On Robust Key Agreement Based on Public Key Authentication (Short Paper)

Feng Hao

Thales E-Security, Cambridge, UK
feng.hao@thales-esecurity.com

Abstract. We describe two new attacks on the HMQV protocol. The first attack raises a serious question on the basic definition of "authentication" in HMQV, while the second attack is generally applicable to many other protocols. In addition, we present a new authenticated key agreement protocol called YAK. Our approach is to depend on well-established techniques such as Schnorr's signature. Among all the related protocols, YAK appears to be the simplest so far. We believe simplicity is an important engineering principle.

1 Introduction

There are two categories of authenticated two-party key agreement protocols: Password Authenticated Key Exchange (PAKE) and Authenticated Key Exchange (AKE) [9]. The former realizes authentication based on a shared password, while the latter based on public key certificates [1,2,4–6]. In this paper, we focus on discussing the second category. To better differentiate it from the first category, we will call it Public Key Authenticated Key Exchange (PK-AKE).

2 Past work

Many PK-AKE protocols claim to be provably secure in a formal model. Among them, the HMQV scheme is perhaps the most well-known example [2]. In this section, we will show two new attacks on HMQV.

The HMQV protocol is modified from MQV [6] with the primary aim for provable security [2]. The most signficant change is that HMQV drops some mandated verification steps in MQV, including the Proof of Possession (PoP) check during the CA registration and the prime-order validation check of the ephemeral public key.

Dropping the public key validations is highly controversial, despite that HMQV has formal security proofs. In one attack, Menezes *et al.* demonstrated disclosing the user's private key without violating the HMQV model definition [7,8]. This attack indicates a flaw in the original design of HMQV.

In acknowledgement of the missing public key validation, Krawczyk revised HMQV in the submission to IEEE P1363 Standards committee [3]. He added the following validation: Alice checks the term YB^e has the correct prime order

and Bob does the same for XA^d (see [2], p. 548, for the definition of symbols.) This change prevents the small subgroup attack in [8], but decreases the claimed efficiency. However, instead of validating the static and ephemeral public keys separately as in MQV, the revision chooses to optimize efficiency by mixing the two operations together. This causes the problem as below.

We now report a new "invalid public key attack" on HMQV. For illustration, we follow the same symbols used in the original description of HMQV (see [2], p. 548). In both the original and revised versions of HMQV, the CA is only required to check the submitted public key is not 0. The attack works as follows. Assume Bob (attacker) registers a small subgroup element $s \in G_w$ as the public key where $w|p-1$. Bob chooses an arbitrary value $z \in Z_q$. Let $Y = g^z \cdot s'$ where s' is an element in the same small subgroup G_w. Exhaustively, Bob tries every element s' in G_w such that $YB^e = g^z \cdot s' \cdot s^e = g^z$. In other words, the small subgroup elements s and s' cancel each other out. Suppose $\bar{H}$ works like a random oracle as assumed in HMQV. Then, for each try of s', the probability of finding $s' \cdot s^e = 1$ is $1/w$. It will be almost certain to find such s' after searching all w elements in G_w (if not then change a different z and repeat the procedure). Following the HMQV protocol, Bob sends $Y = g^z \cdot s'$ to Alice. Alice checks YB^e has the correct prime order and computes the session key $\kappa = H((YB^e)^{x+da}) = H(g^{z\cdot(x+da)})$. Because Bob knows z, he can compute the same session key κ and successfully authenticate himself to Alice.

The fact that an obviously invalid public key is totally undetected by all flows in HMQV is unsettling. This raises a serious question on the basic definition of "authentication" in HMQV – Bob does not even have a private key, yet he is able to successfully pass all authentication checks. In fact, anyone can do the same pre-computation as above and authenticate to Alice as "Bob". In one attacking scenario, Bob (the attacker) may at any time suddenly repudiate all previous authenticated transactions with Alice by telling the judge that his public key is invalid, so anyone can impersonate him. (Bob's certificate is publicly available.) In comparison, MQV does not have this problem.

The other attack on HMQV happens when two parties use the same certificate during self-communication [2]. Self-communication is a useful application in practice. For example, a mobile user and the desktop computer may hold the same static private key (registering two public key certificates costs more). Krawczyk formally proved that self-communication is "secure" in HMQV [2]. However, the formal model in [2] only considers the user talking to one copy of self, but neglects the possibility that the user may talk to multiple copies of self at the same time. This deficiency is common among other formal models too [1,4,11]. The attack works as follows (also see Figure 1):

1. Alice initiates the connection to a copy of herself by sending g^x. The connection is intercepted by Mallory who pretends to be Alice-1.
2. Mallory starts a separate session by pretending to be Alice-2. He initiates the connection by sending to Alice g^x (this is possible because HMQV does not require the sender to know the exponent).
3. Alice responds to Alice-2 by sending g^y.

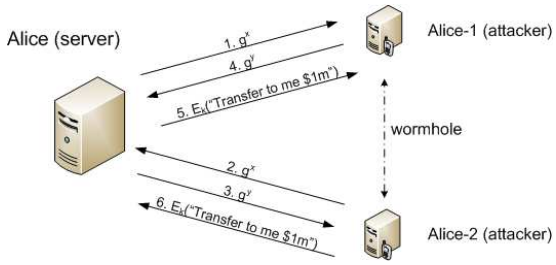

Fig. 1. Wormhole attack on HMQV

4. Mallory replays g^y to Alice as Alice-1.
5. Alice derives a session key and sends an encrypted message to Alice-1, say: "Transfer to me $1m".
6. Mallory replays the encrypted message to Alice. (After receiving money from Alice, Mallory disconnects both connections.)

In the above attack, we only demonstrated the attack against the two-pass HMQV (implicit authentication). For the three-pass HMQV (explicit authentication), the attack works exactly the same. Also, we have omitted the identities in the message flows, because they are all identical according to the HMQV specification [2].

This attack is essentially an unknown key sharing attack. Alice thinks she is communicating to a mobile user with the same certificate, but she is actually communicating to herself. The attacker does not hold the private key, but he manages to establish two fully authenticated channels with Alice (server). The same attack also applies to other PK-AKE schemes, including NAXOS [4], KEA+ [5], CMQV [11], MQV [6], and SIG-DH [1] etc, despite that many of them have formal security proofs.

3 The YAK protocol

In this section, we propose a new PK-AKE protocol called YAK[1]. Let G denote a subgroup of Z_p^* with prime order q in which the Computational Diffie-Hellman problem (CDH) is intractable. Let g be a generator in G. The two communicating parties, Alice and Bob, both agree on (G, g).

3.1 stage 1: public key registration

In stage 1, Alice and Bob register static public keys from a Certificate Authority (CA). Alice selects a random secret $a \in_R Z_q$ as her private key. Similarly, Bob selects $b \in_R Z_q$ as his private key.

[1] The yak lives in the Tibetan Plateau where environmental conditions are extremely adverse.

CA-Registration Alice sends to the CA g^a with a knowledge proof for a. Similarly, Bob sends to the CA g^b with a knowledge proof for b.

The sender needs to produce a valid knowledge proof to demonstrate the Proof of Possession (PoP) of the private key. As an example, we can use Schnorr's signature, which is provably secure in the random oracle model [9]. Let H be a secure hash function. To prove the knowledge of the exponent for $X = g^x$, one sends $\{\text{SignerID}, \text{OtherInfo}, V = g^v, r = v - x \cdot h\}$ where SignerID is the *unique* user identifier (also called Distinguished Name [10]), OtherInfo includes auxiliary information to indicate this is a request for certifying a static public key and may include other practical information such as the name of the algorithm etc, $v \in_R Z_q$ and $h = H(g, V, X, \text{SignerID}, \text{OtherInfo})$. The CA checks that X has prime order q and verifies that $V = g^r X^h$ (computing $g^r X^h$ requires roughly one exponentiation using the simultaneous computation technique [9]).

3.2 stage 2: key agreement

Alice and Bob execute the following protocol to establish a session key. For simplicity of discussion, we explain the case that Alice and Bob have different certificates ($a \neq b$) and will cover self-communication later.

YAK-protocol Alice selects $x \in_R Z_q$ and sends out g^x with a knowledge proof for x. Similarly, Bob selects $y \in Z_q$ and sends out g^y with a knowledge proof for y.

When this round of communication finishes, Alice and Bob verify the received knowledge proof to ensure the other party possesses the ephemeral private key. They also need to ensure the identity (i.e., SignerID) in the knowledge proof must match the one in the public key certificate.

Upon successful verification, Alice computes a session key $\kappa = H((g^y \cdot g^b)^{x+a}) = H(g^{(x+a)(y+b)})$. And Bob computes the same session key: $\kappa = H((g^x \cdot g^a)^{y+b}) = H(g^{(x+a)(y+b)})$.

In YAK, Alice needs to perform the following exponentiations: one to compute an ephemeral public key (i.e., g^x), one to compute the knowledge proof for x (i.e., g^{v_x}), two to verify the knowledge proof for the exponent of $Y = g^y$ (i.e., Y^q and $g^{r_y} Y^{h_y}$) and finally one to compute the session key $(Y \cdot B)^{x+a}$. Thus, that is five in total: $\{g^x, g^{v_x}, Y^q, g^{r_y} Y^{h_y}, (Y \cdot B)^{x+a}\}$.

Among the above operations, some are merely repetitions. To explain this, let the bit length of the exponent be $L = \log_2 q$. Then, computing g^x alone would require roughly $1.5L$ multiplications which include L square operations and $0.5L$ multiplications of the square terms. However, the same square operations need not be repeated for other items with the common base. If we factor this in, it will take $(1+0.5\times 3)L = 2.5L$ to compute $\{g^x, g^{v_x}, g^{r_y}\}$, and another $(1+0.5\times 2)L = 2L$ to compute $\{Y^q, Y^{h_y}\}$ and finally $1.5L$ to compute $(Y \cdot B)^{x+a}$. Hence, that is in total $6L$, which is equivalent to $6L/1.5L = 4$ usual exponentiations. This is quite comparable to the 3.5 exponentiations in MQV (which cannot reuse the square terms since the bases are different).

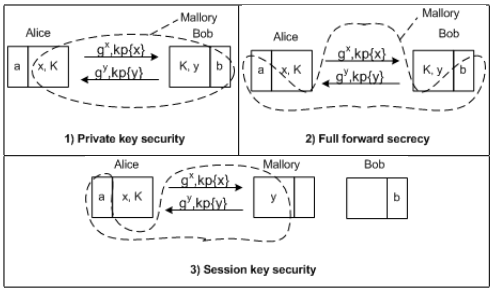

Fig. 2. The oracle diagrams in YAK. Alice is honest.

4 Security analysis

We formulate the following requirements for the PK-AKE protocol.

1. **Private key security**: An attacker cannot learn any useful information about the user's static private key even if he is able to learn all session specific secrets in any session.
2. **Full forward secrecy**: Session keys that were securely established in the past uncorrupted sessions will remain incomputable in the future even when both users' static private keys are disclosed.
3. **Session key security**: An attacker cannot compute the session key if he impersonates a user but has no access to the user's private key.

The first requirement is generally not covered by a formal model, but we think it is crucially important. For example, both the SIG-DH [1] and (original) HMQV [2] protocols have been formally proven secure in the CK model. Yet attacks reported in [4] and [8] show that in both protocols, an attacker is able to disclose the user's private key. In the second requirement, we use "full" to distinguish it from the "half" forward secrecy, which only allows one user's private key to be revealed (e.g., KEA+ [5]). The third requirement has already covered the Key Compromise Impersonation (KCI) attack [6]. The "invalid public key" attack in Section 2 indicates that HMQV does not satisfy this property.

The goal of our design is to make the best use of well-established techniques such as Schnorr's signature. This strategy allows us to leverage upon the provable results of Schnorr's signature (see [9]), and thus greatly simplify the security analysis. In the following, we will provide a simple and intuitive analysis, while leaving detailed proofs to a full paper.

First, let us discuss the private key security. Without loss of generality, we assume Alice is honest. As shown in Figure 2 (1), Mallory totally controls Bob's static and ephemeral private keys. Additionally, he has the extreme power that allows him to learn Alice's transient secrets in an arbitrary session. The only power that he does not have is the access to Alice's private key.

A sketch of the proof goes as follows. The knowledge proofs defined in YAK prove that Mallory (the attacker) knows the value of y and b. He also knows

Alice's public key g^a. By revealing Alice's transient secrets (i.e., x and K) in a session, he learns x and $K = g^{(a+x)(b+y)}$. But learning K does not give Mallory any information, because he can compute it by himself from $\{x, y, b, g^a\}$. Effectively, Mallory can actually simulate the attack all by himself through defining arbitrary values of x, y, and b. Clearly, he does not learn any useful information about Alice's private key through his own simulations.

Next, we discuss the full forward secrecy requirement. The definition (see Section 2) specifies that the past sessions must be "uncorrupted", namely the session-specific transient secrets must remain unknown to the attacker. In YAK, this means x, y and K must remain unknown to the attacker. Obviously, knowing K would have trivially broken the past session. Also, if Mallory can learn any ephemeral exponent x or y in the past session in addition to knowing both parties' static private keys, he has possessed the power to trivially compromise any PK-AKE. Therefore, as shown in Figure 2 (2), we assume the attacker knows both Alice and Bob's private keys, but not any transient secrets in the past session.

We explain the YAK's fulfillment of the full forward secrecy under the Computational Diffie-Hellman (CDH) assumption. To obtain a contradiction, we assume the attacker can compute $K = g^{(a+x)(b+y)}$. The attacker knows the values of a and b (see Figure 2 (2)). The ephemeral public keys g^x and g^y are public information. Therefore, Mallory can compute g^{ab}, g^{ay} and g^{bx}. Now, we can solve the CDH problem as follows: given g^x and g^y where $x, y \in_R Z_q$, we use Mallory as an oracle to compute $g^{xy} = K/(g^{ab} \cdot g^{ay} \cdot g^{bx})$. This, however, contradicts the CDH assumption.

Finally, we discuss the session key security requirement. As shown in Figure 2 (3), Mallory does not hold Bob's private key but he tries to impersonate Bob. We assume the powerful Mallory even knows Alice's private key a. The only power he does not have is the access to Alice and Bob's session states. If Mallory can access Alice's session state, he can impersonate anyone to Alice – he just needs to "steal" the session key that Alice computes in the transient memory. Similarly, if Mallory can access Bob's session state, he can impersonate Bob to anybody by waiting until Bob computes the session key and then stealing it.

In this case, the assumed attacker is less powerful than the one described in the "private key security" argument. Previously, the attacker was able to corrupt an arbitrary session of Alice's or Bob's. He however had learned no useful information than what he can simulate. On discussing the session key security, we assume the attacker no longer has access to either user's session state. This change is necessary, and is consistent with the extreme-adversary principle [4]: the only powers that an attacker does not have are those that would allow him to trivially break any PK-AKE protocol.

The YAK protocol satisfies the session key security requirement under the CDH assumption. As shown in Figure 2 (3), Mallory does not possess Bob's static private key, or have access to either Alice or Bob's session state. To obtain a contradiction, we assume Mallory is able to compute $K = g^{(a+x)(b+y)}$. Bob's public key g^b is public information. Mallory knows Alice's private key a. The

knowledge proof in the protocol proves that Mallory also knows the value y. Hence, he can compute g^{ab}, g^{ay} and g^{xy}. Now, we can solve the CDH problem as follows: given g^b and g^x where $x, b \in_R Z_q$, we use Mallory as an oracle to compute $g^{bx} = Z/(g^{ab} \cdot g^{ay} \cdot g^{xy})$. This, however, contradicts the CDH assumption, which shows YAK satisfies the session key security requirement.

5 Self-communication

The user identity is an important parameter in the protocol definition. In the past literature, almost all PK-AKE protocols use the Distinguished Name (DN) in the user's X.509 certificate as the user identity [1, 2, 4–6]. This practice also carries over to the self-communication mode [2], which causes the "wormhole attack" (see Section 2). In the self-communication mode, the two parties are still distinct entities and hence, naturally require different identities.

To enable self-communication in YAK, we need to ensure the SignerID in the Schnorr's signature remains unique. This is to prevent Bob from replaying Alice's signature back to Alice and vice versa. One solution is to simply attach an additional identifier to the mobile stations using the same certificate. For example, when Alice (server) is communicating to the nth copy of herself (mobile station), Alice uses "Alice" as her SignerID to generate the Schnorr's signature and the nth copy uses "Alice-n" as its SignerID. Thus, Alice-n cannot replay Alice's signature back to Alice and vice versa. This solution is also generically applicable to fix the self-communication problem in past protocols [1, 2, 4–6].

Though self-communication is considered a useful feature [2], one should be careful to enable this feature only when it is really needed. This is because, when enabled, it may have negative impact on the theoretical security. In Section 4, under the "private key security", we have explained that, under normal operations (using different certificates $a \neq b$), an attacker cannot learn $g^{a \cdot a}$ from a corrupted session. However, if self-communication is enabled in YAK, we essentially allow $a = b$, hence the attacker can learn $g^{a \cdot a}$ from a corrupted session. This implies we would need a stronger assumption than CDH to prove the "session key security". This is undesirable, but to our best knowledge, no PK-AKE protocol is reducible to the CDH assumption with the self-communication enabled. In comparison, in HMQV [2], the attacker can learn $g^{a \cdot a}$ from a corrupted session regardless whether the self-communication is enabled.

6 Conclusion

In this paper, we report two new attacks on the HMQV protocol. In addition, we present a new authenticated key agreement protocol called YAK, and analyze its robustness in an extremely adverse condition: the only powers that an attacker does not have are those that would allow him to trivially break any other protocols. Overall, YAK demonstrates robust security under the Computational Diffie-Hellman assumption in the random oracle model.

Acknowledgment

We thank Alfred Menezes and Berkant Ustaoglu for their generous advice and invaluable comments. We thank Lihong Yang for helping improve the readability.

References

1. Canetti, R., Krawczyk, H.: Analysis Of Key-Exchange Protocols And Their Use For Building Secure Channels. In: Pfitzmann, B. (ed.) Eurocrypt 2001. LNCS, vol. 2045, pp. 453-474. Springer, Heidelberg (2001)
2. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546-566. Springer, Heidelberg (2005). The full version (2005), `http://eprint.iacr.org/2005/176.pdf`
3. Krawczyk, H.: HMQV in IEEE P1363. Submission to the IEEE P1363 Standardization Working Group, (2006). `http://grouper.ieee.org/groups/1363/P1363-Reaffirm/submissions/krawczyk-hmqv-spec.pdf`
4. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger Security Of Authenticated Key Exchange. In: Susilo, W. *et al.* (eds.) Provable Security 2007. LNCS, vol. 4784, pp. 1-16. Springer, Heidelberg (2007)
5. Lauter, K., Mityagin, A.: Security Analysis Of KEA Authenticated Key Exchange Protocol. In: Yung, M. (ed.) PKC 2006. LNCS, vol. 3958, pp. 378-394. Springer, Heidelberg (2006)
6. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An Efficient Protocol For Authenticated Key Agreement. Designs, Codes and Cryptography 28 (2), 119-134 (2003)
7. Menezes, A.: Another Look At HMQV. J. of Mathematical Cryptology 1(1), 47-64, (2007)
8. Menezes, A., Ustaoglu, B.: On The Importance Of Public-Key Validation In The MQV And HMQV Key Agreement Protocols. In: INDOCRYPT 2006. Barua, R., Lange, T. (eds.) LNCS, vol. 4329, pp. 133-147. Springer, Heidelberg (2006)
9. Menezes, A., Van Oorschot, P., Vanstone, S.: Handbook Of Applied Cryptography. CRC Press (1996)
10. Mitchell, C.: Security For Mobility. The Institution of Electrical Engineers (2004)
11. Ustaoglu, B.: Obtaining A Secure And Efficient Key Agreement Protocol For (H)MQV And NAXOS. Designs, Codes and Cryptography 46 (3), 329-342 (2008)