Extreme minimality:
Implicitly Authenticated KE Protocols
A natural Authenticated DH Solution (ISO 9796)

A, \(g^x\) → B

B, \(g^y\), SIG\(_B\)(\(g^x, g^y, A\)) → A

SIG\(_A\)(\(g^y, g^x, B\)) → B

Simple, but 3 messages plus signatures [and certificates]
The quest for Authenticated DH

- What is the inherent cost of authentication in Diffie-Hellman? In terms of
  - Communication: number of messages, group elements, authentication information, actual message size
  - Computation: algebraic operations and actual speed
  - Security: *What can we prove?*

- How close can we get to the fundamental limits? And still prove security...
Implicitly Authenticated DH

- 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 \( \rightarrow \) authentication

- The simpler the trickier: many insecure proposals
(Abuse of) Notation

Public key of $A$ (resp. $B$) denoted $A=g^a$ (resp. $B=g^b$)
Some Ideas

- Can we really have a *non-replayable* 2-msg protocol?
  - Remember $A \rightarrow B$: $g^x$, $\text{SIG}_A(g^x, B)$, $A \rightarrow B$: $g^y$, $\text{SIG}_B(g^y, A)$ insecurity

- Combining $A$, $B$, $X$, $Y$:
  - $K=H(g^a, g^x)$: Open to known key and interleaving attacks
  - $K=H(g^a, g^y, g^x, g^y)$ works but open to “KCI attacks”
    (a general weakness of protocols with $g^{ab}$)

- We want that no attack except if learning pair $(x,a)$ or $(y,b)$

- Idea: $K = g^{(a+x)(b+y)}$ (computed by $A$ as $(BY)^{a+x}$, by $B$ as $(AX)^{b+y}$)
  - Doesn’t work: Attacker sends $X^*=g^{x*}/A$, $B$ sends $Y$, $K=(BY)^{x*}$
    (no need to know $A$)
MQV

- Idea: set $K = g^{(a+dx)(b+ey)}$ and define $d, e$ so that attacker cannot control $e$ and $Y$, or $d$ and $X$
- MQV: $d=\text{half bits of } X$, $e=\text{half bits of } Y$
- Does not quite work
- But a simple variation does
The HMQV Protocol

- Basic DH + special key computation

- Notation: \( G=\langle g \rangle \) of prime order \( q \); \( g \) in supergroup \( G' \) (e.g. EC, \( Z^*_p \))
  - Alice's PK is \( A=g^a \) and Bob’s is \( B=g^b \) (private keys are \( a, b \), resp.)
  - Exchanged ephemeral DH values are \( X=g^x, Y=g^y \)

- 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 \( |q| \) bits)
  - Session key \( K=H(\sigma) \) (here \( H \) outputs \( |K| \) bits, say 128)

- Authentication almost for free (1/6 exponentiation, no commun'n)
The HMQV Protocol

- **Basic DH + special key computation**

- **Notation:** $G = \langle g \rangle$ of prime order $q$; $g$ in supergroup $G'$ (e.g. $EC, \mathbb{Z}_p^*$)
  - Alice’s PK is $A = g^a$ and Bob’s is $B = g^b$ (private keys are $a$, $b$, resp.)
  - Exchanged ephemeral DH values are $X = g^x$, $Y = g^y$

- **Each computes** $\sigma = g^{(x+da)(y+eb)}$ 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** $K = H'(\sigma)$  ($H'$ outputs $|K|$ bits, say 128)

- **Almost free authentication:** $\frac{1}{6}$ exponentiation, $=\text{communic'n}$
multi-exponentiation

- **Input**: \( g_0, g_1, e_0=(a_0, a_1, \ldots, a_{t-1}) \)  
  \( e_1=(b_0, b_1, \ldots, b_{t-1}) \)

- **Output**: \( g_0^{e_0} \cdot g_1^{e_1} \)

- **Pre-computation**: \( G_0=1, G_1=g_0, G_2=g_1, G_3=g_0 \cdot g_1, s(i)=a_i+2b_i \)

- **Compute**: \( A:=1; \) For \( i=0 \) to \( t-1 \): \{ \( A:=A \cdot A; \) \( A:=A \cdot G_{s(i)} \) \}

- **Ops**: \( t-1 \) squarings; \( \frac{3}{4} t \) multiplies (\( \frac{3}{4} \) because \( G_{s(0)}=1 \))

- **Compared to full exponentiation**: \( t-1 \) squares, \( \frac{1}{2} t \) multiplies

\[ g_0^{e_0} \cdot g_1^{e_1} \text{ costs } 1 \frac{1}{6} \text{ exponentiations rather than } 2 \]

- Works for any number \( k \) of bases (extra \( 2_k-2 \) mults)
The HMQV Protocol (w/short \(d,e\))

- 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 \(K = H(\sigma)\) (here \(H\) outputs \(|K|\) bits, say 128)

- Authentication for \(\frac{1}{2}\) exponentiation (no multiexp optimiz'n)

- Original formulation and proof (full length \(d, e\) simplifies some aspects of proof)
HMQV Explained

- **HMQV**: basic DH ($X=g^x$, $Y=g^y$), PKs: $A=g^a$, $B=g^b$
  
  - $\sigma=g^{(x+da)(y+eb)}$ as $\sigma = (YB^e)^{x+da} = (XA^d)^{y+eb}$; $K=H(\sigma)$
  
  - $d=H(X,"Bob")$  $e=H(Y,"Alice")$

- No signatures exchanged, authentication achieved via computation of $\sigma$ (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

- Alice $\rightarrow$ Bob: $m$, $X=g^x$

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

  Alice accepts the signature as valid iff $Y(B^e)^x=\sigma$

- Note: Alice could generate the signature by herself! (signature convinces only the challenger – non-transferable -- *bug or feature?*)

- We call this scheme XCR (Xponential Challenge Response)
Security of XCR Signatures

- Theorem: XCR signatures are unforgeable
  - Unforgeability under usual adaptive chosen message attack
  - Only signer and challenger can compute it
  - Assumptions: Computational DH; also H modeled as random oracle

- Idea of proof: “exponential” Schnorr via Fiat-Shamir
  - More later...
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 $X A^d$
  
  Alice generates an XCR sig on $m_B$ under challenge $Y B^e$
- The signature is the same! $\sigma = (Y B^e)^{x+da} = (X A^d)^{y+eb}$
- 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 $b$)
  - Bob $\rightarrow$ Alice: $Y=g^y$
  - Alice $\rightarrow$ Bob: $e \in \mathbb{Z}_q$
  - Bob $\rightarrow$ Alice: $s=eb+y$ (Alice checks $Y^e B^s=g^s$)

- Exponential Schnorr: Bob proves ability to compute $(\cdot)^b$
  - Bob $\rightarrow$ Alice: $Y=g^{y\{0,1\}^{|q|/2}}$
  - Alice $\rightarrow$ Bob: $e \in \mathbb{Z}_q$, $X=g^x$
  - Bob $\rightarrow$ Alice: $\sigma=X^{eb+y}$ (Alice checks $(Y^e B^s)^x=\sigma$)

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; \(Y = g^s/B^e\); \(H(Y, m) \leftarrow e\))
- **When F outputs forgery \((Y_0, m_0, \sigma)\):** (*\(Y_0, m_0\) fresh and \(H(Y_0, m_0)\) queried *)

  Re-run F with new independent oracle responses to \(H(Y_0, m_0)\)

- **If 2\(^{nd}\) run results in forgery \((Y_0, m_0, \sigma')\):** (*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 1\(^{st}\) and 2\(^{nd}\) run, respectively)

**Lemma:** with non-negligible probability \(W = \text{DH}(U, V)\)

**Proof:** \([PS]\) + \(W = (\sigma/\sigma')^{1/c} = \left( (Y_0 B^e)^{x_0} / (Y_0 B^{e'})^{x_0} \right)^{1/c} = \left( (B^c)^{x_0} \right)^{1/c} = B^{x_0}\)
Implications for HMQV (* X → XA^d *)

- We used \( W = \left( \frac{\sigma}{\sigma'} \right)^{1/c} = \frac{(Y_0 B^e)^{x_0}}{(Y_0 B'^e)^{x_0}} \)^{1/c} \\
  But can we divide by \( Y_0 B^e \)? 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 \( G \subseteq R \), then enough to check \( Y_0 \) has inverse in \( R \)
    - E.g: \( G = G_q = \langle g \rangle \subseteq \mathbb{Z}_p^* \); \( R = \mathbb{Z}_p \); simply check \( Y \) in \( \mathbb{Z}_p \) 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 \( G \)) → No need for PoP or PK test in HMQV!
  
  \( (X \text{ becomes } XA^d \text{ and we do not need to check } X \text{ nor } A!) \)

→ Robust security of HMQV without extra complexity
  (no extra exponentiations, PoP’s, PK validation, etc.)
More on Security of HMQV

- Note that each party can start the protocol (no initiator/responder roles) even simultaneously.

- Protocol is not secure against leakage of both \{a,x\} or \{b,y\} but secure against any other pair in \{a,x,b,y\}.
  - Secure against disclosure of \{a,b\} is equivalent to PFS.
  - But does HMQV really achieve PFS?
PFS in HMQV

- **PFS achieved only against passive attackers**

- **Impossibility fact.** If the messages sent in the protocol are computed without knowledge of the long-term keys of the sender then PFS fails to active attackers
  - The attacker chooses the message in the name of A (e.g. it chooses $x$ and sends $g^x$); later it learns the long-term key of A, hence can compute the session key.
  - Thus, authentication specific information must be transmitted to achieve PFS (e.g., via a third “key confirmation” message)
A fundamental question:

- Can one obtain a FULLY authenticated DH protocol with
  - A single message per party (2 message total)
  - A single group element per message
  - NO certificates
  - Minimal computational overhead for authentication
  - AND PFS AGAINST FULLY ACTIVE ATTACKERS

???????????????
A surprising answer: YES!

- By working over cyclic groups modulo a composite (we will refer to the bit size of elements later)
- Resorting to classical Okamoto-Tanaka protocol (1987)
- With simple modifications required for security
- With full proof of security, including proof of PFS against active attackers!!
Modified Okamoto-Tanaka (mOT)
Modified Okamoto-Tanaka Protocol

- Identity-based setting: Key Generation Center (KGC)
  - Chooses safe primes \( p, q \) (\( p=2p'-1, q=2q'-1 \)) and RSA exponents \( e, d \) for \( N=pq \)
  - Chooses generator \( g \) of \( QR_N \) (set of quadratic residues), a cyclic group of order \( p'q' \)
  - Publishes \( N, g, e \) (e.g. \( e=3 \))

- Secret key for party \( I \) is \( S_I=H(I)^d \mod N \) (computed by KGC)

- Ephemeral session values: \( g^x \mod N \) for \( x \) of length twice the security parameter (e.g., between 160-256 bits)
Modified Okamoto-Tanaka (mOT)

\[ S_A = H(I_A)^d \]

\[ S_B = H(I_B)^d \]

\[ I_A, \alpha = S_A \cdot g^x \mod N \]

\[ I_B, \beta = S_B \cdot g^y \mod N \]

\[ K = H(\beta^e / H(I_B))^{2x} = g^{2xye} = K = H(\alpha^e / H(I_A))^{2y} \]

- Two msgs, single group element, no certificate
- Computation: 1 off-line + 1 on-line expon’n (= basic DH) + e-exponentiation (= 2 multiplications) and 1 squaring
- Just 3 mult’s more than a basic DH over composite N !!!
mOT: Minimal Overhead, But is it secure?

- YES!
- With proof of security in the Canetti-Krawczyk model
  - RSA assumption + Random Oracle Model (passive PFS only)
- And proof of PFS security against active attackers
  - With additional “knowledge-of-exponent” type assumptions
- Note: mOT avoids the “implicit-authenticated PFS impossibility” since messages $\alpha, \beta$ depend on the private key of the sender

$$I_A, \alpha = S_A g^x \mod N$$
$$I_B, \beta = S_B g^y \mod N$$
On the Proof

- The basic case: CK model, weak PFS, under plain RSA in ROM follows more or less standard arguments.

- The challenge is in proving full PFS (against active attacks and without additional messages/communication).

- **Good news:** we can do it!
  - In particular, security of past communication if KGC compromise.

- **Less good news:** non-standard assumptions.
  - But a STRONG indication of security!
KEA-type Assumptions

- KEA-DH (a.k.a KEA1): “Computing $g^{xy}$ from $g,g^x,g^y$ can only be done if one knows x or y”

- KEA-DL:
  - Given $y=g^x$ want to compute x with the help of a Dlog oracle D that accepts any input but y
  - Obvious strategy: query D with $y\cdot g$
    More generally: Can query D with $z=y^i g^j$ for any known i,j (and recover x from the oracle’s response $ix+j$)
  - KEA-DL states that this is the most general strategy, i.e., if you find x by querying D(z) then you know i, j such that $z=ix+j$
Real-World Performance

😊 Complexity is essentially the same as the basic unauthenticated DH...

😢 ... but it runs over $\mathbb{Z}_N$ for composite $N$

😊 ... with short exponents (assumes dlog hard w/ such exponent)

😊 ... no certificate transmission and processing

In all, comparable performance to HMQV/ECC for security levels under 2048 bits (for RSA)

The really important point, however, is theoretical: Testing the limits of what is possible
Conclusions and Open Problems

- **Conclusions**
  - It is amazing how little one may need to pay over the basic DH for a fully authenticated exchange *with full PFS*:
    - Over QR groups the ONLY overhead is JUST 3 multiplications!
    - (no communication penalty, not even certificates)

- **Open questions**
  - Achieve the same performance properties with full PFS over other Dlog groups (e.g. Elliptic Curves)
  - Get rid of special assumptions for PFS proof
  - Reduce reliance on secrecy of the ephemeral $g^x$ and $g^y$
One-Pass HMQV and Asymmetric Key-Wrapping
Motivation

- Key wrapping as a basic functionality (e.g. storage systems)

- A good example of:
  
  Optimizing cryptography via "Proof-driven design"

- Proof tells us precisely what elements in the design are essential and which can be avoided
  
  - Avoid unnecessary safety margins
  - Better performance and functionality
  - A great protocol debugging tool
Key Wrapping

- **Key-wrapping or key encapsulation**: Server wraps a symmetric key for transporting it to a client
  - Think of wrapping as a *key encryption mechanism*
  - Encrypting key may be symmetric or asymmetric (AES, RSA)
  - Wrapped key may be a fresh key, or a previously generated one, sometimes bound together with associated data
  - Wrapping typically done off-line and non-interactively.
Example: Encrypted Backup Tapes

Tape encryption:

- Tape sent to KMM (key management module)
- KMM encrypts tape with tape-specific key $K$
  
  wraps $K$ (under another key) and stores wrapped key with tape

Tape decryption

- wrapped key sent to KMM* who unwraps (decrypts) $K$
- $K$ is sent back to tape holder for tape decryption

Notes: Decryption may happen many years after encryption

KMM and KMM* may not be the same (KMM* holds de-wrapping key)
Key Wrapping and Standards

- **Major key management tool:**
  - storage, hardware security modules, secure co-processors, ATM machines, clouds, etc.
  - Complex: long-lived keys & systems, backwards compatibility,…

- **Standards are important:** server and client typically run different systems (and by different vendors)
  - Industry standards: storage systems, financial, HSMs, etc.

- Currently deployed: mainly DES/AES and RSA

- Searching for ECC-based key wrapping techniques
Main Candidate: DHIES Encryption

- Elgamal encryption + RO-based key derivation + Enc/Mac

- $G=<g>$: prime-order $q$  
  $H$: hash function (RO)  
  Enc: symmetric encryption  
  Mac: message auth code

- Receiver’s PK: $A=g^a$, message to be encrypted: $M$

- Sender chooses $y \in \mathbb{Z}_q$, sends: $(Y, C, T)$ where

  1. $Y = g^y$, $\sigma = A^y$, $K = H(\sigma)$  
  2. $K \rightarrow K_1, K_2$  
     $C = Enc_{K_1}(M)$  
     $T = Mac_{K_2}(C)$

- Decryption: $\sigma = Y^a$, $K = H(\sigma)$, etc.

- [ABR01]: Scheme is CCA-secure in the ROM
DHIES as Key Wrapping

- DHIES instantiates the KEM/DEM paradigm: \((Y,C,T)\)
  - Key Encapsulation: \(Y = g^y\) encapsulates key \(K = H(A^y)\) under PK \(A\)
  - Data Encapsulation: \((C,T)\) CCA-encrypts data under \(K\)

- Simple, efficient, functional
  - KEM: Can be used to transmit a random fresh key \(K\)
  - DEM: Can be used to transport a previously defined key (and possible associated data)
    - The message \(M\) (under \(C\)) is the transported key and assoc’d data

- Missing: Sender’s authentication
Authenticated Key Wrapping

- DHIES implicitly authenticates the receiver
  - Only intended receiver can read the key/data
  - This is the case for most key wrapping techniques

- But how about sender’s authentication?
  - Who encrypted the tape? Who can it be decrypted for?

- Authenticated key wrapping: Key wrapping with sender’s authentication (mutual authentication)
Authenticating Key Wrapping

- Solution: Add sender’s signature on wrapper $\text{Sign}_S(Y,C,T)$
- But, is it necessary (performance)?
- Is it sufficient?
- No. Needs to bind signer to key, not just to the wrapper
  - For example, Bob encrypts tape, sends wrapper with signature
  - Charlie strips Bob’s signature and generates its own
  - Alice believes the key is owned by Charlie
  - Thus, she may later decrypt the tape for Charlie

Similar to UKS (or identity-misbinding) attacks on KE protocols
Authenticated Wrapping: Equivalent Notions

- Requirements are essentially of a key exchange protocol (w/replay)
  - Alice will never associate with Charlie a key created by Bob (assuming Bob and Alice are honest)
- Considering just KEM part of key wrapping (fresh key) with sender authentication, the following are equivalent:
  - Authenticated Key Wrapping, Authenticated KEM, One-Pass AKE
- With the DEM part (a “message” encrypted with the KEM key) one obtains a notion of “authenticated encryption” or its equivalent
  - UC-secure message transmission (w/replay) [Gjosteen, Krakmo]
  - Secure signcryption [Gorantla et al, Dent]
Authenticated Wrapping & One-Pass KE

- We can use any one-pass AKE to instantiate authenticated KEM $\Rightarrow$ authenticated key wrapping
  - Want something as simple and as close as possible to DHIES
  - More secure and more efficient than adding sender’s signature

$\Rightarrow$ HOMQV (a One-Pass HMQV KE protocol)
Group $G=<g>$, hash function $H$, sym encryption $Enc$, msg auth $Mac$

**DHIES**

- Receiver’s PK: $A=g^a$
- Sender chooses $y$, sends $(Y,C,T)$ where
  1. $Y = g^y$, $\sigma = A^y$
  2. $K = H(\sigma)$
     (2 expon's)
  3. $K \rightarrow K_1, K_2$
  4. $C = Enc_{K_1}(M)$; $T = Mac_{K_2}(C)$

- Decryption: $\sigma = Y^a$, etc
  (1 expon.)

**Authenticated DHIES**

- Receiver’s PK: $A=g^a$; Sender’s PK $B=g^b$
- Sender chooses $y$, sends $(Y,C,T)$ where
  1. $Y = g^y$, $\sigma = A^{y+be}$, $e = H_{\frac{1}{2}}(Y,id_R)$
     $K = H(\sigma, id_S, id_R, Y)$
     (2 expon's)
  2. $K \rightarrow K_1, K_2$
  3. $C = Enc_{K_1}(M)$; $T = Mac_{K_2}(C)$

- Decryption: $\sigma = (YB^e)^a$, etc
  (1.5 expon.)

* Group membership tests or cofactor exponentiation omitted (more later...)*
HOMQV (Hashed One-pass MQV)

- Functionally optimal
  - Minimal performance overhead: Just extra $\frac{1}{2}$ exp for receiver. Free for sender. No extra communication
  - Backwards compatibility with DHIES: Set $B=1$ $b=0$

- How about security?

- We prove security of HOMQV as one-pass key-exchange
  (→ authenticated key wrapping)
HOMQV

- Sender id$_S$ has public key $B=g^b$
- Receiver id$_R$ has public key $A=g^a$
- $S \rightarrow R$: $Y = g^y$
- $S$ computes $\sigma = A^{y+be}$
- $e = H_{1/2}(Y, id_R)$
- $R$ computes $\sigma = (YB^e)^a$
- Both set $K = H(\sigma, id_S, id_R, Y)$
Theorem

- Under Gap-DH in the ROM, HOMQV is a secure one-pass key-exchange protocol
  - Security of one-pass protocol: Canetti-Krawczyk relaxed to allow for key-replays
  - Guarantees mutual authentication in a strong adversarial model
  - Proof: Reduction to XCR signatures (defined in [HMQV])

- Some important leakage-resilience properties
  - Sender’s Forward Security
  - Resistance to leakage of ephemeral Diffie-Hellman exponents (y-security)
Leakage-resilience Properties

- Sender forward security (disclosure of sender’s secret key b does not compromise past keys and messages)
  - Weak FS: For sessions where attacker was passive
  - For full FS: Add a “key confirmation” $\text{Mac}_K^*(1)$ to sender’s message (in particular, satisfied by the DEM part of DHIES)

- $y$-security: The disclosure of ephemeral secret $y$ does not compromise any keys or messages
  - Not even the key/msg transported using $Y=g^y$
  - Moreover: the disclosure of both $y$ and $b$ reveals the msg sent using $y$ but no other msgs sent by $b$’s owner
On the Proof

- Too technical... for a short presentation
- but amazingly precise: The proof tells exactly what the role of each element in the protocol is
- and what the consequences of leakage are for each such element (a, b, y, σ and their combinations)

- Better security, better efficiency: Proof-driven design
  - Get rid of safety margins
  - Compare DHIES+signature vs HOMQV
  - Would you buy it without a proof?
Additional checks

- Proof tells us exactly what properties of incoming values (Y, B, A, etc.) each party needs to check.

- Need to assure $YB^e$ is of order $q$ (no need for separate $Y,B$ test).

- Can implement more efficiently over elliptic curve by cofactor exponentiation:
  - $s = A^{fy}$ instead of $A^y$ or $A^{f(y+be)}$ instead of $A^{(y+be)}$ where $f = \frac{|G'|}{\text{ord}(g)}$ and $G'$ a supergroup containing $g$ (e.g. $G'$ = ell. curve).

- Note: Same needed for DHIES (Y test), hence $\frac{1}{2}$ expon advantage remains.
For Fun
HMQV application to PAKEs

- What’s a PAKE (Password Authenticated Key Exchange)
  - Peers share a password as the only means of authentication (same as pre-shared key but a low-entropy key)
  - As long as attacker does not guess password, security in full
  - Only attack option: online guessing (one passwd per connection)

- Asymmetric PAKE: user has pwd, server stores H(pwd)
  - Above security requirements PLUS: If server is broken into, finding pwd requires a full offline dictionary attack
OPAQUE: Application of HMQV to aPAKE
(the beauty of minimality)

\[ U \ (pwd) \quad a = H(pwd)^r, \ X = g^x \quad S \ (k, c, p_S) \]

\[ b = a^k, \ c = \text{AuthEnc}_{rwd}(p_U, PK_U, PK_S), \ Y = g^y \]

- \( rwd = H(pwd)^k \leftarrow H(b^{1/r}) \)
- \( p_U, PK_U, PK_S \leftarrow \text{AuthDec}_{rwd}(c) \)
- \( SK = \text{HMQV}(x, p_U, Y, PK_S) \quad SK = \text{HMQV}(y, p_S, X, PK_U) \)