Int. Computer Symposium, Dec. 15-17, 2004, Taipei, Taiwan.

Cryptanalysis of an Enhanced Authentication Key Exchange Protocol

Fuw-Yi Yang
Department of Applied Mathematics
National Chung Hsing University
E-mail:yangfy@ms7.hinet.net

Jinn-Ke Jan
Department of Computer Science
National Chung Hsing University
E-mail:jkjan@cs.nchu.edu.tw

Abstract-*An enhanced authentication key exchange protocol was proposed to exchange multiple session keys between two participants at a time. This paper shows that this enhanced protocol is insecure under the known session key attack, the known long-term private key attack, and the signature forgery attack.*

Keywords: Authentication, Diffie-Hellman key exchange, forward secrecy.

1. Introduction

In order to achieve secret communication over an insecure channel, the messages must be transmitted in cipher. Therefore, two participants must agree on a shared session key before starting to transmit/receive messages. The shared session key is used to encrypt plaintext or decrypt ciphertext. The famous Diffie-Hellman key exchange protocol [1] is often used to establish a shared session key for every protocol execution. However, this protocol does not authenticate the participants engaging in exchanging their session keys. This gives chance to an adversary to impersonate one of the participants. Thus, this protocol is suffered from the middleman attacks.

An enhanced protocol is proposed in [2], henceforth called *H-protocol*. To resist the attack of middleman, *H-protocol* has been furnished with the capability of authenticating participants. In addition, the participants can exchange multiple session keys at one execution of the *H-protocol*. Therefore, the users of *H-protocol* have an efficient way to share a set of session keys.

However, *H-protocol* lacks rigorous treatment on security. Section 3 will present three attacks on the *H-protocol*, *i.e.*, the known session key attack, the known long-term private key attack, and the signature forgery attack. The first two attacks concern information leakage when losing session keys and long-term private key. The third attack considers forging the signatures without the knowledge of user’s signing key. The paper show that *H-protocol* cannot withstand any of these attacks.

2. Review of H-protocol

The system authority chooses a large prime p to initialize the system. Let g be the generator of the finite field $GF(p)$. Assume the participants Alice and Bob have registered at the system. Therefore, Alice has a long-term private key x_a, long-term public key $y_a = g^{x_a} \bmod p$, and a certificate $cert(y_a)$. The certificate $cert(y_a)$ is a signature of a trust third party (TTP) on the public key y_a. Similarly, Bob has a long-term private key x_b, long-term public key $y_b = g^{x_b} \bmod p$, and a certificate $cert(y_b)$. After registering on the system, these two participants can exchange a set of authenticated Diffie-Hellman keys by executing the *H-protocol*. The following steps describe the details of the *H-protocol*.

Step 1. Alice randomly selects two elements, k_{a1} and k_{a2}, from the finite field $GF(p)$. The quantities $r_{a1} = g^{k_{a1}} \bmod p$, $r_{a2} = g^{k_{a2}} \bmod p$, and $s_a = x_a (r_{a1} \oplus r_{a2}) + k_{a1} r_{a2} \bmod p-1$ are computed, respectively. Then, the initiator Alice sends the message $m_{a1} = \{r_{a1}, r_{a2}, s_a, cert(y_a)\}$ to the recipient Bob.

Step 2. Upon receiving the message m_{a1}, Bob first verifies the certificate $cert(y_a)$. Then he starts on verifying the validity of m_{a1} by checking $g^{s_a} = y_a^{r_{a1} \oplus r_{a2}} r_{a1}^{r_{a2}} \bmod p$. A valid verification leads Bob to construct a response message m_{b1}; otherwise, Bob stops this instance of *H-protocol*.

To form a response message, Bob picks two random elements, k_{b1} and k_{b2}, from the finite field $GF(p)$. The quantities $r_{b1} = g^{k_{b1}} \bmod p$, $r_{b2} = g^{k_{b2}} \bmod p$, and $s_b = x_b (r_{b1} \oplus r_{b2}) + k_{b1} r_{b2} \bmod p-1$ are computed, respectively. Then, Bob sends the response message $m_{b1} = \{r_{b1}, r_{b2}, s_b, cert(y_b)\}$ to Alice. While constructing a response message, Bob also computes a set of Diffie-Hellman keys, *i.e.*, the shared session keys $K_1 = r_{a1}^{k_{b1}} \bmod p$, $K_2 = r_{a2}^{k_{b1}} \bmod p$, $K_3 = r_{a1}^{k_{b2}} \bmod p$, and $K_4 = r_{a2}^{k_{b2}} \bmod p$.

Step 3. Alice verifies the certificate $cert(y_b)$ when receiving the message m_{b1}. In order to certify that m_{b1} is sent from Bob, Alice must check whether $g^{s_b} = y_b^{r_{b1} \oplus r_{b2}} r_{b1}^{r_{b2}} \bmod p$ holds true. Alice stops the execution if the check is invalid; otherwise, Alice also computes a set of shared session keys $K_1 = r_{b1}^{k_{a1}} \bmod p$, $K_2 = r_{b1}^{k_{a2}} \bmod p$, $K_3 = r_{b2}^{k_{a1}} \bmod p$, and $K_4 = r_{b2}^{k_{a2}} \bmod p$.

Therefore, Bob and Alice have agreed on a set of four session keys after executing the protocol cooperatively. If both participants have chosen n random elements from the finite field $GF(p)$ during executing the protocol, then they will agree on a set of n^2 session keys. In order to achieve perfect forward secrecy, only n^2-1 session keys are available to participants.

3. Cryptanalysis

In order to investigate the security of *H-protocol*, three famous attacks are mounted to attack it. The details are shown in the following subsections.

3.1 Known session key attack

The known session key attack considers what are the side effects if some previous session keys are disclosed. No secret information of the participants or system must be revealed by the disclosure of previous session keys. In the followings, we show how to compute the long-term Diffie-Hellman key $y_{ab} = g^{x_a x_b} \bmod p$ if the session key K_1 is compromised. Express s_a and s_b in (1) and (2).

$$s_a = x_a (r_{a1} \oplus r_{a2}) + k_{a1} r_{a2} \bmod (p-1) \quad (1)$$

$$s_b = x_b (r_{b1} \oplus r_{b2}) + k_{b1} r_{b2} \bmod (p-1) \quad (2)$$

$$x_a\, x_b (r_{a1} \oplus r_{a2}) (r_{b1} \oplus r_{b2}) = (s_a s_b - k_{a1} r_{a2} s_b - k_{b1} r_{b2} s_a + k_{a1} r_{a2} k_{b1} r_{b2}) \bmod (p-1) \quad (3)$$

$$y_{ab}^{(r_{a1} \oplus r_{a2})(r_{b1} \oplus r_{b2})} = g^{s_a s_b}\, r_{a1}^{-r_{a2} s_b}\, r_{b1}^{-r_{b2} s_a}\, K_1^{r_{a2} r_{b2}} \bmod p \quad (4)$$

$$u = 1 / ((r_{a1} \oplus r_{a2}) (r_{b1} \oplus r_{b2})) \bmod (p - 1) \quad (5)$$

$$y_{ab} = (g^{s_a s_b}\, r_{a1}^{-r_{a2} s_b}\, r_{b1}^{-r_{b2} s_a}\, K_1^{r_{a2} r_{b2}})^u \bmod p \quad (6)$$

Equation (3) is obtained by multiplying (1) by (2). Raising both sides of (3) to the exponentials of the generator g, (4) is obtained. As can be seen in (5) and (6), given the quantity of the session key

K_1, the long-term Diffie-Hellman key y_{ab} is derived, where the quantities s_a, s_b, r_{a1}, r_{a2}, r_{b1}, and r_{b2} are obtained by listening on the public channel.

3.2 Perfect forward secrecy (Known long-term secret key attack)

A very desirable security property of key exchange protocol is the perfect forward secrecy. Communications are usually among insecure channels. The insecure channels have many unacceptable properties, *e.g.*, the adversaries can eavesdrop on, intercept, and modify data over the channels. Therefore, the shared session keys are used to encrypt the confidential messages before putting them in an insecure transmission channel. Suppose that a secure encryption function is used. Then, the adversaries cannot learn any information about the confidential messages since they do not know the session keys used.

Assume that an adversary has recorded some ciphertext from an insecure channel; and further, the exposure of participant's long-term secret key lead the session keys to be revealed. Thus, the adversary is able to decrypt those intercepted ciphertext and thereby reads the confidential messages that were sent in the past sessions. This result would be undesirable. Hence, a stronger security property is required. This is the property of perfect forward secrecy. It requires that the session keys should be concealed even the participant's long-term secret key is disclosed.

From (4), anyone can compute the session key K_1 if y_{ab} is available.

From (7), the adversary listening on the public channel can compute the session key K_1 if y_{ab} is available. The details are as follows.

$$v = 1 / (r_{a2}\, r_{b2}) \bmod (p-1)$$

$$K_1 = (\, y_{ab}^{(r_{a1} \oplus r_{a2})(r_{b1} \oplus r_{b2})}\, g^{-s_a s_b}\, r_{a1}^{r_{a2} s_b}\, r_{b1}^{r_{b2} s_a}\,)^v \bmod p \qquad (7)$$

From (1), the adversary can compute the quantity k_{a1} if Alice's private key x_a is available. Thus the session keys K_1 and K_3 are computed. Similarly, From (2), the adversary can compute the quantity k_{b1} and the session keys K_1 and K_2 if Bob's private key x_b is available.

Therefore the *H-protocol* does not satisfy the requirement of perfect forward secrecy, since the disclosure of either Alice's or Bob's long-term private keys x_a or x_b enables an adversary to compute the shared session key K_1, K_2, or K_3.

3.3 Signature forgeries attack

Bob verifies the received message $m_{a1} = \{r_{a1}, r_{a2}, s_a, cert(y_a)\}$ by checking $g^{s_a} = y_a^{r_{a1} \oplus r_{a2}}\, r_{a1}^{r_{a2}} \bmod p$. Similarly, Alice certifies the received message $m_{b1} = \{r_{b1}, r_{b2}, s_b, cert(y_b)\}$ by the verification equation $g^{s_b} = y_b^{r_{b1} \oplus r_{b2}}\, r_{b1}^{r_{b2}} \bmod p$. Essentially, $\{r_{a1}, r_{a2}, s_a\}$ and $\{r_{b1}, r_{b2}, s_b\}$ are one of variants of ElGamal signatures [3]. The following steps show how to counterfeit signatures so as to pass the verification equation. Assume that an adversary wants to construct a message $m_{a1} = \{r_{a1}, r_{a2}, s_a, cert(y_a)\}$.

Step 1. The certificate $cert(y_a)$ is obtained from a previous intercepted message.

Step 2. Let $r_{a1} = g^v\, y_a^u \bmod p$, where v is chosen randomly from $Z_{(p-1)}$ and $-u = 2 \bmod (p-1)$.

Step 3. Substituting $r_{a1} = g^v\, y_a^u \bmod p$ into verification equation (8), (9) is obtained. Equations (10) and (11) are obtained by combining the terms with the same base in (9).

$$g^{s_a} = y_a^{r_{a1} \oplus r_{a2}}\, r_{a1}^{r_{a2}} \bmod p \qquad (8)$$

$$g^{s_a} = y_a^{r_{a1} \oplus r_{a2}}\ g^{v r_{a2}}\ y_a^{u r_{a2}} \mod p \quad (9)$$

$$r_{a1} \oplus r_{a2} = -u\ r_{a2} = 2\ r_{a2} \mod (p-1) \quad (10)$$

$$s_a = v\ r_{a2} \mod (p-1) \quad (11)$$

Step 4. Assume that the most significant bit of r_{a2} is 0 such that the quantity $2\ r_{a2}$ is derived by merely left shifting one bit on all bits of r_{a2} (the least significant bit of the result is filled by 0). Please note that this assumption occurs with high probability. Then, r_{a2} can be solved from (10) by the following equations. Let $r_{a2}[1]$ and $r_{a2}[|p|]$ denote the least significant bit and the most significant bit of r_{a2}.

$r_{a2}[1] = r_{a1}[1]$,
$r_{a2}[2] = r_{a1}[2] \oplus r_{a2}[1]$,...,
$r_{a2}[j] = r_{a1}[j] \oplus r_{a2}[j-1]$,...,
$r_{a2}[|p|] = r_{a1}[|p|] \oplus r_{a2}[|p|-1]$.
If $r_{a2}[|p|] \neq 0$, redo Step 2.

Therefore, without knowing Alice's long-term private key the adversary has constructed a message $m_{a1} = \{r_{a1}, r_{a2}, s_a, cert(y_a)\}$, which would pass the verification equation $g^{s_a} = y_a^{r_{a1} \oplus r_{a2}}\ r_{a1}^{r_{a2}}$ $mod\ p$. Although the adversary cannot compute the shared session keys, this undesired result may still cause problem, if the shared session keys are used to encrypt random messages and no further key confirmation protocol is used.

4. Conclusion

It is shown that H-proto*col* is vulnerable to the known session key attack, known long-term secret key attack, signature forgery attack.

References

1. W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, Vol. 22, pp. 644-654, 1976.
2. M. S. Hwang, T. Y. Chang, S. C. Lin, and C. S. Tsai, "On the security of an enhanced authentication key exchange protocol," In *Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)*, IEEE, Volume 2, pp. 160-163, 2004.
3. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inform. Theory*, IT-31, (4), pp. 469-472, 1985.