.
- $\mathcal{S}$ chooses[3] $m_2 \in \{0,1\}^*$, $s_B \in_R \mathbb{Z}_q^*$, $d, e \in_R \{0,1\}^l$, computes $Y = G^{s_B} B^{-e}$, and sets $\bar{H}(X, Y, m_1, m_2) = d$, $\bar{H}(Y, X, m_1, m_2) = e$; if $\bar{H}(X, Y, m_1, m_2)$ or $\bar{H}(Y, X, m_1, m_2)$ was defined in a previous query, $\mathcal{S}$ aborts.
- $\mathcal{S}$ provides $\mathcal{A}$ with the signature $((m_2, Y), (XA^d)^{s_B}, s_B)$ (s_B is returned, with the signature).

The simulation of $\hat{B}$'s role is perfect, except with negligible probability. The deviation happens when the same message–challenge pair (m_2, Y) is chosen twice in two signature queries on the same pair (m_1, X). Since Y is chosen uniformly at random in $\mathcal{G}$, this happens with negligible probability. Then if $\mathcal{A}$ succeeds with non–negligible probability in forging a valid and fresh signature σ_0, it succeeds also with non–negligible probability under this simulation. And since $\mathcal{S}$ knows a, using $\mathcal{A}$, it can output with non–negligible success probability

$$\sigma_0 (Y_0 B^e)^{-da} = (Y_0 B^e)^{x_0 + da} (Y_0 B^e)^{-da} = (Y_0 B^e)^{x_0} = X_0{}^{y_0 + eb}.$$

This is valid FXCR forgery on message (m_{1_0}, m_{2_0}) (the concatenation of m_{1_0} and m_{2_0}) and challenge X_0 with respect to the public key B; contradicting Proposition 7. □

4.3 The Fully Hashed MQV Protocol.

We can now derive the Fully Hashed MQV (FHMQV) protocol, which provides all security attributes of the (H)MQV protocol, in addition to resistance to s_A (and s_B) leakage.

Protocol 5 FHMQV key exchange

I) $\hat{A}$ chooses $x \in_R \mathbb{Z}_q^*$, computes $X = G^x$ and sends $(\hat{A}, \hat{B}, X)$ to $\hat{B}$.

II) $\hat{B}$ does the following things:
- *a)* Verify that $X \in \mathcal{G}^*$.
- *b)* Choose $y \in_R \mathbb{Z}_q^*$ and compute $Y = G^y$.
- *c)* Send $(\hat{B}, \hat{A}, Y)$ to $\hat{A}$.
- *d)* Compute $d = \bar{H}(X, Y, \hat{A}, \hat{B})$, $e = \bar{H}(Y, X, \hat{A}, \hat{B})$, $s_B = (y + eb) \bmod q$, $\sigma_B = (XA^d)^{s_B}$, and $K = H(\sigma_B, \hat{A}, \hat{B}, X, Y)$.

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

IV) The shared session key is K.

5 Security Analysis of the FHMQV Protocol

We suppose $n \leqslant \mathcal{P}(|q|)$ (for some polynomial $\mathcal{P}$) parties modeled as probabilistic polynomial time machines. In accord with our security model, the following queries are allowed.
- *Send($\hat{A}$, $\hat{B}$)* which makes $\hat{A}$ perform the step I of Protocol 5.
- *Send($\hat{A}$, $\hat{B}$, X)* which makes $\hat{B}$ perform the step II of Protocol 5.

[3]Notice that the proof remains valid even if the message m_2 is provided by $\mathcal{A}$ at signature query.

- *Send($\hat{A}, \hat{B}, X, Y$)* which makes $\hat{A}$ update the session identifier $(\hat{A}, \hat{B}, X, \star)$ (if any) to $(\hat{A}, \hat{B}, X, Y)$, and perform the step III of Protocol 5; if $\hat{A}$ does not hold a session with identifier $(\hat{A}, \hat{B}, X, \star)$, the call is ignored.

Proposition 9. *Under the CDH assumption in $\mathcal{G}$, and the RO model, the FHMQV protocol is ck–secure.*

5.1 Proof of Proposition 9 (ck–security)

It is immediate from the definition of the FHMQV protocol that, if two honest parties complete matching sessions, then they both compute the same session key. Suppose an adversary $\mathcal{A}$ which succeeds, with probability significantly greater than 1/2, in distinguishing a session key of a ck–fresh session (that we designate by $(\hat{A}, \hat{B}, X_0, Y_0)$ or test session) from random a value chosen under the distribution of session keys. $\mathcal{A}$ can only distinguish a ck–fresh session key from a random value in one of the following ways.

Guessing attack: $\mathcal{A}$ guesses correctly the test session key.

Key replication attack: $\mathcal{A}$ succeeds in making two non–matching sessions have the same session key, it can then query a session key reveal on one of the two sessions and use the other as test session.

Forging attack: $\mathcal{A}$ computes $\sigma_0 = FDSig_{\hat{A},\hat{B}}(\hat{A}, \hat{B}, X_0, Y_0)$ and queries the random oracle H with $(\sigma_0, \hat{A}, \hat{B}, X_0, Y_0)$.

Under the RO model, the probability of guessing correctly the output of the hash function is 2^{-k}; and non–matching sessions cannot have (except with negligible probability) the same session key. It thus remains that if $\mathcal{A}$ succeeds with probability significantly greater than 1/2 in distinguishing a ck–fresh session key, from a random value chosen under the distribution of session keys, then it succeeds with non–negligible probability in forging attack. We thus suppose that $\mathcal{A}$ interacts in an n parties environment, and ends its run with non–negligible probability with an output (sid, σ_0), where sid is a ck–fresh session identifier, and σ_0 a guess of the sid session signature.

Let E denote the event "*$\mathcal{A}$ succeeds in forging the session signature of some ck–fresh session sid.*" The event E divides in: *(a) E.1*: "*$\mathcal{A}$ succeeds in forging the session signature of some ck–fresh session sid, which matching session exits*", and *(b) E.2*: "*$\mathcal{A}$ succeeds in forging the session signature of some ck–fresh session sid, without matching session.*" Since the session *sid* is required to be ck–fresh, in accord with definition 4, the *StaticKeyReveal* query on both $\hat{A}$ and $\hat{B}$ is allowed in *E.1*. In *E.2* the *StaticKeyReveal* query is allowed on $\hat{A}$ (but not on $\hat{B}$). Now since $\mathcal{A}$ succeeds with non–negligible probability, either *E.1* or *E.2* occurs with non–negligible probability. Hence, to prove Proposition 9, it suffices to show that neither *E.1* nor *E.2* can happen with non–negligible probability.

5.1.1 Analysis of *E.1*

Suppose that *E.1* occurs with non–negligible probability. Using $\mathcal{A}$, we build a polynomial time CDH solver which succeeds with non–negligible probability. The solver $\mathcal{S}$ takes as input $X_0, Y_0 \in_R \mathcal{G}^*$ and interacts with $\mathcal{A}$ as described below.

1) $\mathcal{S}$ simulates $\mathcal{A}$'s environment, with n parties $\hat{P}_1, \ldots, \hat{P}_n$; recall that $\mathcal{A}$ is supposed polynomial, we thus suppose that each party is activated at most m times, $m \leqslant \mathcal{P}(|q|)$ for some polynomial $\mathcal{P}$.

2) $\mathcal{S}$ chooses $i, j \in_R \{1, \ldots, n\}$, $i \neq j$ (the case $i = j$, reflection attack, is analyzed in subsection 5.1.3), and $t \in_R \{1, \ldots, m\}$ (with the choice of (i, j, t), $\mathcal{S}$ is guessing the test session, and peers in that session). We refer to $\hat{P}_i$ as $\hat{A}$ and $\hat{P}_j$ as $\hat{B}$. $\mathcal{S}$ assigns to each $\hat{P}_k$ a random static key pair $(p_k, P_k = G^{p_k})$, and answers to $\mathcal{A}$ queries as follows; $\bar{H}$ queries are simulated in the usual way (see the proof of Proposition 8).

3) At $\mathcal{A}$'s $Send(\hat{P}_l, \hat{P}_m)$ query, $\mathcal{S}$ chooses $x \in_R \mathbb{Z}_q^*$, computes $X = G^x$, creates a session state with identifier $(\hat{P}_l, \hat{P}_m, X, \star)$, and provides $\mathcal{A}$ with the outgoing message $(\hat{P}_l, \hat{P}_m, X)$.

4) At $\mathcal{A}$'s $Send(\hat{P}_m, \hat{P}_l, Y)$ $\mathcal{S}$ chooses $x \in_R \mathbb{Z}_q^*$, computes $X = G^x$, creates a session state with identifier $(\hat{P}_l, \hat{P}_m, X, Y)$, provides $\mathcal{A}$ with the outgoing message $(\hat{P}_l, \hat{P}_m, X)$; and completes

the session with identifier $(\hat{P}_l, \hat{P}_m, X, Y)$ as responder (step IId of Protocol 5).

5) At $\mathcal{A}$'s $Send(\hat{P}_l, \hat{P}_m, X, Y)$ query, $\mathcal{S}$ updates the session identifier $(\hat{P}_l, \hat{P}_m, X, \star)$ to $(\hat{P}_l, \hat{P}_m, X, Y)$; and completes the session with identifier $(\hat{P}_l, \hat{P}_m, X, Y)$ as initiator (step IIIb of Protocol 5). (Notice that $\mathcal{S}$ can compute the session FDCR signature of any session different from the t–th session at $\hat{A}$ — and its matching session.)

6) When $\mathcal{A}$ activates the t–th session at $\hat{A}$, if $\hat{A}$'s peer in that session is $\hat{B}$, $\mathcal{S}$ provides $\mathcal{A}$ with the outgoing message $(\hat{A}, \hat{B}, X_0)$; otherwise, $\mathcal{S}$ aborts.

7) When $\mathcal{A}$ activates the session matching t–th session at $\hat{A}$, $\mathcal{S}$ provides $\mathcal{A}$ with $(\hat{B}, \hat{A}, Y_0)$.

8) At $\mathcal{A}$'s digest query on $(\sigma, \hat{P}_l, \hat{P}_m, X, Y)$, $\mathcal{S}$ responds as follows:

- If there exists a completed session with identifier $sid = (\hat{P}_l, \hat{P}_m, X, Y)$ or $sid = (\hat{P}_m, \hat{P}_l, Y, X)$ and with initiator $\hat{P}_l$, and if σ is the sid session's FDCR signature, $\mathcal{S}$ returns the sid session key.
- Else, if the same query was made previously, $\mathcal{S}$ returns the previously returned value.
- Else $\mathcal{S}$ responds with $\pi \in_R \{0,1\}^\lambda$, and sets $H(\sigma, \hat{P}_l, \hat{P}_m, X, Y) = \pi$.

9) If $\mathcal{A}$ issues a *StaticKeyReveal*, *SecretExponentReveal*, *SecretGroupElementReveal*, *SessionKeyReveal*, or *EstablishParty* query, $\mathcal{S}$ answers faithfully.

10) In any of the following situations, $\mathcal{S}$ aborts.

- $\mathcal{A}$ halts with a test session different from the t–th session at $\hat{A}$.
- $\mathcal{A}$ issues a *SecretExponentReveal*, a *SecretGroupElementReveal*, or a *SessionKeyReveal* query on the t–th session at $\hat{A}$ or its matching session.
- $\mathcal{A}$ issues an *EstablishParty* query on $\hat{A}$ or $\hat{B}$.

11) If $\hat{A}$ provides a guess σ_0 of the signature of the t–th session at $\hat{A}$, $\mathcal{S}$ outputs $(\sigma_0)(X_0 A^d)^{-eb} Y_0^{-da}$ as a guess for $CDH(X_0, Y_0)$. Otherwise $\mathcal{S}$ aborts.

Fact. *If $\mathcal{A}$ succeeds with non–negligible probability in E.1, $\mathcal{S}$ outputs with non–negligible probability $CDH(X_0, Y_0)$.*

Proof. The simulation of $\mathcal{A}$'s environment is perfect except with negligible probability; when $\mathcal{A}$ activates the t–th session at $\hat{A}$, the X_0 provided to $\mathcal{A}$ is chosen uniformly at random in $\mathcal{G}^*$, its distribution is the same as that of the real X. The same argument holds for Y_0. The probability of guessing correctly the test session is $(n^2 m)^{-1}$; and if *E.1* happens and $\mathcal{S}$ guesses correctly the test session, $\mathcal{S}$ does not abort. Thus $\mathcal{S}$ succeeds with probability $(n^2 m)^{-1} Pr(E.1)$ where negligible terms are ignored. In addition, when $\mathcal{A}$ outputs a correct guess for the test session, $\mathcal{S}$ outputs

$$(\sigma_0)(X_0 A^d)^{-ea} Y_0^{-da} \;=\; (X_0 A^d)^{y_0 + ae} (X_0 A^d)^{-ea} Y_0^{-da} \;=\; Y_0^{x_0 + da} Y_0^{-da} \;=\; CDH(X_0, Y_0).$$

Moreover if $\mathcal{A}$ is polynomial, $\mathcal{S}$ is also polynomial. This shows that *E.1* cannot happen with non–negligible probability. □

5.1.2 Analysis of *E.2*

If *E.2* occurs with non–negligible probability, using $\mathcal{A}$, we build a polynomial time machine, which forges a valid FDCR signature, with non–negligible success probability. For that, we use the same simulation as in the analysis of *E.1*, with the following modifications.

- $\mathcal{S}$ takes as input $a \in_R \mathbb{Z}_q^*$, and $X_0, B \in_R \mathcal{G}^*$.
- $\hat{A}$'s key pair is set to (a, G^a), and $\hat{B}$'s public key to B ($\hat{B}$'s private key is unknown to $\mathcal{S}$).
- At $\mathcal{A}$'s $Send(\hat{P}_l, \hat{B}, X)$ query, $\mathcal{S}$ answers as follows:
  - $\mathcal{S}$ chooses $s_B \in_R \mathbb{Z}_q^*$, $d, e \in_R \{0,1\}^l$, and sets $Y = G^{s_B} B^{-e}$, $\bar{H}(X, Y, \hat{P}_l, \hat{B}) = d$, and $\bar{H}(Y, X, \hat{P}_l, \hat{B}) = e$; if Y was previously used as outgoing ephemeral key in a session held by $\hat{B}$ with peer $\hat{P}_l$, $\mathcal{S}$ aborts.
  - $\mathcal{S}$ creates a session state with identifier $(\hat{B}, \hat{P}_l, Y, X)$, and provides $\mathcal{A}$ with the message $(\hat{B}, \hat{P}_l, Y)$.
  - $\mathcal{S}$ completes the session $(\hat{B}, \hat{P}_l, Y, X)$ as responder.
- At $\mathcal{A}$'s $Send(\hat{B}, \hat{P}_l)$ query:

- $\mathcal{S}$ chooses $s_B \in_R \mathbb{Z}_q^*$, $d, e \in_R \{0,1\}^l$, and sets[4] $Y = G^{s_B} B^{-e}$, $\bar{H}(\star, Y, \hat{B}, \hat{P}_l) = d$, $\bar{H}(Y, \star, \hat{B}, \hat{P}_l) = e$; if Y was previously used as outgoing ephemeral key in a session at $\hat{B}$ with peer $\hat{P}_l$, $\mathcal{S}$ aborts.
- $\mathcal{S}$ creates a session state with identifier $(\hat{B}, \hat{P}_l, Y, \star)$, and provides $\mathcal{A}$ with the message $(\hat{B}, \hat{P}_l, Y)$.

- If $\mathcal{A}$ activates at $\hat{B}$ a session matching the t–th session at $\hat{A}$, $\mathcal{S}$ aborts.
- If $\mathcal{A}$ issues a *StaticKeyReveal* query on $\hat{B}$, $\mathcal{S}$ aborts.
- If $\hat{A}$ halts with the t–th session at $\hat{A}$, $(\hat{A}, \hat{B}, X_0, Y_0)$ as test session, and with a guess σ_0 of the session's signature, $\mathcal{S}$ outputs the triple $((\hat{A}, \hat{B}), Y_0, \sigma_0)$ as FDCR signature on messages $\hat{A}, \hat{B}$, and challenges X_0, Y_0 with respect to public keys A, B. Otherwise $\mathcal{S}$ aborts.

The simulation of $\mathcal{A}$'s environment is perfect, except with negligible probability. The deviation happens when the same Y is chosen twice as ephemeral key in sessions at $\hat{B}$ with the same peer $\hat{P}_l$, this happens with probability less than m/q, which is negligible. Hence, under this simulation, $\mathcal{A}$ succeeds with non–negligible probability in *E.2*. And when $\mathcal{A}$ outputs a correct forgery, and $\mathcal{S}$ guesses correctly the test session, $\mathcal{S}$ outputs a valid FDCR forgery on messages $\hat{A}$, $\hat{B}$, and challenges X_0, Y_0, with respect to the public keys A and B; contradicting Proposition 8. Hence neither *E.1* nor *E.2*, can happen with non–negligible probability; this shows the Proposition 9.

5.1.3 Resistance to Reflection Attacks

We show here that the FHMQV protocol provides resistance to reflection attacks (for ck–fresh sessions). A session with identifier $sid = (\hat{A}, \hat{A}, X, Y)$ is said to be ck–fresh if none of these conditions hold:

- $\mathcal{A}$ issues a *SecretExponentReveal* query on sid or sid^* (if sid^* exists);
- $\mathcal{A}$ issues a *SecretGroupElementReveal* query on sid or sid^* (if sid^* exists);
- $\mathcal{A}$ issues a *SessionKeyReveal* query on sid or sid^* (if sid^* exists);
- sid^* does not exist and $\mathcal{A}$ makes a *StaticKeyReveal* query on $\hat{A}$.

Guessing and key replication attacks cannot succeed, except with negligible probability. It suffices to show that no polynomially bounded adversary can compute, with non–negligible success probability, the session signature of a ck–fresh session $(\hat{A}, \hat{A}, X_0, Y_0)$.

Let F be the event "*$\mathcal{A}$ succeeds in forging the session signature of some ck–fresh session* $(\hat{A}, \hat{A}, X_0, Y_0)$." The event F divides in *F.1*: "*$\mathcal{A}$ succeeds in forging the session signature of some ck–fresh session* $(\hat{A}, \hat{A}, X_0, Y_0)$*, which matching session exists*", and *F.2*: "*$\mathcal{A}$ succeeds in forging the session signature of some ck–fresh session* $(\hat{A}, \hat{A}, X_0, Y_0)$ *without matching session*." It thus suffices to show that neither *F.1* nor *F.2* can happen with non–negligible probability.

Analysis of *F.1*. If *F.1* happens with non–negligible probability, using $\mathcal{A}$, we build a polynomial time CDH solver which succeeds with non–negligible probability. For that, we use the same simulation as in the analysis of *E.1*, except the following differences.

- $\mathcal{S}$ takes as input a, X_0, Y_0; $a \in_R \mathbb{Z}_q^*$, $X_0, Y_0 \in_R \mathcal{G}^*$.
- $\hat{B}$'s key pair and identity are set to that of $\hat{A}$.

$\mathcal{A}$'s simulated environment remains perfect except with negligible probability; and if $\mathcal{A}$ succeeds with non–negligible probability, in event *F.1*, then under this simulation it outputs with non–negligible probability a valid signature forgery σ_0 on the session $(\hat{A}, \hat{A}, X_0, Y_0)$. And then $\mathcal{S}$ outputs with non–negligible probability $CDH(X_0, Y_0)$ from $\sigma_0, a, d,$ and e. This shows that under the CDH assumption and RO model, *F.1* cannot happen, except with negligible probability.

Analysis of *F.2*. If *F.2* happens with non–negligible probability, using $\mathcal{A}$, we build a polynomial time machine, which given $A = G^a$ outputs with non–negligible probability $G^{(a^2)}$. Such a "squaring" CDH solver can in turn be used as a general CHD solver, which succeeds with non–negligible probability [15].

[4] To simulate consistently the ephemeral exponent leakage on $\hat{B}$, $\mathcal{S}$ has to assign values to $\bar{H}$ query with a partially unknown input, namely the incoming ephemeral key is unknown. For these queries, random values are taken in $\{0,1\}^l$ as $\bar{H}(Y, \star, \hat{B}, \hat{P}_l)$ and $\bar{H}(\star, Y, \hat{B}, \hat{P}_l)$; and when $\mathcal{S}$ is later queried $\bar{H}(Y, X, \hat{B}, \hat{P}_l)$ (resp. $\bar{H}(X, Y, \hat{B}, \hat{P}_l)$, it responds with $\bar{H}(Y, \star, \hat{B}, \hat{P}_l)$ (resp. $\bar{H}(\star, Y, \hat{B}, \hat{P}_l)$).

We simulate $\mathcal{A}$'s environment as in the analysis of *E.1*, except the following differences.

- $\mathcal{S}$ takes as input $A \in_R \mathcal{G}^*$ (recall that $\hat{A} = \hat{B}$).
- $\hat{A}$'s public key is set to A; $\hat{A}$'s roles are simulated in the same way as that of $\hat{B}$ in the analysis *E.2*.
- When $\mathcal{A}$ activates the t–th session at $\hat{A}$, $\mathcal{S}$ chooses $x_0 \in_R \mathbb{Z}_q^*$, and provides $\mathcal{A}$ with the message $(\hat{A}, \hat{A}, X_0 = G^{x_0})$; and if $\mathcal{A}$ activates at $\hat{B}$ a session matching t–th session at $\hat{A}$, $\mathcal{S}$ aborts.
- If $\mathcal{A}$ issues a static key reveal on $\hat{A}$, $\mathcal{S}$ aborts.

The simulation of $\mathcal{A}$'s environment remains perfect, except with negligible probability; and if $\mathcal{A}$ succeeds with non–negligible probability in *F.2*, then under this simulation, it outputs with non–negligible probability a valid forgery σ_0 of the test session's FDCR signature. And then since $\mathcal{S}$ knows x_0 it can output with non–negligible probability

$$\left((\sigma_0)(Y_0 A^e)^{-x_0}\right)^{d^{-1}} = \left((Y_0 A^e)^{x_0+da}(Y_0 A^e)^{-x_0}\right)^{d^{-1}} = \left((Y_0 A^e)^{da}\right)^{d^{-1}} = (Y_0 A^e)^a = A^{y_0+ea}.$$

Hence, given a public key A, $\mathcal{S}$ outputs with non–negligible probability a valid and fresh FXCR signature, on message $(\hat{A}, \hat{B})$ (concatenation of $\hat{A}$ and $\hat{B}$), and challenge A (the challenge equals the public key) with respect to the public key A. Using the "oracle replay" technique (as in the proof of Proposition 7), $\mathcal{S}$ yields a polynomial machine, which given $A = G^a$, outputs with non–negligible probability $A^a = G^{(a^2)}$; contradicting the CDH assumption.

5.2 Ephemeral Private Keys Leakage Resilience (eck–security)

The arguments for this security attribute do not derive from the analysis in section 4. The reason is that we cannot simulate simultaneously and consistently both *SecretExponentReveal* and *EphemeralKeyReveal*.

5.2.1 Hashed Full Dual Challenge Response scheme

Definition 10 (Hashed FDCR (HFDCR) signature scheme)**.** Let $\hat{A}$, $\hat{B}$ be two entities with public keys $A, B \in_R \mathcal{G}^*$. The HFDCR signature of $\hat{A}$ and $\hat{B}$ on messages (m_1, m_2) is

$$HFDCR_{A,B}(X, Y, m_1, m_2) = H(\sigma, m_1, m_2, X, Y),$$

where σ is the FDCR signature of $\hat{A}$ and $\hat{B}$ on messages m_1, m_2, and challenges X, Y.

Definition 11 (Security of the HFDCR signature scheme)**.** Let $\hat{A}, \hat{B}$ be two entities with public keys $A, B \in_R \mathcal{G}^*$. The HFDCR scheme is said to be secure, if given A, B, and $x_0, y_0 \in_R \mathbb{Z}_q^*$, no probabilistic polynomial time attacker, performing queries adaptively, can output with non negligible success probability a triple $(m_{1_0}, m_{2_0}, \pi_0)$ such that: $HFDCR_{A,B}(X_0, Y_0, m_{1_0}, m_{2_0}) = \pi_0$, and π_0 was not obtained from the signing oracle with a query on a quadruple (X', Y', m_1', m_2') such that $X_0 = X'$, $Y_0 = Y'$, and $(m_{1_0}, m_{2_0}) = (m_1', m_2')$.

For the HFDRC scheme security arguments, we need the Gap Diffie–Hellman (GDH) assumption.

Definition 12 ([20])**.** Let $\mathcal{G} = \langle G \rangle$ be a cyclic group. An algorithm is said to be a *Decisional Diffie–Hellman Oracle* (DDO) for $\mathcal{G}$, if on input $G, X = G^x, Y = G^y, Z \in \mathcal{G}$, it outputs 1 if and only if $Z = G^{xy}$. The *Gap Diffie–Hellman* (GDH) assumption is said to hold in $\mathcal{G}$, if given a DDO for $\mathcal{G}$, there exists no polynomially bounded algorithm which solves the CDH problem in $\mathcal{G}$, with non–negligible success probability.

We now prove the security of the HFDCR scheme, under the GDH assumption and the RO model.

Proposition 10. *Under the GDH assumption in $\mathcal{G}$, and the RO model, the HFDCR scheme is secure in the sense of definition 11.*

Proof. Suppose a polynomially bounded attacker $\mathcal{A}$, which given a DDO, $A, B \in_R \mathcal{G}^*$, and $x_0, y_0 \in_R \mathbb{Z}_q^*$, outputs with non–negligible success probability a valid and fresh HFDCR$_{A,B}$ signature on some messages (m_{1_0}, m_{2_0}) with respect to challenges $X_0 = G^{x_0}$, $Y = G^{y_0}$.

Non–matching HFDCR signature queries cannot have the same signature value, except with negligible probability. And guessing the output of the hash function cannot be done with non–negligible success probability. We can thus suppose that $\mathcal{A}$ succeeds with non–negligible probability in forging attack. Using $\mathcal{A}$ and a DDO, we build a polynomial time CDH solver $\mathcal{S}$ which succeeds with non–negligible probability. The solver $\mathcal{S}$ provides $\mathcal{A}$ with random coins and simulates the signature queries; it takes as input $A, B \in_R \mathcal{G}^*$, $x_0, y_0 \in_R \mathbb{Z}_q^*$, and outputs with non–negligible probability $CDH(A, B)$.

1) At $\mathcal{A}$'s $\bar{H}$ digest query on (X, Y, m_1, m_2), $\mathcal{S}$ does the following things:
- If a value is already assigned to $\bar{H}(X, Y, m_1, m_2)$, $\mathcal{S}$ provides $\mathcal{A}$ with the value of $\bar{H}(X, Y, m_1, m_2)$.
- Else $\mathcal{S}$ chooses $d \in_R \{0,1\}^l$, sets $\bar{H}(X, Y, m_1, m_2) = d$, and provides $\mathcal{A}$ with d.

2) At $\mathcal{A}$'s signature query on (X, Y, m_1, m_2), $\mathcal{S}$ responds as follows:
- If a value is already assigned to $HFDCR_{A,B}(X, Y, m_1, m_2)$, then $\mathcal{S}$ returns the value of $HFDCR_{A,B}(X, Y, m_1, m_2)$; else $\mathcal{S}$ takes $\pi \in_R \{0,1\}^\lambda$, sets $HFDCR_{A,B}(X, Y, m_1, m_2) = \pi$, and provides $\mathcal{A}$ with π.
- If no value is assigned to $\bar{H}(X, Y, m_1, m_2)$ (resp. $\bar{H}(Y, X, m_1, m_2)$), $\mathcal{S}$ chooses $d \in_R \{0,1\}^l$ and sets $\bar{H}(X, Y, m_1, m_2) = d$ (resp. $\bar{H}(Y, X, m_1, m_2) = d$).

3) At $\mathcal{A}$'s digest query on (σ, m_1, m_2, X, Y), $\mathcal{S}$ does the following things:
- If a value is assigned to $HFDCR_{A,B}(X, Y, m_1, m_2)$, and if $\sigma = CDH(XA^d, YB^e)$, where $d = \bar{H}(X, Y, m_1, m_2)$, $e = \bar{H}(Y, X, m_1, m_2)$ (if a value is already assigned to $HFDCR_{A,B}(X, Y, m_1, m_2)$, d and e are defined, and the verification is done with the DDO), $\mathcal{S}$ returns the value of $HFDCR_{A,B}(X, Y, m_1, m_2)$.
- Else, *(i)* if the same query was made previously, $\mathcal{S}$ returns the previously returned value; *(ii)* else $\mathcal{S}$ chooses $\pi \in_R \{0,1\}^\lambda$, sets $H(\sigma, m_1, m_2, X, Y) = \pi$, and provides $\mathcal{A}$ with π.
- If no value is assigned to $\bar{H}(X, Y, m_1, m_2)$ (resp. $\bar{H}(Y, X, m_1, m_2)$), $\mathcal{S}$ chooses $d \in_R \{0,1\}^l$ and sets $\bar{H}(X, Y, m_1, m_2) = d$ (resp. $\bar{H}(Y, X, m_1, m_2) = d$); and if $\sigma = CDH(XA^d, YB^e)$ (this is verified with the DDO), $\mathcal{S}$ sets $HFDCR_{A,B}(X, Y, m_1, m_2) = \pi$.

4) If $\mathcal{A}$ halts with a forgery $\pi_0, X_0, Y_0, m_{1_0}, m_{2_0}$, $\mathcal{S}$ verifies that the digests value π_0 was queried from the random oracle, as $H(\sigma_0, m_{1_0}, m_{2_0}, X_0, Y_0)$ for some σ_0, and that $\sigma_0 = CDH(X_0A^d, Y_0B^e)$, where $d = \bar{H}(X_0, Y_0, m_{1_0}, m_{2_0})$, and $e = \bar{H}(Y_0, X_0, m_{1_0}, m_{2_0})$ (if π_0 was queried from the hashing oracle, then $\bar{H}(X_0, Y_0, m_{1_0}, m_{2_0})$ and $\bar{H}(Y_0, X_0, m_{1_0}, m_{2_0})$ are defined).

Under the RO model, $\mathcal{A}$'s simulated environment is perfect except with negligible probability; hence if $\mathcal{A}$ succeeds with non–negligible probability in forging a HFDRC signature, then it succeeds under this simulation with the same probability, except a negligible difference. Since $\mathcal{S}$ knows x_0, y_0, and $\mathcal{A}$ succeeds with non–negligible probability, $\mathcal{S}$ can output with non–negligible probability

$$\begin{aligned}
\left(\left((\sigma_0)(Y_0B^e)^{-x_0}\right)^{d^{-1}}A^{-y_0}\right)^{e^{-1}} &= \left(\left((Y_0B^e)^{x_0+da}(Y_0B^e)^{-x_0}\right)^{d^{-1}}A^{-y_0}\right)^{e^{-1}} \\
&= \left(\left((Y_0B^e)^{da}\right)^{d^{-1}}A^{-y_0}\right)^{e^{-1}} = \left((Y_0B^e)^{a}A^{-y_0}\right)^{e^{-1}} \\
&= \left(A^{y_0+eb}A^{-y_0}\right)^{e^{-1}} = CDH(A, B);
\end{aligned}$$

contradiction the GDH assumption. □

5.2.2 Application to FHMQV

With the HFDCR, we can now prove that the FHMQV protocol is eck–secure; showing that the FHMQV protocol is secure in the sense of definition 5.

Proposition 11. *Under the GDH assumption in $\mathcal{G}$, and the RO model, the FHMQV protocol is eck–secure.*

Proof of Proposition 11. Suppose an adversary $\mathcal{A}$ which succeeds with probability significantly greater that 1/2 in distinguishing an eck–fresh session key from a random value chosen under the distribution of session keys. Since guessing and key replication attacks cannot succeed with non–negligible probability, $\mathcal{A}$ succeeds with non–negligible probability in forging attack.

Let *E'* be the event *"$\mathcal{A}$ succeeds in forging an eck–fresh session (FDCR) signature."* *E'* divides in *E'.1*: *"$\mathcal{A}$ succeeds in forging the session signature of some eck–fresh session sid, which matching session exists"*, and *E'.2*: *"$\mathcal{A}$ succeeds in forging the session signature of some eck–fresh session sid, without matching session."* It thus suffices to show that neither *E'.1* nor *E'.2* can happen with non–negligible probability.

Analysis of *E'.1*. Since the test session *sid* is required to be eck–fresh and *sid** exists, the strongest queries that $\mathcal{A}$ can perform are:*(i)* StaticKeyReveal *queries on both* $\hat{A}$ *and* $\hat{B}$; *(ii)* EphemeralKeyReveal *queries on both sid and sid**; *(iii)* a *StaticKeyReveal* query on $\hat{A}$ and an EphemeralKeyReveal *query on sid**; *(iv)* an EphemeralKeyReveal *query on sid and* a StaticKeyReveal *query on* $\hat{B}$. And from any polynomial time machine which succeeds in *E'.1*, and performs weaker queries than those above, one can build a polynomial time machine, which succeeds with same probability and performs one of the above queries. It thus suffices to show that none of the events
E'.1.1: "*E'.1* $\wedge$ $\mathcal{A}$ *performs* StaticKeyReveal *queries on both* $\hat{A}$ *and* $\hat{B}$",
E'.1.2: "*E'.1* $\wedge$ $\mathcal{A}$ *performs* EphemeralKeyReveal *queries on both sid and sid**",
E'.1.3: "*E'.1* $\wedge$ $\mathcal{A}$ *performs a* StaticKeyReveal *query on* $\hat{A}$ *and an* EphemeralKeyReveal *query on sid**",
E'.1.4: "*E'.1* $\wedge$ $\mathcal{A}$ *performs an* EphemeralKeyReveal *query on sid and a* StaticKeyReveal *query on* $\hat{B}$",
occur with non–negligible probability.

Analysis of *E'.1.1*. Suppose that *E'.1.1* happens with non–negligible probability; using $\mathcal{A}$, we build a polynomial time CDH solver which succeeds with non–negligible probability. For that we use the same simulation as in the analysis of *E.1* with the following modifications (recall that the allowed queries are that from Set 2):

- If $\mathcal{A}$ issues a *SessionKeyReveal*, *EphemeralKeyReveal*, or an *StaticKeyReveal* query, $\mathcal{S}$ answers faithfully.
- In any of the following situations, $\mathcal{S}$ aborts.
  - $\mathcal{A}$ halts with a test session different from the t–th session at $\hat{A}$.
  - $\mathcal{A}$ issues a *SessionKeyReveal* query on the t–th session at $\hat{A}$ or its matching session.
  - $\mathcal{A}$ issues an *EphemeralKeyReveal* on the t–th session at $\hat{A}$ or its matching session.
  - $\mathcal{A}$ issues an *EstablishParty* query on $\hat{A}$ or $\hat{B}$.

The simulated environment remains perfect, except with negligible probability. The probability of guessing correctly the test session is $(n^2 m)^{-1}$. If $\mathcal{A}$ succeeds with non–negligible probability in *E'.1.1*, then under this simulation $\mathcal{A}$ outputs a valid forgery of the t–th session at $\hat{A}$ with non–negligible probability. Hence $\mathcal{S}$ outputs $CDH(X_0, Y_0)$ (from $\mathcal{A}$'s forgery and a, b, d, e) with non–negligible probability.

Analysis of *E'.1.2*.

We reuse the simulation of the analysis of *E.1*, with the following modifications:

- $\mathcal{S}$ takes as input $x_0, y_0 \in_R \mathbb{Z}_q^*$, and $A, B \in_R \mathcal{G}^*$.
- $\hat{A}$ and $\hat{B}$'s public keys are set to A and B (the private keys are unknown to $\mathcal{S}$).
- At $\mathcal{A}$'s $Send(\hat{P}_m, \hat{P}_l, Y)$ query, with $\hat{P}_l = \hat{A}$ or $\hat{B}$, $\mathcal{S}$ responds as follows.
  - $\mathcal{S}$ chooses $x \in_R \mathbb{Z}_q^*$, computes $X = G^x$, creates a session state with identifier $(\hat{P}_l, \hat{P}_m, X, Y)$, provides $\mathcal{A}$ with the outgoing message $(\hat{P}_l, \hat{P}_m, X)$.
  - $\mathcal{S}$ chooses $\pi \in_R \{0,1\}^\lambda$, $d, e \in_R \{0,1\}^l$ and sets $HFDCR_{P_m,P_l}(Y, X, \hat{P}_m, \hat{P}_l) = \pi$, $\bar{H}(Y, X, \hat{P}_m, \hat{P}_l) = d$, and $\bar{H}(X, Y, \hat{P}_m, \hat{P}_l) = e$.
- At $Send(\hat{P}_l, \hat{P}_m, X, Y)$ query, with $\hat{P}_l = \hat{A}$ or $\hat{B}$, $\mathcal{S}$ answers as follows:
  - $\mathcal{S}$ updates the session identifier $(\hat{P}_l, \hat{P}_m, X, \star)$ (if any) to $(\hat{P}_l, \hat{P}_m, X, Y)$.
  - If a value is not assigned to $HFDCR_{P_l,P_m}(X, Y, \hat{P}_l, \hat{P}_m)$, $\mathcal{S}$ chooses $\pi \in_R \{0,1\}^\lambda$ and sets $HFDCR_{P_l,P_m}(X, Y, \hat{P}_l, \hat{P}_m) = \pi$; if a value is not assigned to $\bar{H}(X, Y, \hat{P}_l, \hat{P}_m)$ (resp. $\bar{H}(Y, X, \hat{P}_l, \hat{P}_m)$), $\mathcal{S}$ chooses $d \in_R \{0,1\}^l$ and sets $\bar{H}(X, Y, \hat{P}_l, \hat{P}_m) = d$ (resp. $\bar{H}(Y, X, \hat{P}_l, \hat{P}_m) = d$).
- When $\mathcal{A}$ activates the t–th session at $\hat{A}$, if $\hat{A}$'s peer in that session is not $\hat{B}$, $\mathcal{S}$ aborts; otherwise $\mathcal{S}$ provides $\mathcal{A}$ with the outgoing message $(\hat{A}, \hat{B}, X_0 = G^{x_0})$.
- When $\mathcal{A}$ activates the session matching t–th session at $\hat{A}$, $\mathcal{S}$ provides $\mathcal{A}$ with $(\hat{B}, \hat{A}, Y_0 = G^{y_0})$.

- At $\mathcal{A}$'s digest query on $(\sigma, \hat{P}_l, \hat{P}_m, X, Y)$, with $\hat{P}_l = \hat{A}$ or $\hat{B}$, or $\hat{P}_m = \hat{A}$ or $\hat{B}$, $\mathcal{S}$ does the following things:
  - If a value is assigned to $HFDCR_{P_l,P_m}(X, Y, \hat{P}_l, \hat{P}_m)$, and if $\sigma = CDH(XP_l^d, YP_m^e)$, where $d = \bar{H}(X, Y, \hat{P}_l, \hat{P}_m)$, $e = \bar{H}(Y, X, \hat{P}_l, \hat{P}_m)$ (if $HFDCR_{P_l,P_m}(X, Y, \hat{P}_l, \hat{P}_m)$ is already defined, the values of d and e are already assigned), $\mathcal{S}$ returns the value of $HFDCR_{P_l,P_m}(X, Y, \hat{P}_l, \hat{P}_m)$.
  - Else, *(i)* if the same query was made previously, $\mathcal{S}$ returns the previously returned value; *(ii)* else $\mathcal{S}$ chooses $\pi \in_R \{0,1\}^\lambda$, sets $H(\sigma, \hat{P}_l, \hat{P}_m, X, Y) = \pi$, and provides $\mathcal{A}$ with π.
  - If a value is not assigned to $\bar{H}(X, Y, \hat{P}_l, \hat{P}_m)$ (resp. $\bar{H}(Y, X, \hat{P}_l, \hat{P}_m)$), $\mathcal{S}$ chooses $d \in_R \{0,1\}^\lambda$ and sets $\bar{H}(X, Y, \hat{P}_l, \hat{P}_m)$ (resp. $\bar{H}(Y, X, \hat{P}_l, \hat{P}_m)) = d$; and if $\sigma = CDH(XP_l^d, YP_m^e)$ (this is verified with the DDO), $\mathcal{S}$ sets $HFDCR_{P_l,P_m}(X, Y, \hat{P}_l, \hat{P}_m) = \pi$.
- If $\mathcal{A}$ issues an *EphemeralKeyReveal* query on the t–th session at $\hat{A}$ or its matching session, $\mathcal{S}$ answers faithfully.
- In the following situations $\mathcal{S}$ aborts.
  - $\mathcal{A}$ halts with a test session different from the t–th session at $\hat{A}$.
  - $\mathcal{A}$ issues a *StaticKeyReveal* query on $\hat{A}$ or $\hat{B}$.
  - $\mathcal{A}$ issues an *EstablishParty* query on $\hat{A}$ or $\hat{B}$.

The simulated environment remains perfect, except with negligible probability, and the probability of guessing correctly the test session is $(n^2 m)^{-1}$. If $\mathcal{A}$ succeeds with a FDCR signature σ_0, and $\mathcal{S}$ guesses correctly the test session, $\mathcal{S}$ outputs a valid HFDCR forgery $\pi_0 = H(\sigma_0, \hat{A}, \hat{B}, X_0, Y_0)$, contradicting Proposition 10.

Analysis of *E'.1.3* and *E'.1.4*. Since $\hat{A}$ and $\hat{B}$ roles are symmetrical in *E'.1.3* and *E'.1.4*, it suffices to show that *E'.1.3* cannot happen with non–negligible probability. For that, the simulation used in the analysis of *E.1*, is modified as follows:

- $\mathcal{S}$ takes as input $X_0, B \in_R \mathcal{G}^*$.
- $\hat{B}$'s public key is set to B (the private key is unknown to $\mathcal{S}$), and $\hat{A}$'s key pair is $(a = p_i, G^a), p_i \in_R \mathbb{Z}_q^*$.
- At $\mathcal{A}$'s $Send(\hat{P}_m, \hat{B}, X)$ query, $\mathcal{S}$ responds as follows:
  - $\mathcal{S}$ chooses $y \in_R \mathbb{Z}_q^*$, computes $Y = G^y$, creates a session state with identifier $(\hat{B}, \hat{P}_m, Y, X)$, and provides $\mathcal{A}$ with outgoing message $(\hat{B}, \hat{P}_m, Y)$.
  - $\mathcal{S}$ chooses $\pi \in_R \{0,1\}^\lambda$, $d, e \in_R \{0,1\}^l$, and sets $HFDCR_{P_m,B}(X, Y, \hat{P}_m, \hat{B}) = \pi$, $\bar{H}(X, Y, \hat{P}_m, \hat{B}) = d$, and $\bar{H}(Y, X, \hat{P}_m, \hat{B}) = e$.
- At $Send(\hat{B}, \hat{P}_m, Y, X)$ query:
  - $\mathcal{S}$ updates the session identifier $(\hat{B}, \hat{P}_m, Y, \star)$ to $(\hat{B}, \hat{P}_m, Y, X)$.
  - If no value is assigned to $HFDCR_{B,P_m}(Y, X, \hat{B}, \hat{P}_m)$, $\mathcal{S}$ chooses $\pi \in_R \{0,1\}^\lambda$ and sets $HFDCR_{B,P_m}(Y, X, \hat{B}, \hat{P}_m) = \pi$; and if no value is assigned to $\bar{H}(Y, X, \hat{B}, \hat{P}_m)$ (resp. $\bar{H}(X, Y, \hat{B}, \hat{P}_m)$), $\mathcal{S}$ chooses $d \in_R \{0,1\}^l$ and sets $\bar{H}(Y, X, \hat{B}, \hat{P}_m) = d$ (resp. $\bar{H}(X, Y, \hat{B}, \hat{P}_m) = d$).
- At $\mathcal{A}$'s digest query on $(\sigma, \hat{P}_l, \hat{P}_m, X, Y)$, with $\hat{P}_l = \hat{B}$, or $\hat{P}_m = \hat{B}$, $\mathcal{S}$ responds as follows:
  - If a value is assigned to $HFDCR_{P_l,P_m}(X, Y, \hat{P}_l, \hat{P}_m)$, and if $\sigma = CDH(XP_l^d, YP_m^e)$, where $d = \bar{H}(X, Y, \hat{P}_l, \hat{P}_m)$, $e = \bar{H}(Y, X, \hat{P}_l, \hat{P}_m)$, $\mathcal{S}$ returns the value of $HFDCR_{P_l,P_m}(X, Y, \hat{P}_l, \hat{P}_m)$;
  - Else, *(i)* if the same query was made previously, $\mathcal{S}$ returns the previously returned value; *(ii)* else $\mathcal{S}$ chooses $\pi \in_R \{0,1\}^\lambda$, sets $H(\sigma, \hat{P}_l, \hat{P}_m, X, Y) = \pi$, and provides $\mathcal{A}$ with π;
  - If a value is not assigned to $\bar{H}(X, Y, \hat{P}_l, \hat{P}_m)$ (resp. $\bar{H}(Y, X, \hat{P}_l, \hat{P}_m)$), $\mathcal{S}$ chooses $d \in_R \{0,1\}^\lambda$ and sets $\bar{H}(X, Y, \hat{P}_l, \hat{P}_m) = d$ (resp. $\bar{H}(Y, X, \hat{P}_l, \hat{P}_m) = d$); and if $\sigma = CDH(XP_l^d, YP_m^e)$, $\mathcal{S}$ sets $HFDCR_{P_l,P_m}(X, Y, \hat{P}_l, \hat{P}_m) = \pi$.
- When $\mathcal{A}$ activates the t–th session at $\hat{A}$, if $\hat{A}$'s peer in that session is not $\hat{B}$, $\mathcal{S}$ aborts; otherwise, $\mathcal{S}$ provides $\mathcal{A}$ with the outgoing message $(\hat{A}, \hat{B}, X_0)$.
- When $\mathcal{A}$ activates the session matching t–th session at $\hat{A}$, $\mathcal{S}$ chooses $y_0 \in_R \mathbb{Z}_q^*$, and provides $\mathcal{A}$ with $(\hat{B}, \hat{A}, Y_0 = G^{y_0})$.
- If $\mathcal{A}$ issues an *EphemeralKeyReveal* query on the session matching the t–th session at $\hat{A}$, $\mathcal{S}$ answers faithfully.

- In the following situations $\mathcal{S}$ aborts.
  - $\mathcal{A}$ halts with a test session different from the t–th session at $\hat{A}$.
  - $\mathcal{A}$ issues a *StaticKeyReveal* query on $\hat{B}$.
  - $\mathcal{A}$ issues an *EphemeralKeyReveal* query on the t–th session at $\hat{A}$.
  - $\mathcal{A}$ issues an *EstablishParty* query on $\hat{A}$ or $\hat{B}$.

The simulation remains perfect, except with negligible probability; if $\mathcal{A}$ succeeds with non–negligible probability in event *E'.1.3*, $\mathcal{S}$ outputs with non–negligible probability $CDH(X_0, B)$, from $\mathcal{A}$'s forgery (and a, y_0, d, and e); contradicting the GDH assumption.

Under the GDH assumption and the RO model, none of the events *E'.1.1*, *E'.1.2*, *E'.1.3*, or *E'.1.4* can happen with non–negligible probability, this shows that *E'.1* cannot happen, except with negligible probability.

Analysis of *E'.2* (sketch). The strongest query that $\mathcal{A}$ can perform in *E'.2* are: *a* StaticKeyReveal *query on* $\hat{A}$ or *an* EphemeralKeyReveal *query on sid* (but not both). It thus suffices to show that neither *E'.2.1: "E'.2* $\wedge$ $\mathcal{A}$ *performs a* StaticKeyReveal *query on* $\hat{A}$*"* nor *E'.2.2: "E'.2* $\wedge$ $\mathcal{A}$ *performs an* EphemeralKeyReveal *query on sid"* can happen with non–negligible probability.

To show that *E'.2.1* cannot happen with non–negligible probability, the simulation used in the analysis of *E'.1.3* can be modified such that if $\mathcal{A}$ activates a session matching the t–th session at $\hat{A}$, $\mathcal{S}$ aborts. Since $\mathcal{A}$ succeeds with non–negligible probability, using $\mathcal{A}$, $\mathcal{S}$ outputs with non–negligible probability $(Y_0 B^e)^{x_0}$ from $\mathcal{A}$ forgery and a. Hence, from the forking lemma, using $\mathcal{S}$, we can build a polynomial machine $\mathcal{S}'$ which given X_0, B, outputs with non–negligible probability $CDH(X_0, B)$. This shows that under the RO model and the GDH assumption, *E'.2.1* cannot happen with non–negligible probability.

For the analysis of *E'.2.2*, the simulation used in the analysis of *E'.1.2* can be modified to take as input x_0, A, B, and aborts when $\mathcal{A}$ activates a session matching the t–th session at $\hat{A}$. If $\mathcal{A}$ succeeds in *E'.2.2* (and then under this simulation), using $\mathcal{A}$'s forgery, $\mathcal{S}$ outputs $(Y_0 B^e)^a$; and from the forking lemma, $\mathcal{S}$ can be transformed into an efficient machine which given A, B outputs with non–negligible probability $CDH(A, B)$, contradicting the GDH assumption.

Reflection Attacks Resilience (sketch). With arguments similar to that of subsection 5.1.3, one can show that reflection attacks cannot hold against eck–fresh sessions. Indeed, if $\hat{A} = \hat{B}$, and if the session matching the test session exists, $\mathcal{A}$ is not allowed to perform both StaticKeyReveal *on* $\hat{A}$*, and* EphemeralKeyReveal *on the test session or its matching session.* And if the session matching the test session does not exist, StaticKeyReveal *on* $\hat{A}$ is not allowed. Moreover, the analysis of *E'.1.1* applies also if $\hat{A} = \hat{B}$. And a polynomial time successful attacker in *E'.1.2* when $\hat{A} = \hat{B}$, yields a polynomial machine which given $A = G^a$ outputs $G^{(a^2)}$. Hence under these restrictions on allowed queries, and with minor modifications in the analysis of *E'.1.1* and *E'.1.2*, one can show that under the RO model and the GDH assumption, the FHMQV protocol provides resistance to reflection attacks for eck–fresh sessions.

5.3 Main Differences between FHMQV and HMQV Security Arguments

We summarize the most important differences between the HMQV and FHMQV security arguments.

Building blocks and adversary model. The design of FHMQV relies on the FXCR and FDCR signature schemes. While in the XCR scheme as in the FXCR scheme, both s_A and x leakages in the same session imply $\hat{A}$'s private key discloser. In the FXCR scheme, an adversary who has learned s_A is unable to forge A's signature. The FHMQV adversary model allows s_A leakage. The impersonation and man in the middle attacks we present in section 2 do not hold against FHMQV.

Key replication attacks resilience. At session key derivation in FHMQV, ephemeral keys and peers identities are hashed with the session's FDCR signature ($K = H(\sigma, \hat{A}, \hat{B}, X, Y)$). Since non matching sessions cannot have (except with negligible probability) the same ephemeral keys, and non matching digest queries cannot have (except with negligible probability) the same digest value, the analysis of key replication attacks is immediate for the FHMQV protocol.

Ephemeral private keys leakage resilience. To show this security attribute for FHMQV,

we define the Hashed FDCR signature scheme; for the HMQV protocol, it is used a hashed variant of the XCR, namely the HCR signature scheme [11]. While both HFDCR and HCR security arguments rely on the GDH assumption, for the HFDCR scheme the Knowledge of Exponent Assumption (KEA1) [4] is not needed, whereas it is required for the HCR scheme.

6 The FHMQV–C Protocol

As shown in [11], no two message key agreement protocol can meet the perfect forward secrecy security attribute; key confirmation security attribute (for both peers) cannot be achieved also. Nevertheless these security attributes may be desirable; for that the FHMQV protocol can be added with a third message, yielding the FHMQV–C protocol, we describe in Protocol 6; KDF_1 and KDF_2 are key derivation functions, and MAC a message authentication code [8].

Protocol 6 FHMQV–C key exchange

I) $\hat{A}$ chooses $x \in_R \mathbb{Z}_q^*$, computes $X = G^x$, and sends $(\hat{A}, \hat{B}, X)$ to $\hat{B}$.

II) $\hat{B}$ does the following things:
  a) Verify that $X \in \mathcal{G}^*$.
  b) Choose $y \in_R \mathbb{Z}_q^*$, compute $Y = G^y$, $d = \bar{H}(X, Y, \hat{A}, \hat{B})$, and $e = \bar{H}(Y, X, \hat{A}, \hat{B})$.
  c) Compute $s_B = (y + eb) \bmod q$, $\sigma_B = (XA^d)^{s_B}$.
  d) Compute and $K_1 = KDF_1(\sigma_B, \hat{A}, \hat{B}, X, Y)$ and $t_B = MAC_{K_1}(\hat{B}, Y)$.
  e) Send $(\hat{B}, \hat{A}, Y, t_B)$ to $\hat{A}$.

III) $\hat{A}$ does the following things:
  a) Verify that $Y \in \mathcal{G}^*$, compute $d = \bar{H}(X, Y, \hat{A}, \hat{B})$, and $e = \bar{H}(Y, X, \hat{A}, \hat{B})$.
  b) Compute $s_A = (x + da) \bmod q$, $\sigma_A = (YB^e)^{s_A}$.
  c) Compute $K_1 = KDF(\sigma_A, \hat{A}, \hat{B}, X, Y)$, $t'_B = MAC_{K_1}(\hat{B}, Y)$, and verify that $t'_B = t_B$.
  d) Compute $t_A = MAC_{K_1}(\hat{A}, X)$, and send t_A to $\hat{B}$.
  e) Compute $K_2 = KDF_2(\sigma_B, \hat{A}, \hat{B}, X, Y)$

IV) $\hat{B}$ computes $t'_A = MAC_{K_1}(\hat{A}, X)$, verifies that $t'_A = t_A$, and computes $K_2 = KDF_2(\sigma_B, \hat{A}, \hat{B}, X, Y)$.

V) The shared session key is K_2.

When a party $\hat{A}$ completes a FHMQV–C session with some honest peer $\hat{B}$, and with incoming ephemeral key Y, he is guaranteed that Y was chosen and authenticated by $\hat{B}$, and that $\hat{B}$ can compute the session key he derives. The FHMQV–C protocol provides also perfect forward secrecy, the compromise of $\hat{A}$'s static private key, does not compromise the session keys established in previous runs. This can be shown when the analysis of FHMQV is completed with the *session–key expiration* notion [5].

7 Concluding Remarks

We propose a complementary analysis of the Exponential Challenge Response and Dual Exponential Challenge Response signature schemes, which are the building blocks of the HMQV protocol. On the basis of this analyze, we show how impersonation and man in the middle attacks can be done against the HMQV protocol, when some session specific information leakages happen. The attacks we present apply (with some modifications) to the (C)MQV [13, 26], and ECKE–1N [27] protocols.

We propose the Full Exponential Challenge Response (FXCR) and Full Dual Exponential Challenge Response (FDCR) signature schemes, with security arguments. Using these schemes, we define the Fully Hashed MQV (FHMQV) protocol, which preserves the efficiency and security attributes of the (H)MQV protocol, in addition to resistance to the ephemeral information s_A (and s_B) leakage. Similar works can be done for the CMQV and ECKE–1N protocols.

We define a Canetti–Krawczyk type security model, based on session's tree of computations, which allows a fine granularity analysis of session specific information leakages. We aim with this approach to capture both *ephemeral key reveal* and *session state reveal*. The FHMQV protocol is shown secure in this model. The FHMQV protocol can be added with a third message, yielding

the FHMQV–C protocol, which provides all the security attributes of the FHMQV protocol, added with key confirmation and perfect forward secrecy.

In a forthcoming stage, we will be interested in the analysis of relations between the security model we propose and the Canetti–Krawczyk and extended Canetti–Krawczyk security models.

Acknowledgments. The authors would like to thanks Netheos R&D for supporting this work. We would like to thank also the EuroPKI 2009 reviewers for their useful comments.

References

[1] ANSI X9.42: Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography. 2001.

[2] ANSI X9.63: Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport using Elliptic Curve Cryptography. 2001.

[3] Basin D., Cremers C.: From Dolev–Yao to Strong Adaptive Corruption: Analyzing Security in the Presence of Compromising Adversaries. Cryptology ePrint Archive, Report 2009/079, 2009.

[4] Bellare M., Palacio A.: The Knowledge–of–Exponent Assumptions and 3–round Zero–Knowledge Protocols. Lecture Notes in Computer Science, vol. 3152, 273–289, Springer–Verlag, 2004.

[5] Canetti R., Krawczyk H.: Analysis of Key–Exchange Protocols and Their Use for Building Secure Channels. Cryptology ePrint Archive, Report 2001/040, 2001.

[6] Cremers C.: Session-state Reveal is stronger than Ephemeral Key Reveal: Attacking the NAXOS key exchange protocol. Lecture Notes in Computer Science, vol. 5536, 20–33, Springer–Verlag, 2009.

[7] Gopalakrishnan K., Thériault N., Yao C. Z.: Solving Discrete Logarithms from Partial Knowledge of the Key. Lecture Notes in Computer Science, vol. 4859, 224–237, Springer–Verlag, 2007.

[8] Hankerson D., Menezes A., Vanstone S.: Guide to Elliptic Curve Cryptography. Springer–Verlag, 2003.

[9] IEEE 1363: Standard Specifications for Public Key Cryptography. 2000.

[10] ISO/IEC IS 9798-3: Information Technology – Security techniques : Cryptography techniques based on elliptic curves – Part 3 : Key Establishment. 2002.

[11] Krawczyk H.: HMQV: A Hight Performance Secure Diffie–Hellman Protocol. Cryptology ePrint Archive, Report 2005/176, 2005.

[12] LaMacchia B., Lauter K., Mityagin A.: Stronger Security of Authenticated Key Exchange. Lecture Notes in Computer Science, vol. 4784, 1–16, Springer–Verlag, 2007.

[13] Law L., Menezes A., Qu M., Solinas J., Vanstone S.: An Efficient Protocol for Authenticated Key Agreement. Designs, Codes and Cryptography, vol. 28(2), 119–134, Kluwer Academic Publishers, 2003.

[14] Leadbitter P. J., Smart N. P.: Analysis of the Insecurity of ECMQV with Partially Known Nonces. Lecture Notes in Computer Science, vol. 2851, 240–251, Springer–Verlag, 2003.

[15] Maurer U. M., Wolf S.: Diffie–Hellman Oracles. Lecture Notes in Computer Science, vol. 1109, 268–282, Springer–Verlag, 1996.

[16] Menezes A.: Another Look at HMQV. Journal of Mathematical Cryptology, vol. 1, 148-175, Walter de Gruyter, 2007.

[17] Menezes A., Ustaoglu B.: On the Importance of Public-Key Validation in the MQV and HMQV Key Agreement Protocols, Lecture Notes in Computer Science, vol. 4329, 133–147, Springer–Verlag, 2006.

[18] Menezes A., Ustaoglu B.: On Reusing Ephemeral Keys in Diffie–Hellman Key Agreement Protocols. Preprint, 2008 (available at http://www.cryptolounge.net/10.research.shtml).

[19] NIST Special publication 800-56: Recommendation on Key Establishment Schemes. 2003.

[20] Okamoto T., Pointcheval D.: The Gap–Problems: A new class of problems for the security of cryptographic schemes. Lecture Notes in Computer Science, vol. 1992, 104-118, Springer–Verlag, 2001.

[21] Pointcheval D.: Les preuves de connaissances et leurs preuves de sécurité. PhD thesis, Université de Caen, 1996; (in french, available at http://www.di.ens.fr/users/pointche/).

[22] Pointcheval D., Stern J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology, vol. 13, 361–396, Springer–Verlag, 2000.

[23] Pollard J. M.: Kangaroos, Monopoly and Discrete Logarithms. Journal of Cryptology, vol. 13, 437–447, Springer–Verlag, 2000.

[24] Teske E.: Square-root Algorithms for the Discrete Logarithm Problem (A survey). Public Key Cryptography and Computational Number Theory, 283–301, Walter de Gruyter, 2001.

[25] Teske E.: On Random Walks for Pollard's Rho Method. Mathematics of Computation, vol. 70, 809–825, American Mathematical Society, 2001.

[26] Ustaoglu B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Designs, Codes and Cryptography, vol. 46(3), 329–342, Kluwer Academic Publishers, 2008.

[27] Wang S., Cao Z., Strangio M. A., Wang L.: Cryptanalysis and Improvement of an Elliptic Curve Diffie–Hellman Key Agreement Protocol. Communications Letters, vol 12, 149–151, IEEE, 2008.