PAPER

Key Agreement Protocols Resistant to a Denial-of-Service Attack

Shouichi HIROSE[†] ***and*** **Kanta MATSUURA**[††], ***Regular Members***

SUMMARY In this manuscript, two key agreement protocols which are resistant to a denial-of-service attack are constructed from a key agreement protocol in [9] provably secure against passive and active attacks. The denial-of-service attack considered is the resource-exhaustion attack on a responder. By the resource-exhaustion attack, a malicious initiator executes a key agreement protocol simultaneously as many times as possible to exhaust the responder's resources and to disturb executions of it between honest initiators and the responder. The resources are the storage and the CPU. The proposed protocols are the first protocols resistant to both the storage-exhaustion attack and the CPU-exhaustion attack. The techniques used in the construction are stateless connection, weak key confirmation, and enforcement of heavy computation. The stateless connection is effective to enhancing the resistance to the storage-exhaustion attack. The weak key confirmation and the enforcement of heavy computation are effective to enhancing the resistance to the CPU-exhaustion attack.

key words: *key agreement protocol, denial-of-service (DoS) attack, resource-exhaustion attack, Diffie-Hellman key agreement protocol*

1. Introduction

In using private key encryption for secure communication, it is necessary for the participants to share a common key securely in advance with some key agreement scheme. Key agreement schemes can be classified into two types: non-interactive schemes and interactive schemes. In this manuscript, the interactive schemes are discussed, which are called the key agreement protocols (KAP's).

A three-pass KAP was proposed in [9] which is based on the Diffie-Hellman KAP [4]. This protocol is provably secure against passive and active attacks in the random oracle model [2] on the assumptions that the Diffie-Hellman problem is intractable and that the secret pieces of information of users are selected at random and independently of each other. This protocol also provides key confirmation, and the method for providing it is different from those of the other protocols such as in [3], [5], [10], [12]. This protocol provides key confirmation without using the shared key explicitly. Each participant only shows the other participant that he can compute the shared key. This kind of key confirmation is called weak key confirmation.

In this manuscript, two KAP's which are resistant to a denial-of-service attack are constructed from the KAP in [9]. The denial-of-service (DoS) attack considered is a resource-exhaustion attack on a responder. A malicious initiator launches as many (bogus) requests as possible one after another without establishing connection with the target responder in order to exhaust the responder's resource and to disturb executions with honest initiators. The resources are the storage and the CPU. The amount of storage of a responder determines the upper limit of the number of initiators who are able to execute a KAP with the responder simultaneously. The computation load on the CPU of a responder determines the throughput. An anti-clogging token called Cookie [11] is an efficient way to avoid DoS with IP-address spoofing. However, when it is used in public-key based authenticated protocols like corresponding modes of IKE (the Internet Key Exchange) [8], the responder must perform expensive public-key based operation *before* he/she becomes sure of the initiator's identity. By contrast, the proposed protocols are the first authenticated protocols that are resistant both to the storage-exhaustion attack and to the CPU-exhaustion attack.

The techniques used in our transformation are stateless connection [1], weak key confirmation, and enforcement of heavy computation. The stateless connection is effective to enhancing the resistance to the storage-exhaustion attack. It releases responders from keeping connection states, which is kept by initiators. Once released, the responders are confronted with a risk of replay-flooding attack. In a comparison of the replay-flooding attack against stateless protocols with the storage-exhaustion attack against stateful ones, however, the stateless protocols are shown to provide a better performance [1].

The weak key confirmation and the enforcement of heavy computation are effective to enhancing the resistance to the CPU-exhaustion attack. In the proposed KAP's, the most time-consuming computation is a modular exponentiation. The weak key confirmation enables a responder to compute the shared key, which needs a modular exponentiation, only if the received messages are valid.

The validity verification starts with a light step

Manuscript received August 16, 1999.
Manuscript revised November 15, 2000.

[†]The author is with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.
[††]The author is with the Institute of Industrial Science, the University of Tokyo, Tokyo, 106–8558 Japan.

of hashing before main heavy computation on the responder's side. Since the light step can tell whether the initiator has really completed the exponentiation, a malicious initiator will fear a "falling-together" nightmare: he/she has to compute modular exponentiations in order to make the responder compute modular exponentiations for checking the validity of the received messages. The basic concept of this trick is "attackers must pay a lot for the attack." A *pricing function* by Dwork and Naor [6] is based on the same concept. The pricing function was, however, introduced for combatting junk mails in one-pass protocols *without* authentication mechanisms. On the other hand, we studied three-pass protocols equipped *with* authentication.

The proposed KAP's are still practical in computation although they require the participants to exchange the connection state and additional authenticators: they cost encryption/decryption of a private-key cipher and one-way hashing, which are computationally inexpensive in comparison with modular exponentiations, the main cost of the prototype KAP. In addition, depending on the operation mode, the required number of modular exponentiations in effect can be reduced by the use of simultaneous multiple exponentiation. The proposed KAP's also remain secure and one of them almost inherits the provable security, except that they become vulnerable to the replay attack.

This paper is constructed as follows. Section 2 first overviews the KAP in [9]. DoS-resistant KAP's are then proposed in Sect. 3, with a discussion on the resistance. Section 4 includes the performance analysis of the proposed protocols. Conclusion is given in Sect. 5.

2. The Base Key Agreement Protocol

In this section, the KAP in [9] is reviewed, which is the base of the DoS-resistant KAP's proposed in the next section. It is an authenticated KAP based on the Diffie-Hellman KAP. In this manuscript, it is called the base KAP.

Let p and q be large primes such that $q \,|\, p-1$ and $g \in \mathrm{GF}(p)$ of order q. Let $h : \{0,1\}^* \to \mathbf{Z}_q$ be a collision-free hash function. p, q, g, h are public and shared among all of the users. Let $s_i \in \mathbf{Z}_q$ be a secret piece of information of the user i and $v_i = g^{-s_i} \bmod p$ be a public piece of information of the user i. Let I_i be the ID of the user i. It is assumed that the users know the public pieces of information of the other users or that the public piece of information of each user is contained in his ID.

The base KAP

1. The initiator A randomly selects $k_A \in \mathbf{Z}_q$ and computes $u_A = g^{-k_A} \bmod p$.
2. A sends u_A, I_A to B.
3. The responder B randomly selects $k_B \in \mathbf{Z}_q$ and computes $u_B = g^{-k_B} \bmod p$. B also selects $r_B \in \mathbf{Z}_q$ at random and computes $x_B = g^{r_B} \bmod p$. B computes $e_B = h(x_B, u_B, u_A)$ and $w_B = r_B + e_B k_B + e_B^2 s_B \bmod q$.
4. B sends u_B, e_B, w_B, I_B to A.
5. A computes $z_B = g^{w_B} u_B^{e_B} v_B^{e_B^2} \bmod p$ and checks if $e_B = h(z_B, u_B, u_A)$. If it does not hold, then A terminates the execution. Otherwise, A randomly selects $r_A \in \mathbf{Z}_q$ and computes $x_A = g^{r_A} \bmod p$. Then, A computes $e_A = h(x_A, u_A, u_B)$ and $w_A = r_A + e_A k_A + e_A^2 s_A \bmod q$.
6. A sends e_A, w_A to B.
7. 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 $e_A = h(z_A, u_A, u_B)$. If it does not hold, then B terminates the execution. Otherwise, B computes $K_B = u_A^{-k_B} \bmod p$.

As is mentioned in Introduction, this protocol provides key confirmation by the novel method which is different from those of the previous protocols. This protocol provides it without using the shared key explicitly. Each participant only shows the other participant that he can compute the shared key. This kind of key confirmation is called the weak key confirmation.

A signature scheme called the redundant signature scheme is used for providing the weak key confirmation. It is an extension of Schnorr's signature scheme [13]. For example, (u_B, e_B, w_B) is able to be regarded as a signature for u_A, which is used as a nonce. By checking the validity of (u_B, e_B, w_B), that is, by computing z_B and checking that $e_B = h(z_B, u_B, u_A)$ in Step 5, A is able to make sure that B received u_A, that B sent u_B in response to the receipt of u_A and that B knows k_B. Thus, A is able to make sure that B is able to compute K_B which is equal to K_A. B does not use K_B in computing (u_B, e_B, w_B).

This protocol is provably secure against passive and active attacks in the random oracle model [2] on the assumptions that the Diffie-Hellman problem is intractable and that the secret pieces of information of users are selected at random and independently of each other [9]. All of these attacks are assumed to be known-key attacks: it is provably secure even if all the previously shared keys are disclosed to the attacker. In addition, this protocol provides forward secrecy: the security against passive eavesdropping is proved on the assumption that the attacker knows the secret pieces of information of the participants.

3. Key Agreement Protocols Resistant to a Denial-of-Service Attack

3.1 Denial-of-Service Attack

The DoS attack considered is the resource-exhaustion attack on a responder by a malicious initiator. The resources are the storage and the CPU. KAP's usually require the storage which keeps connection states. The amount of storage of a responder determines the upper limit of the number of initiators who are able to execute a KAP with the responder simultaneously. On the other hand, the computation load on the CPU of a responder determines the throughput.

By the DoS attack, a malicious initiator tries to disturb executions of the KAP between honest initiators and a target responder. The malicious initiator launches as many (bogus) requests as possible one after another without establishing connections with the responder by using arbitrary ID's. In the worst case, the storage of the responder is exhausted and new requests are refused. Even if the storage is not exhausted, the throughput is degraded by the computation load on the responder. Thus, it is preferable that the amount of the storage and the computation load required by each execution are as small as possible.

The base KAP is unfortunately vulnerable to the DoS attack, regarding both the storage and the CPU. In the DoS attack against the base KAP, a malicious initiator X uses arbitrary ID I_X, and sends u_X, I_X to the target responder B. Then, X receives u_B, e_B, w_B, I_B from B and stops sending e_X, w_X or postpones it until just before a certain expiration time. X repeats this procedure as many times as possible. For each request of this attack, B has to keep (I_X, u_X, k_B, u_B) until the expiration or until he receives e_X, w_X. Moreover, he has to compute two modular exponentiations in Step 3, that is, $u_B = g^{-k_B} \bmod p$ and $x_B = g^{r_B} \bmod p$. B also has to compute modular exponentiations in Step 6, that is, $z_X = g^{w_X} u_X^{e_X} v_X^{e_X^2} \bmod p$, if he receives e_X, w_X to check the validity of u_X, e_X, w_X. On the other hand, X need not compute any modular exponentiation during the attack. X need not check the validity of u_B, e_B, w_B nor compute valid u_X, e_X, w_X. His only purpose is to make the responder be over-loaded, and he achieves it with much less computation than B.

In the next subsection, the base KAP is transformed to the protocols more resistant to the DoS attack. The techniques used in the transformation are the stateless connection [1], the weak key confirmation, and the enforcement of heavy computation.

3.2 Protocols

Two DoS resistant protocols are presented in this subsection.

The first DoS resistant protocol can be constructed by applying the idea of the stateless connection directly to the base KAP. The protocol is shown below. Let E_B (D_B) be an encryption (a decryption) function of B. These are assumed to be the functions of some symmetric key encryption scheme.

<u>DoS-resistant KAP1</u>

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, I_A to B.
2. B computes $e_B = h(x_B, u_B, u_A)$ and $w_B = r_B + e_B k_B + e_B^2 s_B \bmod q$. B also computes $c_B = E_B(k_B, x_B)$.
3. B sends u_B, e_B, w_B, c_B, I_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 $e_B = h(z_B, u_B, u_A)$. If it does not hold, then A terminates the execution. Otherwise, A computes $a_A = h(z_B, u_A, u_B)$. A also computes $e_A = h(x_A, u_A, u_B)$ and $w_A = r_A + e_A k_A + e_A^2 s_A \bmod q$.
5. A sends $u_A, e_A, w_A, a_A, u_B, e_B, c_B, I_A$ to B.
6. A computes $K_A = u_B^{-k_A} \bmod p$.
   B recovers $(k_B, x_B) = D_B(c_B)$ and checks if $a_A = h(x_B, u_A, u_B)$ and $e_B = h(x_B, u_B, u_A)$. If they do 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 $e_A = h(z_A, u_A, u_B)$. If it does not hold, then B terminates the execution. Otherwise, B computes $K_B = u_A^{-k_B} \bmod p$.

This protocol is also shown in Fig. 1. It requires the participants to pass the connection state and its authenticators back and forth, which requires additional bandwidth. The computational costs of the authenticators are an encryption/decryption of a private-key cipher and a few one-way hashings, which are computationally inexpensive.

Precisely speaking, for DoS-resistant KAP1, the provable confidentiality of the shared key of the base KAP is ruined in general, because k_B is encrypted with E_B and the ciphertext is transformed over an insecure channel. Furthermore, because k_B is encrypted with E_B, the leak of the secret key of E_B causes serious damage. Thus, the management of the secret key is very important for this protocol. An example of the scheme for the key management is as follows: B keeps the master key K^* securely and uses $h(K^*, t)$ as a secret key of E_B, where h is a collision-free hash function and t is a nonce. Although t need not be kept secret, it should be changed every certain period of time.

DoS-resistant KAP2 in Fig. 2 solves the above problem of KAP1. This protocol assumes that the responder B makes use of the same u_B in many executions. This assumption does not ruin the provable security of the base KAP. In this case, B can use the same k_B to compute shared keys in many executions of the protocol and he can keep k_B in his own storage. In this protocol, x_B is used as a nonce for A instead of u_B, and $e_A = h(x_A, u_A, u_B, x_B)$.

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

Fig. 1 DoS-resistant KAP1.

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

Fig. 2 DoS-resistant KAP2. B can use the same u_B and k_B in many executions: B updates k_B (and consequently u_B) in not every execution of the protocol.

3.3 Resistance to the DoS Attack

In the following, resistance to the DoS attack is discussed only for the DoS-resistant KAP1. The discussion for the DoS-resistant KAP2 is similar to it.

First, let us consider the storage-exhaustion attack. By the stateless connection for the responder B, the connection state (I_A, u_A, k_B, u_B) need not be kept by B during an execution of the protocol, and the storage-exhaustion attack has no effect on DoS-resistant KAP1. e_B and c_B guarantee the integrity and confidentiality of the state information. In Step 6 of DoS-resistant KAP1, B can confirm that u_A and u_B are used together for computing the shared key if $e_B = h(x_B, u_B, u_A)$. B can recover k_B by decrypting c_B. Since $c_B = E_B(k_B, x_B)$ and $e_B = h(x_B, u_B, u_A)$, x_B implies to B that $-k_B$ is the discrete logarithm of u_B. Hence, B need not check whether $u_B = g^{-k_B} \bmod p$.

Next, let us consider the CPU-exhaustion attack. The computation of modular exponentiations are mainly considered, because it is the most time-consuming computation. $u_B = g^{-k_B} \bmod p$ and $x_B = g^{r_B} \bmod p$ are able to be computed off-line; Step 0 can be implemented as a precomputation step. The shared key need be computed only if a_A, e_B and u_A, e_A, w_A are valid because this protocol provides the weak key confirmation. Hence, B need not compute any modular exponentiation in Step 2 though this protocol is based on the Diffie-Hellman KAP. Furthermore, due to the enforcement of heavy computation, B need not compute any modular exponentiations on-line if $a_A \neq h(x_B, u_A, u_B)$ or $e_B \neq h(x_B, u_B, u_A)$. This means that, in order to make B compute $z_A = g^{w_A} u_A^{e_A} v_A^{e_A^2} \bmod p$ in Step 6, the malicious initiator has to pay quite similar computational cost in Step 4 for $z_B = g^{w_B} u_B^{e_B} v_B^{e_B^2} \bmod p$ and $a_A = h(z_B, u_A, u_B)$: heavy computation of z_B is required for providing the correct extra hash a_A which will be inexpensively checked by the responder at the beginning of Step 6. This property somewhat discourages the attackers from applying the CPU-exhaustion attack. We shall recommend readers to compare this cost on the attacker's side with that on the responder's, which will be discussed later in the next section.

The KAP with stateless connection such as those proposed in this manuscript is intrinsically vulnerable to the replay-flooding attack. Looking at a comparison of the replay-flooding attack against stateless protocols with the storage-exhaustion attack against stateful ones, however, we can find that the stateless protocols perform better [1].

4. Performance Analysis

In an actual network, one application may differ from another in requirements on the efficiency of a key agreement protocol. For example, a network-layer application may probably have much more stringent requirements than an upper-layer application. This section analyzes the computational costs of the proposed versions of KAP.

4.1 Basic Analysis

Since the main source of the computational cost is modular exponentiation, Step 4 and Step 6 have to be analyzed both in DoS-resistant KAP1 and in KAP2: specifically, the computation of z_B and z_A, and the key establishment (K_A on the initiator's side and K_B on the responder's side).

The cost of the key establishment is trivial: one modular exponentiation on each side. The exponentiation is carried out by using one-time exponent and one-time base: none of (u_A, u_B, k_A, k_B) is a long-term value. The rest of this section is devoted to the analysis of the cost of z_B and z_A.

A simple implementation may pay three exponentiations for computing z_B. This, however, can be reduced to 2.17 exponentiations by the use of the simultaneous multiple exponentiation technique which was attributed by ElGamal [7] to Shamir[†]. $z_B = g^{w_B} v_B^{e_B^2} u_B^{e_B} \bmod p$, where the first two bases g and v_B are known in advance. Since we can use Shamir's trick to $g^{w_B} v_B^{e_B^2} \bmod p$, the total cost is given by $(2/3)(2-(1/2)^2)+1 = 2.17$ modular exponentiations.

On the responder's side, fully conforming to the DoS consideration, computation of z_A costs three modular exponentiations.

Table 1 Comparison of the three modes on the responder's side. The cost is measured by the equivalent number of modular exponentiations; the cost in Step 4 on the initiator's side is 2.17. After precomputation of simultaneous multiple exponentiation, Light Mode requires additional storage of four precomputed results, while Moderate Mode one.

Mode	Additional storage	Additional precomputation	Cost in Step 6
Light Mode	(u_A, I_A)	4 multiplications	1.25
Moderate Mode	I_A	1 multiplication	2.17
Simple Mode	0	0	3

4.2 Optional Saving

One approach for saving in computational cost on the responder's side is a DoS-resistance-dependent use of three different modes.

The lightest mode requires the responder to keep the state information carried by the message in Step 1. This makes the protocol less resistant against the storage-exhaustion attack by the storage of (u_A, I_A). Since both u_A and v_A can be regarded as a pre-known value in this case, the computation of z_A in Step 6 can use the simultaneous multiple exponentiation for three bases: g, u_A, and v_A. Thus the cost in Step 6 can be reduced to $(2/3)(2-(1/2)^3) = 1.25$ exponentiations. It should be noted that the precomputation for simultaneous multiple exponentiation here costs only four modular multiplications. We refer to this mode as "Light Mode."

The second one requires the responder to keep the state information I_A. Since v_A can be regarded as a pre-known value in this case, the computation of z_A in Step 6 can use the simultaneous multiple exponentiation for two bases: g and v_A. Thus the cost in Step 6 can be reduced to $(2/3)(2-(1/2)^2)+1 = 2.17$ exponentiations. It should be noted that the precomputation for simultaneous multiple exponentiation here costs only one modular multiplication. We refer to this mode as "Moderate Mode."

The third one is the simple mode with the cost of three exponentiations mentioned in the previous subsection. We refer to this mode as "Simple Mode."

These three modes are summarized in Table 1. Depending on the real-time situation of the required DoS care[††] or traffic condition, the mode can be switched from one to another.

[†]This algorithm is summarized in Appendix.

[††]Precomputed values require additional storage as state information.

4.3 DoS-Resistance Evaluation

The DoS-resistance of KAP1 is compared with that of the other protocols with key confirmation. These protocols are

STS	Station-to-station protocol [5] with Schnorr's signature scheme [13],
JV	Protocol IIA of Just and Vaudenay [10],
BJM	Protocol 2 of Blake-Wilson, Johnson and Menezes [3],
LMQSV	Protocol 3 of Law, Menezes, Qu, Solinas and Vanstone [12].

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. All protocols are assumed to be executed in Light Mode.

Stateless connection is a general technique which is applicable to other protocols. Thus, in this subsection, only the CPU-exhaustion attack is considered, and the number of modular exponentiations required to a responder under the CPU-exhaustion attack is evaluated. 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 in this subsection. One is a valid request, another is an invalid request with a last message and the other is an invalid request without a last message. The last message, for example, for DoS-resistant KAP1 or KAP2, is the one sent in Step 5 by an initiator. The last two types of requests are from malicious initiators who try to make the CPU-exhaustion attack. Malicious initiators are expected to tend to make invalid requests without last messages, because they need not keep the status of their past requests in mind and are able to concentrate on launching as many new invalid requests as possible. 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 2 shows the numbers of all modular exponentiations for the three types of requests. For BJM, the computational load of a MAC involved in it is not considered here. KAP1 is the least efficient for a good request. However, weak key confirmation of KAP1 reduces the number of exponentiations of KAP1 for a bad request. For a minor bad request, KAP1 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 KAP1 is the most efficient protocol.

Notice that the enforcement of heavy computation of KAP1 makes minor bad requests less applicable to it. The number of exponentiations for a minor bad request is equal to that for a major bad one unless a responder receives valid a_A, which requires a (malicious) initiator to compute 1.25 exponentiations, $z_B = g^{w_B} u_B^{e_B} v_B^{e_B^2} \bmod p$.

Table 3 shows the lower bounds of the ratios of bad requests to all requests for the average number of modular exponentiations of KAP1 to be smaller than those of the other protocols. These are evaluated based on Table 2. Good ∨ MinorBad represents each request is a good request or a minor bad request. Good ∨ MajorBad represents each request is a good request or a major bad request. In the case of Good ∨ MajorBad,

Table 2 The number of modular exponentiations required to a responder for a good request, a minor bad request and a major bad request. STS is the station-to-station protocol [5] with Schnorr's signature scheme [13], JV is Protocol IIA of Just and Vaudenay [10], BJM is Protocol 2 of Blake-Wilson, Johnson and Menezes [3], and LMQSV is Protocol 3 of Law, Menezes, Qu, Solinas and Vanstone [12]. All protocols are in Light Mode. For BJM, the computational load of MAC involved in it is not considered here.

request \ protocol	KAP1	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 3 The lower bounds of the ratios(%) of bad requests to all requests for the average number of modular exponentiations of KAP1 to be smaller than those of protocols compared. These are evaluated based on Table 2. Good ∨ MinorBad represents each request is a good request or a minor bad request. Good ∨ MajorBad represents each request is a good request or a major bad request. "xxx" means that the average number of modular exponentiations of KAP1 is always larger.

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

Table 4 The number of on-line modular exponentiations required to 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	KAP1	STS	JV	BJM	LMQSV
good request	2.25	2.17	3	2	1.17
minor bad request	1	2.17	3	2	1.17
major bad request	0	1	2	2	1.17

Table 5 The lower bounds of the ratios(%) of bad requests to all requests for the average number of on-line modular exponentiations of KAP1 to be smaller than those of protocols compared. These are evaluated based on Table 4.

request \ protocol	STS	JV	BJM	LMQSV
Good $\vee$ MinorBad	6.4	0	20.0	86.4
Good $\vee$ MajorBad	7.4	0	11.1	48.0

KAP1 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 KAP1 is smaller than that of STS. In the case of Good $\vee$ MinorBad, the average number of exponentiations of KAP1 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 logarithm.

Table 4 shows the numbers of on-line modular exponentiations for the three types of requests. For a bad request, KAP1 is the most efficient protocol in terms of on-line computation. Especially, for a major bad request, KAP1 requires no on-line exponentiations to a responder.

The average number of on-line modular exponentiations of KAP1 can be smaller than those of the other protocols when some of the requests are bad. Table 5 shows the lower bounds of the ratios of bad requests to all requests for the average number of exponentiations of KAP1 to be smaller than those of the other protocols. These are evaluated based on Table 4.

5. Conclusion

Two key agreement protocols which are resistant to a DoS attack have been proposed in this manuscript. The denial-of-service attack considered is the resource-exhaustion attack on a responder, where the resources are the storage and the CPU. The proposed KAP's are the first protocols resistant both to the storage-exhaustion attack and to the CPU-exhaustion attack.

Their DoS-resistance is evaluated with a comprehensive analysis on multiple modular exponentiation. This may simplify DoS-related bandwidth-control mechanisms (*e.g.* connection timeout for buffers) in different layers: without the evaluation, it would be too hard to achieve balance between security and latency demands.

Future work is to design a provably secure and DoS-resistant key agreement protocol.

Acknowledgement

The authors would like to thank Dr. P. Nikander for giving them the motivation and for fruitful discussions.

References

[1] T. Aura and P. Nikander, "Stateless connections," Information and Communications Security (ICICS'97), pp.87–97, 1997. Lecture Notes in Computer Science 1334.

[2] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," Proc. 1st ACM Conference on Computer and Communications Security, pp.62–73, 1993.

[3] S. Blake-Wilson, D. Johnson, and A. Menezes, "Key agreement protocols and their security analysis," Cryptography and Coding (IMA'97), pp.30–45, 1997. Lecture Notes in Computer Science 1355.

[4] W. Diffie and M. E. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theory, vol.IT-22, no.6, pp.644–654, 1976.

[5] W. Diffie, P. C. van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," Designs, Codes and Cryptography, vol.2, no.2, pp.107–125, 1992.

[6] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," CRYPTO'92, pp.139–147. Springer-Verlag, 1993. Lecture Notes in Computer Science 740.

[7] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. Inf. Theory, vol.IT-31, no.4, pp.469–472, 1985.

[8] D. Harkins and D. Carrel, The internet key exchange (IKE). RFC2409, 1998.

[9] S. Hirose and S. Yoshida, "An authenticated Diffie-Hellman key agreement protocol secure against active attacks," PKC'98, pp.135–148, 1998. Lecture Notes in Computer Science 1431.

[10] M. Just and S. Vaudenay, "Authenticated multi-party key agreement," ASIACRYPT'96, pp.36–49, 1996. Lecture Notes in Computer Science 1163.

[11] P. Karn and W. Simpson, Photuris: Session-key management protocol, RFC2522, 1999.

[12] 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.

[13] C. P. Schnorr, "Efficient identification and signatures for smart cards," CRYPTO'89, pp.239–252, 1990. Lecture Notes in Computer Science 435.

Appendix: Simultaneous Multiple Exponentiation

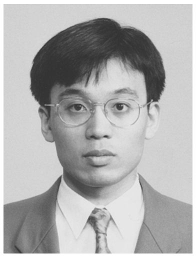
The followings are a brief description of a simultaneous multiple exponentiation algorithm, which was attributed by ElGamal [7] to Shamir.

Input: group elements $g_0, g_1, \cdots, g_{k-1}$ and non-negative t-bit integers $e_0, e_1, \cdots, e_{k-1}$.
Output: $g_0^{e_0} g_1^{e_1} \cdots g_{k-1}^{e_{k-1}}$.
Assumption: $g_0, g_1, \cdots, g_{k-1}$ are known in advance and thus can be used in precomputation.

1. For $i = 1$ to $(2^k - 1)$, $G_i := \prod_{j=0}^{k-1} g_j^{i_j}$ where $i = (i_{k-1} \cdots i_0)_2$ (*Precomputation*).
2. Let I_i be the i-th column of an exponent array (binary representation).
3. $A := 1$.
4. For $i = 1$ to t, $A := A \cdot A$ and then $A := A \cdot G_{I_i}$. If $I_i = 0$ (and hence $G_{I_i} = 1$), the latter multiplication is trivial.
5. Return(A).

Expected Cost: $\left\{2 - \left(\frac{1}{2}\right)^k\right\} t$ non-trivial multiplications when the exponents are independently and randomly generated. Assuming that the cost of "one modular exponentiation" means the expected number of non-trivial modular multiplications in the case of "square and multiply" method, the simultaneous multiple exponentiation reduces the cost from k exponentiations down to

$$\frac{\left\{2 - \left(\frac{1}{2}\right)^k\right\} t}{\frac{3}{2} t} = \frac{2}{3}\left\{2 - \left(\frac{1}{2}\right)^k\right\}. \qquad \text{(A$\cdot$1)}$$



Shouichi Hirose was born in Kyoto, Japan, on December 30, 1965. He received the B. E., M. E. and D. E. degrees in information science from Kyoto University, Kyoto, Japan, in 1988, 1990 and 1995, respectively. From 1990 to 1997, he was a research associate at Faculty of Engineering, Kyoto University. From 1998, he is a lecturer at the Graduate School of Informatics, Kyoto University. His current interests include cryptography, information security and digital mobile communications. He received Young Engineer Award from IEICE in 1997. He is a member of ACM, IEEE, IACR and IPSJ.

Kanta Matsuura was born in Osaka, Japan, in 1969. He received B.Eng. degree in electrical engineering in 1992, M.Eng. degree in electronics in 1994, and Ph.D. degree in electronics in 1997, all from the University of Tokyo, Japan. He is at present a Lecturer of Institute of Industrial Science, the University of Tokyo. His current research interests include network security, protocol-optimization techniques, and Internet technologies. He is a member of IEEE.