HMQV: A High-Performance Secure Diffie-Hellman Protocol

(Extended Abstract) [⋆]

Hugo Krawczyk[⋆⋆]

Abstract. The MQV protocol of Law, Menezes, Qu, Solinas and Vanstone is possibly the most efficient of all known authenticated Diffie-Hellman protocols based on public-key authentication. In addition to great performance, the protocol has been designed to achieve a remarkable list of security properties. As a result MQV has been widely standardized, and has recently been chosen by the NSA as the key exchange mechanism underlying *"the next generation cryptography to protect US government information"*.

One question that had not been settled so far is whether the protocol can be proven secure in a rigorous model of key-exchange security. In order to provide an answer to this question we analyze the MQV protocol in the Canetti-Krawczyk formal model of key exchange. Disappointingly, we show (by presenting explicit attacks) that virtually none of the stated security goals of MQV is actually satisfied in this model.

On the basis of these findings, we present HMQV, a carefully designed variant of MQV, that provides with the same superb performance and functionality of the original protocol but for which all the MQV's security goals can be formally proven to hold in the random oracle model under the computational Diffie-Hellman assumption.

We base the design and proof of HMQV in a new form of "challenge-response signatures" that have the property that both the challenger and signer can compute the *same* signature; the former by having chosen the challenge and the latter by knowing the private signature key. Hence, by simply exchanging challenges (in the form of DH values), two parties can compute the *same* signature from which a common authenticated key is then derived.

1 Introduction

The classic Diffie-Hellman (DH) key-exchange protocol that marked the birth of modern cryptography has since been one of the main pillars of both theory and practice of cryptography. While the basic protocol as originally proposed (i.e., two parties $\hat{A}$ and $\hat{B}$ exchange values g^x and g^y, and compute a secret shared

[⋆] This is a very partial and informal version of the full paper available at `http://eprint.iacr.org/2005/`

[⋆⋆] IBM T.J.Watson Research Center, PO Box 704, Yorktown Heights, NY 10598, USA. Email: hugo@ee.technion.ac.il

key as g^{xy}) is believed to be secure against an eavesdropping-only attacker, the quest for an "authenticated Diffie-Hellman" protocol that resists active, man-in-the-middle, attacks has resulted in innumerable ad-hoc proposals, many of which have been broken or shown to suffer from serious weaknesses. Fortunately, with the development of rigorous security models for key exchange in the last years, we are now in a much better position to judge the security of authenticated DH protocols as well as to develop designs that provably withstand realistic active attacks.

In addition to the need for sound security, the many practical applications of key exchange have driven designers to improve on the performance cost associated with authentication mechanisms, especially those based on public key. One ambitious line of investigation, initiated by Matsumoto, Takashima and Imai in 1986 [31], is to design DH protocols whose communication is identical to the basic DH protocol (i.e., no explicit authentication added except for the possible transmission of PK certificates), yet they are implicitly authenticated by the sole ability of the parties to compute the resultant session key (i.e, rather than agreeing on the key g^{xy}, the parties would agree on a key that combines g^x, g^y with their public/private keys). Not only can this approach generate protocols that are very efficient communication-wise, but the combination of authentication with the key derivation procedure can potentially result in significant computational savings. For these reasons, several of these "implicitly authenticated" protocols have been standardized by major national and international security standards.

Of these protocols, the most famous, most efficient and most standardized is the MQV protocol of Law, Menezes, Qu, Solinas and Vanstone [33, 30]. This protocol has been standardized by many organizations, e.g. [2, 3, 19, 21, 35], and has recently been announced by the US National Security Agency (NSA) as the key exchange mechanism underlying "the next generation cryptography to protect US government information" [36]. Indeed, MQV appears to be a remarkable protocol that not only is the most efficient and versatile authenticated DH protocol in existence, but it has also been designed to satisfy an impressive array of security goals.

Yet, in spite of its attractiveness and success, MQV has so far eluded any formal analysis in a well-defined model of key exchange. The present work was initially motivated by the desire to provide such an analysis. Our findings, however, have been disappointing: we found that when formally studied virtually none of the stated MQV goals can be shown to hold (specifically, we carried this study in the computational key exchange model of Canetti and Krawczyk [11]). This raises clear concerns about the security of the protocol and triggers a natural question: Do we have a replacement for MQV with the same superb performance and versatility but for which the MQV security goals can be guaranteed in a well analyzed, provable way?

The main contribution of this paper is in identifying the various analytical shortcomings of the MQV design and proposing a "hashed variant", which we call HMQV, that provides the same (almost optimal) performance of MQV but

Key Computation in the MQV and HMQV Protocols

Both protocols: $\hat{A}$ and $\hat{B}$ exchange $X = g^x, Y = g^y$ (via a basic DH run)

$\hat{A}$ computes $\sigma_{\hat{A}} = (YB^e)^{x+da}$, $\quad \hat{B}$ computes $\sigma_{\hat{B}} = (XA^d)^{y+eb}$

MQV: $d = \bar{X},\ e = \bar{Y},\ K = \sigma_{\hat{A}} = \sigma_{\hat{B}}$ ($\bar{X}$ and $\bar{Y}$ are defined in the text)

HMQV: $d = \bar{H}(X, \hat{B}),\quad e = \bar{H}(Y, \hat{A}),\quad K = H(\sigma_{\hat{A}}) = H(\sigma_{\hat{B}})$

Fig. 1. Computation of the session key K in each of the two protocols ($A = g^a$ and $B = g^b$ are $\hat{A}$'s and $\hat{B}$'s public keys, respectively.)

also delivers, in a provable way, the original security goals of MQV (and even more).

Organization. Due to space limitations most of this proceedings version is devoted to a high-level informal description of our results (Sections 2 and 3). The full version of the paper [28] contains a detailed presentation of our results and their proofs. The only technical section in this extended abstract is Section 4 which presents XCR signatures, the main technical tool developed here as a basis for the proof of the HMQV protocol (see the end of that section for a 1-paragraph rationale of HMQV's design). We end with some discussion of related work in Section 5 and concluding remarks in Section 6.

Note on groups and notation. All the protocols and operations discussed in this paper assume a cyclic group G of prime order q generated by a generator g. We denote by $|q|$ the bit length of q (i.e., $|q| = \lceil \log_2 q \rceil$), and use this quantity as an implicit security parameter. The parameters G, g and q are assumed to be fixed and known in advance to the parties (this is usually the case in practice, e.g., [20]; alternatively, one could include these values in certificates, etc.).
We use the multiplicative representation of group operations but our treatment is equally applicable to additive (prime order) groups such as elliptic curves.
In our protocols, public keys (denoted by upper case letters) are elements in the group G, and the private keys (denoted by the corresponding lower case letters) are elements in Z_q. For example, to a public key $A = g^a$ corresponds a private key a. The party having A as its public key will be denoted by $\hat{A}$ (in general, the "hat notation" is used to denote the identities of parties in the protocol, possibly including the party's PK certificate).

2 The MQV Protocol and its Security Shortcomings

The communication in the MQV protocol is identical to the basic unauthenticated DH protocol except that the identities $\hat{A}, \hat{B}$ may include a public-key certificate. The computation of the session key is shown in Figure 1 where: party $\hat{A}$ possesses a long-term private key $a \in Z_q$ and corresponding public key $A = g^a$, $\hat{B}$'s private/public key pair is $(b, B = g^b)$, and the ephemeral DH values exchanged in the protocol are $X = g^x, Y = g^y$ (x, y chosen by $\hat{A}, \hat{B}$, respectively).

The computation of the session key also uses the values $d = \bar{X}$ and $e = \bar{Y}$, where $\bar{X} = 2^\ell + (X \bmod 2^\ell)$ and $\bar{Y} = 2^\ell + (Y \bmod 2^\ell)$ for $\ell = |q|/2$. The computation of the session key by $\hat{A}$ (and similarly by $\hat{B}$) involves the exponentiations $X = g^x$, B^e, and $(YB^e)^{x+da}$. Note, however, that e is of length $|q|/2$ and hence B^e counts as "half exponentiation" (i.e. half the number of modular multiplication relative to a regular exponentiation of g). Also, note that $X = g^x$ can be pre-computed. This sums up to an impressive performance: same communication as the basic DH protocol and just half exponentiation more than the basic protocol, i.e. *a mere 25% increase in computation to achieve an authenticated exchange!* This is significantly better than any of the proven DH protocols that rely on digital signatures or public key encryption for authentication (which involve more expensive operations and increased bandwidth), and is also the most efficient of the implicitly-authenticated DH protocols (the closest are the "Unified Model" protocols [8, 23] that require three full exponentiations and offer substantially less security features – see Section 5).

2.1 Stated Security Goals of the MQV Protocol

The designers of MQV clearly, albeit informally, stated the security goals behind the MQV design (see [33, 30] and related publications). This includes the resistance to a variety of explicit attack strategies such as guessing attacks, impersonation attacks, known-key attacks, key-compromise impersonation (KCI) attacks, and the provision of perfect forward secrecy (PFS).

While resistance to guessing attacks (in which the attacker learns information about a key exchanged between honest parties) and impersonation attacks are basic and obvious security requirements, it is worth expanding on the meaning of the other attacks. They all represent realizations of the same fundamental security principle: 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.

In the case of known key attacks, one is concerned with the realistic possibility that some session-specific information, such as a session key (or the ephemeral secrets that led to the computation of that key), will leak to an attacker. This can happen in a variety of ways ranging from the simple mishandling of information to a temporary break-in into a computer system or the malicious action of an insider. In this case, one does not expect the exposed session to remain secure, but a well-designed key-exchange protocol needs to guarantee that such a failure will *only* affect the specific compromised session. Other sessions, by the same or other parties, should not be endangered by this leakage. The resistance to known-key attacks enforces other basic security principles as well; most importantly, that keys from different sessions should be fully "computationally independent" (meaning that from learning one session key nothing can be implied about the value of other session keys).

The properties of PFS and KCI resistance are also concerned with limiting the effects of eventual failures, in this case the disclosure of long-term keys. Clearly, the discovery by an attacker $\mathcal{M}$ of the long-term authentication key of party $\hat{A}$ allows $\mathcal{M}$ to impersonate $\hat{A}$ and establish its own sessions in the name of $\hat{A}$. A protocol is said to have PFS if session keys established (and deleted from memory) before the compromise of the long-term key cannot be recovered (even with the use of this key). In the case of KCI, the question is whether the knowledge of $\hat{A}$'s private key allows $\mathcal{M}$ not only to impersonate $\hat{A}$ to others but also *to impersonate other, uncorrupted, parties to $\hat{A}$.* A protocol that prevents this form of "reverse impersonation" is said to resist KCI attacks. In other words, in such a protocol the only way $\mathcal{M}$ can take advantage of the knowledge of $\hat{A}$'s private key is by active impersonation of $\hat{A}$. Any session established by $\hat{A}$, without being actively controlled by $\mathcal{M}$, remains secure. Resistance to KCI attacks is a very significant security property that has added much to the attractiveness of MQV as it is not offered by other implicitly-authenticated protocols, such as the unified-model protocols of [8, 23] (see Section 5), that use the static DH key g^{ab} for authentication (this key functions as a long-term shared key and hence cannot resist a KCI attack).

2.2 Weaknesses of the MQV Protocol

In spite of the ambitious security goals described above, it turns out that when casting these goals in a well-defined formal setting as the one in [11], the MQV protocol falls short of delivering the intended security. We point out to analytical shortcomings ranging from basic authentication failures due to non-standard group representations to the vulnerability of the protocol to combined KCI and known-key attacks. Due to page limitations we only present a summary of these findings here; please refer to [28] for a detailed account (which also includes a succinct description of the formal model from [11]).

Group representation attacks. We first observe that MQV's security is strongly susceptible to the specific way the group elements are represented in the protocol. We show how some representations render the protocol totally insecure. While ordinary group representations may not have such an extreme effect on the security of the protocol, this result shows that any attempt at proving MQV would need to involve restricted group representations. Moreover, the inherent weaknesses of the protocol discussed below show that the protocol cannot be proven secure even for specific groups.

UKS attacks. We study the vulnerability of the protocol to "unknown key share" (UKS) attacks which were also listed as a security consideration in MQV. A successful UKS attack [16] is one in which two parties compute the same session key but have different views of who the peer to the exchange was (this attack represents both an authentication failure as well as a vulnerability to known-key attacks). Originally, it was thought that MQV (at least when the registrants of public keys are required to prove "possession" of the corresponding private keys) was immune to these attacks; later it was shown by Kaliski [24] that even with such proofs of possession MQV fails to a UKS attack. Since then it has been believed that augmenting the protocol with a "key confirmation" step (which adds a third message to the protocol) would solve the problem. Here we show that this is *not* the case. Indeed, the 3-message variant of MQV is still vulnerable to this form of attack if the attacker can access ephemeral secret session-state information for sessions different than the session attacked.

KCI attacks. We demonstrate that MQV does not provide resistance to KCI attacks by showing how an attacker $\mathcal{M}$ can use the knowledge of the private key of a party $\hat{A}$ to learn the value of a session s established between $\hat{A}$ and an uncorrupted peer $\hat{B}$, without $\mathcal{M}$ having to ever control or actively impersonate $\hat{A}$ (the attack uses a known-key attack against a session, different than s, established between $\hat{B}$ and the attacker $\mathcal{M}$).

Lack of PFS. MQV does not provide Perfect Forward Secrecy (PFS). This, however, is not just a failure of MQV but it's an inherent limitation of implicitly-authenticated 2-message protocols based on public-key authentication. Indeed no such protocol can provide PFS. We present a generic attack against any such protocol where an active attacker $\mathcal{M}$ causes the establishment of a session key K at party $\hat{A}$ with peer $\hat{B}$ such that a later corruption of $\hat{B}$ (even after K was erased) allows $\mathcal{M}$ to find K.

Vulnerability to leaked ephemeral DH exponents. Another important question regarding any DH protocol, in particular MQV, is the vulnerability of the protocol to attacks that use the knowledge of the secret exponent x of an ephemeral (session-specific) DH value $X = g^x$. This is particularly significant since one of the computational advantages of these protocols is that the ephemeral (x, X) pairs can be computed off-line; but in this case these stored pairs are more vulnerable to leakage than long-term static secrets (the latter may be stored in a hardware-protected area while the ephemeral pairs will be typically stored on disk and hence more available to a temporary break or to a malicious user of the system). Also here we show that MQV fails to provide sufficient defense by demonstrating how the leakage of an exponent x can compromise the security of sessions that do not even use X.

Prime-order checks. Our description in Figure 1 omitted an element from the session-key computation in MQV: In case where the group G generated by g is a subgroup of a larger group G' (which is the case in typical mod p and elliptic curve groups), MQV specifies that the key be computed as $K = (\sigma_{\hat{A}})^h = (\sigma_{\hat{B}})^h$ where h is the co-factor $|G'|/|G|$. This measure is used to ensure that the key K belongs to the group G, and has been added to MQV as a safeguard against potential attacks resulting from the lack of explicit authentication of the DH values, e.g., small-group attacks. We note, however, that this addition is of no help against the vulnerabilities mentioned above (and as we will see is not needed to provide security in HMQV). Moreover, lacking a proof that the above counter-measure really works, several standards defining MQV, as well as various descriptions of the protocol in the literature, often specify (or at least recommend) that the parties to MQV explicitly check that the DH value presented by the peer is of prime order q. This adds a costly extra exponentiation to each peer and takes significantly from the almost-optimal performance of MQV. As we will see, HMQV "provably dispenses" of the need for this costly check.

3 The HMQV Protocol and its Proven Security

The HMQV protocol ("H" is for Hash) is a simple but powerful variant of MQV. As in MQV, its communication is identical to the basic DH exchange with the possible addition of certificates. The computation of the session key (Figure 1) differs from MQV's in two ways: (i) the computation of d and e involves the hashing of the party's own DH value and the peer's identity, and (ii) the output key is computed by hashing the value $\sigma_{\hat{A}} = \sigma_{\hat{B}}$. The output of the hash in (i) is $\ell = |q|/2$ bits while in (ii) it is of the length, denoted k, of the desired key. We denote the hash function with ℓ bits of output by $\bar{H}$ and the one with k bits by H. In practice the same hash function can be used with different output lengths. (As a mnemonics, note that the bar in $\bar{H}$ indicates that the output of the function is used as an exponent).

From this description one can see that HMQV preserves the outstanding performance of MQV (both in terms of communication and computation). At the same time, *HMQV overcomes all the mentioned security shortcomings of MQV to the largest possible extent in a 2-message protocol.* We prove that in the random oracle model [5], and under the Computational Diffie-Hellman (CDH) assumption [15], the protocol is secure in the Canetti-Krawczyk security model [11]. In particular, this establishes the security of the protocol against impersonation attacks, known-key attacks, and UKS attacks. We also prove the resistance of HMQV to KCI attacks (which we formally define) as well as its resistance to the exposure of ephemeral secret DH exponents (see Section 5 for comparison to other work). For this latter property, and only for it, we need to resort to two strong assumptions: Gap Diffie-Hellman [37] and Knowledge of Exponent (KEA1) [14, 18, 4]; in return we get a guarantee that not even the session key computed using the exposed exponent is compromised by this leakage.

Another important performance gain ensured by our analysis of HMQV is the avoidance of costly prime-order tests on the DH values transmitted in the protocol; our proof shows that the only way an attacker can benefit from the choice of rogue DH values is by choosing these to be zero, and thus a simple non-zero check is all is required (hence, there is no need for prime-order tests or for the co-factor h used in MQV).

Regarding forward secrecy, we said earlier that PFS cannot be achieved by any implicitly authenticated 2-message protocol, including HMQV. Yet, the following limited forward secrecy property holds for HMQV: any session key established without the active intervention of the attacker (except for eavesdropping the communication) is guaranteed to be irrecoverable by the attacker once the session key is erased from memory. (This is the case even if the attacker knew the private keys of both peers when the session was established.)

For applications that require full PFS we present a 3-message variant of HMQV which adds a third message and a MAC computation by each party and guarantees full PFS. This 3-message protocol, called HMQV-C, also provides "key confirmation" to both parties, i.e., the assurance that the assumed peer indeed participated in the protocol and that it computed the same session key. Another advantage of HMQV-C is that it can be proven secure in the stronger *universally composable* (UC) KE security model of [13] which ensures the security of key-exchange protocols when run concurrently with other applications. We note that while HMQV-C requires an extra message, its computational cost is essentially the same as HMQV as the MAC computation is negligible relative to the exponentiation cost. On the other hand, [23] note that 2-message symmetric protocols such as the basic HMQV allow for simultaneous initiation of a session by both $\hat{A}$ and $\hat{B}$, a desirable property in some network settings.

Another variant of HMQV is a one-pass authenticated key-exchange protocol in which $\hat{A}$ sends a single message to $\hat{B}$ after which both parties share a secret key. We show also this protocol to be secure (under CDH and in the random oracle model) in a security model adapted from [11] to one-pass protocols (the only difference is that we cannot prevent the adversarial replay of a message from $\hat{A}$ to $\hat{B}$ and, of course, cannot provide PFS). In particular, this one-pass protocol provides the functionality of public-key based deniable authentication as well as an authenticated CCA encryption scheme (in the random oracle model) in a more efficient way than existing alternatives.

We end by noting an important property of our analysis: all results in this paper hold under a strong adversarial model in which the attacker is allowed to register arbitrary public key for all corrupted parties (and at any time during the protocol). This may include a public key that is identical, or related, to the public key of another (possibly uncorrupted) party; in particular, the attacker may not know the private key corresponding to the public key it chose. In practical terms this means that the security of our protocols does not depend on the certification authority requiring registrants of public keys to prove knowledge of the corresponding private keys. This is important since in many practical settings such "proofs of possession" are not required or performed by the CA (for contrast, see the comparison with [23] in Section 5).

4 Exponential Challenge-Response Signatures

Here we introduce the Exponential Challenge-Response (XCR) Signature Scheme which is the main building block used in the design and analysis of the HMQV protocol. As in a regular digital signature scheme, in a challenge-response signature scheme a signer has a pair of private and public keys used for generation and verification, respectively, of signatures. However, in contrast to regular signatures, challenge-response signatures are inherently interactive and require the recipient (i.e., the verifier) of a signature to issue a *challenge* to the signer before the latter can generate the signature on a given message. A *secure* challenge-response signature scheme needs to guarantee that no one other than the legitimate signer be able to generate a signature that will convince the challenger to accept it as valid (in particular, a signature is not only message-specific but also challenge-specific). On the other hand, we are only interested to ensure verifiability of the signature by the challenger, and thus we make no assumptions or requirements regarding the transferability, or verifiability by a third party, of the signature. Moreover, in the scheme described below the party that chooses the challenge can always generate a signature, on any message, which is valid with respect to that particular challenge. What is even more important for our application (and differentiates our scheme from other interactive signatures) is the fact that the verifier can compute, using the challenge, the *same signature string* as the signer.

While the above description may serve as a basis for a general definition of challenge-response signatures, we omit here such a general treatment in favor of a more focused description of the specific challenge-response signature used in this work. In particular, the definition of security is simplified by tailoring it to this specific scheme.

As before, we use g to denote a generator of a group G of prime order q, and $\bar{H}$ to denote a hash function that outputs $\ell = |q|/2$ bits. Our results require the following assumption (our treatment of polynomial-time, asymptotics, etc. is very informal, and uses $|q|$ as an implicit security parameter).

The CDH Assumption. For two elements $U = g^u, V = g^v$ in G we denote by $CDH(U, V)$ the result of applying the Diffie-Hellman computation (wrt to generator g) to U and V, i.e., $CDH(U, V) = g^{uv}$. An algorithm is called a CDH solver for G if it takes as input pairs of elements (U, V) in G and a generator g of G and outputs the Diffie-Hellman result $CDH(U, V)$ wrt g. We say that the Computational Diffie-Hellman (CDH) assumption holds in the group $G = \langle g \rangle$ if for all probabilistic polynomial-time CDH solvers for G, the probability that on a pair (U, V), for $U, V \in_{\mathrm{R}} G$, the solver computes the correct value $CDH_g(U, V)$ is negligible.

4.1 Definition of the XCR Signature Scheme

Definition 1. The exponential challenge-response (XCR) signature scheme. *The signer in a* XCR *scheme, denoted by $\hat{B}$, possesses a private key $b \in_R Z_q$ and a public key $B = g^b$. A verifier (or challenger), denoted $\hat{A}$, provides a message m for signature by $\hat{B}$ together with a challenge X which $\hat{A}$ computes as $X = g^x$ for $x \in_R Z_q$ (x is chosen, and kept secret, by $\hat{A}$). The signature of $\hat{B}$ on m using challenge X is defined as a pair $(Y, X^{y+\bar{H}(Y,m)b})$, where $Y = g^y$ and $y \in_R Z_q$ is chosen by $\hat{B}$. The verifier $\hat{A}$ accepts a signature pair (Y, σ) as* valid *(for message m and with respect to challenge $X = g^x$) if and only if it holds that $Y \neq 0$ and $(YB^{\bar{H}(Y,m)})^x = \sigma$).*

Notation: For message m, challenge X, and value Y we define $XSIG_{\hat{B}}(Y, m, X) \stackrel{\text{def}}{=} X^{y+\bar{H}(Y,m)b}$ (i.e., $XSIG_{\hat{B}}$ denotes the second element in an XCR *signature pair).*

Relation between XCR and Schnorr's scheme. The main motivation for introducing the XCR scheme comes from its use in our design and analysis of the HMQV protocol [28]. By now, however, it may be illustrative to motivate this scheme via its relation to the Schnorr's identification scheme from which the XCR scheme is derived. We sketch this relation next. Schnorr's (interactive) identification scheme consists of a proof of knowledge of the discrete logarithm b for a given input $B = g^b$. Let $\hat{B}$ denote the prover in this scheme (that possesses

b) and $\hat{A}$ the verifier (that is given the input B). The basic Schnorr's identification consists of three messages: (i) $\hat{B}$ chooses $y \in_{\text{R}} Z_q$ and sends $Y = g^y$ to $\hat{A}$; (ii) $\hat{A}$ responds with a random value $e \in_{\text{R}} Z_q$; and (iii) $\hat{B}$ sends $\hat{A}$ the value $s = y + eb$. $\hat{A}$ accepts if and only if $g^s = YB^e$ holds. This protocol is a Arthur-Merlin zero-knowledge proof of knowledge (of b) for an honest verifier $\hat{A}$ (i.e., one that chooses e uniformly at random). Therefore, it can be transformed via the Fiat-Shamir methodology into a signature scheme, namely $\text{SIG}_{\hat{B}}(m) = (Y, y + \bar{H}(Y, m)b)$, that is provably secure in the random oracle model [38].

Now consider the following 4-message variant of Schnorr's protocol in which a first message from $\hat{A}$ to $\hat{B}$ is added. In this first message $\hat{A}$ sends to $\hat{B}$ a value $X = g^x$. Then the 3 messages from Schnorr's scheme follow, except that in message (iii) (the fourth message in the modified protocol) rather than sending $s = y + eb$ to $\hat{A}$, $\hat{B}$ sends $S = X^s$. $\hat{A}$ accepts if and only if $S = (YB^e)^x$. It can be shown that this protocol is a proof of the "ability" of $\hat{B}$ to compute $CDH(B, X)$ for any value $X \in G$. Moreover, the protocol is zero-knowledge against a verifier $\hat{A}$ that chooses e at random (while X may be chosen arbitrarily). Now, note that applying the Fiat-Shamir transformation to this protocol one obtains the challenge-response signature XCR.[1] This also explains why we use the term "exponential" in naming the XCR scheme: it refers to the replacement of $s = y + eb$ in the Schnorr scheme with X^s in the last message of the protocol.

Next we establish our security requirement from the XCR scheme.

Definition 2. Security of the XCR signature scheme. *We say that the* XCR *challenge-response signature scheme is secure if no polynomial-time machine $\mathcal{F}$ can win the game in Figure 2 with non-negligible probability.*

Remarks on Definition 2.

1. Note that in order to be successful the forger has to use the input X_0 in its forgery. This captures the fact that XCR signatures are only unforgeable with respect to a challenge not chosen by the attacker.
2. According to our definition, if $\hat{B}$ outputs a signature (Y, σ) on a message m wrt challenge X, and the forger can find a signature (Y', σ') for the same m wrt the same challenge X but with $Y' \neq Y$, then we consider $\mathcal{F}$ successful (namely, finding a second signature for the same message is considered a valid forgery). In some sense, we look at these signatures as signing both m and Y. This property may not be essential in other applications of challenge-response signatures but it is crucial for the application to HMQV security in this paper.
3. We do not ask that $\mathcal{F}$ will always output good forgeries; it can output "fail" or even invalid triples. The only requirement is that with non-negligible

[1] Note that if in Schnorr's protocol one chooses e in $\{0,1\}^\ell$ (rather than from Z_q) the protocol remains valid except that the soundness is limited by $2^{-\ell}$. This is the basis for our choice of $\ell = |q|/2$ as the output length of $\bar{H}(Y, m)$, namely, a trade-off between efficiency and security. See Remark 1 for a more accurate discussion.

Forger $\mathcal{F}$ in Definition 2

1. $\mathcal{F}$ is given values B, X_0 where $B, X_0 \in_{\mathrm{R}} G$.
2. $\mathcal{F}$ is given access to a signing oracle $\hat{B}$ (representing a signer $\hat{B}$ with private key b and public key B) which on query (X, m) outputs a pair $(Y, XSIG_{\hat{B}}(Y, m, X))$ where $Y = g^y, y \in_{\mathrm{R}} Z_q$, is chosen by $\hat{B}$ afresh with each query.
3. $\mathcal{F}$ is allowed a polynomial number of queries to $\hat{B}$ where the queries are chosen (possibly adaptively) by $\mathcal{F}$.
4. $\mathcal{F}$ halts with output "fail" or with a *guess* in the form of a triple (Y_0, m_0, σ).

$\mathcal{F}$'s guess is called a (successful) *forgery* if the following two conditions hold:

(a) The pair (Y_0, σ) is a valid XCR signature of $\hat{B}$ on message m_0 with respect to challenge X_0 (i.e., $Y_0 \neq 0$ and $\sigma = XSIG_{\hat{B}}(Y_0, m_0, X_0)$; also note that the value of X_0 is the one received by $\mathcal{F}$ as input).

(b) The pair (Y_0, m_0) did not appear in any of the responses of $\hat{B}$ to $\mathcal{F}$'s queries.

We say that $\mathcal{F}$ wins the game (or simply *forges*) if it outputs a successful forgery.

Fig. 2. Forgery Game for XCR Signatures

probability (over the distribution of inputs to $\mathcal{F}$, the choices by the random oracle $\bar{H}$, the coins of $\mathcal{F}$, and the coins of $\hat{B}$) $\mathcal{F}$ output a successful forgery.

4. Note that we only restricted Y_0 to be non-zero. In particular, we require no check that Y_0 be of prime order q (only that it represents a non-zero element for which the group operation is defined). This is an important aspect of XCR signatures. In particular, the requirement to run a prime-order test would translate into an additional exponentiation for each party in the HMQV protocol, thus degrading significantly the "almost optimal" performance of the protocol.

4.2 Proof of Unforgeability for the XCR Signature Scheme

The following theorem states the security of the XCR scheme in the random oracle model under the CDH assumption; it constitutes the basis for the proof of security of protocol HMQV.

Theorem 1. *Under the CDH assumption, the* XCR *signature scheme is secure (according to Definition 2) in the random oracle model.*

Proof. Given an efficient and successful forger $\mathcal{F}$ against the XCR signature scheme (i.e., $\mathcal{F}$ wins the forgery game from Definition 2 with non-negligible probability), we build an efficient solver $\mathcal{C}$ for the CDH problem, namely, $\mathcal{C}$ gets as input a pair of random elements U, V in G, and outputs the value $CDH(U, V)$ with non-negligible probability. Unsuccessful runs of $\mathcal{C}$ may end with "fail" or just the wrong value of $CDH(U, V)$. Using results by Maurer and Wolf [32] and Shoup [40] such a “faulty CDH solver” can be transformed, using the self-reducibility properties of the CDH problem, into an efficient algorithm that solves CDH for every input U, V with only negligible probability of error.

Algorithm $\mathcal{C}$ is presented in Figure 3, and it follows a mostly standard argument for Fiat-Shamir type signatures. The idea is that if $\mathcal{F}$ can succeed in forging a signature with a pair (Y_0, m_0) and a given value $\bar{H}(Y_0, m_0)$ output by the function $\bar{H}$, then $\mathcal{F}$ is likely to succeed also when $\bar{H}(Y_0, m_0)$ is set to a *different* random value. Using this property, we construct $\mathcal{C}$ such that after running $\mathcal{F}$ twice, $\mathcal{C}$ obtains (with non-negligible probability) two forgeries with the same pair (Y_0, m_0) but different values of $\bar{H}(Y_0, m_0)$. Now, using these two forgeries $\mathcal{C}$ is able to compute $CDH(U, V)$.

Examining the specification of $\mathcal{C}$ in more detail, first note that in the run of $\mathcal{F}$ by $\mathcal{C}$ all queries to the signer $\hat{B}$ are answered by $\mathcal{C}$ without knowledge of the private key b, and without access to an actual signing oracle for $\hat{B}$. Instead all these answers are simulated by $\mathcal{C}$ in steps S1-S3. It is easy to see that this is a perfect simulation of the XCR signature generation algorithm under private key b except for the following deviation that happens with negligible probability: In step S3 of the simulation, $\mathcal{C}$ does not complete the run of $\mathcal{F}$ if the value (Y, m) was queried earlier from $\bar{H}$. However, since the value Y, as generated by $\mathcal{C}$, is distributed uniformly over G and chosen independently of previous values in the protocol, then the probability that the point (Y, m) was queried earlier from $\bar{H}$ is at most Q/q where Q is an upper bound on the number of queries to $\bar{H}$ that occur in a run of $\mathcal{C}$.

Therefore, the probability of $\mathcal{F}$ outputting a successful forgery in the run under $\mathcal{C}$ is the same, up to a negligible difference, as in a real run of $\mathcal{F}$, and therefore non-negligible. In particular, when such a successful forgery is output by $\mathcal{F}$ then conditions F1 and F2 checked by $\mathcal{C}$ necessarily hold. Condition F3 also holds except for probability $2^{-\ell}$, i.e., the probability that $\mathcal{F}$'s forgery is correct when it did not query $\bar{H}(Y_0, m_0)$. To see this, note that if one fixes the pair (Y_0, m_0) and the challenge X_0, then the signature produced with $e = \bar{H}(Y_0, m_0)$ is necessarily different than the signature produced with $e' = \bar{H}(Y_0, m_0)$ if $e \neq e' \pmod{q}$. Hence, the probability that $\mathcal{F}$ will guess the right signature without querying $\bar{H}(Y_0, m_0)$ is at most as the probability of guessing the value of $\bar{H}(Y_0, m_0)$, i.e., $2^{-\ell}$. Since conditions F1-F3 determine the run of the ”repeat experiment” then in all we have that the simultaneous probability that $\mathcal{F}$ outputs a correct forgery in a run under $\mathcal{C}$ AND that $\mathcal{C}$ executes the “repeat experiment” in that run is non-negligible.

Now, using the Forking Lemma[2] from [38] we obtain that the probability that in the “repeat experiment” $\mathcal{F}$ will output a correct forgery for the pair (Y_0, m_0) given that $\mathcal{F}$ did so in the first run is non-negligible. Moreover, in such a case

[2] We note that the formulation of the Forking Lemma in [38] is very general and fits exactly our application: namely, it applies to signatures of the form $(m, Y, \bar{H}(Y, m), \sigma)$ derived via the Fiat-Shamir methodology, provided that the signer is simulatable without knowledge of its private key (the use of the challenge X in our case does not affect the validity of the Forking Lemma argument).

Building a CDH solver $\mathcal{C}$ from an XCR forger $\mathcal{F}$

Setup. Given a successful XCR-forger $\mathcal{F}$ we build an algorithm $\mathcal{C}$ to solve the CDH problem. The inputs to $\mathcal{C}$ are random values $U = g^u, V = g^v$ in G. $\mathcal{C}$'s goal is to compute $CDH(U, V) = g^{uv}$.

$\mathcal{C}$'s actions. $\mathcal{C}$ sets $B = V$ and $X_0 = U$, and runs the forger $\mathcal{F}$ on input (B, X_0) against a signer $\hat{B}$ with public key B. $\mathcal{C}$ provides $\mathcal{F}$ with a random tape and provides random answers to the $\bar{H}$ queries generated in the run (if the same $\bar{H}$ query is presented more than once $\mathcal{C}$ answers it with the same response as in the first time).

Each time $\mathcal{F}$ queries $\hat{B}$ for a signature on values (X, m) chosen by $\mathcal{F}$,
$\mathcal{C}$ answers the query for $\hat{B}$ as follows (note that $\mathcal{C}$ does not know b):

S1. Chooses $s \in_{\mathrm{R}} Z_q, e \in_{\mathrm{R}} \{0, 1\}^{\ell}$.
S2. Sets $Y = g^s/B^e$.
S3. Sets $\bar{H}(Y, m) = e$ (if $\bar{H}(Y, m)$ was defined by a previous query to $\bar{H}$, $\mathcal{C}$ aborts its run and outputs "fail").
$\mathcal{C}$ responds to $\mathcal{F}$'s query with the signature pair (Y, X^s)

When $\mathcal{F}$ halts $\mathcal{C}$ checks whether the three following conditions hold:

F1. $\mathcal{F}$ output a guess (Y_0, m_0, σ), $Y_0 \neq 0$.
F2. The pair (Y_0, m_0) was not used as the (Y, m) pair in any of the signatures generated by $\hat{B}$.
F3. The value $\bar{H}(Y_0, m_0)$ was queried from the random oracle $\bar{H}$.

If the three conditions hold, then $\mathcal{C}$ proceeds to the "repeat experiment" below; in all other cases $\mathcal{C}$ halts and outputs "fail".

The repeat experiment. $\mathcal{C}$ runs $\mathcal{F}$ again for a second time under the same input (B, X_0) and using the same coins for both $\mathcal{C}$ and $\mathcal{F}$. The difference between the two runs is in the way in which $\mathcal{C}$ answers the $\bar{H}$ queries during the second run. Specifically, all queries to $\bar{H}$ performed before the $\bar{H}(Y_0, m_0)$ query are answered identically as in the first run. The query $\bar{H}(Y_0, m_0)$, however, is answered with a new independent value $e' \in_{\mathrm{R}} \{0, 1\}^{\ell}$. Subsequent queries to $\bar{H}$ are also answered at random from $\{0, 1\}^{\ell}$, independently of the responses provided in the first run.

Output. If at the end of the second run, $\mathcal{F}$ outputs a guess (Y_0, m_0, σ') (with same (Y_0, m_0) as in the first run) and $e \neq e'$, then $\mathcal{C}$ computes the value $W = (\sigma/\sigma')^{(e-e')^{-1}}$ and outputs W as its guess for $CDH(U, V)$; otherwise $\mathcal{C}$ outputs "fail".

Fig. 3. Proof of Theorem 1: Reduction from CDH to XCR forgeries

we are guaranteed that the forgeries in the first and second runs use *different* values e, e' of $\bar{H}(Y_0, m_0)$.

We now proceed to show that in the case that both the first run and the repeat experiment end up with two valid forgeries for the same pair (Y_0, m_0), and $e \neq e'$ (which happens with probability $1 - 2^{-\ell}$), then the value W computed by $\mathcal{C}$ equals $CDH(X_0, B)$. Indeed, a simple computation shows that, if $Y_0 \neq 0$

as necessary for a valid forgery, then writing $X_0 = g^{x_0}$ we have:

$$W = \left(\frac{\sigma}{\sigma'}\right)^{\frac{1}{e-e'}} = \left(\frac{(YB^e)^{x_0}}{(YB^{e'})^{x_0}}\right)^{\frac{1}{e-e'}} = (B^{(e-e')x_0})^{\frac{1}{e-e'}} = B^{x_0} = CDH(X_0, B).$$

Now, since X_0 and B are, respectively, the inputs U and V provided to $\mathcal{C}$, then we get that in this case (which happens with non-negligible probability) $\mathcal{C}$ has successfully computed $CDH(U, V)$. □

Remark 1. (Number of bits in $\bar{H}(Y, m)$). Let ℓ be the number of bits in the output of $\bar{H}(Y, m)$. Clearly, the smaller ℓ the more efficient the signature scheme is; on the other hand, a too small ℓ implies a bad security bound (since once the exponent $\bar{H}(Y, m)$ is predictable the signature scheme is insecure). But how large a ℓ do we need for security purposes? Here we see that setting $\ell = \frac{1}{2}|q|$, as we specified for XCR signatures (and for its application to the HMQV protocol), provides the right performance-security trade-off. In order to assess the level of security that ℓ provides to the XCR signatures (and consequently to HMQV), we note that there are two places in the above proof where this parameter ℓ enters the analysis. One is when bounding the probability that the attacker could forge a signature with parameters (Y_0, m_0) without querying $\bar{H}$ on this pair. As we claimed the probability in this case is $2^{-\ell}$ (or $1/\sqrt{q}$ using the fact that we defined $\ell = |q|/2$). The other use of ℓ is in the proof of the Forking Lemma by Pointcheval and Stern [38]. When written in terms of XCR signatures Lemma 9 and Theorem 10 from [38] show that given a forger against XCR signatures that works time T, performs Q queries to $\bar{H}$ and forges with probability ε, one can build a CDH solver that runs expected time $c\frac{Q}{\varepsilon}T$ provided that $\varepsilon \geq \frac{c'Q}{2^\ell}$ $(c, c'$ are constants). Now, since we know how to build CDH solvers that run time $\sqrt{q}$ (e.g., Shanks algorithm) then the above analysis tells us something significant only when $c\frac{Q}{\varepsilon}T << \sqrt{q}$, in particular $Q/\varepsilon << \sqrt{q}$. From this and the condition $\varepsilon \geq c'Q/2^\ell$ we get that we need $Q/\varepsilon < \min\{\sqrt{q}, 2^\ell\}$. Since this is the only constraint on ℓ we see that choosing ℓ such that $2^\ell > \sqrt{q}$ does not add to the security of the scheme, and therefore setting $\ell = \frac{1}{2}|q|$ provides the best trade-off between security and performance. (Note that the same length consideration applies to the parameter e in the modified Schnorr's identification scheme described following Definition 1). We also comment, independently from the above considerations on ℓ, that the above constraint $Q < \varepsilon\sqrt{q}$ also guarantees that the simulation error in step S3 of Figure 3 (which we showed in the proof of Theorem 1 to be at most Q/q) is no more than $1/\sqrt{q}$.

Remark 2. (*A non-interactive* XCR *variant.)* XCR signatures can be made non-interactive, but verifier-specific, by putting $X = A$, where A is a public key of the verifier. In this case the signature will be a pair (Y, t) where t is a MAC tag computed on the signed message using a key derived by hashing $XSIG_{\hat{A}}(Y, \text{“}\hat{A}\text{”}, A)$. This provides for a very efficient non-interactive verifier-specific deniable authentication mechanism. It does *not* provide for a universally-verifiable non-repudiable signature.

Remark 3. (*HCR and DSS signatures.*) We do not know whether XCR signatures remain secure if the exponent y corresponding to a value Y used by $\hat{B}$ in a signature is revealed to the forger (note that in this case the simulation steps S1-S3 in Figure 3 do not work). On the other hand, if one modifies the definition of XCR such that the $XSIG_{\hat{B}}$ component is replaced with a hash of this value (note that the signature is still verifiable by the challenger) then one obtains a signature scheme in which revealing y does not help the forger. More precisely, in [28] we study these signatures, which we call HCR, in detail and show that under the Gap Diffie-Hellman and KEA1 assumptions they are unforgeable in the random oracle model even if y is revealed to the attacker. As a result, HCR signatures provide for a more secure alternative to DSS signatures as they resolve the main DSS vulnerability by which the disclosure of a single ephemeral exponent (i.e., k in the component $r = g^k$ of a DSS signature) suffices to reveal the signature key. On the other hand, HCR signatures are verifier-specific and require interaction (or the possession of a public key by the verifier as in Remark 2), and do not provide for third-party verifiability (a property that may be a bug or a feature of HCR depending on the application).

4.3 Dual XCR Signatures (DCR)

An important property of XCR signatures is that the challenger (having chosen the challenge) can compute the signature by itself. Here we show how to take advantage of this property in order to derive a related challenge-response signature scheme (which we call the "dual XCR scheme", or DCR for short) with the property that any two parties, $\hat{A}, \hat{B}$, can interact with each other with the dual roles of challenger and signer, and each produce a signature that no third party can forge. Moreover, *and this is what makes the scheme essential to the HMQV protocol,* the resultant signatures by $\hat{A}$ and by $\hat{B}$ have the *same value.* (More precisely, they have the same *XSIG* component.)

Definition 3. The dual (exponential) challenge-response (DCR) signature scheme. *Let $\hat{A}, \hat{B}$ be two parties with public keys $A = g^a, B = g^b$, respectively. Let m_1, m_2 be two messages. The* dual XCR signature (DCR for short) *of $\hat{A}$ and $\hat{B}$ on messages m_1, m_2, respectively, is defined as a triple of values: X, Y and $DSIG_{\hat{A},\hat{B}}(m_1, m_2, X, Y) \stackrel{\text{def}}{=} g^{(x+da)(y+eb)}$, where $X = g^x, Y = g^y$ are challenges chosen by $\hat{A}$ and $\hat{B}$, respectively, and the symbols d and e denote $\bar{H}(X, m_1)$ and $\bar{H}(Y, m_2)$, respectively.*

As said, a fundamental property of a DCR signature is that after exchanging the values X and Y (with x and y chosen by $\hat{A}$ and $\hat{B}$, respectively), both $\hat{A}$ and $\hat{B}$ can compute (and verify) the **same** signature $DSIG_{\hat{A},\hat{B}}(m_1, m_2, X, Y)$. This can be seen from the identities:

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

Moreover, as shown next the attacker cannot feasibly compute this signature.

The security of DCR signatures. Roughly speaking, a dual signature is an XCR signature by $\hat{A}$ on message m_1, under challenge YB^e, and at the same time an XCR signature by $\hat{B}$ on message m_2, under challenge XA^d. More precisely, since the values d and e are determined during the signature process (via the possibly adversarial choice of messages m_1, m_2), then we will say that a DCR signature of $\hat{B}$ is secure (with respect to A) if no efficient attacker can win, with non-negligible probability, the game of Figure 2 with the following modifications. In step 2, the queries to $\hat{B}$ are of the form (X, m, m_1) and the signature by $\hat{B}$ is the pair $(Y, \mathit{XSIG}_{\hat{B}}(Y, m, XA^d))$ where Y is chosen by $\hat{B}$ and $d = \bar{H}(X, m_1)$. A successful forgery is a quadruple (Y_0, m_0, m_1, σ) where $\sigma = \mathit{XSIG}_{\hat{B}}(Y_0, m_0, X_0 A^d)$, $Y_0 \neq 0$, the pair (Y_0, m_0) satisfies the validity requirement (b) from Figure 2, and m_1 is an arbitrary message chosen by $\mathcal{F}$. We say that the dual signature of $\hat{B}$ is secure if it is secure *with respect to any value $A = g^a$ not chosen by the attacker.*

Theorem 2. *Let $\hat{A}, \hat{B}$ be two parties with public keys $A = g^a, B = g^b$, resp. Under the CDH assumption, the* DCR *signature of $\hat{B}$ with respect to A is secure even if the forger is given the private key a of $\hat{A}$ (but not the private key of $\hat{B}$).*

Proof. The same proof of unforgeability of XCR (Theorem 1) works here with a modified computation of W as specified below. First note that since the DCR signature of $\hat{B}$ now involves the value $d = \bar{H}(X_0, m_1)$, where m_1 is a message that $\mathcal{F}$ may choose at will, then the value of m_1 chosen by $\mathcal{F}$ before the repeat experiment may differ from the value of m_1 chosen during the repeat experiment. In this case we get two different values, d, d', used in the σ and σ' signatures. Also, note that even with a single value of d the specification of W in the proof of Theorem 1 would result in the value $(XA^d)^b$ rather than X^b (as required in order to solve the CDH problem on inputs $U = X, V = B$). We deal with these two issues by redefining W as follows (here we use the fact that a, the private key of $\hat{A}$, is known to $\mathcal{C}$):

$$W = \left(\frac{\sigma/(YB^e)^{da}}{\sigma'/(YB^{e'})^{d'a}} \right)^{\frac{1}{e-e'}} \tag{1}$$

The rest of the proof remains unchanged. □

HMQV explained. The HMQV protocol consists of an exchange between parties $\hat{A}$ and $\hat{B}$ of DH values $X = g^x$ and $Y = g^y$ that serve as challenges from which both parties compute the dual XCR signature $\mathit{DSIG}_{\hat{A},\hat{B}}(``\hat{A}", ``\hat{B}", X, Y) = g^{(x+da)(y+eb)}$. The session key is then derived by hashing this value. In this way the signature itself need not be transmitted: it is the uniqueness of the signature that ensures a common derived value for the session key, and it is the *ability to compute the key (equivalently, the signature)* that provides for a proof that the exchange was carried by the alleged parties $\hat{A}$ and $\hat{B}$. Moreover, since the messages m_1, m_2 on which the signature is computed are the identities of the peers, both parties get assurance that the key they computed is uniquely bound to the correct identities (this is essential to avoid some authentication failures such as the UKS attacks). We end by noting that while the casting of the HMQV design in terms of DCR signatures is the main conceptual contribution of our work, showing that this idea indeed works for proving the security of the protocol turns out to be technically challenging (see [28] for the gory details).

5 Related Work.

There is a large number of implicitly-authenticated DH protocols discussed in the literature. See [9, 34, 7, 8] for some surveys which also include the discussion of desirable security goals for these protocols as well as some of the weaknesses of specific proposals. Two works that study such protocols in a formal model are those of Blake-Wilson et al [8] and Jeong et al [23]. They both treat very similar protocols referred to in the literature as "unified model". In these protocols, parties $\hat{A}$ and $\hat{B}$ use their corresponding public keys g^a, g^b to generate a shared key g^{ab} that they then use to authenticate a DH exchange (as in [26]).

The variant studied in [8] is shown to be open to interleaving and known-key attacks and hence insecure (unfortunately, this variant has been widely standardized [2, 3, 19]). One main flaw of this protocol is that it does not explicitly authenticate (or includes under the key derivation) the ephemeral DH values exchanged by the parties. [23] studies the version in which the DH values are included under the key derivation and shows this protocol to be secure in the random oracle model.[3] However, the protocol does not provide resistance to KCI and is open to a UKS attack if the CA does not enforce a proof-of-possession check at the time of certificate issuance. Lack of KCI is one aspect of a more substantial drawback of these protocols, namely, the use of the keys g^{ab} as long-term shared keys between the parties; these keys become particularly vulnerable when cached for efficiency.

In all, HMQV is a much stronger protocol which, in particular, does not use g^{ab} as a long-term key, does not require (even for efficiency) to cache this value, and even if the value of g^{ab} is ever learned by the attacker it is of no help for impersonating either $\hat{A}$ or $\hat{B}$, or for learning anything about their session keys. On top of all its security advantages (which hold without relying on proofs of possession performed by CAs or prime-order tests performed by the parties), HMQV is more efficient than the unified model protocols that take 3 exponentiations per-party.

Finally, we mention the works of Shoup [41] and Jeong et al [23] that present 2-message authenticated DH exchanges with *explicit* authentication (via signatures and MAC, respectively) that they show to satisfy the security definitions from [41, 6] in the standard (non-random-oracle) model. In these protocols, however, it is sufficient for the attacker to learn a single ephemeral exponent x of a DH value g^x, exchanged between parties $\hat{A}$ and $\hat{B}$, to be able to impersonate $\hat{A}$ to

[3] [23] claim their protocol to enjoy perfect forward secrecy (PFS), but what they actually show is a weaker and non-standard notion (the same we outlined in Section 3) where PFS holds only for sessions created without active intervention by the attacker. As we pointed out, this is an inherent limitation of any 2-message implicitly-authenticated DH exchange, including the one-round protocols from [23].

$\hat{B}$ indefinitely, and without ever having to learn $\hat{A}$'s or $\hat{B}$'s private keys. This is a serious security weakness which violates the basic principle that the disclosure of ephemeral session-specific information should not compromise other sessions (see Section 2.2). The reason that these protocols could be proven secure in [41, 6] is that the models of key exchange security considered in these works do not allow the attacker to find any session-specific information beyond the session key itself. The above vulnerability, however, indicates that such models are insufficient to capture some realistic attack scenarios. In contrast, the model of [11], used as the basis for our analysis, captures such attacks via state-reveal queries (see [28]).

6 Concluding Remarks

The results in this paper show multiple vulnerabilities of MQV to known-key and other attacks (in the case of the 3-message variant of MQV some of these vulnerabilities depend on the ability of the attacker to access ephemeral state information for incomplete sessions). The extent to which these weaknesses are exploitable in practice depends of course on the application, the computing and communication environment, threat model, etc. These results do *not* mean that applications already using MQV are necessarily insecure in their specific environments. At the same time, the identified weaknesses should not be dismissed as theoretical only, especially when considering that MQV has been (and is being) standardized as a key-exchange protocol for use in heterogeneous and unknown scenarios. This is particularly true when, as shown here, these weaknesses are not inherent to the problem being solved nor to the formal analytical setting.

Indeed, HMQV provides the same functionality with the same (or even better) performance of MQV while enjoying a full proof of security. Two caveats regarding this proof are the use of the idealized random oracle methodology and the significant (though polynomially bounded) reduction cost. Other proven protocols (such as SKEME [26], ISO [22, 11] and SIGMA [27, 12]), while less efficient, enjoy less expensive reductions and do not directly require random oracles. We note, however, that dispensing of random oracles in these protocols requires the use of the more expensive (and seldom used in practice) signature and encryption schemes that do not rely on random oracles, and also requires the stronger DDH assumption. Certainly, coming up with a protocol that offers the many attractive security and performance properties of HMQV and does not rely on the random oracle model in its analysis is an important open question.

We end by stressing that in spite of the weaknesses demonstrated here, the MQV protocol contains some remarkable ideas without which the design of HMQV would have not been possible. Nor would this design have been possible without the rigorous examination of these ideas in a formal framework such as the one in [11]. The design of HMQV is a demonstration of the strength of "proof-driven designs" which guide us in choosing the necessary design elements of the protocol while dispensing of unnecessary "safety margins". As a result, one obtains solutions that are not only cryptographically sound but are also more efficient

References

1. M. Abdalla, M. Bellare and P. Rogaway, “The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES”, *CT-RSA 2001.* LNCS 2020.
2. American National Standard (ANSI) X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography.
3. American National Standard (ANSI) X9.63: Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport using Elliptic Curve Cryptography.
4. M. Bellare and A. Palacio, “The Knowledge-of-Exponent Assumptions and 3-round Zero-Knowledge Protocols”, *Crypto'04*, LNCS 3152, 2004, pp. 273–289.
5. M. Bellare and P. Rogaway, “Random Oracles are Practical: A Paradigm for Designing Efficient Protocols”, *First ACM Conference on Computer and Communications Security*, pp. 62–73, 1993.
6. M. Bellare and P. Rogaway, “Entity authentication and key distribution”, *Crypto'93*, LNCS 773, 1994, pp. 232-249.
7. S. Blake-Wilson and A. Menezes, “Authenticated Diffie-Hellman Key Agreement Protocols”, *Proceedings of SAC '99*, Lecture Notes in Computer Science, 1556, 1999.
8. S. Blake-Wilson, D. Johnson and A. Menezes, “Key exchange protocols and their security analysis,” *6th IMA International Conf. on Cryptography and Coding*, 1997.
9. C. Boyd and A. Mathuria, *Protocols for Authentication and Key Establishment*, Springer, 2003.
10. R. Canetti, “Universally Composable Security: A New paradigm for Cryptographic Protocols”, *42nd FOCS,* 2001.
11. Canetti, R., and Krawczyk, H., “Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels”, Eurocrypt'2001, LNCS Vol. 2045.
    Full version in: `http://eprint.iacr.org/2001/040`.
12. Canetti, R., and Krawczyk, H., “Security Analysis of IKE's Signature-based Key-Exchange Protocol”, *Crypto 2002.* LNCS Vol. 2442.
13. Canetti, R., and Krawczyk, H., “Universally Composable Notions of Key Exchange and Secure Channels”, *Eurocrypt 02,* 2002. Full version available at `http://eprint.iacr.org/2002/059.`
14. I. Damgård, “Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks”, Crypto'91, LNCS Vol. 576.
15. W. Diffie and M. Hellman, “New Directions in Cryptography”, IEEE Trans. Info. Theor. 22, 6 (Nov 1976), pp. 644–654.
16. W. Diffie, P. van Oorschot and M. Wiener, “Authentication and authenticated key exchanges”, *Designs, Codes and Cryptography*, 2, 1992, pp. 107–125.
17. C. Dwork, M. Naor and A. Sahai, “Concurrent Zero-Knowledge”, *STOC'98*, pp. 409–418.
18. Hada, S. and Tanaka, T., “On the Existence of 3-round Zero-Knowledge Protocols”, Crypto'98, LNCS Vol. 1462.
19. IEEE 1363-2000: Standard Specifications for Public Key Cryptography.
20. D. Harkins and D. Carrel, “The Internet Key Exchange (IKE)”, RFC 2409, Nov. 1998.
21. ISO/IEC IS 15946-3 “Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 3: Key establishment”, 2002.

22. ISO/IEC IS 9798-3, "Entity authentication mechanisms — Part 3: Entity authentication using asymmetric techniques", 1993.
23. Ik Rae Jeong, Jonathan Katz, Dong Hoon Lee, "One-Round Protocols for Two-Party Authenticated Key Exchange", ACNS 2004: 220-232
24. B. Kaliski, "An unknown key-share attack on the MQV key agreement protocol", *ACM Transactions on Information and System Security (TISSEC)*, Vol. 4 No. 3, 2001, pp. 275–288.
25. J. Katz, "Efficient and Non-Malleable Proofs of Plaintext Knowledge and Applications", EUROCRYPT'03, LNCS 2656.
26. H. Krawczyk, "SKEME: A Versatile Secure Key Exchange Mechanism for Internet,", *1996 Internet Society Symposium on Network and Distributed System Security,* pp. 114-127, Feb. 1996.
27. H. Krawczyk, "SIGMA: The 'SiGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols", *Crypto '03*, LNCS No. 2729. pp. 400–425, 2003.
28. H. Krawczyk, "HMQV: A High-Performance Secure Diffie-Hellman Protocol" (full version). `http://eprint.iacr.org/2005/`
29. H. Krawczyk, "On the Security of Implicitly-Authenticated Diffie-Hellman Protocols", work in progress.
30. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "An efficient Protocol for Authenticated Key Agreement", *Designs, Codes and Cryptography, 28, 119-134, 2003.*
31. T. Matsumoto, Y. Takashima, and H. Imai, "On seeking smart public-key distribution systems", *Trans. IECE of Japan*, 1986, E69(2), pp. 99-106.
32. U. Maurer and S. Wolf, "Diffie-Hellman oracles", CRYPTO '96, LNCS, vol. 1109. 1996, pp. 268-282.
33. A. Menezes, M. Qu, and S. Vanstone, "Some new key agreement protocols providing mutual implicit authentication", *Second Workshop on Selected Areas in Cryptography (SAC 95)*, pp. 22–32, 1995.
34. A. Menezes, P. Van Oorschot and S. Vanstone, "Handbook of Applied Cryptography," CRC Press, 1996.
35. NIST Special Publication 800-56 (DRAFT): Recommendation on Key Establishment Schemes. Draft 2, Jan. 2003.
36. "NSAs Elliptic Curve Licensing Agreement", presentation by Mr. John Stasak (Cryptography Office, National Security Agency) to the IETF's Security Area Advisory Group, Nov 2004.
`http://www.machshav.com/~smb/saag-11-2004/NSA-EC-License.pdf`
37. T. Okamoto and D. Pointcheval, "The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes", *PKC 2001*, LNCS 1992. pp 104–118, 2001.
38. D. Pointcheval and J. Stern, "Security Arguments for Digital Signatures and Blind Signatures", *J.Cryptology* (2000) 13:361-396.
39. M.O. Rabin, "Digitalized Signatures", Foundations of Secure Computing, DeMillo-Dobkins-Jones-Lipton, editors, 155-168, Academic Press, 1978.
40. V. Shoup, "Lower Bounds for Discrete Logarithms and Related Problems", Eurocrypt'97, LNCS 1233, pp. 256-266.
41. V. Shoup, "On Formal Models for Secure Key Exchange", Theory of Cryptography Library, 1999. `http://philby.ucsd.edu/cryptolib/1999/99-12.html`.