HMQV in IEEE P1363

Hugo Krawczyk[*]

July 7, 2006

Abstract

This constribution contains a proposal for including HMQV as a key agreement primitive in IEEE P1363. It includes an informal description of the protocol and a discussion of its security and performance properties (in particular as they compare to MQV), as well as a detailed formal specification for inclusion in the coming revision of the P1363 standard.

The HMQV primitive is a variant of the MQV protocol that enjoys several advantages including reduced complexity, improved performance, and a full array of attractive security properties all backed by formal analysis. In particular, it offers a simple two-message key agreement primitive that provides full authentication without depending on external mechanisms.

1 Introduction

This document presents a full specification of the HMQV protocol intended for consideration, and possible adoption, by the IEEE P1363 standardization group as a key agreement primitive. For full details on the design and analysis of HMQV the reader is referred to [4]. Here we start with an informal description of the protocol and a discussion of its main security properties (including a comparison to the MQV protocol on which HMQV is based), and then follow with a detailed specification ready for use in P1363.

1.1 Informal Specification

HMQV is a variant of the MQV primitive included in P1363 [3]. In both protocols a pair of interacting parties exchange ephemeral Diffie-Hellman values and compute a shared secret as a function of these values and of their long-term private and public keys. The shared secret is then input to a KDF or hash function to produce a session key. In the following

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

informal description we assume a finite field setting; adaptation to the elliptic curve setting is straightforward (here the notation is taken from P1363 and is different from then one used in [4]).

Let $GF(q)$ be a finite field, r be a prime that divides $q-1$ (with multiplicity 1), and g be an element with multiplicative order r in $GF(q)$. In both MQV and HMQV, each party has a long term key pair of the form $(s, w = g^s)$. When two parties wish to establish a shared session key K they proceed as follows. Let the identities of the parties be id and id', and their long-term key pairs be $(s, w = g^s)$ and $(s', w' = g^{s'})$, respectively. The parties then compute ephemeral key pairs, $(u, v = g^u)$ and $(u', v' = g^{u'})$, respectively, exchange the values v and v', and compute a shared secret z defined as $z = g^{(st+u)(s't'+u')} = (w^t v)^{(s't'+u')} = (w'^t v')^{(st+u)}$ where t and t' are $|q|/2$-bit numbers (here $|q|$ denotes the length in bits of the number q). MQV and HMQV differ in the way the values t and t' are defined. In the case of MQV, t and t' are defined as the truncated numerical representation of the ephemeral DH values w and w'. In the case of HMQV, we have $t = H(v, id')$ and $t' = H(v', id)$ where H is a hash function that outputs $|q|/2$ bits. In both MQV and HMQV, the final session key K is computed as $K = KDF(z)$; however, while MQV [5] does not specify the requirements from KDF, the HMQV analysis shows that this has to be a strong one-way hash function (one that is modeled as a random oracle in the analysis).

We stress that the two elements added by HMQV to the computation of the t-values, namely the involvement of the parties' identities and a hash function, are essential to achieve the provability properties of the HMQV protocol; in particular, they are essential to achieve full authentication in a two-message exchange and without depending on additional authentication mechanisms (such as depending on the parameters to the key derivation function or key confirmation).

Another important difference between MQV and HMQV is their requirements for the testing of private and public keys. MQV requires full proof of possession (PoP) for private keys (namely, the owner of a public key needs to "prove" that it knows the corresponding private key). Therefore, a party must run a PoP upon receiving another party's public key or must assume that a certification authority (CA) has performed such a PoP. MQV also requires either testing that public keys (long-term and ephemeral) are of the correct prime order r or augmenting the protocol with a 'co-factor exponentiation" [5]. These operations may be of significant cost (especially when MQV is chosen for performance reasons and when the cofactor, i.e. the number $(q-1)/r$, is large). In contrast, HMQV fully dispenses of proofs of possession and requires prime-order verification only in specific attack scenarios. Specifically, such tests are required only in settings where ephemeral exponents are more vulnerable to attack than long-term secrets. In all other cases, i.e., where ephemeral and long-term secrets are equally protected[1], HMQV can safely skip these tests, thus providing superior performance especially when the cofactor is large. Both MQV and HMQV need

[1]This is the case in many practical implementations and is used as an essential assumption in some cryptographic primitives. For example, in the context of the DSA and ECDSA signature schemes, the disclosure of a single ephemeral secret value reveals the whole long-term private key.

to check that ephemeral and long-term public keys are (non-zero) elements in $GF(q)$ (or points in a given elliptic curve), but these are inexpensive operations.

1.2 Security and Performance Properties

Here we summarize the main security and performance properties of HMQV. As said, HMQV is a variant of the MQV key agreement protocol [5] that preserves the outstanding performance of the original MQV protocol while offering a wide array of well-defined security properties all backed by formal analysis and computational proofs (in the random oracle model). Moreover, due to its detailed analysis, HMQV is able to provide provable security while dispensing of several of the safeguards required in MQV. This results in a more robust protocol, much less dependent on system and infrastructure assumptions, and even in a more efficient protocol. Specifically: (i) HMQV does not depend on a CA (or other parties) performing "proofs of possession" of private keys at the time of public key registration or verification; (ii) HMQV does not require the testing for prime order of long-term and ephemeral public keys except for specific attack scenarios; (iii) HMQV provides entity- and key- authentication as a built-in mechanism without depending on the parameters to the key derivation function or other external mechanisms. In particular, the basic two-message HMQV protocol already provides full authentication, including provable resistance to UKS and KCI attacks[2].

We note that the relaxation of the testing requirements on the private and public keys, as mentioned in points (i) and (ii), greatly simplifies the deployment of the protocol and makes it more secure in many practical scenarios such as in the common real-life cases in which proofs of possession are not performed at all by the certification entity. Moreover, even when PoP is enforced (and implemented with secure protocols – a very challenging task), its very execution may lead to vulnerabilities. (For example, a private key may be generated for exclusive use in a high-security setting and its exposure to a CA application interface may be undesirable or just prohibited.) In addition, in many practical scenarios HMQV may fully dispense of prime order tests and co-factor exponentiation, thus making it more efficient than MQV (the exact cost of these tests depends on the underlying group). Specifically, it is proven that, in a model where the attacker has no access to ephemeral exponents, HMQV is secure without these tests. On the other hand, in a setting where ephemeral exponents are significantly more vulnerable than long-term secrets, also HMQV requires co-factor exponentiation or a single prime-order test (there is no need to test separately the order of long-term and ephemeral public keys). In other words, in any security setting where ephemeral secrets are protected as well as long-term ones, HMQV offers better performance than MQV. As said before, such settings are common (and even

[2]Failure to UKS attacks [1] is a serious authentication concern (especially in the setting of "known-key attacks") where two honest parties compute the same session key but have inconsistent views of who the peer to the exchange is. Resistance to KCI attacks means that even if the attacker learns the long-term private key of Alice, he still cannot impersonate other parties to Alice. Resistance to both UKS and KCI attacks are listed as explicit goals of MQV but only partially achieved by the original protocol [5].

mandatory) in many applications; for example, in a DSA signature scheme ephemeral and long-term secrets must be equally protected since the exposure of a single ephemeral secret value compromises the whole private signing key.

We end this discussion by stressing that HMQV mandates the hashing of the shared secret z to produce a session key. This is necessary for the security of the protocol since the disclosure of z to the attacker may lead not only to the inevitable compromise of the session using z but also to the vulnerability of other sessions (the same is true for MQV). In contrast, it is proven that in HMQV if an attacker learns $K = H(z)$ but not z, the only session that is compromised is the one using K.

2 Formal Specification of the HMQV Primitive

This specification builds on the MQV specifications in the IEEE P1363/D13 document dated Nov 12, 1999 [2] (which we refer to as D13). We keep the structure of the MQV specification in D13 and borrow as much text as possible. Changed or added text is highlighted in San-Serif font. Explanatory notes, not intended to be part of the final spec, are presented as footnotes.

2.1 DLSVDP-HMQV

The following text defines DLSVDP-HMQV building on the definition of DLSVDP-MQV in section 6.2.3 with as few textual changes as possible. (Some text is also borrowed from section 6.2.4, DLSVDP-MQVC.) We refer to this new section as 6.2.3' although this is not proposed as a replacement for 6.2.3 but as an additional section. The changes include: (i) replacing the computation of t and t' from a "truncated DH value" as in MQV to a hashed value as in HMQV, (ii) involving the identities of the parties and a hash function in the computation of t and t', and (iii) dispensing of the requirement for prime-order validation of public keys which becomes an optional step.

6.2.3' DLSVDP-HMQV

DLSVDP-HMQV is Discrete Logarithm Secret Value Derivation Primitive, HMQV version. It is based on the work of [LMQ98] and [Kra05]. This primitive derives a shared secret value from one party's two key pairs and another party's two public keys, where all the keys have the same set of DL domain parameters. The computation of the shared secret also involves the identities of the parties and a hash function. If two parties execute this primitive with corresponding keys and identities as inputs, they will produce the same output. This primitive can be invoked by a scheme to derive a shared secret key; specifically, it may be used with the scheme DLKAS-MQV[3]. It does not assume the validity of the input public

[3]We leave DLKAS-MQV as the generic name for schemes based on MQV variants. In Section 9.4, where these schemes are defined, we add DLSVDP-HMQV and DLSVDP-HMQVC as optional primitives on which

keys, but does assume that the public keys are elements of $GF(q) \setminus \{0,1\}$.[4] (See the NOTES at the end.)

In this primitive, let $h = \lceil (log_2 r)/2 \rceil$. Note that h depends only on the DL domain parameters, and hence can be computed once for a given set of domain parameters.

Input:

– the DL domain parameters q, r and g associated with the keys s, $(u, v), w'$, and v' (the domain parameters shall be the same for these keys)

– the party's own first private key s

– the party's own second key pair (u, v)

– the other party's public key w'

– the other party's second public key v'

– the party's identity, denoted id, and the other party's identity, denoted id'

– a specified hash function H with output of length $\geq h$

Assumptions: private key s, key pair (u, v), and DL domain parameters q, r and g are valid; private key s and key pair (u, v) are associated with the domain parameters ; w' and v' are in $GF(q) \setminus \{0,1\}$.

Output: the derived shared secret value z or "invalid public key"

Operation. The shared secret value z shall be computed by the following or an equivalent sequence of steps.[5] [6]

1. Compute an integer $t = OS2IP(H(FE2OSP(v)||id'))$
2. Compute an integer $t' = OS2IP(H(FE2OSP(v')||id))$
3. Compute an integer $e = ts + u \ mod \ r$.
4. Compute a field element $z = exp(v' * exp(w', t'), e)$.
5. If $z = 1$ or $z = 0$, output "invalid public key" and stop.
6. Output z as the shared secret value.

Conformance region recommendation. A conformance region should include:

– at least one valid set of DL domain parameters q, r and g

to base the DLKAS-MQV scheme.

[4] Consider having a notation for this set, say $\overline{GF}(q) \overset{\text{def}}{=} GF(q) \setminus \{0,1\}$.

[5] Steps 1, 2 replace steps 1-4 in DLSVDP-MQV and they implement the computation of $t = H(v||id')$ and $t' = H(v'||id)$.

[6] IMPORTANT NOTE: In the computation of the t-values one may replace the direct use of the hash function H with the KDF2 primitive (from the P1363a spec). In this case, one passes the value v (or v') as the Z input to KDF2 and id (or id') as the octet string P. The result is essentially the same as applying H directly, except that KDF2 allows to specify the output length. However, this equivalence is due to the very specific form of KDF2; other key derivation functions may not be well-suited to replace H. Another issue to be specified relates to whether identities are of fixed size, or are preceded by a length parameter.

– at least one valid private key s for each set of domain parameters

– all valid key pairs (u, v) associated with the same set of domain parameters as s

– all elements w' and v' in $GF(q) \setminus \{0,1\}$, where q is from the domain parameters of s

NOTES (intended to be part of the specification).

1. DLSVDP-HMQV does not require full validation of (ephemeral and long-term) public keys but only requires checking that these public keys are in $GF(q) \setminus \{0,1\}$ (namely, any element in $GF(q)$ except for the zero and identity elements). In particular, it does not require testing the prime order r of the public keys, an operation that may be costly depending on the underlying group and which is often performed to prevent the so called small-group attacks (see Annex D.5.1.6). In the case of HMQV, small-group attacks cannot help an attacker except in cases in which the ephemeral key u (but not the private key v) is learned by the attacker. Thus, any system or application where all secrets are equally protected (this is not uncommon; for example, in the DSA or ECDSA signature schemes ephemeral secrets are as valuable for an attacker as long-term ones and hence require equal protection) can benefit from the superior performance of DLSVDP-HMQV. Applications that desire to add protection in case of disclosure of ephemeral secrets can either test that the value $v' * exp(w', t')$ is of order r (note that there is no need to test the v' and w' keys separately) or use the DLSVDP-HMQVC primitive from Section 6.2.4'. The computational cost of these two options is an ℓ-bit exponentiation where ℓ depends on the parameters r and q: in the first case $\ell = \min\{|r|, |(q-1)/r|\}$ and in the latter $\ell = |(q-1)/r|$.

2. A secure key agreement protocol must ensure that, even in the presence of an adversary that controls the communication channels and may corrupt individual parties and sessions, keys computed at uncorrupted parties and associated with peers that are also uncorrupted remain unguessable by the attacker. In particular, the attacker should not be able to force such keys to fall in a small subset of values. DLSVDP-HMQV guarantees this property as follows: any shared value z established at an uncorrupted party and associated with a peer's identity id', where id' identifies an uncorrupted party, must be of order (in $GF(q)$) that is divisible by r, and hence its set of possible values is at least of size r. In the case in which a party performs an r-order test, as discussed in point 1 above, it is guaranteed that z is of *exact* order r. Yet, the latter property is of no advantage in this case since the value z itself is not used directly but only after applying to it a hash (or key derivation) function as discussed next. Note that keys established by corrupted parties, or with corrupted parties, may belong to arbitrary sets; there is no security guarantee provided on such keys (even if such key may a-priori fall in a large set, the key is still insecure; for example, the corrupted peer can choose its private exponents from a small set, or simply leak the key, etc.).

3. It is imperative for the security of key agreement protocols based on DLSVDP-HMQV (and the same is true for the original MQV protocols) that the shared secret z not be used as a key but only as the input to a KDF (actually a strong KDF modeled as a random oracle in analysis). Indeed, it is shown in [Kra05] that if the value z corresponding to an execution of the protocol (a "session") is revealed to the attacker then the attacker can

use this value to attack the keys of *other* sessions, thus compromising the security of the key agreement protocol.[7] In the case of HMQV, the hashing of z via the KDF is sufficient to guarantee the security of the protocol, independently of possible additional parameters involved in the hashing of z.

4. Since the protocol requires to test that both public keys contributed by the other party (the peer), namely v' and w', are in $GF(q) \setminus \{0, 1\}$ then z cannot be 0 and can be 1 with negligible probability only. Thus an output of either 0 or 1 is regarded as a strong indication of a failure in the testing of public keys and hence the corresponding abort message "invalid public key" output by the protocol in this case. This event may trigger a new run of the primitive with new inputs, and it should be logged as a potential security breach.

5. The computation of z involves the identities of the peers. Identities are octet strings that uniquely identify an entity (in this case, a participant in the protocol) for the purpose of authentication. Identities may include names as well as other attributes that are relevant to the unique identification and authentication of an entity. When digital certificates are used to bind entities to public keys, the identities used in the protocol will usually be the same as, or related to, the identity appearing in the certificate (also known as the "distinguished name" in the X.509 jargon). The specific meaning of identities and the decisions made on the basis of such identities vary by scenario and application, and are usually determined by policy rules, authorization and access control mechanisms, etc.

2.2 DLSVDP-HMQVC

The following text defines DLSVDP-HMQVC building on the definition of DLSVDP-MQVC (section 6.2.4) with as few changes as possible; we refer to this section as 6.2.4' although it is anticipated that this will not replace 6.2.4 but will be added as a new section. The main changes relative to DLSVDP-MQVC are, as in section 6.2.3, the use of a hash function and the parties identities in the computation of the values t and t'.

6.2.4' DLSVDP-HMQVC

DLSVDP-HMQVC is Discrete Logarithm Secret Value Derivation Primitive, HMQV version with cofactor exponentiation. It is based on the work of [LMQ98], [Kal98a], and [Kra05]. This primitive derives a shared secret value from one party's two key pairs and another party's two public keys, where all the keys have the same set of DL domain parameters. The computation of the shared secret also involves the identities of the parties and a hash function. If two parties execute this primitive with corresponding keys and identities as inputs, they will produce the same output. This primitive can be invoked by a scheme to derive a shared secret key; specifically, it may be used with the scheme DLKAS-MQV (see section 9.4.). It does not assume the validity of the input public keys, but does assume that the public key is an element of $GF(q) \setminus \{0, 1\}$ (see also Section 6.2.3').

[7]For this reason I would be happier to define that the output of all theses protocols (both the MQV and HMQV versions) be defined as $KDF(z)$ rather than z.

In this primitive, let $h = \lceil (log_2 r)/2 \rceil$. Note that h depends only on the DL domain parameters, and hence can be computed once for a given set of domain parameters.

Input:

– the DL domain parameters q, r, g and the cofactor k associated with the keys s, (u, v), w', and v' (the domain parameters shall be the same for these keys)

– the party's own first private key s

– the party's own second key pair (u, v)

– the other party's public key w'

– the other party's second public key v'

– the party's identity id and the other party's identity id'

– a specified hash function H with output of length $\geq h$

– an indication as to whether compatibility with DLSVDP-HMQV is desired.

Assumptions: private key s, key pair (u, v), and DL domain parameters q, r, g and k are valid; private key s and key pair (u, v) are associated with the domain parameters; w' and v' are in $GF(q) \setminus \{0, 1\}$; GCD(k,r)=1

Output: the derived shared secret value, which is a nonzero field element z in $GF(q)$; or "invalid public key"

Operation. The shared secret value z shall be computed by the following or an equivalent sequence of steps[8]:

1. Compute an integer $t = OS2IP(H(FE2OSP(v)||id'))$
2. Compute an integer $t' = OS2IP(H(FE2OSP(v')||id))$

5. If compatibility with DLSVDP-HMQV is desired, then compute an integer $e = k^{-1}(ts + u)\ mod\ r$; otherwise compute an integer $e = ts + u\ mod\ r$.
6. Compute a field element $z = exp(v' * exp(w', t'), ke)$.
7. If $z = 0$ or $z = 1$, output "invalid public key"; else, output z as the shared secret value.

Conformance region recommendation. A conformance region should include:

– at least one valid set of DL domain parameters q, r, g and k

– at least one valid private key s for each set of domain parameters

– all valid key pairs (u, v) associated with the same set of domain parameters as s

– all elements w' and v' in $GF(q) \setminus \{0, 1\}$, where q is from the domain parameters of s

– compatibility with DLSVDP-HMQV may be preset by the implementation, or given as an input flag

NOTES (intended to be part of the specification).

[8]Steps 1, 2 replace steps 1-4 in DLSVDP-MQVC and they implement the computation of $t = H(v||id')$ and $t' = H(v'||id)$. See also footnote 6.

1. DLSCDP-HMQVC does not require full validation of public keys (ephemeral and long-term) but only requires checking that these public keys are in $GF(q) \setminus \{0, 1\}$, that is, that they are elements in the field $GF(q)$ except for the zero and identity elements. The primitive explicitly addresses small subgroup attacks (see Annex D.5.1.6) via the cofactor exponentiation. This is useful in the case of HMQV to prevent damage from such attacks in a scenario where an attacker can learn ephemeral secrets used by a party but not the long term secret (as pointed out in section 6.2.3', small group attacks against HMQV are relevant only in this scenario). The added cost relative to DLSDVP-HMQV is a $|(q-1)/r|$-bit exponentiation. Therefore, the choice between DLSDVP-HMQV and DLSDVP-HMQVC depends on performance considerations and on the security model. Note that even when disclosure of ephemeral secrets is deemed more practical than leakage of long-term keys, the option of performing DLSDVP-HMQV with a prime-order test as described in the Notes of section 6.2.3' may be more efficient than performing cofactor exponentiation (specifically, this is the case whenever $r < \sqrt{q}$).

2. The cofactor k depends only on the DL domain parameters and is equal to $(q-1)/r$. Hence, it can be computed once for a given set of domain parameters and stored as part of the domain parameters. Similarly, in the compatibility case, the value k^{-1} can be computed once and stored with the domain parameters. An equivalent way to compute the integer e in this case is as $(tk^{-1}s + k^{-1}u) \ mod \ r$, where $k^{-1}s \ mod \ r$ and $k^{-1}u \ mod \ r$ can be computed once for each given private key s and u.

3. Since the protocol requires to test that both public keys contributed by the other party (the peer), namely v' and w', are in $GF(q) \setminus \{0, 1\}$ then z cannot be 0 and can be 1 with negligible probability only. Thus an output of either 0 or 1 is regarded as a strong indication of a failure in the testing of public keys and hence the corresponding abort message "invalid public key" output by the protocol in this case. This event may trigger a new run of the primitive with new inputs, and it should be logged as a potential security breach. In all other cases, it is guaranteed that z is of order q; in particular, it will not be in a small group.

4. In the compatibility case, DLSVDP-HMQVC computes the same output as DLSVDP-HMQV, so an implementation that conforms with DLSVDP-HMQVC in the compatibility case also conforms with DLSVDP-HMQV.

5. It is imperative for the security of key agreement protocols based on DLSVDP-HMQVC (and the same is true for HMQV and the original MQV protocols) that the shared secret z not be used directly as a key but only as the input to a KDF (actually a strong KDF modeled as a random oracle in analysis). See more on this in note number 3 in Section 6.2.3'.

2.3 Elliptic Curve HMQV Primitives

Sections 7.2.3 and 7.2.4 of D13 define the counterparts of the above primitives where the underlying group is an elliptic curve. The modifications needed to obtain the corresponding

HMQV EC flavors are similar to those shown above for the DL case and omitted from this first draft. Full text will be added (as sections 7.2.3' and 7.2.4') after receiving the group's feedback on the above DL specification.

2.4 HMQV Key Agreement Schemes

Section 9.4 of D13 defines a generic key agreement scheme using any one of the defined MQV flavors. To accommodate HMQV the only required change in that section is to list the DL/ECSVDP-HMQV and DL/ECSVDP-HMQVC primitives among those listed in that scheme. The optional steps for validation will be the same as those listed there for the MQVC variants. (A note can be added that in the case of the DL/ECSVDP-HMQV scheme full validation – namely, prime order validation – may be performed if defense against leaked ephemeral exponents is desired.) In addition, Section 9.4 should have an explicit note explaining that the value of the shared secret z should never be revealed to a non-secure interface since revealing this value not only compromises the specific session using the key derived from z but it can potentially compromise other sessions. This note needs to be there regardless of the addition of HMQV since it also applies to MQV. Also, a note about key confirmation, especially as a way to ensure full forward secrecy (against active attacks) should be added to 9.4 (again, irrespective of HMQV adoption). Finally, text about the checking of certificates, or other ways to validate the authenticity of public keys, should be added too.

References

[1] W. Diffie, P. van Oorschot and M. Wiener, "Authentication and authenticated key exchanges", *Designs, Codes and Cryptography*, 2, 1992, pp. 107–125.

[2] IEEE P1363/D13 (Draft Version 13): Standard Specifications for Public Key Cryptography, Nov. 12, 1999.

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

[4] H. Krawczyk, "HMQV: A High-Performance Secure Diffie-Hellman Protocol", `http://eprint.iacr.org/2005/176` (Preliminary version in Crypto'05. LNCS 3621. 2005.)

[5] 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.*