HMQV and Provable Security

Hugo Krawczyk

IBM Research

See http://eprint.iacr.org/2005/176

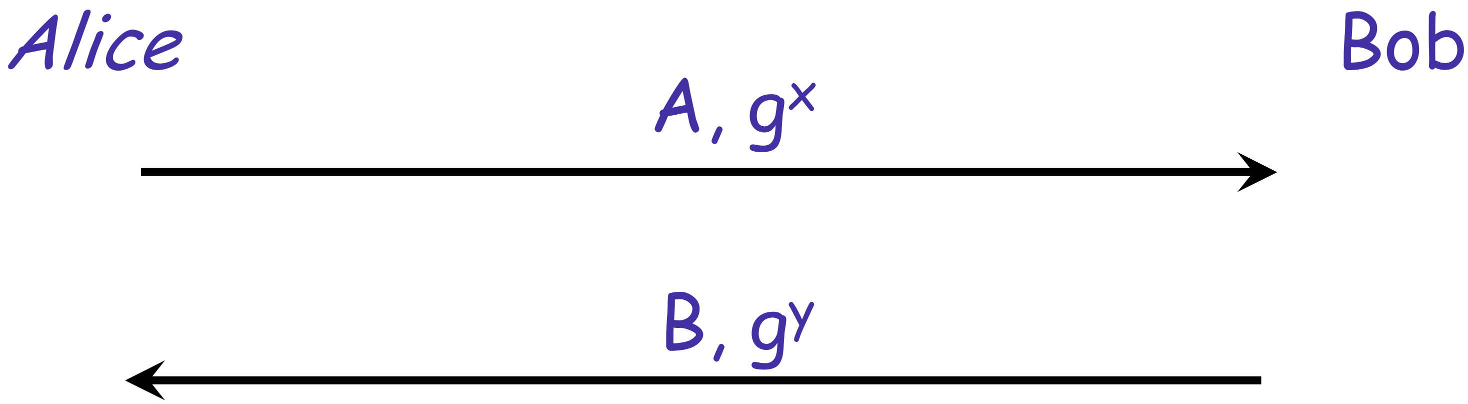
Talk Motivation

- Why this topic (HMQV and provable security)?
  - Conceptually and technically challenging; the beauty of simplicity and the trickiness of understanding it and proving it; and the practical applications of course
  - Also because of the debate around *"provable security"* (e.g. Koblitz-Menezes)
- Goal: illustrate the central (and indispensable) role of provable security as BOTH analysis and *design* tools!!
  - Also: encourage YOU to be proactive about the design, standardization and deployment of GOOD cryptography (e.g., NIST's SP 800-56)

Talk Plan

- Introduction to MQV (most efficient authenticated DH)
- MQV's wish list: is it achieved?
- HMQV: a provable variant of MQV
- On the analysis of HMQV
- Illustrating the power of proofs: design and analysis (even cryptanalysis); the proof-driven design concept
- Some concluding remarks

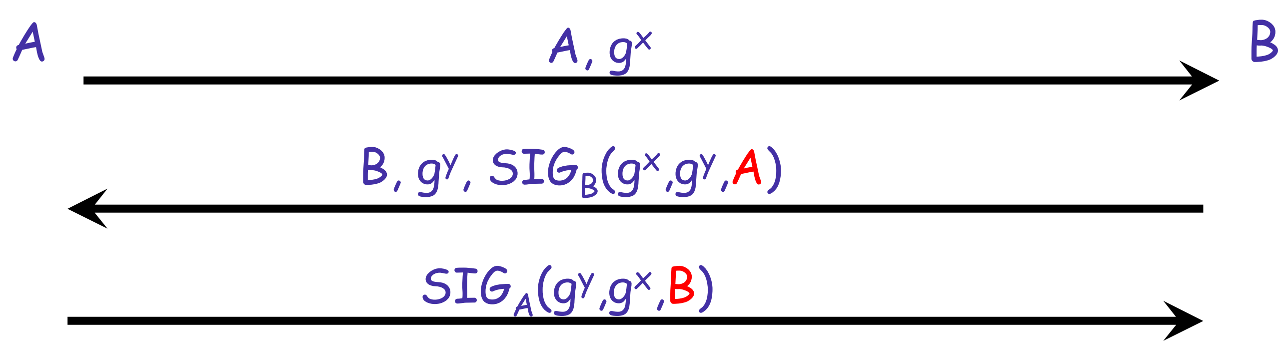
Diffie-Hellman Exchange [DH'76]



- both parties compute the secret key $K=g^{xy}=(g^x)^y=(g^y)^x$
- assumes authenticated channels (+ DDH assumption)
- **open to m-i-t-m** in a realistic unauthenticated setting

The Challenge of Authenticated DH

- Many failed attempts, few are secure
- Some secure protocols: add flows with authentication information. For example, an ISO variant:



- The fundamental element: bind session key to identities!
- Can we avoid the extra flows/info and still be secure?

Implicitly Authenticated DH [MTI'86]

- Minimalist approach: Keep a plain (2-msg) DH exchange, but give Alice and Bob public keys (possibly with certificates)
- Authentication via session key computation
  - No transmitted signatures, MAC values, etc
  - Session key must involve long-term and ephemeral keys:

    $K=F(PK_A,PK_B,SK_A,SK_B\,,g^x,g^y,x,y)$
  - Ability to compute key ⇒ authentication
- Possible and simple but tricky: many insecure proposals/ standards (e.g NIST's "unified model" proven insecure in [BJM97])

MQV [MQV'95,LMQSV'00]

- Most attractive among implicitly authenticated DH; some beautiful ideas (builds on MTI'86, Arazi'92, Nyberg-Rueppel'93)
  - Performance: just ½ exponentiation (25%) more than DH (with NO added bandwidth except if public keys transmitted)
  - Broad array of security goals considered: m-i-t-m, known-key attacks, UKS, PFS, KCI (non-trivial with implicit auth),…
- Widely standardized: ANSI, IEEE, ISO, NIST
- NSA: "next generation cryptography" (including protection of "classified or mission critical national security information")
- But is MQV secure? In what sense? Can be improved?

The MQV Protocol

- Basic DH + special key computation
- Notation: $G=\langle g\rangle$ of prime order q; g in supergroup G' (eg. EC, Z^*_p)
  - Alice's PK is $A=g^a$ and Bob's is $B=g^b$,
  - Exchanged ephemeral DH values are $X=g^x$, $Y=g^y$
  - From which two values are computed: $d=LSB(X)$, $e=LSB(Y)$ where $LSB(X)= 2^L + X \bmod 2^L$ for $L=|q|/2$ (this is the ½ exponentiation)
- Both compute $\sigma=g^{(x+da)(y+eb)}$ as $\sigma = (YB^e)^{x+da} = (XA^d)^{y+eb}$
- Session key is $K=KDF(\sigma)$ (KDF unspecified but OW not required)
- **Magic, isn't it?** Is it secure? Why? Can it be formally analyzed?

The MQV Protocol (cont.)

- Actual computation of σ involves co-factor $h=|G'|/q$
  - $\sigma' = (YB^e)^{x+da} = (XA^d)^{y+eb}$; $\sigma = (\sigma')^h$
  - Adds an exponentiation: typically small for ECC, large for Z^*_p, significant in high-performance scenarios (can replace w/ q-order test)
  - I omitted it in the basic description for simplicity: does not help against weaknesses discussed here
- Other requirements in MQV: "Proof of Possession" (PoP) by CA and PK validation (prime order) Adds significant complexity and trust dependency!
  - Note: PoP not always done and hard to get it right (especially with non-signature keys, e.g. SP 800-56)
  - Minimizing trust/reliance in CA is an important consideration!
  - Ex: PK A certified by Alice herself! *(cert(Alice,PK_{Alice}), sig_{Alice}(A))*

MQV's Wish List [LMQSV]

- Authentication and Secrecy (the "obvious" meaning)
- Known-key attacks (attacker may learn some session keys)
- PFS (session keys secure even if private keys eventually found)
- Resistance to special attack forms:
  - UKS (unknown key share): Alice and Bob compute the same K, but Alice binds it to Bob while Bob binds it to Eve (a serious auth'n failure even if Eve does not learn K)
  - KCI (key-compromise impersonation) : Using Alice's private key, Eve cannot impersonate _*other*_ parties to Alice (reverse is unavoidable)
  - Disclosure of ephemeral DH exponents x, y breaks single session
- Avoid using hash functions or OWF's as KDFs

Are these properties achieved?

- This question motivated my work
  - [LMQSV] offer no proof or formal definitions; little rationale, ambiguous language
  - Trying to prove MQV reveals weaknesses (practical significance varies but enough to show the protocol cannot be proven secure)
  - More interestingly: proof-driven design results in a simpler, more practical and more efficient protocol
- Next: some MQV properties that do not hold

Are these properties achieved?

- UKS failure (even with "Proof of Possession" by CA [Kal])
  - Essential binding key-identities missing (may even fail w/ KC)
- KCI not achieved if KDF(σ) not OW (hash is essential!)
- Similarly: w/o strong hashing of σ, exposure of x, y breaks the protocol (even if prime order tests performed!)
- MQV sensitive to "element representation": security bound by entropy of LSB's (group/representation dependent)
- PFS: achieved only against passive attackers (wPFS) (unavoidable in 2 rounds, requires key confirmation; also HMQV)

Note: None of these prevented with PoPs, PK validation, prime tests, etc

HMQV: A secure MQV variant

(Hashed — brace over "MQV")

- As in MQV: basic DH ($X=g^x$, $Y=g^y$), PKs: $A=g^a$, $B=g^b$
- Both compute $\sigma=g^{(x+da)(y+eb)}$ as $\sigma = (YB^e)^{x+da} = (XA^d)^{y+eb}$
- d=H(X,"Bob") e=H(Y,"Alice") (here H outputs |q|/2 bits)
- Session key K=H(σ) (here H outputs |K| bits, say 128)
- Differences with MQV
  - Definition of d, e: binds id's, randomizes representation
  - H(σ): integral (and essential) part of the protocol (OW,RO)
  - No need for PoP or PK validation by CA!
  - PROVABLE SECURITY and even better performance!!

HMQV Analysis

- In the KE model of Canetti and Krawczyk [CK'01]
- Attacker may access private keys, session keys, session-state information ("exposed session")
- Any unexposed session is secure (key is indist from random)
- In addition: extensions to capture PFS, KCI [K'05]
- [CK'01] Prove that secure KE in this model Ł secure communications ("secure channels")
- Note: protocol must specify what resides in state and what in protected memory (such as private keys)

Part I: x,y as protected as a,b

- The DSA case: sig = $(g^k, k^{-1}h+ag^k)$, single exposed k à a
- KE model without state reveal
- The case of x,y leakage requires a more complex analysis and even an extra protocol operation (later)

Basic Security of HMQV

- Thm: Under the CDH assumption and in the random oracle model, HMQV (basic 2-msg or 3-msg with KC) is a secure KE protocol in the Canetti-Krawczyk KE model
  - The thm applies when σ and the ephemeral x,y are specified to be in protected memory, same as the private key (as in DSA)
- Theorem includes wPFS (full with KC) and resistance to KCI, UKS, known-key attacks, key recovery, etc
- No need for prime-order testing, co-factor exponent'n, PoP's, PK validation by CA or special KDF's

  (significant security and performance advantages; in particular wrt MQV)

HMQV Analysis

- HMQV: basic DH ($X=g^x$, $Y=g^y$), PKs: $A=g^a$, $B=g^b$
  - Compute $\sigma=g^{(x+da)(y+eb)}$ as $\sigma = (YB^e)^{x+da} = (XA^d)^{y+eb}$
  - d=H(X,"Bob") e=H(Y,"Alice") (H outputs ≥|q|/2 bits)
  - Session key K=H(σ) (e.g., 128 bits)
- No signatures exchanged, authentication achieved via computation of σ (must ensure: only Alice and Bob can compute it)
- Idea: $(YB^e)^{x+da}$ is a sig of Alice on the pair (X, "Bob") and, at the same time, $(XA^d)^{y+eb}$ is a sig of Bob on (Y, "Alice")
  - Two signatures by two different parties (different priv/publ keys) on different msgs but with the same signature value!

Underlying Primitive: Challenge-Response Signatures

- Bob is the signer (PK is $B=g^b$), Alice is the verifier (no PK)
  - Alice sends a "challenge" ($X=g^x$) and a msg m to Bob, who responds with a "challenge-specific" signature on m (sig depends on b, X, m)
  - Alice uses her "challenge trapdoor" (x) to verify the signature
- Aliceà Bob: m, $X=g^x$

  Bobà Alice: $Y=g^y$, $\sigma=X^{y+eb}$ where $e=H(Y,m)$

  Alice accepts the signature as valid iff $(YB^e)^x= \sigma$
- We call this scheme XCR (Xponential Challenge Response)

Security of XCR Signatures

- Theorem: no forger can generate a new signature of Bob that will be accepted by a honest verifier
  - Unforgeability under usual adaptive chosen message attack
  - Assumptions: Computational DH and H modeled as random oracle
- Note: Alice could generate the signature by herself! (signature convinces only the challenger – non-transferable)
- Idea of proof: "exponential" Schnorr via Fiat-Shamir (in a minute…)

Dual XCR (DCR) Signatures

- Alice and Bob act as signers and verifiers simultaneously
- Alice has PK $A=g^a$, Bob has PK $B=g^b$
- Alice and Bob exchange values $X=g^x$, $Y=g^y$ and msgs m_A,m_B
- Bob generates an XCR sig on m_A under challenge XA^d

  Alice generates an XCR sig on m_B under challenge YB^e
- The signature is the same! $\sigma = (YB^e)^{x+da} = (XA^d)^{y+eb}$
- This is exactly HMQV if one puts m_A="Alice", m_B="Bob" (since sig is the same value it needs not be transmitted!)

Proof of HMQV

- Reduction from breaking HMQV as KE (in the CK model) to forging DCR
  - Not a trivial step
  - Great at showing the necessity of all elements in the protocol: drop any element and the proof shows you an attack (e.g. MQV)
- Reduction from forging DCR to forging XCR
  - Quite straightforward
- Reduction from forging XCR to solving CDH in RO model
  - I expand on this next

XCR Proof via "Exponential Schnorr"

- Schnorr's protocol (given $B=g^b$, Bob proves knowledge of b)
  - Bobà Alice: $Y=g^y$
  - Aliceà Bob: $e \in_R Z_q$
  - Bobà Alice: $s=eb+y$ (Alice checks $YB^e=g^s$)

[FS]: ZK for honest verifier (Alice) ⇒ (Y,s=eb+y) w/ e=H(m,Y) is a RO sig on m

- Exponential Schnorr: Bob proves *ability to compute* $()^b$
  - Bobà Alice: $Y=g^y$
  - Aliceà Bob: $e \in_R$ ~~Z_q~~ $\{0,1\}^{|q|/2}$, $X=g^x$
  - Bobà Alice: $\sigma=X^{eb+y}$ (Alice checks $(YB^e)^x=\sigma$)

ZK for honest verifier (& any X) ⇒ (Y, $\sigma=X^{eb+y}$) w/ e=H(m,Y) is a RO XCR sig on m

Theorem: XCR is strongly CMA-unforgeable (CDH + RO)

Proof: A CDH solver C from XCR forger F

- Input: U, V in G=<g> (a CDH instance; goal: compute g^{uv})
- Set B = V X_0 = U (B is signer's PK, X_0 is challenge to forger)
- Run F; for each msg m and challenge X queried by F (*a CMA attack*) *simulate* signature pair (Y, X^s) (random s, e; $Y = g^s/B^e$; H(Y,m) ß e)
- When F outputs forgery (Y_0, m_0, σ): (* (Y_0, m_0) fresh and $H(Y_0, m_0)$ queried *)

  Re-run F with new independent oracle responses to $H(Y_0, m_0)$
- If 2nd run results in forgery (Y_0, m_0, σ') (* same (Y_0, m_0) as before! *) then C outputs $W = (\sigma/\sigma')^{1/c}$ where c=(e-e') mod q .
  (e, e' are the responses to $H(Y_0, m_0)$ in 1st and 2nd run, respectively)

Theorem: with non-negligible probability W=DH(U,V)

Proof: [PS] + $W = (\sigma/\sigma')^{1/c} = ((Y_0B^e)^{x0} / (Y_0B^{e'})^{x0})^{1/c} = ((B^c)^{x0})^{1/c} = B^{x0}$

Implications for HMQV (* X à XA^d *)

- We used W $(\sigma/\sigma')^{1/c} = ((Y_0B^e)^{x0} / (Y_0B^{e'})^{x0})^{1/c}$

  But can we divide by $Y_0B^{e'}$? Yes if B and Y_0 in G (have inverses)

- B in G always true (chosen by honest signer) but what about Y_0 which is chosen by forger?
  - Do we need to check that Y_0 in G? (An extra exponentiation?)
  - No. If $G \subset R$, then enough to check Y_0 has inverse in R
    - E.g: $G = G_q = \langle g \rangle \subset Z_p^*$; $R = Z_p$; simply check Y in Z_p and $Y \neq 0$

- HMQV needs no prime order verification! (later: only if exponent leak)

- Forger can query arbitrary msgs with arbitrary challenges X (even challenges not in group G) à No need for PoP or PK test in HMQV!

  (X becomes XAd and we do not need to check X nor A!)

- **Robust security of HMQV without extra complexity (no extra exponentiations, PoP's, PK validation, etc.)**

Part II: ensuring security even if x,y revealed

- Not needed is systems supporting ECDSA (typical for MQV settings)
- Needed if x,y less protected (e.g. computed overnight)
- Desirable but a price to pay: extra exponentiation (cheap if small co-factor, expensive otherwise)
  - Clear security-performance trade-off
  - Also a more complex proof (and stronger assumptions)

Security in the face of ephemeral disclosure

- Under Gap-DH, KEA1 and in the random oracle model HMQV is secure also if ephemeral x,y disclosed provided that parties test XA^d and YB^e in $G=\langle g \rangle$ (= prime-order test or cofactor)
  - Test adds ONE exponentiation; cost depends on the group (MQV <u>always</u> performs such exponentiation: test or cofactor)
    - Note: Still no need for PoP or PK validation (CA out of the loop)
  - Establishes a clear security/performance trade-off
    - Possible only with analysis
- Plus all goodies: UKS, KCI, wPFS (* KC Ł PFS & UC *)
- "Maximal Security": HMQV secure with the disclosure of any pair from {a,b,x,y} except for (a,x), (b,y) $(\sigma = (YB^e)^{x+da} = (XA^d)^{y+eb})$

On the proof…

- Under Gap-DH, KEA1 and in the random oracle model HMQV is secure also if ephemeral x,y disclosed provided that parties test X and Y in G=<g> (= prime-order test)
- Stronger assumptions/ complex proof ("hashed XCR")
- Shows that Alice must check that YB^e is in G (else Lim-Lee)
  - Very subtle: input to a DDH oracle! [Menezes]
  - But note: no need for separate tests for Y and B! (more efficient, less trust in CA)

HMQV: Summary

- Plain DH exchange (no additional bandwidth except for cert's)
- 2.5 exponentiation per party: just 25% increase over plain DH
- Original "wish list" in MQV proven to hold for HMQV
- No performance penalty. Actually better/simpler!
  - Minimizes prime-order tests, minimizes CA dependency and trust (no PoP or PK validations), independent of KDF, "self contained"
  - Fastest authenticated and fully functional DH protocol to date
- Proof-driven design (proof as a design guide)!

Caveats

- Models/assumptions/random oracle/reduction cost
- Proofs need verification (Thanks, Alfred)
- Reduction cost: huge but fine if
  1. XCR as primitive (the way we assume DSA w/o going through P-S)
  2. Small scale vs. large scale attacks (nodes involved in attack)
  - Compare MQV: the thermometer story…
- Random oracle: can it be avoided?
  - XCR as primitive (e.g., using DFN'05) and "Hashed DH"
- "Structural security": Huge progress relative to hand-waved (often wrong and not well defined) arguments

Cryptography as a Science!

- Intuition, ideas, cryptanalysis, new attacks…
  all necessary and important but:
- Formal analysis as main <u>confidence</u> tool
  - Not a Panacea: never stronger than the model it is based on
  - But well-defined mechanisms and properties: can be verified (not just "trust me, I have not been able to break it")
  - Even a cryptanalysis tool (e.g. UKS, LimLee attacks, KCI w/o hash,…)
- Formal analysis as main <u>design</u> tool
  - Guides us to choose secure mechanisms, compose them right, discern between the essential, desirable and dispensable
  - Result is efficiency, simplicity, rationale, even impl'n guidance!
- **Provable security: a strong weapon! (use with care!)**

Final Remark

From invited talk Crypto'03

- The KE area has matured to the point in which there is no reason to use unproven protocols
  - Addressing practicality does not require (or justify) giving up on rigorous analysis (ISO and SIGMA) *and HMQV*
  - Proofs **not an absolute guarantee** (relative to the security model), but the best available assurance
  - It is easy to design simple and secure key-exchange protocols, but it is easier to get them wrong…
- Message to standards: go for proven protocols (secure and efficient, no need to compromise in quality, efficiency or analysis)

Did I mention NIST SP 800-56??

ThAnKs

J

http://eprint.iacr.org/2005/176