

# An Authenticated Diffie-Hellman Key Agreement Protocol

Shouichi Hirose<sup>1</sup>      Susumu Yoshida

Department of Communications and Computer Engineering,  
Graduate School of Informatics, Kyoto University  
Kyoto, 606-8501 Japan

E-mail: [hirose@kuee.kyoto-u.ac.jp](mailto:hirose@kuee.kyoto-u.ac.jp),  
Tel: +81 75 753 5316, Fax: +81 75 753 4982

**Abstract** A two-party authenticated Diffie-Hellman key agreement protocol is proposed. The protocol is practical and provably secure against passive eavesdropping, interference and unknown-key share in the random oracle model. The results hold even if the above attacks are known key attacks. This protocol also provides forward secrecy.

---

<sup>1</sup>Supported by The Grant-in-Aid for Encouragement of Young Scientists of the Ministry of Education, Science, Sports and Culture of Japan and by The Telecommunications Advancement Foundation

# 1 Description of the cryptographic technique

Let  $p$  and  $q$  be large primes and  $q$  divide  $p - 1$ . Let  $g$  be an element of  $\text{GF}(p)$  whose order is  $qD$ . Let  $h : \{0, 1\}^* \rightarrow \mathbf{Z}_q$  be a collision free hash function, where  $\mathbf{Z}_q = \{0, 1, \dots, q - 1\}$ .  $p, q, g, h$  are public and shared among all of the users.

Let  $s_i \in \mathbf{Z}_q$  be a secret key of the user  $i$  and  $v_i = g^{-s_i} \bmod p$  be a public key of the user  $i$ .  $I_i$  represents the ID of the user  $i$ .

The proposed protocol is shown below. This is called KAP in this manuscript. It is assumed that each of the participants knows the public key of the other participant in advance.

## KAP

### 0. (Precomputation)

The initiator  $A$  randomly selects  $k_A, r_A \in \mathbf{Z}_q$  and computes  $u_A = g^{-k_A} \bmod p$  and  $x_A = g^{r_A} \bmod p$ .

The responder  $B$  randomly selects  $k_B, r_B \in \mathbf{Z}_q$  and computes  $u_B = g^{-k_B} \bmod p$  and  $x_B = g^{r_B} \bmod p$ .

1.  $A$  sends  $u_A$  to  $B$ .

2.  $B$  computes  $e_B = h(x_B, u_B, u_A, I_B, I_A)$  and  $w_B = r_B + e_B k_B + e_B^2 s_B \bmod q$ .

3.  $B$  sends  $u_B, e_B, w_B$  to  $A$ .

4.  $A$  computes  $z_B = g^{w_B} u_B^{e_B} v_B^{e_B^2} \bmod p$  and checks if  $e_B = h(z_B, u_B, u_A, I_B, I_A)$ . If it does not hold, then  $A$  terminates the execution. Otherwise,  $A$  computes  $e_A = h(x_A, u_A, u_B, I_A, I_B)$  and  $w_A = r_A + e_A k_A + e_A^2 s_A \bmod q$ .

5.  $A$  sends  $e_A, w_A$  to  $B$ .

6.  $A$  computes  $K_A = u_B^{-k_A} \bmod p$ .

$B$  computes  $z_A = g^{w_A} u_A^{e_A} v_A^{e_A^2} \bmod p$  and checks if  $e_A = h(z_A, u_A, u_B, I_A, I_B)$ . If it does not hold, then  $B$  terminates the execution. Otherwise,  $B$  computes  $K_B = u_A^{-k_B} \bmod p$ .

It is clear that  $A$  and  $B$  can compute  $K_A$  and  $K_B$  such that  $K_A = K_B$  respectively if they behave honestly.

KAP is also shown in Fig. 1.  $\in_R$  represents a random selection of an element of the set on its right side.

KAP is also able to be constructed over elliptic curves.

Step	Initiator: $A$		Responder: $B$
0	$k_A, r_A \in_{\mathbb{R}} \mathbf{Z}_q$ $u_A = g^{-k_A} \bmod p$ $x_A = g^{r_A} \bmod p$		$k_B, r_B \in_{\mathbb{R}} \mathbf{Z}_q$ $u_B = g^{-k_B} \bmod p$ $x_B = g^{r_B} \bmod p$
1	$\Rightarrow (u_A) \Rightarrow$		
2			$e_B = h(x_B, u_B, u_A, I_B, I_A)$ $w_B = r_B + e_B k_B + e_B^2 s_B \bmod q$
3	$\Leftarrow (u_B, e_B, w_B) \Leftarrow$		
4	$z_B = g^{w_B u_B^{e_B} v_B^{e_B^2}} \bmod p$ $e_B = h(z_B, u_B, u_A, I_B, I_A) ?$ $e_A = h(x_A, u_A, u_B, I_A, I_B)$ $w_A = r_A + e_A k_A + e_A^2 s_A \bmod q$		
5	$\Rightarrow (e_A, w_A) \Rightarrow$		
6	$K_A = u_B^{-k_A} \bmod p$		$z_A = g^{w_A u_A^{e_A} v_A^{e_A^2}} \bmod p$ $e_A = h(z_A, u_A, u_B, I_A, I_B) ?$ $K_B = u_A^{-k_B} \bmod p$

Figure 1: The proposed protocol: KAP

## 2 Claimed attributes and advantages of the technique

### 2.1 The design principle and a redundant signature scheme

Fig. 2 presents the Diffie-Hellman key agreement protocol. For the Diffie-Hellman key agreement protocol, if, for example,  $A$  gets the information which guarantees the following three points, then  $A$  makes sure that  $B$  is really able to compute  $K_B$ , which is equal to  $K_A$ .

- $B$  received  $u_A$ .
- $B$  sent  $u_B$  to  $A$  in response to the receipt of  $u_A$ .
- $B$  knows  $-k_B$ , the discrete logarithm of  $u_B$ .

In KAP,  $(u_B, e_B, w_B)$  can be regarded as a signature for  $(u_A, I_B, I_A)$ . This signature contains the information which guarantees the above three points.

The new signature scheme is called a redundant signature scheme in this manuscript. It is an extension of Schnorr's signature scheme [5].

**Redundant signature scheme** Let  $s_i \in \mathbf{Z}_q$  be a secret key of the user  $i$  and  $v_i = g^{-s_i} \bmod p$  be a public key of the user  $i$ . Let  $h : \{0, 1\}^* \rightarrow \mathbf{Z}_q$  be a public collision free hash function.

Let  $k_B \in \mathbf{Z}_q$  and  $u_B = g^{-k_B} \bmod p$ . Suppose that  $B$  knows  $k_B$ .  $B$ 's signature  $(u_B, e_B, w_B)$  for  $Msg$  is computed with the following procedure.

Step	A		B
1	$k_A \in_{\mathbb{R}} \mathbf{Z}_q$ $u_A = g^{-k_A} \bmod p$		
2		$\implies (u_A) \implies$	
3			$k_B \in_{\mathbb{R}} \mathbf{Z}_q$ $u_B = g^{-k_B} \bmod p$
4		$\longleftarrow (u_B) \longleftarrow$	
5	$K_A = u_B^{-k_A} \bmod p$		$K_B = u_A^{-k_B} \bmod p$

Figure 2: Diffie-Hellman key agreement protocol

1.  $B$  selects  $r_B \in \mathbf{Z}_q$  at random and computes  $x_B = g^{r_B} \bmod p$ .
2.  $B$  computes  $e_B = h(x_B, u_B, Msg)$  and  $w_B = r_B + e_B k_B + e_B^2 s_B \bmod q$ .

The procedure for checking the validity of the signature  $(u_B, e_B, w_B)$  for  $Msg$  is as follows:

1. Let  $z_B = g^{w_B} u_B^{e_B} v_B^{e_B^2} \bmod p$ .
2. The signature is valid if and only if  $e_B = h(z_B, u_B, Msg)$ .

Notice that the redundant signature scheme provides key confirmation while it does not contain the computation of  $K_A$  or  $K_B$ . This novel key confirmation is called weak key confirmation in this manuscript.

## 2.2 Extensions

### 2.2.1 A key sharing scheme with no communication

With the redundant signature scheme, a key sharing scheme can be constructed, which requires no communication between the users.

First, each user  $i$  follows the next procedure.

1. He selects  $k_i \in \mathbf{Z}_q$  at random and computes  $u_i = g^{-k_i} \bmod p$ . He also determines  $ExpDate_i$ , the expiration date of  $u_i$ .
2. He selects  $r_i \in \mathbf{Z}_q$  at random and computes  $x_i = g^{r_i} \bmod p$ . Then, he computes  $e_i = h(x_i, u_i, I_i, ExpDate_i)$  and  $w_i = r_i + e_i k_i + e_i^2 s_i \bmod q$ .
3. He writes  $(I_i, ExpDate_i, u_i, e_i, w_i)$  in publicly accessible board.

When  $A$  would like to share a session-key with  $B$ , first  $A$  checks the validity of  $(I_B, ExpDate_B, u_B, e_B, w_B)$ . If it is valid, then  $A$  computes  $K = u_B^{-k_A} \bmod p$ .

### 2.2.2 A multiple key agreement protocol

A multiple key agreement protocol can be constructed which enables the participants to share two or more keys in one execution of the protocol. This protocol is called MKAP in this manuscript.

#### MKAP

0. (Precomputation)

$A$  randomly selects  $k_{A1}, k_{A2}, \dots, k_{An} \in \mathbf{Z}_q$  and computes  $u_{Ai} = g^{-k_{Ai}} \bmod p$  for  $i = 1, 2, \dots, n$ .  $A$  also selects  $r_A \in \mathbf{Z}_q$  at random and computes  $x_A = g^{r_A} \bmod p$ .

$B$  randomly selects  $k_{B1}, k_{B2}, \dots, k_{Bn} \in \mathbf{Z}_q$  and computes  $u_{Bj} = g^{-k_{Bj}} \bmod p$  for  $j = 1, 2, \dots, n$ .  $B$  also selects  $r_B \in \mathbf{Z}_q$  at random and computes  $x_B = g^{r_B} \bmod p$ .

1.  $A$  sends  $u_{A1}, u_{A2}, \dots, u_{An}$  to  $B$ .

2.  $B$  computes  $e_B = h(x_B, u_{B1}, \dots, u_{Bn}, u_{A1}, \dots, u_{An}, I_B, I_A)$  and  $w_B = r_B + \sum_{j=1}^n e_B^j k_{Bj} + e_B^{n+1} s_B \bmod q$ .

3.  $B$  sends  $u_{B1}, u_{B2}, \dots, u_{Bn}, e_B, w_B$  to  $A$ .

4.  $A$  computes  $z_B = g^{w_B} \left( \prod_{j=1}^n u_{Bj}^{e_B^j} \right) v_B^{e_B^{n+1}} \bmod p$  and checks if  $e_B = h(z_B, u_{B1}, \dots, u_{Bn}, u_{A1}, \dots, u_{An}, I_B, I_A)$ . If it does not hold, then  $A$  terminates the execution. Otherwise,  $A$  computes  $e_A = h(x_A, u_{A1}, \dots, u_{An}, u_{B1}, \dots, u_{Bn}, I_A, I_B)$  and  $w_A = r_A + \sum_{i=1}^n e_A^i k_{Ai} + e_A^{n+1} s_A \bmod q$ .

5.  $A$  sends  $e_A, w_A$  to  $B$ .

6.  $A$  computes  $K_{Ai} = u_{Bi}^{-k_{Ai}} \bmod p$  for  $i = 1, \dots, n$ .

$B$  computes  $z_A = g^{w_A} \left( \prod_{i=1}^n u_{Ai}^{e_A^i} \right) v_A^{e_A^{n+1}} \bmod p$  and checks if  $e_A = h(z_A, u_{A1}, \dots, u_{An}, u_{B1}, \dots, u_{Bn}, I_A, I_B)$ . If it does not hold, then  $B$  terminates the execution. Otherwise,  $B$  computes  $K_{Bj} = u_{Aj}^{-k_{Bj}} \bmod p$  for  $j = 1, \dots, n$ .

Fig. 3 shows an example of MKAP. In this example, each of the participants generates three random numbers and shares two keys with the other participant.

Unfortunately, the authors have not yet found any good applications of MKAP.

### 2.3 Resistance against a denial-of-service attack

The resistance against a denial-of-service attack (DoS attack) of KAP is compared with that of the other protocols with key confirmation. These protocols are

STS: Station-to-station protocol [2] with Schnorr's signature scheme [5],

JV: Protocol IIA of Just and Vaudenay [3],

BJM: Protocol 2 of Blake-Wilson, Johnson and Menezes [1],

LMQSV: Protocol 3 of Law, Menezes, Qu, Solinas and Vanstone [4].

Step	A		B
0	$k_{A1}, k_{A2}, r_A \in_{\mathbb{R}} \mathbf{Z}_q$ $u_{Ai} = g^{-k_{Ai}} \bmod p \ (i = 1, 2)$ $x_A = g^{r_A} \bmod p$		$k_{B1}, k_{B2}, r_B \in_{\mathbb{R}} \mathbf{Z}_q$ $u_{Bj} = g^{-k_{Bj}} \bmod p \ (j = 1, 2)$ $x_B = g^{r_B} \bmod p$
1	$\implies (u_{A1}, u_{A2}) \implies$		
2			$e_B = h(x_B, u_{B1}, u_{B2}, u_{A1}, u_{A2}, I_B, I_A)$ $w_B = r_B + e_B k_{B1} + e_B^2 k_{B2} + e_B^3 s_B \bmod q$
3	$\longleftarrow (u_{B1}, u_{B2}, e_B, w_B) \longleftarrow$		
4	$z_B = g^{w_B} u_{B1}^{e_B} u_{B2}^{e_B^2} v_B^{e_B^3} \bmod p$ $e_B = h(z_B, u_{B1}, u_{B2}, u_{A1}, u_{A2}, I_B, I_A) ?$ $e_A = h(x_A, u_{A1}, u_{A2}, u_{B1}, u_{B2}, I_A, I_B)$ $w_A = r_A + e_A k_{A1} + e_A^2 k_{A2} + e_A^3 s_A \bmod q$		
5	$\implies (e_A, w_A) \implies$		
6	$K_{Ai} = u_{Bi}^{-k_{Ai}} \bmod p \ (i = 1, 2)$		$z_A = g^{w_A} u_{A1}^{e_A} u_{A2}^{e_A^2} v_A^{e_A^3} \bmod p$ $e_A = h(z_A, u_{A1}, u_{A2}, u_{B1}, u_{B2}, I_A, I_B) ?$ $K_{Bj} = u_{Aj}^{-k_{Bj}} \bmod p \ (j = 1, 2)$

Figure 3: MKAP: An example

These protocols as well as KAP are three-pass protocols. KAP is provably secure in the random oracle model on the assumption that the Diffie-Hellman problem is intractable, which is discussed in the next section. BJM is provably secure in the random oracle model on the assumptions that the Diffie-Hellman problem is intractable and that there exists a secure MAC (Message Authentication Code). On the other hand, STS, JV and LMQSV are not provably secure.

The DoS attack considered here is a kind of resource-exhaustion attack. By this attack, a malicious initiator launches as many (bogus) requests as possible one after another without establishing connections with the responder in order to prevent him executing KAP with honest initiators. The main resources are a memory and a CPU, and two kinds of resource-exhaustion attacks are called the memory-exhaustion attack and the CPU-exhaustion attack. In this manuscript, only the CPU-exhaustion attack is considered because the cost of a CPU is more expensive than that of a memory.

The number of modular exponentiations required of a responder under the CPU-exhaustion attack is evaluated. A modular exponentiation is much more time-consuming than any other operations in key agreement protocols based on discrete logarithms. First, the number of all modular exponentiations is considered. Then, the number of on-line modular exponentiations is considered.

Three types of requests to a responder are considered; a valid request, an invalid request with a last message and an invalid request without a last message. The last two types of requests are from malicious initiators who try to make the CPU-exhaustion attack. Malicious initiators tend to make invalid requests without last messages, because he is able to forget his past requests and concentrate on making new invalid requests. Thus, in this manuscript, an invalid request with a

last message is called a minor bad request and an invalid request without a last message is called a major bad request. A valid request is called a good request.

Table 1 shows the numbers of all modular exponentiations for the three types of requests. For BJM, the computational load of a MAC involved in it does not considered here. KAP is the least efficient for a good request. However, weak key confirmation of KAP reduces the number of exponentiations of KAP for a bad request. For a minor bad request, KAP is more efficient than STS and JV, but still less efficient than BJM and LMQSV. For a major bad request, a responder need not check the authenticity of an initiator, and KAP is the most efficient protocol.

Table 2 shows the lower bounds of the ratios of bad requests to all requests for the average number of modular exponentiations of KAP to be smaller than those of the other protocols. These are evaluated based on Table 1. Good  $\vee$  MinorBad represents each request is a good request or a minor bad request. Good  $\vee$  MajorBad represents each request is a good request or a major bad request. In the case of Good  $\vee$  MajorBad, KAP can be more efficient than the other protocols. For example, if each request is a good or major bad request and 7.4% or more of all requests are the latter one, then the average number of exponentiations of KAP is smaller than that of STS. In the case of Good  $\vee$  MinorBad, the average number of exponentiations of KAP is always larger than those of BJM and LMQSV. Notice that the computational load of MAC in BJM is not considered here and that LMQSV provides no provable security. The computational load of BJM gets much larger if its MAC is implemented based on discrete logarithms.

Table 3 shows the numbers of on-line modular exponentiations for the three types of requests. For a bad request, KAP is the most efficient protocol in terms of on-line computation. Especially, for a major bad request, KAP requires no on-line exponentiations to a responder.

The average number of on-line modular exponentiations of KAP can be smaller than those of the other protocols when some of the requests are bad. Table 4 shows the lower bounds of the ratios of bad requests to all requests for the average number of exponentiations of KAP to be smaller than those of the other protocols. These are evaluated based on Table 3.

KAP is able to be transformed to a protocol to which a minor bad request is less applicable. The load caused by this transformation is one check of equality to a responder. The new protocol is called DoS-resistant KAP.

**DoS-resistant KAP** Steps 0 through 4 are same as those of KAP. Step 5 and 6 of this protocol is as follows.

5.  $A$  sends  $z_B, e_A, w_A$  to  $B$ .

6.  $A$  computes  $K_A = u_B^{-k_A} \bmod p$ .

$B$  checks if  $z_B = x_B$ . If it does not hold, then  $B$  terminates the execution. Otherwise,  $B$  computes  $z_A = g^{w_A} u_A^{e_A} v_A^{e_A^2} \bmod p$  and checks if  $e_A = h(z_A, u_A, u_B, I_A, I_B)$ . If it does not hold, then  $B$  terminates the execution. Otherwise,  $B$  computes  $K_B = u_A^{-k_B} \bmod p$ .

In Step 6 of DoS-resistant KAP, a responder computes  $z_A = g^{w_A} u_A^{e_A} v_A^{e_A^2} \bmod p$  only if  $z_B = x_B$ . An initiator has to compute  $z_B = g^{w_B} u_B^{e_B} v_B^{e_B^2} \bmod p$  in order to send valid  $z_B$  to

Table 1: The number of all modular exponentiations required of a responder for a good request, a minor bad request and a major bad request. STS is the station-to-station protocol [2] with Schnorr’s signature scheme [5], JV is Protocol IIA of Just and Vaudenay [3], BJM is Protocol 2 of Blake-Wilson, Johnson and Menezes [1], and LMQSV is Protocol 3 of Law, Menezes, Qu, Solinas and Vanstone [4]. For BJM, the computational load of MAC involved in it is not considered here.

request \ protocol	KAP	STS	JV	BJM	LMQSV
good request	4.25	4.17	4	3	2.17
minor bad request	3.25	4.17	4	3	2.17
major bad request	2	3	3	3	2.17

Table 2: The lower bounds of the ratios(%) of bad requests to all requests for the average number of modular exponentiations of KAP to be smaller than those of protocols compared. These are evaluated based on Table 1. Good  $\vee$  MinorBad represents each request is a good request or a minor bad request. Good  $\vee$  MajorBad represents each request is a good request or a major bad request. “xxx” means that the average number of modular exponentiations of KAP is always larger.

request \ protocol	STS	JV	BJM	LMQSV
Good $\vee$ MinorBad	8.0	25.0	xxx	xxx
Good $\vee$ MajorBad	7.4	20.0	55.6	92.4

a responder. A minor bad request with invalid  $z_B$  is almost equivalent to a major bad request, because an responder need not compute any modular exponentiations in Step 6 if  $z_B \neq x_B$ .

Table 3: The number of on-line modular exponentiations required of a responder for a good request, a minor bad request and a major bad request. For BJM, the computational load of MAC involved in it is not considered here.

request \ protocol	KAP	STS	JV	BJM	LMQSV
good request	2.25	2.17	3	2	1.17
minor bad request	1.25	2.17	3	2	1.17
major bad request	0	1	2	2	1.17

Table 4: The lower bounds of the ratios(%) of bad requests to all requests for the average number of on-line modular exponentiations of KAP to be smaller than those of protocols compared. “xxx” means that the average number of on-line modular exponentiations of KAP is always larger. These are evaluated based on Table 3.

request \ protocol	STS	JV	BJM	LMQSV
Good $\vee$ MinorBad	8.0	0	25.0	xxx
Good $\vee$ MajorBad	7.4	0	11.1	48.0

### 3 Security assessment and considerations

It is shown that the proposed protocol is provably secure against a passive attack and active attacks in the random oracle model.

The length of  $q$  is used for the security parameter and is denoted by  $l$ . It is assumed that the length of  $p$  is  $O(l)$ .

An execution of KAP by an initiator  $A$  and a responder  $B$  is denoted by  $(A, B)$ .

Let  $t_{sig}$  be the number of steps for generating a signature and  $t_{ver}$  be that of steps for verifying a signature with the redundant signature scheme.

#### 3.1 Security against a passive attack

In this subsection, it is proved that KAP is secure against passive eavesdropping. It is shown that the security against passive eavesdropping is based on the intractability of the Diffie-Hellman problem.

A passive eavesdropper only eavesdrops the messages exchanged by the target participants and tries to obtain the session-key shared between them. He does not alter, delete or insert any messages.

First, the Diffie-Hellman problem and a problem corresponding to passive eavesdropping are defined.

##### Definition 1 (DHP: Diffie-Hellman Problem)

**input:**  $p, q, g, a, b$ , where  $a, b$  are randomly selected from  $\{g^0, \dots, g^{q-1}\}$ .

**output:**  $a^{\log_g b \bmod p} \bmod p$ . □

##### Definition 2 (PEP: Passive Eavesdropping Problem)

**input:**  $p, q, g, I_A, I_B, (s_A, v_A), (s_B, v_B), (u_A; u_B, e_B, w_B; e_A, w_A)$ , where

- $(s_A, v_A)$  and  $(s_B, v_B)$  are randomly selected keys of  $A$  and  $B$ , respectively,
- $(u_A; u_B, e_B, w_B; e_A, w_A)$  is messages exchanged in  $(A, B)$ .

**output:**  $u_A^{\log_g u_B \bmod p} \bmod p$ . □

Notice that the secret keys of the target participants are given as input of PEP.

**Theorem 1** If there exists some probabilistic Turing machine  $M'$  that solves PEP with at most  $t$  steps and with probability at least  $\epsilon$ , then there exists some probabilistic Turing machine  $M$  that solves DHP with at most  $t + 2t_{ver}$  steps and with probability at least  $\epsilon$ .

(Proof) For a given instance of DHP,  $(p, q, g, a, b)$ ,  $M$  operates in the following way:

1. Set  $u_A = a$  and  $u_B = b$ .
2.  $M$  randomly selects  $(s_A, v_A)$  and  $(s_B, v_B)$ .
3.  $M$  randomly selects  $e_B, w_B \in \mathbf{Z}_q$ , computes  $x_B = g^{w_B u_B^{e_B} v_B^{e_B^2}} \bmod p$ , and determines  $e_B = h(x_B, u_B, u_A, I_B, I_A)$ .
4.  $M$  randomly selects  $e_A, w_A \in \mathbf{Z}_q$ , computes  $x_A = g^{w_A u_A^{e_A} v_A^{e_A^2}} \bmod p$ , and determines  $e_A = h(x_A, u_A, u_B, I_A, I_B)$ .
5.  $M$  simulates  $M'$  with input  $p, q, g, I_A, I_B, (s_A, v_A), (s_B, v_B), (u_A; u_B, e_B, w_B; e_A, w_A)$ .
6.  $M$  outputs the output of  $M'$ .

$M$  solves DHP with probability at least  $\epsilon$ , because  $M'$  solves PEP with probability at least  $\epsilon$ . The number of steps of  $M$  is at most  $t + 2t_{ver}$ . □

It follows from Theorem 1 and the definition of PEP that KAP provides forward secrecy, that is, the session-keys shared before cannot be compromised even if the secret keys of the users are leaked.

## 3.2 Security against active attacks

In this subsection, KAP is shown to be provably secure against two active attacks; interference and unknown-key share.

### 3.2.1 Interference

Interference is defined as an active attack that alters the messages exchanged by participants in order to try to obtain some information on secret keys of the participants or to try to make them fail to share a common session-key. It is a general attack and it includes impersonation and active eavesdropping.

First, the discrete logarithm problem and a signature-forgery problem for the redundant signature scheme are defined.

**Definition 3 (DLP: Discrete Logarithm Problem)**

**input:**  $(p, q, g, u)$ , where  $u$  is randomly selected from  $\{g^0, \dots, g^{q-1}\}$ .

**output:**  $\log_g u \bmod p$ . □

**Definition 4 (SFP: Signature-Forgery Problem)**

**input:**  $p, q, g, v$ , where  $v$  is randomly selected from  $\{g^0, \dots, g^{q-1}\}$ .

**output:**  $Msg, u, e, w$ , where  $Msg$  is a message and  $(u, e, w)$  is a valid signature for  $Msg$  with respect to the public key  $v$ . □

SFP assumes adaptive chosen-message existential forgery. It is further assumed that an attacker obtains a signature  $(u', e', w')$  and the discrete logarithm of  $u'$  for a message  $Msg'$  that he selects.

The following lemma shows that SFP is intractable if DLP is intractable.

**Lemma 1** If there exists some probabilistic Turing machine  $M'$  that solves SFP with at most  $t$  steps, with at most  $\alpha$  legitimate message-signature pairs, with at most  $\beta$  queries to the random oracle and with probability at least  $\epsilon$ , then there exists some probabilistic Turing machine  $M$  that solves DLP with at most  $\alpha t_{sig} + \lceil 6\beta/\epsilon \rceil (t + t_{ver}) + O(t^3)$  steps and with probability at least  $\epsilon/(4\beta^2)$ .

(Proof) Without loss of generality, it is assumed that  $M'$  does not ask the same query twice or more. It is also assumed that, for some  $x$ ,  $M'$  asked  $(x, u, Msg)$  to the random oracle and obtains  $e$  as the answer if  $M'$  outputs  $Msg, u, e, w$  and solves SFP. Let  $Q_i$  be the  $i$ -th query of  $M'$  to the random oracle and  $\rho_i$  be the answer to  $Q_i$  from the random oracle.

For every  $c$  such that  $1 \leq c \leq \beta$ , let  $M'_c$  be a probabilistic Turing machine which behaves following the steps below.

1.  $M'_c$  simulates  $M'$  and obtains the output of  $M'$ ,  $Msg, u, e, w$ .
2. If the output is a valid signature,  $Q_c = (x, u, Msg)$ ,  $\rho_c = e$  and  $x = g^w u^e v^{e^2} \bmod p$ , then  $M'_c$  outputs  $(Msg, u, e, w)$ , otherwise outputs "Fail."

Since the success probability of  $M'$  is at least  $\epsilon$ , there exists some  $c$  such that  $M'_c$  solves SFP with probability at least  $\epsilon/\beta$ .

For a given instance  $(p, q, g, v)$  of DLP,  $M$  behaves in the following way:

1.  $M$  randomly selects  $c$  such that  $1 \leq c \leq \beta$ .
2.  $M$  simulates  $M'_c$  until  $M'_c$  asks  $c$ -th query to the random oracle.
3. For  $i = 1, 2, \dots, \lceil 6\beta/\epsilon \rceil$ ,  $M$  repeats the following steps:  $M$  simulates  $M'_c$  just after the  $c$ -th query and obtains its output. To generate  $\rho_c, \rho_{c+1}, \dots, \rho_\beta$ ,  $M$  uses a different part of its random tape for each  $i$ .

4. In Step 3, if  $M$  obtains at most two valid signatures for different  $e$ 's, then it outputs "Fail." Otherwise,  $M$  selects three valid signatures,  $(Msg, u, e_{(1)}, w_{(1)})$ ,  $(Msg, u, e_{(2)}, w_{(2)})$ ,  $(Msg, u, e_{(3)}, w_{(3)})$ , where  $e_{(1)}, e_{(2)}, e_{(3)}$  are different from each other. For these signatures,  $M$  solves the following equation:

$$\begin{pmatrix} w_{(1)} \\ w_{(2)} \\ w_{(3)} \end{pmatrix} = \begin{pmatrix} 1 & e_{(1)} & e_{(1)}^2 \\ 1 & e_{(2)} & e_{(2)}^2 \\ 1 & e_{(3)} & e_{(3)}^2 \end{pmatrix} \begin{pmatrix} r \\ k \\ s \end{pmatrix}.$$

5.  $M$  outputs  $s$ .

For a valid signature,  $(Msg, u, e, w)$ ,  $Q_c = (x, u, Msg)$ ,  $\rho_c = e$ , where  $x = g^w u^e v^{e^2} \pmod p$ . The determinant of the matrix of the equation in Step 4 of  $M$  is Vandermonde's determinant, and the matrix is regular. Thus,  $M$  is able to obtain  $s$  by solving the equation.

The probability that  $M$  selects  $c$  such that  $M'_c$  succeeds in solving SFP with probability at least  $\epsilon/\beta$  is at least  $1/\beta$ . If  $M'_c$  succeeds in solving SFP with probability at least  $\epsilon/\beta$ , then, in Step 2 of  $M$ , the probability that  $M'_c$  asks  $Q_c$  such that  $M'_c$  is able to obtain a valid signature with probability at least  $\epsilon/(2\beta)$  is at least  $\epsilon/(2\beta)$ . The probability that  $M$  obtains at least 3 valid signatures from  $M'_c$  in Step 3 is at least  $1/2$  if  $M'_c$  obtains a valid signature for  $\rho_c$  with probability at least  $\epsilon/(2\beta)$ . Thus, the success probability of  $M$  is at least  $(1/\beta)(\epsilon/(2\beta))(1/2) = \epsilon/(4\beta^2)$ .

The number of steps of Step 2 and Step 3 of  $M$  is at most  $\alpha t_{sig} + \lceil 6\beta/\epsilon \rceil (t + t_{ver})$  and that of Step 4 of  $M$  is at most  $O(l^3)$ . Thus, the number of steps of  $M$  is at most  $\alpha t_{sig} + \lceil 6\beta/\epsilon \rceil (t + t_{ver}) + O(l^3)$ .  $\square$

In the following parts, it is shown that the proposed protocol is secure against interference if DLP is intractable. A problem which corresponds to interference is defined. For interference, only the alteration of  $u$ 's is considered.

**Definition 5 (IP: Interference Problem)**

**input:**  $p, q, g, v$ , where  $v$  is randomly selected from  $\{g^0, \dots, g^{q-1}\}$ .

**output:**  $I, I', u', u, e, w$ , where

- $u' \in \{g^0, \dots, g^{q-1}\}$ ,
- $I$  and  $I'$  are  $\{0, 1\}$ -sequences,
- $(u, e, w)$  is a valid signature for  $(u', I, I')$ .  $\square$

IP assumes that an attacker is allowed an adaptive chosen-message attack on a target user whose public key is  $v$ . It is further assumed that the attacker obtains  $(u, e, w)$ 's and the discrete logarithms of  $u$ 's for his queries to the target.

If an attacker succeeds in interference, then he is able to solve IP. IP is at most as difficult as interference in that, for IP,  $I, I', u'$  may be selected arbitrarily by the attacker, while, for interference,  $I'$  is predetermined and  $u'$  is determined by the target user.

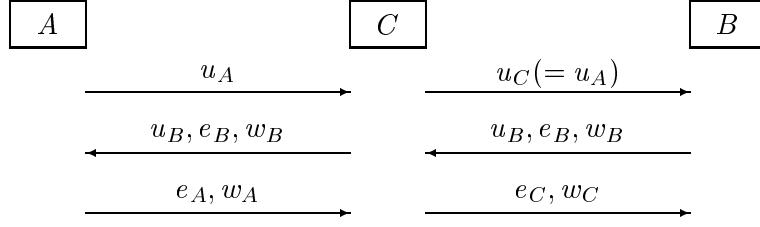


Figure 4: An example of unknown-key share by  $C$  on  $(A, B)$ .  $C$  tries to make  $B$  believe that the session-key is shared between  $C$  and  $B$  though it is actually shared between  $A$  and  $B$ .

**Lemma 2** If there exists some probabilistic Turing machine  $M'$  that solves IP with at most  $t$  steps, with at most  $\alpha$  legitimate message-signature pairs, with at most  $\beta$  queries to the random oracle and with probability at least  $\epsilon$ , then there exists some probabilistic Turing machine  $M$  that solves SFP with at most  $t$  steps, with at most  $\alpha$  legitimate message-signature pairs, with at most  $\beta$  queries to the random oracle and with probability at least  $\epsilon$ .

(Proof) For a given instance  $p, q, g, v$  of SFP,  $M$  first simulates  $M'$  and obtains the output of  $M'$ ,  $(I, I', u', u, e, w)$ . Then,  $M$  outputs  $(u', I, I', u, e, w)$ .  $\square$

To succeed in interference, the attacker has to construct a matching conversation [2] with some public key whose secret key is not known to him. For KAP, construction of a matching conversation is forgery of a signature with the redundant signature scheme.

From Lemma 1 and Lemma 2, the following theorem can be obtained easily.

**Theorem 2** If there exists some probabilistic Turing machine  $M'$  that solves IP with at most  $t$  steps, with at most  $\alpha$  legitimate message-signature pairs, with at most  $\beta$  queries to the random oracle and with probability at least  $\epsilon$ , then there exists some probabilistic Turing machine  $M$  that solves DLP with at most  $\alpha t_{sig} + \lceil 6\beta/\epsilon \rceil (t + t_{ver}) + O(t^3)$  steps and with probability at least  $\epsilon/(4\beta^2)$ .  $\square$

### 3.2.2 Unknown-key share

It is shown that KAP is provably secure against unknown-key share if DLP is intractable.

Fig. 4 shows an example of unknown-key share by  $C$  on  $(A, B)$ . This figure shows the situation where  $C$  tries to make  $B$  believe that the session-key is shared between  $C$  and  $B$  though it is actually shared between  $A$  and  $B$ .  $C$  succeeds in unknown-key share if  $(u_A, e_C, w_C)$  is a valid signature of  $C$  for  $(u_B, I_C, I_B)$ . Hence, unknown-key share does not involve any alterations of  $u_A$  and  $u_B$  and  $C$  does not know the value of the session-key.

First, a problem corresponding to unknown-key share is defined.

#### Definition 6 (UKSP: Unknown-Key Share Problem)

**input:**  $p, q, g, I', I'', s', v', u', u'', e', w'$ , where

- $I'$  and  $I''$  are  $\{0, 1\}$ -sequences,

- $s'$  is randomly selected from  $\mathbf{Z}_q$  and  $v' = g^{-s'} \bmod p$ ,
- $u'$  and  $u''$  are randomly selected from  $\{g^0, \dots, g^{q-1}\}$ ,
- $(u', e', w')$  is a valid signature for  $(u'', I', I'')$  of a signer whose public key is  $v'$ .

**output:**  $I, v, e, w$ , where

- $I$  is a  $\{0, 1\}$ -sequence such that  $I \neq I'$ ,
- $v \in \{g^0, \dots, g^{q-1}\}$ ,
- $(u', e, w)$  is a valid signature for  $(u'', I, I'')$  of a signer whose public key is  $v$ . □

If an attacker succeeds in unknown-key share, then he is able to solve UKSP. UKSP is at most as difficult as unknown-key share, because, for UKSP, the secret key of one of the target users is given as input and  $I$  may be selected arbitrarily by the attacker.

The following theorem shows that UKSP is intractable if DLP is intractable. The theorem can be proved almost in the same way as Lemma 1, and the proof is omitted.

**Theorem 3** If there exists some probabilistic Turing machine  $M'$  that solves UKSP with at most  $t$  steps, with at most  $\beta$  queries to the random oracle and with probability at least  $\epsilon$ , then there exists some probabilistic Turing machine  $M$  that solves DLP with at most  $\lceil 6\beta/\epsilon \rceil(t + t_{ver}) + O(l^3)$  steps and with probability at least  $\epsilon/(4\beta^2)$ . □

## 4 Known limitations and disadvantages

An apparent disadvantage is that each participant has to generate two random numbers from  $\mathbf{Z}_q$  in one execution.

Another disadvantage is that it requires participants slightly more modular exponentiations than other protocols, which has already shown in Table 1. Under a denial-of-service attack, however, the proposed protocol requires less modular exponentiations than many of them.

## 5 Intellectual property issues

The proposed technique is not patented. There are no patent applications on it either.

## Acknowledgements

The first author would like to thank Dr. P. Nikander of Helsinki University of Technology for introducing the research on denial-of-service attacks to him. He also would like to thank Dr. K. Matsuura of the University of Tokyo and Dr. P. Nikander for their fruitful discussions on the resistance against denial-of-service attacks. The original idea of DoS-resistant KAP is from Dr. K. Matsuura.

## References

- [1] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *Cryptography and Coding (IMA'97)*, pages 30–45, 1997. Lecture Notes in Computer Science 1355.
- [2] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.
- [3] M. Just and S. Vaudenay. Authenticated multi-party key agreement. In *ASIACRYPT'96*, pages 36–49, 1996. Lecture Notes in Computer Science 1163.
- [4] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. Technical Report CORR98-05, Department of C&O, University of Waterloo, 1998.
- [5] C. P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO'89*, pages 239–252, 1990. Lecture Notes in Computer Science 435.