An Enhanced and Secure Protocol for Authenticated Key Exchange

Fuw-Yi Yang and Jinn-Ke Jan*

Department of Applied Mathematics, National Chung Hsing University

Taichung Taiwan 402, R.O.C., E-mail: yangfy@ms7.hinet.net

*Department of Computer Science, National Chung Hsing University

Taichung Taiwan 402, R.O.C., 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, known long-term private key attack, signature forgery attack, and replay attack. This paper also proposes an enhanced and secure key agreement protocol for exchanging multiple session keys in one run of the protocol. The protocol is secure against the attacks mentioned above. Besides, a formal proof is given to guarantee the security of the proposed protocol under other potential attacks.*

Keywords *Authentication, Diffie-Hellman key exchange, perfect forward secrecy, session key.*

1. INTRODUCTION

In order to achieve secret communication over an insecure channel, the messages must be transmitted in cipher. Therefore, the 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 well-known Diffie-Hellman key exchange protocol proposed in [1] is often used to establish a shared session key. Assume that Alice and Bob have agreed on a large prime p and g, such that g is a primitive element in the multiplicative group Z_p^*. Alice randomly chooses an element x from the additive group $Z_{(p-1)}$, computes $X = g^x \bmod p$, and sends X to Bob. Similarly, Bob chooses a random element y from $Z_{(p-1)}$, computes $Y = g^y \bmod p$, and sends Y to Alice. Then, Alice computes the shared session key $K_A = Y^x = g^{xy} \bmod p$; Bob computes the shared session key $K_B = X^y = g^{yx} \bmod p$. Both K_A and K_B are equal, since $g^{xy} = g^{yx} \bmod p$. Although the quantities X and Y are transmitted over an insecure channel, no one listening on the channel can compute the shared key. The protocol's security is based on the assumption that g^x and g^y are known making it difficult to compute the quantity $g^{xy} \bmod p$. However, this protocol does not authenticate the participants engaging in exchanging their session keys.

This allows an adversary to impersonate one of the participants. Thus, this protocol is vulnerable to the middleman attack.

An enhanced protocol was proposed in [2], henceforth called *H-protocol*. To resist the attack of the middleman, the *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 *H-protocol* provides a more efficient way to share session keys and is more secure than that of the original Diffie-Hellman key exchange protocol.

However, the *H-protocol* is still insecure. This paper will present four attacks on the *H-protocol*, *i.e.*, the known session key attack, the known long-term private key attack, the signature forgery attack, and the replay attack.

In the first attack, if an adversary obtains a shared session key, then the adversary can compute the long-term Diffie-Hellman key shared between Alice and Bob, *i.e.* $y_{ab} = g^{x_a x_b} \bmod p$, where x_a and x_b are Alice and Bob's long-term private keys. In the second attack, when obtaining the long-term private key, the adversary can compute the previous session keys and thus decrypt those ciphertext that have been transmitted over a public channel. The third attack demonstrates that an adversary can forge the signatures (messages exchanged) without knowing the participant's long-term private key. The *H-protocol* is unprotected under the fourth attack — the replay attack. This attack is simply a retransmission of a previous message.

After cryptanalysis, the paper proposes a secure protocol for authenticated key exchange, which provides the same functionality as that of the *H-protocol*. The paper shows that the proposed protocol is secure under the attacks mentioned above. Furthermore, the paper provides a formal proof to guarantee the protocol's security under other unknown attacks. Thus the proposed protocol is not merely able to mend the security leakage of the *H-protocol*; it is intended to provide a secure way to exchange multiple session keys in one run of the protocol.

The paper is organized as follows. Section 2 reviews the *H-protocol*. Section 3 demonstrates that the *H-protocol* is vulnerable under the four attacks mentioned above. The proposed protocol will be described in Section 4. Section 5 investigates the proposed protocol's security, and finally Section 6 concludes the paper.

2. REVIEW OF THE *H-PROTOCOL*

The system authority chooses a large prime p. Let g be the primitive root in the finite field $GF(p)$. Assume the participants Alice and Bob have registered on 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 trusted third party (TTP) on the public key y_a and the identity of Alice. 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 additive group $Z_{(p-1)}$. 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. 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 checking $g^{s_a} = y_a^{r_{a1} \oplus r_{a2}} \; r_{a1}^{r_{a2}} \bmod p$ to verify the message m_{a1}. A valid verification leads Bob to construct a response message; otherwise, Bob stops this stage of the *H-protocol*.

To construct a response message, Bob chooses two random elements, k_{b1} and k_{b2}, from the additive group $Z_{(p-1)}$. 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. Then, Bob sends the response message $m_{b1} = \{r_{b1}, r_{b2}, s_b, cert(y_b)\}$ to Alice. After constructing the 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 additive group $Z_{(p-1)}$ 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. The property of perfect forward secrecy will be discussed in Section 3.2.

3. CRYPTANALYSIS

In order to investigate the security of the *H-protocol,* four well-known attacks — the known session key, known long-term key, signature forgery, and replay attack are mounted to attack it. The details are shown in the following subsections.

3.1 Known session key attack

The known session key attack examines 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 following calculation, it is shown 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. First, express s_a and s_b in (1) and (2).

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

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

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

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

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

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

Equation (3) is obtained by multiplying (1) by (2). Raising both sides of (3) to the exponentials of the primitive root 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 private key attack)

A very desirable security property of key exchange protocol is the perfect forward secrecy. Communications are usually on insecure channels. The insecure channels have many unacceptable properties, *e.g.,* the adversaries can eavesdrop on, intercept, and modify the messages transmitted 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 has been used to encrypt the plaintext or to decrypt the ciphertext. Then, the adversaries cannot glean 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 the exposure of a participant's long-term private key leads the shared session keys to be revealed. Thus, the adversary is able to decrypt those intercepted cipher texts and thereby read 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 though the participant's long-term secret key is disclosed.

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

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

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

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 keys K_1, K_2, or K_3.

3.3 Signature forgery 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 verifies 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, the triplet (r_{a1}, r_{a2}, s_a) is a signature of Alice on the message r_{a2}, and the triplet (r_{b1}, r_{b2}, s_b) is a signature of Bob on the message r_{b2}, using the scheme of ElGamal signature [3]. The original ElGamal signature is well-known to be existentially forgeable. Assume that an adversary wants to construct a message $m_{a1} = \{r_{a1}, r_{a2}, s_a, cert(y_a)\}$. The following steps show how to forge signatures so as to pass the verification equation.

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).

$$g^{s_a} = y_a^{r_{a1} \oplus r_{a2}} r_{a1}^{r_{a2}} \bmod p \tag{8}$$

$$g^{s_a} = y_a^{r_{a1} \oplus r_{a2}} g^{v r_{a2}} y_a^{u r_{a2}} \bmod p \tag{9}$$

$$r_{a1} \oplus r_{a2} = -u\, r_{a2} = 2\, r_{a2} \bmod (p-1) \tag{10}$$

$$s_a = v\, r_{a2} \bmod (p-1) \tag{11}$$

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

3.4 Replay attack

The adversary sends $m_{a1} = \{r_{a1}, r_{a2}, s_a, cert(y_a)\}$ obtained from a previous intercepted message to Bob. Bob would recognize that Alice is trying to establish a new session with him, since the message m_{a1} is really constructed by Alice. Like the attack of signature forgery, the adversary cannot compute the shared session keys. Note that this replay attack is inherent in the key exchange protocol implemented in only two rounds (one round trip). This type of attack can be avoided if the participants cache all the messages received or use a global timestamp. However, caching all messages would require an unlimited capacity of storage.

4. THE PROPOSED PROTOCOL

Let p be a large prime number such that $(p - 1)$ has a large prime factor q. The element g in the multiplicative group Z_p^* has order q. $e \in_R G$ and represents that the element e is randomly chosen from the group G. $|b|$ denotes the bit length of the string b. $h(.) : \{0, 1\}^* \rightarrow \{0, 1\}^l$ is a collision-free hash function, where l is a security parameter, *i.e.* $l = 160$ or $l = |q|$ for a practical cryptographic setting [4]. Alice has a long-term private key $x_a \in_R Z_q^*$, long-term public key $y_a = g^{x_a} \bmod p$, and a certificate $cert(y_a)$. Similarly, Bob has a long-term private key $x_b \in_R Z_q^*$, long-term public key $y_b = g^{x_b} \bmod p$, and a certificate $cert(y_b)$.

The following steps describe the details of the proposed scheme.

Step 1. Alice randomly selects three elements $k_a, k_{a1}, k_{a2} \in_R Z_q$. The quantities $r_a = g^{k_a} \bmod p$, $r_{a1} = g^{k_{a1}} \bmod p$, and $r_{a2} = g^{k_{a2}} \bmod p$ are computed. Then, the initiator Alice sends the message $m_{a1} = \{r_a, r_{a1}, r_{a2}, cert(y_a)\}$ to the recipient Bob.

Step 2. Upon receiving the message m_{a1}, Bob first verifies the certificate $cert(y_a)$. A valid verification leads Bob to construct a response message; otherwise, Bob stops this instance of key exchange protocol.

To form a response message, Bob chooses three random elements k_b, k_{b1}, $k_{b2} \in_R Z_q$ and computes the quantities $r_b = g^{k_b} \bmod p$, $r_{b1} = g^{k_{b1}} \bmod p$, $r_{b2} = g^{k_{b2}} \bmod p$ and the signing equation

$$s_b = k_b\, h(r_b, r_{b1}, r_{b2}, r_{a1}, r_{a2}, cert(y_b)) + x_b\, r_b \bmod q. \tag{12}$$

Then, Bob sends the response message $m_b = \{r_b, r_{b1}, r_{b2}, s_b, cert(y_b)\}$ to Alice.

Step 3. Alice verifies the certificate $cert(y_b)$ when receiving the message m_b. Then, Alice verifies the message m_b by checking $g^{s_b} = r_b^{h(r_b, r_{b1}, r_{b2}, r_{a1}, r_{a2}, cert(y_b))}\, y_b^{r_b} \bmod p$. Alice stops the execution if the check is invalid; otherwise, Alice uses the following equation to construct the response message $m_a = \{s_a\}$ and sends it to Bob.

$$s_a = k_a\, h(r_a, r_{a1}, r_{a2}, r_{b1}, r_{b2}, cert(y_a)) + x_a\, r_a \bmod q \tag{13}$$

While constructing a response message, Alice also computes a set of Diffie-Hellman keys, *i.e.*, the 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 $K_4 = r_{b2}^{k_{a2}} \bmod p$.

Step 4. Upon receiving the message m_a, Bob verifies m_a by checking $g^{s_a} = r_a^{h(r_a, r_{a1}, r_{a2}, r_{b1}, r_{b2}, cert(y_a))}\, y_a^{r_a} \bmod p$. Bob stops the key exchange protocol if the check is invalid.

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 $K_4 = r_{a2}^{k_{b2}} \bmod p$.

Therefore, Bob and Alice have agreed on a set of four session keys after executing the protocol cooperatively. If both Alice and Bob have chosen n random numbers from the group Z_q during execution of the protocol, then they will agree on a set of $(n-1)^2$ session keys.

4.1 Security of the signature scheme

Let m be the message and x be the signer's secret key. Then, (s, r) is an ElGamal signature on the message m, where $s = k^{-1} (m - x\, r) \bmod (p - 1)$, $r = g^k \bmod p$ and $k \in_R Z_{p-1}$. However, the original ElGamal signature scheme [3] is well-known to be existentially forgeable. The signature forgery attack described in Section 3.3 is an example.

The signature scheme in [5, 6] replaces the earlier signing equation $s = k^{-1} (m - x\, r) \bmod (p - 1)$ with $s = k^{-1} (h(m, r) - x\, r) \bmod (p - 1)$, where $r = g^k \bmod p$ and $k \in_R Z_{p-1}$. This modified version of ElGamal signature is provably secure against the adaptive chosen message attack proposed in [7] under the random oracle model [8]. In this model of attack, it is assumed that an adversary has access to a signing oracle, which generates the signatures. The adversary is allowed to collect the signatures by asking the signing oracle as he wishes, except for the one that the adversary is forging.

In the paper, the proposed key exchange protocol uses $s = (h(m, r)\, k + x\, r) \bmod q$ as the signing equation, which has the advantage of saving an inverse computation. The work in [9] proved that this variant of ElGamal signature scheme is also secure against the adaptive chosen message attack. Thus the following Theorem is obtained without proof.

Theorem 1. The signature scheme used in the proposed key exchange protocol (Section 4) is secure against the adaptive chosen message attack.

Proof. Please refer to [9].

5. SECURITY ANALYSIS

The sub-sections 5.1-5.4 demonstrate that the proposed protocol is secure under the attacks described in Section 3. Sub-section 5.5 provides a formal proof for the protocol's security.

5.1 Security under known session key attack

The discrete logarithms of random numbers r_{a1}, r_{a2}, r_{b1}, and r_{b2} are not used in computing the quantities s_a and s_b. Thus an adversary cannot solve the long-term Diffie-Hellman key y_{ab} in the same way as described in Section 3.1.

5.2 Security under known long-term private key attack

From (12) and (13), compromise of private keys x_a and x_b can reveal k_a and k_b. However, these values are irrelevant to the computations of the session keys. Therefore, the proposed protocol possesses the perfect forward secrecy.

5.3 Security under signature forgery attack

Both Alice and Bob check on the messages m_b and m_a, which are signatures generated by Bob and Alice, respectively. By Theorem 1, the signatures are secure against the adaptive chosen message attack. Thus, the proposed protocol is secure under the forgery attack described in 3.3.

5.4 Security under replay attack

For each execution of the proposed protocol, Bob generates two fresh random numbers r_{b1} and r_{b2}. The check of fresh random numbers is performed by the computation of hash value $h(r_{a1}, r_{a2}, r_{b1}, r_{b2}, cert(y_a))$. Similarly, Alice does the same check. Thus the proposed protocol is secure under the replay attack.

5.5 Security proof of the proposed protocol

This sub-section investigates the security of the proposed protocol by adopting the security measure and those attack models used in [10, 11]. Assume that an adversary with total control over the communication channels can mount parallel attacks, and is told the previous session keys. A key exchange protocol is secure if the following requirements are satisfied.

1. If both participants execute the protocol honestly, then the session key is $K_{se} = K_{AB} = K_{BA}$, where K_{AB} is the session key computed by Alice and K_{BA} is the session key computed by Bob.
2. No one can calculate the session key K_{se} except the participants Alice and Bob.
3. The session key is indistinguishable from a truly random number.

Lemma 2. The proposed protocol satisfies the first security requirement.

Proof. Both participants have agreed on the random numbers r_{a1}, r_{a2}, r_{b1}, and r_{b2}, because these random numbers are included in the message signed by Alice and Bob. By Theorem 1,

the signatures are secure against the adaptive chosen message attack. Thus, with overwhelming probability, the random numbers r_{b1} and r_{b2} received by Alice are originally sent from Bob, and r_{a1} and r_{a2} received by Bob are random numbers sent by Alice. Therefore, $K_1 = r_{a1}^{k_{b1}} = r_{b1}^{k_{a1}} = g^{k_{a1}k_{b1}} \bmod p$, $K_2 = r_{a2}^{k_{b1}} = r_{b1}^{k_{a2}} = g^{k_{a2}k_{b1}} \bmod p$, $K_3 = r_{b2}^{k_{a1}} = r_{a1}^{k_{b2}} = g^{k_{a1}k_{b2}} \bmod p$, and $K_4 = r_{b2}^{k_{a2}} = r_{a2}^{k_{b2}} = g^{k_{a2}k_{b2}} \bmod p$, by the commutative law of the multiplicative group Z_p^*.

Since the Computational Diffie-Hellman assumption (CDH) and Decisional Diffie-Hellman assumption (DDH) are required in proving Lemma 3 and Theorem 4, a brief description follows. For further details, refer to the detailed descriptions of cryptographic primitives in [12]. Suppose that G is a group with a large prime order q and $g \in G$ generating the group G. The CDH assumption implies that computing g^{xy} from g^x and g^y is difficult [1].

Let g_1, g_2, r_1, and r_2 be elements of the group G. The Diffie-Hellman Pair function $DHP(g_1, g_2, r_1, r_2)$ is defined to be *1* if an $x \in Z_q$ exists such that $r_1 = g_1^x$ and $r_2 = g_2^x$; otherwise, *0* is assigned to the function *DHP()*. A good algorithm for *DHP()* is a polynomial bounded algorithm that correctly decides whether $DHP(g_1, g_2, r_1, r_2)$ is *1* or *0* for all elements g_1, g_2, r_1, and r_2 randomly selected from G, with negligible error probability. The DDH assumption is that there is no good algorithm for *DHP()*. By letting $g_1 = g$, $g_2 = g^x$, $r_1 = g^y$ and $r_2 = g^{xy}$, the quadruple form of $DHP(g_1, g_2, r_1, r_2)$ can be expressed by the triple form of $DHP(g^x, g^y, g^{xy})$. In the triple form, the first argument g is implicitly implied. The latter form is used in this paper.

Lemma 3. The proposed protocol satisfies the second security requirement.

Proof. Assume that an adversary is trying to compute the session keys. The adversary cannot obtain random numbers k_{a1}, k_{a2} , k_{b1}, and k_{b2}, since Alice and Bob generate these random numbers secretly and do not disclose them. Thus, the adversary does not know the discrete logarithms of r_{a1}, r_{a2} , r_{b1}, and r_{b2}. The adversary is challenged to compute $K_1 = g^{k_{a1}k_{b1}} \bmod p$, $K_2 = g^{k_{a2}k_{b1}} \bmod p$, $K_3 = g^{k_{a1}k_{b2}} \bmod p$, and $K_4 = g^{k_{a2}k_{b2}} \bmod p$, with knowledge of r_{a1}, r_{a2} , r_{b1}, and r_{b2}. The adversary will fail to compute the session keys, since asked to break the CDH assumption.

Theorem 4. The proposed protocol satisfies the third security requirement.

Proof. Assume that an adversary *S* can distinguish one of the session keys, *e.g.* K_4, from a truly random number with non-negligible probability. Then the adversary *S* is also a good algorithm for *DHP()*. The following processes show that the adversary *S* is used to calculate $DHP(r_{a2}, r_{b2}, R)$.

Process 1. Alice and Bob cooperatively perform the steps 1, 2, 3, and 4 in Section 4.

Process 2. Select $r \in_R G$ and $c \in_R \{0, 1\}$. Compute $R = (K_4)^c (r)^{1-c}$.

The adversary *S* is able to answer whether *c* is *0* or *1,* because it is assumed that *S* can distinguish the session key K_4 from a truly random number. This conclusion contradicts the assumption of DDH. This completes the proof of Theorem 4.

6. CONCLUSIONS

It is shown that *H-protocol* is vulnerable to the known session key attack, known long-term private key attack, signature forgery attack, and replay attack. A secure protocol is proposed and shown that it is resistant to those attacks presented in the paper. To resist other possible attacks, Section 5 provides a formal proof to guarantee the proposed protocol's security. Therefore, the proposed protocol is not only to mend the flaws in *H-protocol*, but also to provide a secure and efficient method to exchange multiple session keys between participants.

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.
4. A. Lenstra and E. Verheul, "Selecting cryptographic key sizes," The Third International Workshop on Practice and Theory in Public Key Cryptography (PKC2000), LNCS 1751, pp. 446-465, 2000.

5. D. Pointcheval and J. Stern, “Security proofs for signature schemes”, *Advances in Cryptology- EUROCRYPT’96*, LNCS 1070, pp. 387-398, 1996.
6. D. Pointcheval and J. Stern, “Security arguments for digital signatures and blind signatures,” *Journal of Cryptology*, Vol. 13, N0. 3, pp. 361-396, 2000.
7. S. Goldwasser, S. Micali, and R. Rivest, “A digital signature scheme secure against adaptive chosen-message attacks,” SIAM journal of computing, Vol. 17, No. 2, pp. 281-308,1988.
8. M. Bellare and P. Rogaway, “Random oracles are practical: a paradigm for designing efficient protocols”, Proc. of the 1st ACM Conference on Computer and Communications Security CCS’93, ACM press, pp. 62-73, 1993.
9. F. Y. Yang and J. K. Jan, ”A provable access control using smart cards”, IEEE Transactions on Consumer Electronics, 49, (4), pp. 1223-1226, 2003.
10. M. Bellare and P. Rogaway, “Entity authentication and key distribution,” *Advances in Cryptology- CRYPTO’93*, LNCS 773, pp. 232-249, 1993.
11. R. Canetti and H. Krawczyk, “Analysis of key-exchange protocols and their use for building secure channels,” *Advances in Cryptology- EUROCRYPT’01*, LNCS 2045, pp. 453-474, 2001.
12. V. Shoup, “On formal models for secure key exchange,” *IBM Research Report RZ 3120 Version 4*, 1999.