Key Exchange Protocol: HMQV and Its Security Proof

Speaker: Yangguang Tian

September 12, 2014

Hugo Krawczyk

HMQV: A High-Performance Secure Diffie-Hellman Protocol

July 5, 2005 Crypto

UNIVERSITY OF WOLLONGONG

Outline

- DH $\rightarrow$ MTI $\rightarrow$ MQV $\rightarrow$ HMQV.
- Relevant signatures: Exponential Challenge-Response (XCR) Signature, Dual CR signature, Hash CR signature and its relevant security proof.
- CK (Canetti-Krawczyk) model.
- Basic security of HMQV.
- Further security of HMQV, like KCI, weak PFS, or Key confirmation.
- Conclusion

Evolution of HMQV

- MTI: Adding authentication to DH. It can not maintain PFS and KCI attack together. For example: MTI (C).
  - A $\rightarrow$ B: $K_b^{r_a}$
  - B $\rightarrow$ A: $K_a^{r_b}$
  - Session key computed by A:
    $SK = (K_a^{r_b})^{r_a/x_a} = (g^{x_a*r_b})^{r_a/x_a} = g^{r_a*r_b}$
- It maintained Forward Secrecy! But it suffers to KCI attack: x_a is known to $\mathcal{A}$, and $\mathcal{A}$ impersonate B to A successfully.
  - $\mathcal{A}$ modify $K_a^{r_b}$ to $K_b^{x_a*r_e}$ where r_e is chosen by $\mathcal{A}$.
  - A computes $SK = (K_b^{x_a*r_e})^{r_a/x_a} = (g^{x_b*x_a*r_e})^{r_a/x_a} = g^{x_b*r_e*r_a}$.
  - $\mathcal{A}$ computes $SK = (K_b^{r_a})^{r_e}$, same as party A.

UNIVERSITY OF WOLLONGONG

Evolution of HMQV

- MQV protocol:
  - $\widehat{A}$ and $\widehat{B}$ exchange $X_a = g^{r_a}$ and $Y_b = g^{r_b}$;
  - $\widehat{A}$ computes $\sigma_a = (Y_b + e * K_b)^{(r_a + d * x_a)} = g^{(r_b + e * x_b)} * (r_a + d * x_a)$, $\widehat{B}$ computes $\sigma_b = (X_a + d * K_a)^{(r_b + e * x_b)}$; Both parties generate $K_{ab} = H(\sigma_a) = H(\sigma_b)$
  - $d = 2^l + (X_a mod 2^l)$, $e = 2^l + (Y_b mod 2^l)$, where $l = |q|/2$. Trade-off between performance and security.
- It achieved both Forward secrecy and KCI, but suffer to UKS attack.

Evolution of HMQV

- $\mathcal{A}$ modify $X_a = g^{r_a}$ to $Y_e = X_a * K_a^d * g^{-u}$, where d same as before, u is chosen by $\mathcal{A}$. But $K_e = g^{u/d_e}$ where $d_e = 2^l + (Y_e mod 2^l)$, it has been registered to CA ($\mathcal{A}$ has to register public key to CA each time).
- B computes $SK_{\mathcal{A}B} = (Y_e * K_e^{d_e})^{r_b + x_b * e} = (X_a * K_a^d * g^{-u} * (g^{u/d_e})^{d_e})^{r_b + x_b * e} = \underline{(X_a * K_a^d)^{r_b + x_b * e}}$. This value same as session key generated by A in MQV. That means B thinks he is communicate with A, but in fact, he is talking to $\mathcal{A}$. In other words, session keys are disclosed to third party (e.g., $\mathcal{A}$) other than A and B.

HMQV

- How to prevent UKS attack? Solution: Binding identity (signed message) and exchanged DH values. For example: $\widehat{A}$ computes $e = H(X_a||\widehat{B})$ and $d = H(Y_b||\widehat{A})$.
- It also achieves KCI, weak PFS, disclose of $g^{x_a * x_b}$ or $g^{r_a * r_b}$, etc.

Description of HMQV

- $\widehat{A}$ and $\widehat{B}$ exchange $X_a = g^{r_a}$ and $Y_b = g^{r_b}$;
- $\widehat{A}$ computes $\sigma_a = (Y_b + e * K_b)^{(r_a + d * x_a)} = \underline{g^{(r_b + e * x_b) * (r_a + d * x_a)}}$, $\widehat{B}$ computes $\sigma_b = (X_a + d * K_a)^{(r_b + e * x_b)}$; Both parties generate $K_{ab} = H(\sigma_a) = H(\sigma_b)$
- $d = H(X_a, \widehat{B})$, $e = H(Y_b, \widehat{A})$
- Additional security-Key confirmation (Actually, it is not necessary!).
  - $MAC_{K_{ab}}(0)$. You can add this to either second message or both second and third message.

XCR signature

- Schnorr identification signature $\rightarrow$ Exponential Challenge Response signature (XCR) and Dual XCR. (Modified Schnorr Identification Signature)
- Comparison between them:
  1. B sends $Y_b = g^{r_b}$ to A;
  2. A sends challenge e to B; (A sends challenge X_a to B)
  3. B computes $s = r_b + x_b * e$; (B computes $X_a^s = X_a^{r_b + x_b * e}$, where $e = H(X_a || m)$ and return X_a^s to A. It changed to signature protocol by using Fait-Shamir transformation)
  4. A accepts if $g^s = Y_b * K_b^e$. (A check $X_a^s = (Y_b * K_b^e)^{x_a}$)
  - This is exponential challenge response signature protocol within bracket.

Reduction for XCR

- Game between forger $\mathcal{F}$ (target singer $\widehat{B}$) and $\mathcal{S}$.
- Setup: $\mathcal{S}$ input (X_0, B) output g^{b*x_0}; $\mathcal{S}$ sets B as pubic key of signer $\widehat{B}$.
- Simulate signing oracle (based on (X, m) chosen by $\mathcal{A}$):
  - Choose $s \in_R \mathcal{Z}_q$, $e \in_R \{0,1\}^l$;
  - Sets $Y = g^s/B^e$;
  - Sets $H(Y||m) = e$. (Random oracle controlled by $\mathcal{S}$)
  - Verification: $\mathcal{S}$ returns (Y, X^s). $\mathcal{A}$ checks whether $X^s = Y * B^e$?
- Repeat experiment: $\mathcal{F}$ outputs (Y_0, m_0, σ), satisfied following conditions: 1, (Y_0, m_0) not used as signature generation; 2, (Y_0, m_0) was queried in random oracle. If all satisfied, then go to repeat experiment. In the end, $\mathcal{F}$ will output another (Y_0, m_0, σ'). Key point here: $H(Y_0, m_0)$ will get e and e' in two experiment. This is Forking technique.
- Solution to $CDH(X_0, B) = (\sigma/\sigma')^{(e-e')^{-1}} = g^{x_b * r_0}$.

DCR

- The challenge chosen by $\mathcal{A}$ changed from X to $X * K_a^d$. Here is how to simulate signer $\widehat{B}$:
  - Choose $s \in_R \mathcal{Z}_q$, $e \in_R \{0,1\}^l$;
  - Sets $Y = g^s / B^e$;
  - Sets $H(Y||m) = e$.
  - Verification: $\mathcal{S}$ returns $(Y, (X * K_a^d)^s)$. $\mathcal{A}$ checks whether $(X * K_a^d)^s = (Y * B^e)^{r_a + d * x_a}$?
- As for forking lemma technique, signed message m' chosen by $\mathcal{S}$, it might generates two different $d = H(m'||X)$ and d' between two experiments. The solution changed to: $CDH(X_0, B) = ((\sigma/(Y * K_b^e)^{d * x_a})/(\sigma'/(Y * K_b^{e'})^{d' * x_a}))^{1/e-e'}$. Here we can see x_a is known to $\mathcal{S}$.
- It is a special case of XCR.

Hashed CR signature

- Compare to previous signatures:
  - $\widehat{B}$ as signer, $\mathcal{S}$ provides outgoing DH value (state) y_i to $\mathcal{A}$. In order to maintain the Consistency, we need DDH oracle. Specifically, check $\mathrm{CDH}(X_i, B) = (\sigma_i / X_i^{y_i})^{1/e_i}$, where $e_i = H(Y_i || m_i)$, X_i is challenge by $\mathcal{A}$.
  - $\mathcal{S}$ changes simulation (simulate signer $\widehat{B}$) slightly, he needs to simulate random value r (in place of $H(\sigma)$).
  - Even if ephemeral DH exponents are leaked, still secure! It can be reduced to KEA1 and GDH problem (two stronger assumptions).

UNIVERSITY OF WOLLONGONG

Reduction for HCR

- Simulate signer $\widehat{B}$:
  - $\mathcal{A}$ sends m_i to $\mathcal{S}$ for signing, then $\mathcal{S}$ will choose $y_i \in_R Z_q$, sets $e_i = H(Y_i||m_i)$, return y_i, e_i to $\mathcal{A}$.
  - $\mathcal{A}$ sends $(Y_i = g^{y_i}, m_i)$ along with challenge X_i to $\mathcal{S}$, then DDH oracle checked by $\mathcal{S}$. If yes, set r_i as response of $H(\sigma)$, otherwise, r_i is random value. Eventually, return r_i to $\mathcal{A}$.
- Upon $\mathcal{S}$ forgery guess (Y_0, m_0, r_0)
  - First two conditions same as before.
  - r_0 was queried in random oracle $H(\sigma_0)$.
- Repeat experiment: Same as before, $H(Y_0, m_0)$ will get e and e' in two experiment. This is Forking technique. Eventually, $\mathcal{S}$ finds solution to CDH same as XCR.

Reduction for HCR

- Consider collision forgery. $\mathcal{A}$ chooses previous response r_i as guess forgery. That means $X_0^{y_0+e_0*b} = X_i^{y_i+e_i*b}$. It can be reduced to GDH problem and KEA1 problem under random oracle model. It is the further security of HMQV.
- KEA1 assumption: One algorithm is given that input (g, g^a), output (C, C^a). There must be another algorithm that given same input, it needs to choose b and computes $(g^b, (g^a)^b)$ satisfy above output.
- Collision forgery (Y_0, m_0, r_0) and collision signature (Y_i, m_i, r_i), where $r_0 \neq r_i$. We wanna proof the probability of this collision is negligible.

CK model

- Game between $\mathcal{A}$ and $\mathcal{S}$:
  - Activate query (party): $\mathcal{S}$ will return exchanged DH exponents (ephemeral public key) and peer identity.
  - State-reveal query (incomplete session): $\mathcal{S}$ will return ephemeral secret key, e.g., x_a.
  - Corrupt query (party): $\mathcal{S}$ will return long term secret key, e.g., x_a.
  - Session key query (complete session): $\mathcal{S}$ will return session key of that session except g-session (used for test session).
  - Test query (fresh session): $\mathcal{S}$ either return session key SK or random value.
- Freshness (Clean): It relates to test session. Variant of CK, define (r_a, x_a, r_b, x_b), any pair of them corrupted except one party's ephemeral and long term key together (r_a, x_a) can be reduced to hard problems.
- Security: $Pr(b' = b) = 1/2 + Adv_{\mathcal{A}}^{HMQV}$

Basic Security of HMQV

- CK model adversary related to forging attack. Due to $SK = H(\sigma)$, we have:
  - If $\mathcal{A}$ could forge σ, then she could distinguish SK and $r \in_R \{0,1\}^k$; If $\mathcal{A}$ could not forge correct σ, then she will not distinguish since random oracle used $H(r)$.
  - Indistinguishability (IND) $\Rightarrow$ Unforgeability.
- CK model adversary related to key replication attack. Which means, $\mathcal{A}$ forces one particular session generates SK that equal to SK of test session. $\mathcal{A}$ issues session key query to that session without forging attack. Trivial attack.

UNIVERSITY OF WOLLONGONG

High-level Analyze

- Consider test session $(\widehat{A}, \widehat{B}, X_0, Y_0)$ and test signature $\pi(\widehat{A}, \widehat{B}, X_0, Y_0)$. Y_0 is controlled by $\mathcal{A}$. Based on Y_0, it implies following cases:
  - C_1 Y_0 not output by $\widehat{B}$;
  - C_2 Y_0 generated by $\widehat{B}$ in a matching session $(\widehat{B}, \widehat{A}, Y_0, X_0)$;
  - C_3 Y_0 generated by $\widehat{B}$ at a session $(\widehat{B}, \widehat{A}^*, Y_0, X^*)$;
  - C_4 Y_0 generated by $\widehat{B}$ at a session $(\widehat{B}, \widehat{A}, Y_0, X^*)$, where $X^* \neq X_0$.
- C_1, C_2, C_3 can be simulated by $\mathcal{F}$, C_4 be simulated by $\mathcal{F}'$

Simulation by $\mathcal{F}$

- Goal of $\mathcal{F}$: input are (X_0, K_b) and signing oracle $\widehat{B}$, output is $g^{r_0 * x_b}$.
- Setup stage: In (n-party, m-session) group setting, $\mathcal{F}$ picks party $\widehat{B}$ and sets public key K_b. Picks party $\widehat{A}$ and sets t-th session as guess-session.
- Training stage: $\mathcal{F}$ answer all queries made by $\mathcal{A}$, especially, $\mathcal{F}$ answer signature and session key together for session key query (stronger assumption).
  - Activate query (party): $\mathcal{F}$ returns $(X_i, Y_i, peer - ID)$;
  - State reveal query (incomplete session): $\mathcal{F}$ returns r_i;
  - Session key query (complete session): $\mathcal{F}$ returns SK_i and σ_i;
  - Corrupt query (party): $\mathcal{F}$ returns x_i.

Simulation by $\mathcal{F}$

- Challenge stage: If $\mathcal{A}$ selects $\widehat{A}$ g-session as test session and peer is $\widehat{B}$, then $\mathcal{F}$ assigns X_0 to $\mathcal{A}$ as outgoing value.
- Conditions for aborting:
  - $\mathcal{A}$ halts with test session that different to g-session.
  - $\mathcal{A}$ corrupts $\widehat{A}$ or $\widehat{B}$.
  - $\mathcal{A}$ issues state-reveal query or session key reveal query to g-session.
  - $\mathcal{A}$ issues state-reveal query or session key reveal query to matching session of g-session.
- Guess stage: Assume $\mathcal{F}$ not abort, then if $\mathcal{A}$ outputs a guess signature π, then $\mathcal{F}$ outputs $(Y_0, \widehat{A}, \pi)$ as a forgery on message m_0 (on challenge X_0) of DCR.

Analyze of $\mathcal{F}$

- IND between real attack and simulation;
  - $\mathcal{F}$ has full information of group parties except $\widehat{B}$, so that simulation is perfect.
  - As for $\widehat{B}$, $\mathcal{F}$ simulates above related queries by using signing oracle $\widehat{B}$.
  - As long as $\mathcal{F}$ does not abort, we can assume it is perfect simulation.
- Simulation covered C_1, C_2, C_3, check validity of triple.

C_1 Obviously, $(Y_0, \widehat{A}, \pi)$ is valid.

C_2 Y_0 only exist in test session, not session queried before. Still valid.

C_3 Y_0 exist in a session $(\widehat{B}, \widehat{A}^*, Y_0, X^*)$, and signer $\widehat{B}$ might answer $\pi(\widehat{B}, \widehat{A}^*, Y_0, X^*)$. But pair $(Y_0, \widehat{A})$ still clean. π is valid too.

UNIVERSITY OF WOLLONGONG

Simulation of $\mathcal{F}'$

- Define session $H(\widehat{A}, \widehat{B}, Y_0, X^*)$ as s^* and π^* as its signature. Simulation almost same as previous one, the differences are:
  - Setup stage: $\mathcal{F}'$ sets party $\widehat{B}$ l-th session that outgoing value is Y_0.
  - Training stage: If $\mathcal{A}$ issues session key query to l-th session, $\mathcal{F}'$ returns random value to $\mathcal{A}$. $\mathcal{F}'$ will never query π^* .
  - Rewind stage: If $\mathcal{A}$ halts without query s^*, $\mathcal{F}'$ same as $\mathcal{F}$; If $\mathcal{A}$ queried s^*, then rewind to the querying point, pick the previous query to random oracle (e.g., v queried by $\mathcal{A}$) and compute $H(v)$ as answer to this session key query (s^*).

Analyze of $\mathcal{F}'$

- IND between real attack and simulation;
  - $\mathcal{A}$ not query s^*; Obviously, same as before.
  - $\mathcal{A}$ query s^*, but not π^*; Random value as response can be accepted by $\mathcal{A}$, still IND.
  - $\mathcal{A}$ query both s^* and π^*. The π^* must be queried by $\mathcal{A}$ (forged by $\mathcal{A}$) and stored in random oracle, and the response of $\mathcal{F}'$ also can be accepted by $\mathcal{A}$.
- The π^* was never queried by $\mathcal{F}'$. Even if s^* queried, but return either random value or previous output of H. So, forgery $(Y_0, \widehat{A}, \pi)$ is valid forgery to $\widehat{B}$.

Further security

- Key Compromise Impersonation attack. We assume $\mathcal{A}$ knows $\widehat{A}$ private key x_a when forging $\widehat{B}$ signature. Remove one specific condition of abort (simulation by $\mathcal{F}$).
- weak Perfect Forward secrecy. HMQV only achieve weak PFS, it suffer to active attack. In order to achieve full PFS, add key conformation so that guarantee DH values are chosen by peers.
- In HCR, it proved that: even if both ephemeral DH exponents are disclosed, it is still secure since reducing to GDH problem. It is proven in HCR reduction.

Conclusion

- Building block is proven secure, the proposed protocol (scheme) reduce to it! Here first proof modified Schnorr signatures are secure, then HMQV can be reduced to unforgeability under CK model. The relationship between IND and unforgeability is key point (random oracle). In other words, forging is one of means to distinguish, the other way is key replication attack.
- Slightly different original CK model, this variant CK model allow $\mathcal{A}$ to access much more info. For example, Hashed CR signature is going to proof the HMQV still secure even if exchanged DH values are leaked.

Thanks for you time!
Question?