INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
*Int. J. Network Mgmt* 2005; **15**: 193–202
Published online 28 February 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nem.561

# Efficient authenticated key agreement protocols resistant to a denial-of-service attack

By Yuh-Min Tseng*,†

***Malicious intruders may launch as many invalid requests as possible without establishing a server connection to bring server service to a standstill. This is called a denial-of-service (DoS) or distributed DoS (DDoS) attack. Until now, there has been no complete solution to resisting a DoS/DDoS attack. Therefore, it is an important network security issue to reduce the impact of a DoS/DDoS attack. A resource-exhaustion attack on a server is one kind of denial-of-service attack. In this article we address the resource-exhaustion problem in authentication and key agreement protocols. The resource-exhaustion attack consists of both the CPU-exhaustion attack and the storage-exhaustion attack. In 2001, Hirose and Matsuura proposed an authenticated key agreement protocol (AKAP) that was the first protocol simultaneously resistant to both the CPU-exhaustion attack and the storage-exhaustion attack. However, their protocol is time-consuming for legal users in order to withstand the DoS attack. Therefore, in this paper, we propose a slight modification to the Hirose–Matsuura protocol to reduce the computation cost. Both the Hirose–Matsuura and the modified protocols provide implicit key confirmation. Also, we propose another authenticated key agreement protocol with explicit key confirmation. The new protocol requires less computation cost. Because DoS/DDoS attacks come in a variety of forms, the proposed protocols cannot fully disallow a DoS/DDoS attack. However, they reduce the effect of such an attack and thus make it more difficult for the attack to succeed.  Copyright © 2005 John Wiley & Sons, Ltd.***

## 1. Introduction

Recently, a loss of availability through denial-of-service (DoS) or distributed DoS (DDoS) attacks has become an important network security issue.[1,2] DoS/DDoS attacks can result in significant loss of profits for many enterprises and organizations. A server needs some resources to provide services for legitimate users. These resources include network con-

*Yuh-Min Tseng received the B.S. degree in Computer Science and Engineering from National Chiao Tung University, Taiwan, Republic of China, in 1988; and the M.S. degree in Computer and Information Engineering from National Taiwan University in 1990 and the Ph.D. degree in Applied Mathematics from National Chung-Hsing University in 1999. He is currently an Associate Professor in the Department of Mathematics, National Changhua University of Education, Taiwan. He is a member of IEEE Communications Society and the Chinese Cryptology and Information Security Association (CCISA). His research interests include cryptography, communication security, network security, and mobile communications.*

**Correspondence to: Yuh-Min Tseng, Information Security Laboratory, Department of Mathematics, National Changhua University of Education, Jin-De Campus, Chang-Hua 500, Taiwan*
†*E-mail: ymtseng@cc.ncue.edu.tw*

*Contract/grant sponsor: National Science Council, Taiwan, R.O.C.; contract/grant number: NSC9 3-2213-E-018-011*

nectivity between a user and a server, server's memory space and CPU time. An intruder is able to consume the network bandwidth using a large number of request packets sent to a server over the Internet to prevent legitimate users from connecting to the server. Malicious intruders may also launch as many invalid requests as possible without establishing a server connection to bring the services provided by the server to a standstill. Until now, there has been no complete solution to resisting a DoS/DDoS attack. A resource-exhaustion attack on a server is a kind of denial-of-service attack. The resource-exhaustion attack consists of the CPU-exhaustion attack and the storage-exhaustion attack. In this paper, we address the resource-exhaustion problem in authentication and key agreement protocols.

***Loss of availability through denial-of-service attacks has become an important network security issue.***

To achieve secure communication over an open network, two principals use a key agreement or a key distribution protocol to establish a session key to encrypt the communication data between them. In 1976, Diffie and Hellman[3] first proposed a secure key agreement protocol. However, their protocol does not allow two principals to authenticate each other. Their protocol therefore requires an authentication channel to exchange the public keys. Afterwards, many authenticated key agreement protocols were proposed to integrate authentication into key agreement protocols. According to technical categories of the authentication approach, these key agreement protocols may be classified into two categories: public-key-based key agreement protocols[4–8] and password-based key agreement protocols.[9,10] A public-key-based key agreement protocol adopts digital signature schemes[11,12] into a key agreement protocol to provide mutual authentication. A password-based key agreement protocol allows both principals to share a secret password in advance to provide the authentication property. In this article, we address key agreement protocols based on public key systems.

In the past, several desired security goals and properties were considered for authenticated key agreement protocols.[13–15] An authenticated key agreement protocol (AKAP) should provide security goals and properties. In the following, first we describe two kinds of security goals. An authenticated key agreement protocol must achieve one of two security goals to withstand some attacks.

1. *Implicit (Weak) key confirmation*. This means that each principal shows to the other principal who can compute the session key.
2. *Explicit (Strong) key confirmation*. This means that a principal is assured that another principal has actually computed the session key.

In addition, an authenticated key agreement protocol must also have four security properties as follows:

1. *Known-key security*. Suppose that a session key established by two principals is disclosed. The adversary must be unable to learn future session keys between them.
2. *Forward secrecy*. If both secret keys of two principals are disclosed, the adversary must be unable to derive any old session keys established by them.
3. *Key-compromise impersonation*. Assume that entities $A$ and $B$ are two principals. If $A$'s secret key is disclosed, obviously, an adversary that knows this secret key can impersonate $A$ to other principals. However, it is desired that in some situation this disclosure does not allow the adversary to impersonate other principals to $A$.
4. *Unknown key-share*. When principal $B$ believes the key is shared with some principal $C \neq A$, and $A$ believes that the key is shared with $B$. This scenario cannot be permitted.

An authenticated key agreement protocol is designed to provide the mutual authentication function between two communication parties and establish a session key. This is one of the fundamental protocols for many applications in Internet or network systems. Although authenticated key agreement protocols ensure that no malicious intruders can obtain the messages transmitted over an open channel, the previously proposed key agreement protocols[4–6,8] are unable to withstand the denial-of-service attack.

In 2001, Hirose and Matsuura proposed a key agreement protocol that was the first protocol simultaneously resistant to both the CPU-exhaustion attack and the storage-exhaustion attack. In this paper, first we propose a slight modification to the Hirose–Matsuura protocol to reduce the server computation cost. We then propose a new authenticated key agreement protocol with explicit key confirmation, in which the computation cost can be further reduced. Our proposed protocols meet the security requirements for an authenticated key agreement protocol. Although the proposed protocols do not fully resist a DoS/DDoS attack, both protocols reduce the effect of such an attack and thus make it more difficult for the attack to succeed.

The remainder of this article is organized as follows. In the next section, some design principles and strategies for withstanding a DoS attack are presented. In Section 3, we briefly review the Hirose–Matsuura protocol. Section 4 gives our proposed protocols and security analysis. In Section 5, performance comparisons among our proposed protocols and the Hirose–Matsuura protocol are presented. Section 6 gives our conclusions.

***Availability is defined as valid access for data and services to an authorized user within a reasonable time.***

## 2. Design Principles

Availability is defined as valid access for data and services to an authorized user within a reasonable time.[16] Malicious intruders may launch as many invalid requests as possible without establishing server connection to bring the services provided by the server to a standstill: thus, the server cannot provide services for authorized users. The resource-exhaustion attack consists of the CPU-exhaustion attack and the storage-exhaustion attack. Therefore, in the following we will describe strategies for preventing the CPU-exhaustion attack and the storage-exhaustion attack.

Let us first consider how to withstand the CPU-exhaustion attack. The fundamental design approach is that the client computation load should be higher than or equal to that of the server. We provide some design rules as follows.

1. In an authenticated key agreement protocol, several interactive steps occur between a client and a server. Initially, a client sends a request to the server. Malicious intruders may send random messages to the server. After receiving a request, the server should have no on-line modular exponentiation computations. That is, the pre-computation is independent of the client's identity.
2. Malicious intruders may compute exponentiations to ensure that the server also computes some exponentiations to validate the received message. When the client increases the computation cost to carry out the attack in other steps, the increased computation cost to the server should be less than or equal to that of the client. The concept is that malicious intruders must carry a great computation load to initiate the attack. This concept is called the pricing function.[17]
3. The fundamental design approach is that the computation cost to a client should be higher than or equal to that of the server. However, intentionally increasing the computation cost to a client is not permitted because it will affect legitimate clients at the same time.

In the following, let us consider the storage-exhaustion attack on an authenticated key agreement protocol. The server need not maintain the connection state before establishing a correct connection. Therefore, this connection is called the 'stateless connection'.[18] To achieve this goal, the server must combine the request message with some pre-computed values into an encrypted material and then send it to the client. This process is not recorded by the server. Afterwards, in the next step the client must re-send the encrypted material to the server.

According to the above design principles for preventing a DoS attack, two-pass authenticated key agreement protocols (such as, unified model[4] and MQV[6]) are unable to withstand the DoS attack. That is because the client only sends a request message to the server and the server must compute the session key according to the request message. Meanwhile, the server must also validate the client. As a result, on-line computation of modular exponentiations is required.

## 3. Brief Review of the Hirose–Matsuura Protocol

In this section, we briefly review the Hirose–Matsuura protocol.[19] In their system, there exist two large prime numbers $p$ and $q$, where $q$ is a factor of $(p - 1)$. Let $g$ be a generator with order $q$ in $GF(p)$. Let $h$ be a collision-free hash function.[20] The system publishes $p$, $q$, $g$ and $h$, which are shared among all users. Each user $i$ with the identity information $I_i$ selects a secret key $x_i$ in $Z_q$ and computes the corresponding public key $y_i = g^{-x_i} \bmod p$. Assume that the user $A$ is the client and the user $B$ is the server. Thus, $A$ and $B$ carry out the following procedure to generate the session key.

Step 0. The client $A$ selects two random numbers $a_1, a_2 \in Z_q$, and computes $u_A = g^{-a_1} \bmod p$ and $r_A = g^{a_2}$ mod p. The server $B$ also selects two random numbers b$_1$, $b_2 \in Z_q$, and computes $u_B = g^{-b_1} \bmod p$ and $r_B = g^{b_2} \bmod p$.

Step 1. $A$ sends $u_A$ and $I_A$ to $B$.

Step 2. $B$ computes $e_B$ and $w_B$ as follows.

$e_B = h(r_B, u_B, u_A)$

$s_B = b_2 + e_B b_1 + e_B^2 S_B \bmod q$

$B$ also computes $c_B = E_B(b_1, r_B)$, where $E_B(\cdot)$ is a symmetric encryption function of $B$ and $D_B(\cdot)$ is the corresponding decryption function. And $B$ keeps the master key $K$ securely and uses $h(K,t)$ as the secret key of $E_B(\cdot)$, where $t$ is changed every certain period of time.

Step 3. $B$ sends $u_B$, $e_B$, $s_B$, $c_B$ and $I_B$ to $A$.

Step 4. $A$ computes $r_B' = g^{s_B} \cdot u_B^{e_B} \cdot y_B^{e_B^2} \bmod p$, where $r_B'$ should be equal to $r_B$ if $B$ is honest to compute $u_B$, $e_B$, $s_B$ and $c_B$. Then, $A$ checks if $e_B = h(r_B', u_B, u_A)$. If it holds, $A$ computes the following:

$$V = h(r_B', u_A, u_B)$$

$$e_A = h(r_A, u_A, u_B)$$

$$s_A = a_2 + e_A a_1 + e_A^2 x_A \bmod q$$

Step 5. $A$ sends $u_A$, $e_A$, $s_A$, $V$, $u_B$, $e_B$, $c_B$ and $I_A$ to $B$. Meanwhile, $A$ can compute the common key $CK = u_B^{-a_1} \bmod p$.

Step 6. $B$ recovers $(b_1, r_B) = D_B(c_B)$ and checks if $V = h(\mathrm{r}_B, u_A, u_B)$ and $e_B = h(r_B, u_B, u_A)$. If they do not hold, $B$ terminates the execution. Otherwise, $B$ computes $r_A' = g^{s_A} \cdot u_A^{e_A} \cdot y_A^{e_A^2} \bmod p$ and checks if $e_A = h(r_A', u_A, u_B)$. Finally, $B$ can get the common key $CK = u_A^{-b_1} \bmod p$.

## 4. Proposed Authenticated Key Agreement Protocols

This section describes two authenticated key agreement protocols resistant to a DoS attack. The first is a slight modification of the Hirose–Matsuura protocol. As in the Hirose–Matsuura protocol, the improved protocol provides the implicit key confirmation property. The second protocol is a new authenticated key agreement protocol resistant to a DoS attack. It has the explicit key confirmation property and the computation cost can be further reduced.

### —4.1. Improved Hirose–Matsuura Protocol—

In the system, there are two large prime numbers $p$ and $q$, where $q$ is a factor of $(p - 1)$. Let $g$ be a generator with order $q$ in $GF(p)$. Let $h$ be a collision-free hash function. The system publishes $p$, $q$, $g$ and $h$, which are shared among all users. Each user $i$ with the identity information $I_i$ selects a secret key $x_i$ in $Z_q$ and computes the corresponding public key $y_i = g^{x_i} \bmod p$. Assume that the user $A$ is the client and the user $B$ is the server, they own the secret/public key pairs $(x_A, y_A)$ and $(x_B, y_B)$, respectively. Thus, $A$ and $B$ carry out the following steps to generate the session key.

Step 0. The client $A$ selects two random numbers $a_1, a_2 \in Z_q$, and computes $u_A = g^{a_1} \bmod p$ and $r_A = g^{a_1} \cdot g^{-a_2} \bmod p$. The server $B$ also selects two random numbers b$_1$, $b_2 \in Z_q$, and computes $u_B = g^{b_1} \bmod p$ and $r_B = g^{b_1} \cdot g^{-b_2} \bmod p$.

Step 1. $A$ sends $r_A$ and $I_A$ to $B$.

Step 2. $B$ computes $e_B$ and $s_B$ as follows.

$$e_B = h(u_B, r_B, r_A, y_B)$$

$$s_B = b_2 - e_B x_B \bmod q$$

$B$ also computes $c_B = E_B(u_B, b_1)$, where $E_B(\cdot)$ is a symmetric encryption function of $B$ and $D_B(\cdot)$ is the corresponding decryption function. And $B$ keeps the master key $K$ securely and uses $h(K,t)$ as the secret key of $E_B(\cdot)$, where $t$ is changed every certain period of time.

Step 3. $B$ sends $r_B$, $e_B$, $s_B$, $c_B$ and $I_B$ to $A$.

Step 4. $A$ computes $u_B' = r_B \cdot g^{s_B} \cdot y_B^{e_B} \bmod p$. Then, $A$ checks if $e_B = h(u_B', r_B, r_A, y_B)$. If it holds, $A$ computes the following:

$$V = h(u_B', r_A, r_B)$$

$$e_A = h(u_A, r_A, r_B, y_A)$$

$$s_A = a_2 - e_A x_A \mod q$$

Step 5. $A$ sends $V$, $e_A$, $s_A$, $r_A$, $r_B$, $c_B$ and $I_A$ to $B$.

Step 6. $B$ recovers $(u_B, b_1) = D_B(c_B)$ and checks if $V = h(u_B, r_A, r_B)$. If it does not hold, $B$ terminates the execution. Otherwise, $B$ computes $u_A' = r_A \cdot g^{s_A} \cdot y_A^{e_A} \bmod p$ and checks if $e_A = h(u_A', r_A, r_B, y_A)$. Finally, $B$ can get the session key $CK = h((u_A')^{b_1} \bmod p)$. Meanwhile, $A$ can also compute the session key $CK = h((u_B')^{a_1} \bmod p)$.

The improved authenticated key agreement protocol (Improved-AKAP) is depicted in Figure 1. The improved protocol provides the implicit key confirmation using a signature scheme that is a variation of the Nyber–Ruppel signature scheme with message recovery.[21] In Step 3, the message $(r_B, e_B, s_B)$ is regarded as a signature that is used to recover $u_B$. By computing $u_B' = r_B \cdot g^{s_B} \cdot y_B^{e_B} \bmod p$ and then checking $e_B = h(u_B', r_B, r_A, y_A)$, $A$ is able to ensure that $B$ received $r_A$ and knows $u_B$ and $b_1$. Thus, $A$ is able to ensure that $B$ can compute the session key $CK$.

*Security analysis*—Let us discuss the security properties described in Section 1 as follows.

1. *Known-key security*. If the session key $CK$ is disclosed, an adversary is unable to obtain the value $g^{a_1 b_1}$ because it is protected by a hash function. Even though $g^{a_1 b_1}$ is disclosed,

| Step | A | | B |
|---|---|---|---|
| 0 | $a_1, a_2 \in_R Z_q$<br>$u_A = g^{a_1} \bmod p$<br>$r_A = g^{a_1} \cdot g^{-a_2} \bmod p$ | | $b_1, b_2 \in_R Z_q$<br>$u_B = g^{b_1} \bmod p$<br>$r_B = g^{b_1} \cdot g^{-b_2} \bmod p$ |
| 1 | | $\Longrightarrow (r_A, I_A) \Longrightarrow$ | |
| 2 | | | $e_B = h(u_B, r_B, r_A, y_B)$<br>$s_B = b_2 - e_B \cdot x_B \bmod q$<br>$c_B = E_B(u_B, b_1)$ |
| 3 | | $\Longleftarrow (r_B, e_B, s_B, c_B, I_B) \Longleftarrow$ | |
| 4 | $u_B' = r_B \cdot g^{s_B} \cdot y_B^{e_B} \bmod p$<br>? $e_B = h(u_B', r_B, r_A, y_B)$<br>$V = h(u_B', r_A, r_B)$<br>$e_A = h(u_A, r_A, r_B, y_A)$<br>$s_A = a_2 - e_A x_A \bmod q$ | | |
| 5 | | $\Longrightarrow (V, e_A, s_A, r_A, r_B, c_B, I_A) \Longrightarrow$ | |
| 6 | $CK = h((u_B')^{a_1} \bmod p)$ | | $(u_B, b_1) = D_B(c_B)$<br>? $V = h(u_B, r_A, r_B)$<br>$u_A' = r_A \cdot g^{s_A} \cdot y_A^{e_A} \bmod p$<br>? $e_A = h(u_A', r_A, r_B, y_A)$<br>$CK = h((u_A')^{b_1} \bmod p)$ |

Figure 1. Improved authenticated key agreement protocol (Improved AKAP)

the protocol may also withstand the known-key attack, this is because $a_1$ and $b_1$ are selected randomly in each session. Thus, knowing $g^{a_1b_1}$ is no value to deriving the new session keys in future sessions.

2. *Forward secrecy*. If both secret keys of $A$ and $B$ are disclosed, the adversary is unable to compute $a_1$ and $b_1$ from $s_A$ or $s_B$ because neither $s_A$ nor $s_B$ include the values. Suppose that the adversary directly tries to find $a_1$ and $b_1$ from $u'_A$ and $u'_B$, respectively. This is equivalent to solving the discrete logarithm problem.[11] That is, the adversary knows both secret keys of $A$ and $B$, but is unable to derive any old session keys established by the two principals.
3. *Key-compromise impersonation*. Suppose that the secret key of $B$ is disclosed. An adversary that knows this secret key tries to impersonate some entity $A$ to $B$. Impersonating $A$ requires computing the signature message $(r_A,e_A,s_A)$ using the secret key $x_A$ of $A$. In this case impersonating $A$ to $B$ is impossible. Therefore, the proposed protocol can withstand the key-compromise impersonation attack.
4. *Unknown key-share*. Because this protocol provides the implicit key confirmation property, it can withstand the unknown key-share attack.[19]

## —4.2. New Authenticated Key Agreement Protocol—

In this subsection, we propose a new authenticated key agreement protocol. The initialization is the same as that of the improved protocol. The new authenticated key agreement protocol (New-AKAP) is depicted in Figure 2. The detailed steps are presented as follows.

Step 0. The client $A$ selects a random number $a \in Z_q$, and computes $r_A = g^a \bmod p$. The server $B$ also selects a random number, $b \in Z_q$, and computes $r_B = g^b \bmod p$, $s_B = b + x_B \cdot h(r_B,y_B) \bmod q$ and $M_B = g^{s_B} \bmod p$.

Step 1. $A$ sends $I_A$ to $B$.

Step 2. $B$ computes $c_B = E_B(r_B,M_B,s_B)$, where $E_B(\cdot)$ is a symmetric encryption function of $B$ and $D_B(\cdot)$ is the corresponding decryption function. And $B$ keeps the master key $K$ securely and uses $h(K,t)$ as the secret key of $E_B(\cdot)$, where $t$ is changed every certain period of time.

Step 3. $B$ sends $r_B$, $c_B$ and $I_B$ to $A$.

Step 4. $A$ computes $M'_B = r_B \cdot y_B^{h(r_B,y_B)} \bmod p$. Then, $A$ computes the following:

$$V = h(M'_B, r_A, r_B, y_A)$$

$$s_A = a + x_A \cdot V \bmod q$$

$$M_A = g^{s_A} \bmod p$$

$$e = h(M_A, r_A)$$

$$K = (M'_B)^{s_A} \bmod p$$

$$T_A = h(r_A, r_B, K)$$

Step 5. $A$ sends $V$, $T_A$, $r_A$, $e$, $c_B$ and $I_A$ to $B$.

Step 6. $B$ recovers $(r_B,M_B,s_B) = D_B(c_B)$ and checks if $V = h(M_B,r_A,r_B,y_A)$. If it does not hold, $B$ terminates the execution. $B$ computes $M'_A = r_A \cdot y_A^V \bmod p$ and checks if $e = h(M'_A,r_A)$. If it does not hold, $B$ terminates the execution. Otherwise, $B$ computes $K = (M'_A)^{s_B} \bmod p$ and checks if $T_A = h(r_A,r_B,K)$. Finally, $B$ computes $T_B = h(r_B,r_A,K)$.

Step 7. $B$ sends $T_B$ to $A$.

Step 8. $A$ checks $T_B = h(r_B,r_A,K)$. Then, $A$ computes the session key $CK = h(K)$. $B$ also computes the session key $CK = h(K)$.

Note that both $A$ and $B$ may obtain the session key $CK$ since $K = (M'_B)^{s_A} \bmod p = g^{s_A s_B} \bmod p = (M'_A)^{s_B} \bmod p$. Moreover, $B$ checks the equation $T_A = h(r_A,r_B,K)$ in Step 7 and $A$ checks the equation $T_B = h(r_B,r_A,K)$ in Step 8, which may provide the security goal of explicit key confirmation.

*Security analysis*—In the following, we discuss our new protocol and how to hold the security properties described in Section 1.

1. *Known-key security*. If the session key $CK$ is disclosed, an adversary is unable to obtain the value $K$ because it is protected by a hash function. Suppose that the adversary has obtained a value $K_1$ established between $A$ and B. Since $K_1 = (M'_{B1})^{s_{A1}} \bmod p = g^{s_{A1}s_{B1}} \bmod p = (M'_{A1})^{s_{B1}} \bmod p$, we have

| Step | A | | B |
|---|---|---|---|
| 0 | $a \in_R Z_q$<br>$r_A = g^a \bmod p$ | | $b \in_R Z_q$<br>$r_B = g^b \bmod p$<br>$s_B = b + x_B \cdot h(r_B, y_B) \bmod q$<br>$M_B = g^{s_B} \bmod p$ |
| 1 | | $==> (I_A) ==>$ | |
| 2 | | | $c_B = E_B(r_B, M_B, s_B)$ |
| 3 | | $<== (r_B, c_B, I_B) <==$ | |
| 4 | $M_B' = r_B \cdot y_B^{h(r_B, y_B)} \bmod p$<br>$V = h(M_B', r_A, r_B, y_A)$<br>$s_A = a + x_A$ ?$V \bmod q$<br>$M_A = g^{s_A} \bmod p$<br>$e = h(M_A, r_A)$<br>$K = (M_B')^{s_A} \bmod p$<br>$T_A = h(r_A, r_B, K)$ | | |
| 5 | | $==> (r_A, V, e, T_A, c_B, I_A) ==>$ | |
| 6 | | | $(r_B, M_B, s_B) = D_B(c_B)$<br>? $V = h(M_B, r_A, r_B, y_A)$<br>$M_A' = r_A \cdot y_A^V \bmod p$<br>? $e = h(M_A', r_A)$<br>$K = (M_A')^{s_B} \bmod p$<br>? $T_A = h(r_A, r_B, K)$<br>$T_B = h(r_B, r_A, K)$ |
| 7 | | $<== (T_B) <==$ | |
| 8 | ? $T_B = h(r_B, r_A, K)$<br>$CK = h(K)$ | | $CK = h(K)$ |

Figure 2. New authenticated key agreement protocol (New AKAP)

$$\begin{aligned} K_1 &= g^{s_{A1} s_{B1}} \bmod p \\ &= g^{(a_1 + x_{A1} V_1)(b_1 + x_{B1} h(r_{B1}, y_B))} \bmod p \\ &= g^{a_1 b_1 + x_A V_1 b_1 + a_1 x_B h(r_{B1}, y_B) + x_A x_B V_1 h(r_{B1}, y_B)} \bmod p \end{aligned}$$

Suppose that there is another value $K_2$ established by $A$ and $B$. For the same reason, we have $K_2 = g^{a_2 b_2 + x_A V_2 b_2 + a_2 x_B h(r_{B2}, y_B) + x_A x_B V_2 h(r_{B2}, y_B)} \bmod p$. Suppose that the adversary knows $K_1$, but is unable to compute $s_A$ or $s_B$ because this is equivalent to solving the discrete logarithm problem, and the adversary also obtains no information (such as, $a_1 b_1$, $x_A x_B$, $x_A b_1$ and $x_B a_1$) from $K_1$. Certainly, the adversary does not find another session key $K_2$ from $K_1$. Therefore, the proposed protocol can withstand the known-key attack.

2. *Forward secrecy*. If both secret keys of $A$ and $B$ are disclosed, the adversary tries to find $s_A$ or $s_B$ and then computes $K = (M_B')^{s_A} \bmod p$ or $K = (M_A')^{s_B} \bmod p$. However, to find $s_A$ or $s_B$ you must derive $a$ and $b$ from $r_A$ or $r_B$, respectively. In this case, this will be equivalent to solving the discrete logarithm problem.

Therefore, the proposed protocol provides forward secrecy.

3. *Key-compromise impersonation*. Suppose that the secret key of $B$ is disclosed. An adversary that knows this secret key tries to impersonate some entity $A$ to $B$. Impersonating $A$ requires computing $s_A$ that must be computed using the secret key $x_A$ of $A$. In this case impersonating $A$ to $B$ is impossible. Therefore, the proposed protocol can withstand the key-compromise impersonation attack.
4. *Unknown key-share*. Because this protocol provides the explicit key confirmation property, it can withstand the unknown key-share attack.[14]

## 5. Discussion and Performance Comparisons

In this section, we discuss the proposed protocols on how to resist the DoS attack. Performance comparisons between the proposed protocols and the Hirose–Matsuura protocol are presented.

First, the storage-exhaustion attack is considered. In the improved protocol, the values $(I_A,r_A,u_B,b_1)$ are the connection state. Since $u_B$ and $b_1$ are encrypted and transmitted to the client $A$, then $A$ forwards these values to $B$ in Step 5. Thus, the values $u_B$ and $b_1$ need not be maintained by the server $B$. Therefore, this protocol can withstand the storage-exhaustion attack. By checking if $V = h(u_B,r_A,r_B)$ in Step 6, $B$ can confirm that $r_A$ and $r_B$ are used to compute the correct $u_B$ because of $e_B = h(u_B,r_B,r_A,y_B)$ and $u_B = r_B \cdot g^{s_B} \cdot y_A{}^{e_B}$. The above technique is called the stateless connection for the server.

As for the new proposed protocol, we also adopt the same technique. The values $s_B$ and $r_B$ are the connection state. Because $s_B$ and $r_B$ are encrypted and transmitted to the client $A$, then $A$ forwards these to $B$ in Step 5, in this case the values $s_B$ and $r_B$ need not be kept by the server $B$. Therefore, the new protocol can withstand the storage-exhaustion attack. By checking if $V = h(M_B,r_A,r_B,y_A)$ in Step 6, $B$ can confirm that $B$ has computed the correct $M_B$.

In the following, let us consider the CPU-exhaustion attack, in which the server need not keep the connection state. The computation cost of modular exponentiations is considered as the main factor because it is a time-consuming computation. Three kinds of requests to a server are analyzed. One is a valid request. Another is an invalid request with a last message, which is called a minor invalid request. The other is an invalid request without a final message. We call it a major invalid request, that is, the client sends the first message to the server to perform the CPU-exhaustion attack.

***The computation cost of modular exponentiations is considered as the main factor because it is a time-consuming computation.***

In our improved protocol, computing $u_B$ and $r_B$ in Step 0 requires two modular exponentiations and it is able to be pre-computed. That is, it can be computed off-line. When the client has computed a valid $u_B'$, they are able to compute the correct $V$ in Step 4. Thus, the client sends the messages $(V,e_A,s_A,r_A,r_B,c_B,I_A)$ to the server in Step 5. Therefore, when the client spends two modular exponentiations to compute the correct $V$, the checking for $V = h(u_B,r_A,r_B)$ to the server is valid. Then, the server must compute $u_A' = r_A \cdot g^{s_A} \cdot y_A{}^{e_A} \bmod p$ that requires two modular exponentiations. If the client is a legal client, the server can compute the correct $u_A'$. Then, the checking for $e_A = h(u_A',r_A,r_B,y_A)$ is valid and the server must compute $CK = h((u_A')^{b_1} \bmod p)$ that requires one modular exponentiation. Finally, we summarize the above descriptions as follows. In a valid request, it requires five modular exponentiations to the server. In a major invalid request, a malicious client only sends $(r_A,I_A)$ to the server, then two modular exponentiations are required to compute $u_B$ and $r_B$ to the server. For a minor invalid request, a malicious client computes the correct $V$ and sends it to the server, four modular exponentiations are required. In addition, considering the off-line pre-computation, the required modular exponentiations are three, two and zero for the valid request, the minor invalid request and the major invalid request, respectively.

In our new protocol, when a client sends a request to the server, the server requires two modular exponentiations to compute $r_B$ and $M_B$ in Step 0. When the client has computed $M_B'$ and the correct $V$ in Step 4. Thus, the client enables the server to check $V = h(M_B, r_A, r_B, y_A)$ and compute $M'_A = r_A \cdot y_A^V \bmod p$ that requires one modular exponentiation. Meanwhile, a malicious client may compute $M_A = r_A \cdot y_A^V \bmod p$ for passing the check of $e = h(M'_A, r_A)$ on the server. Thus, this enables the server to compute $K = (M'_A)^{s_B} \bmod p$, which requires one modular exponentiation. Finally, we also summarize the above descriptions as follows. In a valid request, it requires four modular exponentiations to the server. In a major invalid request, a malicious client only sends a request to the server, then two modular exponentiations are required to compute $r_B$ and $M_B$ to the server. For a minor invalid request, a malicious client computes the correct $V$ and $e$, and sends these values to the server, four modular exponentiations are required. Note that $r_B$ and $M_B$ in Step 0 can be computed off-line. Therefore, considering the off-line precomputation, the required modular exponentiations are two, two and zero for the valid request, the minor invalid request and the major invalid request, respectively.

The number of modular exponentiations required by a server for three kinds of requests among the Hirose–Matsuura protocol and our proposed protocols is given in Table 1. Table 2 presents the number of on-line modular exponentiations required by a server and a client for three kinds of requests. Obviously, our proposed protocols have better performance than the Hirose–Matsuura protocol for withstanding the CPU-exhaustion attack. Both the Hirose–Matsuura protocol and the improved protocol provide the property of implicit key confirmation with requiring three communication steps. Although the new proposed protocol requires four communication steps, this protocol provides explicit key confirmation and the computation cost can be further reduced.

From Table 2, we know that the computation cost on the server side is less than or equal to that on the client side. If a client with limited resources wants to deny a powerful server, the attack cannot succeed. When a large number of clients simultaneously launch thousands of requests at the target server, this attack actually is difficult to withstand. A notable example is the Yahoo and Amazon web sites attacked by a large number of compromised agent clients in February 2000. This attack crippled their services. The proposed protocols cannot fully disallow a DoS/DDoS attack, but they can reduce the effect of such an attack and thus make it more difficult to succeed. Our protocols increase the capability for resisting this attack. A server requires some resources to provide services for legitimate users including network connectivity between users and the server, server memory space and CPU time. DoS/DDoS attacks have many types: consumption of network bandwidth, exhaustion of server resources, and configuration information destruction. Here, we address only

| Request\Protocol | HM-AKAP | Improved AKAP | New AKAP |
|---|---|---|---|
| Valid request | 6 | 5 | 4 |
| Minor invalid request | 5 | 4 | 4 |
| Major invalid request | 2 | 2 | 2 |
| Communication steps | 3 | 3 | 4 |
| Key confirmation | Weak | Weak | Strong |

Table 1. The number of modular exponentiations required for a server for three kinds of requests, between the Hirose–Matsuura protocol and our proposed protocols

| Request\Protocol | HM-AKAP | | Improved AKAP | | New AKAP | |
|---|---|---|---|---|---|---|
| | Client | Server | Client | Server | Client | Server |
| Valid request | 4 | 4 | 3 | 3 | 3 | 2 |
| Minor invalid request | 3 | 3 | 2 | 2 | 2 | 2 |
| Major invalid request | 0 | 0 | 0 | 0 | 0 | 0 |

Table 2. The number of *on-line* modular exponentiations required for a server and a client for three kinds of requests

the resource-exhaustion problem occurring in authentication and key agreement protocols. For defenses against DoS/DDoS attacks, many prevention and detection mechanisms must also be involved in the system, such as router filters.

## 6. Conclusions

We have proposed two authenticated key agreement protocols resistant to a denial-of-service attack. Although the proposed protocols cannot fully resist a DoS/DDos attack, both protocols reduce the effect of such an attack and thus make it more difficult to succeed. By reducing the computation cost on the server side relative to the client side and using the stateless connection technique, both proposed protocols are able to simultaneously resist both the CPU-exhaustion attack and the storage-exhaustion attack. One is an improved protocol on the Hirose–Matsuura key agreement protocol. The proposed protocol reduces the computation cost compared with the Hirose–Matsuura protocol. The other new protocol provides explicit key confirmation and further reduces the computation cost. Our proposed protocols are efficient in resisting the CPU-exhaustion attack. Several design principles resistant to a DoS attack were also presented for designing an authenticated key agreement protocol or authentication protocol. These design principles increase the system's capability to resist DoS attacks.

## References

1. Needham RM. Denial of services: An example. *Communications of ACM* 1994; **37**(11):42–46.
2. Leiwo J, Aura T, Nikander P. Towards network denial of service resistant protocols. *Proceedings of the Sixteenth Annual Working Conference on Information Security*, IFIP Series, Vol.175, Beijing, China, 2000.
3. Diffie W, Hellman ME. New directions in cryptography. *IEEE Transactions on Information Theory* 1976; IT-**22**(6):644–654.
4. Ankney R, Johnson D, Matyas M. *The Unified Model*. Contribution to ANSI X9F1, 1995.
5. Lee WB, Chang CC. Integrating authentication in public key distribution system. *Information Processing Letters* 1996; **57**:49–52.
6. Menezes AJ, Qu M, Vanstone SA. Some key agreement protocols providing implicit authentication. *2nd Workshop Selected Areas in Cryptography*, 1995.
7. Tseng YM. Multi-party key agreement protocols with cheater identification. *Applied Mathematics and Computation* 2003; **145**(2–3):551–559.
8. Tseng YM. On the security of an efficient two-pass key agreement protocol. *Computer Standards and Interfaces* 2004; **26**(4):371–374.
9. Jablon DP. Extended password key exchange protocols. *WETICE Workshop on Enterprise Security* 1997; 248–255.
10. Kwon T, Song J. Secure agreement scheme for $g^{xy}$ via password authentication. *Electronics Letters* 1999; **35**(11):892–893.
11. ElGamal T. A public key cryptosystem and signature scheme based on discrete logarithm. *IEEE Transactions on Information Theory* 1985; **31**(4):469–472.
12. NIST. Digital signature standard (DSS), FIPS PUB XX, 1993.
13. Blake-Wilson S, Johnson D, Menezes A. Key agreement protocols and their security analysis. *Proceedings of the Sixth IMA International Conference on Cryptography and Coding*, Lecture Notes in Computer Science 1355, 1997; 30–45.
14. Blake-Wilson S, Menezes A. Authenticated Diffie–Hellman key agreement protocols. *Proceedings of the 5th Annual Workshop on Selected Areas in Cryptography (SAC '98)*, Lecture Notes in Computer Science 1556, 1999; 339–361.
15. Diffie W, Van Oorschot PC, Wiener MJ. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography* 1992; **2**:107–125.
16. ITSEC. *Information Technology Security Evaluation Criteria*. Version 1.2, COM(92) 298 final, Brussels, 1992.
17. Dwork C, Naor M. Pricing via processing or combating junk mail. *Advances in Cryptology—Crypto'92*, Lecture Notes in Computer Science 740, Springer-Verlag, 1993; 139–147.
18. Aura T, Nikander P. Stateless connections. *Information and Communications Security (ICICS'97)*, Lecture Notes in Computer Science 1334, 1997; 87–97.
19. Hirose S, Matsuura K. Key agreement protocols resistant to a denial-of-service attack. *IEICE Transactions on Information and Systems* 2001; E-**84**-D(4):477–484.
20. Dobbertin H. The status of MD5 after a recent attack. *CryptoBytes* 1996; **2**(2):1–6.
21. Nyberg K, Rueppel RA. Message recovery for signature schemes based on the discrete logarithm. *Designs Codes and Cryptography* 1996; **7**(1–2): 61–81. ■