

Security and Efficiency in Authentication Protocols Resistant to Password Guessing Attacks

Tae-kyung Kwon Jooseok Song

*Department of Computer Science
Yonsei University, Seoul 120-749, Korea
ktk@emerald.yonsei.ac.kr*

Abstract

Cryptographic protocols for authentication and key exchange are necessary for secure communications. Most protocols have assumed that a strong secret for authentication should be shared between communicating participants in the light of a threat of dictionary attacks. But a user-chosen weak secret, i.e. password, is typically used for authentication. Since most users want to use an easily memorizable password, which tends to be easy to guess, several authentication protocols that protect such a weak secret from password guessing attacks, have been developed. However, those security-oriented protocols are more expensive in terms of the number of random numbers, cipher operations, and protocol steps than the previous protocols which are not resistant to guessing attacks. We propose new authentication and key exchange protocols, which are efficient considerably in protecting a poorly-chosen weak secret from guessing attacks.

1 Introduction

To achieve secure data communications, participants should be authenticated when setting up a communication session and a new session key to be used for encrypting data must be shared between authenticated participants. How to authenticate participants is a chronic problem of computer network security. For authentication and key distribution, a cryptographic protocol is necessary. Since the new session key is utilized to encrypt or decrypt the information of communication participants, the cryptographic protocol is very important. Such a protocol is typically an initial step when a communication session is set up. Therefore, it is critical for overall security of the communications. Furthermore, most of the important properties of the protocol do not depend on the underlying cipher systems, but rather on the structure of the messages and procedures of the protocol. That means the protocol should be designed and verified

carefully.

In most systems, user-chosen secrets, i.e. passwords, are used for authentication. Clearly, cryptographically long passwords or long secrets chosen at random would be better for security only if ordinary users could remember them. But most users want to use a small secret and choose an easy-to-remember password because well-chosen passwords are also quite unmemorable [5, 6]. Such a user-chosen secret is called a poorly-chosen secret or a weakly-shared secret because it is easy to be guessed by others [8]. The weak secret makes authentication protocols vulnerable to guessing attacks.

While, in terms of security, most of the previous protocols are vulnerable to guessing attacks, some protocols [7, 8, 9] to prevent them are more expensive in terms of efficiency. Following the novel approach of [9], we propose new authentication and key exchange protocols which are efficient in protecting weak secrets from guessing attacks using a one-time pad and a strong one-way hash function. We apply a one-time pad and a strong one-way hash function to our cryptographic protocol for security and efficiency. Our protocol is more efficient than other related protocols [7, 8, 9].

In the rest of this section, notations to be used are summarized. In the next section, we describe authentication protocols and guessing attacks. In section 3, we define guessing attacks more clearly and show how to defeat them efficiently. In section 4, we introduce our basic protocol which ensures both security and efficiency, and analyze it using a formal logic. Basic protocol was also proposed in our previous work [1]. Section 4 also introduces advanced protocols which are modified versions of basic protocol. Section 5 discusses security of our protocols. Section 6 compares our protocols with other related protocols and the final section concludes this paper.

1.1 Notations

To describe our protocol clearly we summarize notations to be used. A and S are system principals of communication participants. A shared secret and a public key of A are denoted as P_A and K_A , respectively. The weakly-shared secret P_A corresponds to password of user A . K_S stands for a public key of S and K for a new session key. R_x^i means an i -th random value of principal x . For instance, R_A^2 is the second random value of principal A and R_S is a unique random value of principal S . R_x^i also corresponds to the nonce for a session. The nonce means a random value which is used only for a corresponding session. $[M]_K$ denotes a ciphertext of message M under an encryption key K . $H()$ is a strong one-way hash function and $|X|$ a bit-wise length of X . \oplus and \parallel represent a bit-wise XOR operator and a concatenator, respectively. $A \rightarrow S : M$ denotes that A sends a message M to S . OP means a one-time pad.

2 Guessing Attacks on Authentication Protocols

Since the landmark protocol which uses encryption to achieve authenticated communications, was shown in [3], a lot of protocols have been proposed and used in many network systems. Security in them is based on a pre-shared secret value. Thus, most of the classical protocols assumed that a strong remote authentication is possible only if a cryptographically long or random secret is shared between the participants, in the light of an eternal threat of a dictionary attack or guessing attack [4]. But in most systems, user-chosen secrets, i.e. passwords, are used for authentication. Clearly, cryptographically long passwords or long secrets chosen at random would be better for security only if ordinary users could remember them. But most users want to use a small secret and choose an easy-to-remember password because well-chosen passwords are also quite unmemorable [5, 6]. Such a user-chosen secret is called a poorly-chosen secret or a weakly-shared secret because it is easy to be guessed by others [8]. The long secret chosen at random by the system is to be kept in some external storage. Therefore, it would result in other serious problems and couldn't be accepted wide. Otherwise, additional hardware components which are not ubiquitous, are required. They are not the points to be considered in this paper. It is certain that, for convenience of users, the use of weak secret is inevitable in most systems.

Guessing attack, which is performed through an iterative guessing and verification, on the weak secret is surprisingly successful [8]. The goal of guessing attack is to find out the shared weak secret. If the shared

secret is known to others, it is clear that secure communication is impossible. Such attacks are classified into off-line and on-line methods.

- *Off-line guessing attacks*: An attacker eavesdrops protocol messages and store them in a local storage for verification. Therefore, any other participants cannot notice the off-line attack. After guessing a password included in the eavesdropped messages, the attacker tries to verify the guessed password iteratively in off-line manners. A common method is to compare the eavesdropped messages with the artificial messages which are reconstructed by using the guessed password [8, 10].
- *On-line guessing attacks*: An attacker should access to another participant who shares the weak secret legitimately. Therefore, it is not difficult to notice the on-line guessing attack. After guessing a target password, the attacker tries to use a guessed password iteratively in on-line manners, for instance by replaying eavesdropped messages or by impersonating other users with the guessed password. For impersonation, the attacker should construct a bogus message through the guessed password. Using the messages resulted from on-line guessing attack, the attacker can proceed with guessing in off-line manners. Unless any other participants could find authentication failures of attackers, the on-line attack couldn't be noticed by others [11].

In terms of security, most of the previous protocols are not sufficient. Therefore, many protocols to prevent the guessing attack have been explored and studied. [7] proposes a five step protocol which is an augmented version of their previous work, the Encrypted Key Exchange, to resist the Denning-Sacco attack [12]. [8] describes both three party and two party protocols. Authors introduced a confounder which is a sufficiently large random number. [9] shows a three step protocol called optimal protocol. But all of them are more expensive in terms of the computation and communication costs than the previous protocols. In terms of efficiency, they are not sufficient while they satisfy security. More efficient protocol resistant to guessing attack is required earnestly for communication networks [1]. We introduce our protocols as solutions for security and efficiency in authentication.

3 How to Defeat Guessing Attacks

3.1 Defining Guessing Attacks

Guessing attacks could be known-plaintext attacks or verifiable-text attacks [10]. Guessing attacks explore

the fact that the weak secret is usually chosen from a relatively small space. Theoretically, a bit-wise length $|P_A|$ can make $2^{|P_A|}$ different passwords. But in practice, the $2^{|P_A|}$ space is reduced considerably because users have to use an alphabetic keyboard or a numeric keypad to choose them. The worse is that the space is reduced more since users choose each password from a remindful word space. Therefore, the weak secret P_A cannot help having much bit redundancy. It means that P_A is chosen from a smaller space than $2^{|P_A|}$. Let $|G(P_A)|$ be the bit length reduced to represent a practical P_A without the bit redundancy. We can say that the real P_A is chosen from $2^{|G(P_A)|}$ space. It means that an entropy to determine P_A is close to $|G(P_A)|$ rather than to $|P_A|$. Therefore, at most $2^{|G(P_A)|}$ operations are necessary to find P_A by a brute force search in effect. Here we have $2^{|G(P_A)|} \ll 2^{|P_A|}$. For example, 8 character long password is general for authentication. Thus, the ordinary maximum size of P_A is said to be 64 bits. But the 64 bits have much bit redundancy and 2^{64} space may be reduced considerably. While it is very difficult to perform guessing attacks within $2^{|P_A|}$ complexity for 64 bit long P_A , it is computable within $2^{|G(P_A)|}$. Therefore, to prevent guessing attacks in authentication protocols, a complexity to guess P_A should be made over $2^{|P_A|}$ at least. But, for enlarging the complexity, the number of random values, cryptographic operations or protocol steps has been made abnormally large in other related protocols [7, 8, 9]. However, using a one-time pad and a strong one-way hash function, we solve the efficiency problem. [1] explains several requirements for making a protocol efficient and resistant to guessing attacks.

3.2 Defeating Guessing Attacks

Our basic idea to defeat guessing attacks efficiently is very simple. It is summarized that:

- Never use a weak secret as an encryption key for messages that include verifiable data. [10] explains verifiable-text attacks deliberately. If the weak secret is used as an encryption key for those messages, additional random values such as a confounder have been required in other works [8, 9]. However, for messages which do not include verifiable data, the weak secret can be used as an encryption key. Actually, we encrypt a new public key under the weak secret in section 5.2. The public key is so used carefully as to make other data unverifiable in our protocol.
- Mask the weak secret with nonce in a message for authentication. For example, $[R_A, P_A \oplus R_A]_{K_S}$ would be better than $[R_A, [R_A]_{P_A}]_{K_S}$ in terms of

efficiency. To reconstruct $[R_A, P_A \oplus R_A]_{K_S}$ using guessed P_A , at least $2^{|R_A|}$ complexities are required.

- Use a one-time pad to encrypt a new session key rather than a conventional block cipher algorithm. The one-time pad must be chosen at random and it can never be reused. The length of it must be equal to that of message which is going to be encrypted. While encryption is to XOR the message with the one-time pad, decryption is to XOR the ciphertext with the same one-time pad. It provides very simple and secure encryption. The one-time pad system is the only cryptosystem that achieves perfect secrecy [17]. It means that the ciphertext yields no possible information about the plaintext except its bit-wise length. We define that OP should be made by XORing two random values of corresponding principal, i.e. $R_A^1 \oplus R_A^2$, in our protocol. Since R_A^1 and R_A^2 correspond to nonce values for a session, they can be utilized to make a one-time pad for that session. Since we use a one-time pad to encrypt a new session key, the size of OP must be equal to that of K , and it cannot be reused for another session. $OP \oplus K$ would be better than $[K]_{OP}$ in terms of efficiency.
- Use a strong one-way hash function to exchange Challenge-Response messages for confirming a new session key. Challenge-Response messages are necessary for each participant to make a counterpart confirm that the new key is exchanged securely. In other protocols, a conventional block cipher algorithm has been used for the purpose. In general, the hash function is more efficient than the conventional cipher algorithm. The other advantage of the hash function is that it produces an output in a same fixed length regardless of the length of its input. For example, MD5 is always for 128 bit long output and SHA for 160 bit long output.
- Minimize the number of random values. For example, we introduce a new term F_A in section 5. We call F_A a partial value that is a partial bit stream of corresponding message. We explain the partial value in section 5.5.

We can say that a main goal of our idea is to reduce the number of random values and protocol steps, the amount of encryption, and the size of each message in defeating the guessing attacks. It means that we

attempt to provide both security and efficiency in authentication protocols.

4 Secure and Efficient Protocol

4.1 Basic Protocol : two-party K1P

We introduce our basic protocol which is efficient in defeating guessing attacks. This protocol was called two-party K1P(K-on Protocol) in our previous work[1]. Since several advanced protocols will be proposed on the basis of two-party K1P, we call this the basic protocol:

1. $A \rightarrow S : [R_A^1 \parallel R_A^2 \parallel P_A \oplus R_A^1]_{K_S}$
2. $S \rightarrow A : OP \oplus K \parallel H(P_A \oplus R_A^1 \parallel K \parallel R_A^2)$ (1)
3. $A \rightarrow S : H(P_A \oplus R_A^2 \parallel K \parallel R_A^1)$

It is assumed that a weak secret of A , P_A , is shared between A and S . In step 1, A generates two random values R_A^1 and R_A^2 , masks P_A with R_A^1 , and encrypts them using S 's public key K_S . A sends the message 1 to S . After decryption, S gets the random values and the masked value, and uses them to authenticate A and to check whether the message is readable. By calculating $R_A^1 \oplus (P_A \oplus R_A^1)$ from the message and by comparing the result with P_A of local memory, S can authenticate A and check readability of the message. Each participant makes a same one-time pad OP by XORing the random values, i.e. $R_A^1 \oplus R_A^2$. The size of OP should be equal to that of a session key as we mentioned in section 3.2. In step 2, S generates a new session key K and encrypts it using OP . S also computes a hash value using the known values as shown above. The random values endow a randomness on the input image of $H()$ and serve as nonce values. S sends the message 2 to A . After decryption, A gets and uses K to compute hash value along with $P_A \oplus R_A^1$ and R_A^2 . Comparing the hash values, A can verify the integrity and freshness of K and confirm the authentication result. All of them are based on the freshness and readability of the random values made by A itself. In step 3, A replies with a new hash value to inform an admission of K . Hash values of message 2 and 3 also serve as a Challenge-Response for K . A distinctive feature of our protocol is that the weak secret is not used as a temporal cryptographic key for exchanging a session key.

4.2 Formal Analysis of Basic Protocol

Researchers have found errors and flaws in seemingly secure cryptographic protocols proposed many years ago. Therefore, systematic methods are necessary for proving security of protocols. These formal approaches are classified into four classes[14]. The first

of these is to model and verify the protocol using general specification languages. But this approach fails to detect many flawed protocols because it attempts to prove a correctness only. The second is to develop expert systems which a protocol designer can use to try out different scenarios. But it neither guarantees security nor provides techniques for developing attacks. The third approach is to model the requirements of a protocol formally using logics developed for the analysis of knowledge and belief. Since the first formal logical technique (the BAN logic) for the analysis of authentication protocols was introduced in [15], several variants of the logic have been proposed and applied to the protocol analysis[16], for instance, GNY, AT, SVO, and etc. In the meantime, BAN-like logics have become the most widely used formal method in the analysis of cryptographic protocols despite its well-known limitations such as their lack of a well-defined semantics. The fourth is to develop a formal model based on the algebraic term-rewriting properties of cryptographic systems. All the details are described well in [14]. In the meantime, BAN-like logics have become the most widely used formal method in the analysis of cryptographic protocols despite its well-known limitations. We use the GNY logic[16] that is a variant of the BAN logic for analyzing our protocol.

First of all, each message of our protocol should be presented in an idealized form through the GNY's parser algorithm. The parser algorithm would produce the following description of the protocol:

1. $S \triangleleft: \{ *R_A^1, *R_A^2, *(\langle P_A \rangle \oplus *R_A^1)_{+K_S} \}$
 $\sim A \mid \equiv A \xrightarrow{P_A} S$
2. $A \triangleleft: *(R_A^1 \oplus R_A^2 \oplus *K) \sim S \mid \equiv A \xrightarrow{K} S,$ (2)
 $*H(\langle P_A \rangle \oplus R_A^1 \parallel K \parallel R_A^2) \sim S \mid \equiv A \xrightarrow{K} S$
3. $S \triangleleft: *H(\langle P_A \rangle \oplus R_A^2 \parallel K \parallel R_A^1) \sim A \mid \equiv A \xrightarrow{K} S$

A weak secret P_A used for identification purposes is denoted $\langle P_A \rangle$ for applying the GNY's message interpretation rules.

We assume that the following holds at the beginning of every run of the protocol:

$$\begin{aligned} A \ni P_A; A \ni R_A^1; A \ni R_A^2; A \mid \equiv \sharp(R_A^1); A \mid \equiv \sharp(R_A^2); \\ A \mid \equiv \phi(P_A); A \mid \equiv \phi(R_A^1); A \mid \equiv \phi(R_A^2); A \mid \equiv \overset{+K_S}{\phi} S; \\ A \mid \equiv A \xrightarrow{P_A} S; A \mid \equiv S \Rightarrow S \mid \equiv *; A \mid \equiv S \Rightarrow (A \xrightarrow{K} S); \end{aligned}$$

That is, A possesses his or her password and believes it is a secret between him(her)self and S . A also possesses new random numbers, and believes their freshness and recognizability. The jurisdiction of S over a

new session key is also believed by A .

$$\begin{aligned} S \ni P_A; S \ni -K_S; S \ni K; S \models \phi(P_A); S \models \phi(K); \\ S \models \#(K); S \models A \stackrel{K}{\leftrightarrow} S; S \models \otimes(S); S \models \pm^{K_S} S; \\ S \models A \stackrel{K}{\leftrightarrow} S; S \models A \Rightarrow A \models *; \end{aligned}$$

S believes that he or she shares A 's password and that K is a suitable key. The fact that S already has K and believes its validity might be included in the final results of protocol analysis.

$$\begin{aligned} A \ni (R_A^1 \oplus R_A^2); A \models \phi(R_A^1 \oplus R_A^2); A \models \#(R_A^1 \oplus R_A^2); \\ A \ni (P_A \oplus R_A^1); A \models \phi(P_A \oplus R_A^1); A \models \#(P_A \oplus R_A^1); \end{aligned}$$

That is, A possesses a one-time pad and a masked password for a corresponding session, and believes their freshness and recognizability.

For any run of the protocol:

Message 1 : applying being-told rule(T1) and possession rules(P1,P3,P8) we obtain $S \ni R_A^1, S \ni R_A^2$, and $S \ni (< P_A > \oplus R_A^1)$. That is, S decrypts message 1.

Applying recognizability rule(R1), being-told rule(T1), and message interpretation rule(I2) we obtain $S \models \phi(R_A^1, R_A^2, K \oplus R_A^1)$ and $S \models A \sim (R_A^1, R_A^2, K \oplus R_A^1)$. That is, A is authenticated and message 1 is recognized.

Message 2 : applying being-told rule(T1) and possession-rules(P1,P3,P5) we obtain $A \ni K$. That is, A possesses K .

Applying recognizability rule(R1), possession rules(P2,P4), recognizability rules(R1,R5), freshness rules(F1,F10), and message interpretation rule(I3) we obtain $A \models \#(H(< P_A > \oplus R_A^1, K, R_A^2))$ and $A \models S \sim (H(< P_A > \oplus R_A^1, K, R_A^2))$.

Applying jurisdiction rules(J2,J3) we obtain $A \models S \models A \stackrel{K}{\leftrightarrow} S$. That is, A ensures that S believes K .

Applying jurisdiction rule(J1) we obtain $A \models A \stackrel{K}{\leftrightarrow} S$. That is, A also believes K .

Message 3 : applying recognizability rules(R1,R5), freshness rules(F1,F10), and message interpretation rule(I3) we obtain $S \models \#(H(< P_A > \oplus R_A^2, K, R_A^1))$ and $S \models A \sim (H(< P_A > \oplus R_A^2, K, R_A^1))$.

Applying jurisdiction rules(J2,J3) we obtain $S \models A \models A \stackrel{K}{\leftrightarrow} S$. That is, S also ensures that A believes K .

From the protocol analysis above summarize the results as follows.

$$S \ni K, S \models A \stackrel{K}{\leftrightarrow} S, S \models A \models A \stackrel{K}{\leftrightarrow} S$$

$$A \ni K, A \models A \stackrel{K}{\leftrightarrow} S, A \models S \models A \stackrel{K}{\leftrightarrow} S$$

That is, A and S shares K and ensures the authenticity of it and each other.

4.3 Advanced Protocols

We introduce more protocols which are modified versions of our basic protocol for various purposes.

4.3.1 Mutual Key Generation

By simple modification in our protocol, it is possible to generate a new session key through participants' mutual negotiation. For this purpose, a new session key K can be generated by several methods. For instance, we can use a one-way hash function or a discrete logarithm problem. The protocol is as follows:

1. $A \rightarrow S$: $[R_A^1 \parallel R_A^2 \parallel P_A \oplus R_A^1]_{K_S}$
2. $S \rightarrow A$: $OP \oplus R_S \parallel H(P_A \oplus R_A^1 \parallel K \parallel R_S)$
3. $A \rightarrow S$: $H(P_A \oplus R_S \parallel K \parallel R_A^1)$

Message 1 is equal to that of basic protocol. In step 2, S generates a new session key K as $K = f(H(R_A^2, R_S))$ or $K = f(g^{xy} \text{ mod } n)$. In the latter case, since its security is based on the difficulty of calculating discrete logarithms in a finite field, two random values should satisfy $R_A^2 = g^x \text{ mod } n$ and $R_S = g^y \text{ mod } n$ where n is a large prime and g is a primitive root[2]. Function $f()$ is required for optimizing the size of key. x and y are secret random values of A and S , respectively. The hash value is appended to ensure the integrity and freshness of K . Message 3 is sent to S for ensuring that A received R_S and generated K correctly.

4.3.2 Using A 's Public Key

Our protocol can be modified to use A 's public key rather than S 's as follows:

1. $A \rightarrow S$: $[K_A]_{P_A}$
2. $S \rightarrow A$: $[R_S \parallel K \parallel P_A \oplus R_S]_{K_A}$
3. $A \rightarrow S$: $H(P_A \oplus R_S \parallel K \parallel R_S)$

In step 1, A 's new public key K_A is encrypted under P_A and sent to S . Let's discuss the use of P_A as an encryption key. Let K'_A be a guessed K_A which resulted from message 1 decrypted under P'_A , a guessed P_A . An attacker cannot determine whether K'_A is correct in all messages because they cannot find a correct private key for both K_A and K'_A , and a random value R_S . As a result, we can say that K_A is unforgeable in our protocol. Since K_A is unforgeable data, the weak secret P_A can be used as an encryption key for K_A . In step 2, S replies with a new session key which are encrypted under K_A . R_S generated by S and the masked P_A are included to ensure the freshness in authentication. Message 3 is sent to S to ensure that A

received K correctly. Using P_A which is masked by R_S and the fact that A decrypted message 2 correctly, S is able to confirm the authenticity of A . Though A is authenticated by S in step 3, the modified version is as secure as basic protocol.

4.3.3 Not Using Public Key

We can modify our protocol not to use a complete public key algorithm. This solves a problem of export limitation. The following protocol is only based upon the difficulty of calculating discrete logarithms:

1. $A \rightarrow S : [R_A]_{P_A}$
2. $S \rightarrow A : [R_S]_{P_A} \parallel H(R_A \parallel R_{AS})$ (5)
3. $A \rightarrow S : H(R_{AS} \parallel R_S)$

In this protocol, random values should be generated under the condition of $R_A = g^x \bmod n$ and $R_S = g^y \bmod n$ where n is a large prime and g is a primitive root. R_{AS} denotes $g^{xy} \bmod n$. We define that a new session key should be produced from $H(R_{AS})$ rather than from R_{AS} . That means $K = f(H(g^{xy} \bmod n))$. We mentioned the use of x , y , and $f()$ in section 5.1. A distinctive feature in generating K is that a one-way hash function is used to make R_{AS} un-verifiable even through K . Since R_{AS} is un-verifiable data, an attacker cannot succeed in guessing and in verification in spite of finding K . In step 1, A generates R_A and sends message 1. In step 2, S replies with R_S which is encrypted under P_A . Though K is made from R_A and R_S , nobody can go inverse operation to find R_A and R_S only through K . This is the property and security of the discrete logarithm problem and the one-way hash function. Since R_A and R_S are un-verifiable, P_A can be used as a cryptographic key for encrypting them. Hash value is appended to ensure the integrity and freshness of K . Message 3 is sent to S for ensuring that A received R_S and generated K correctly.

4.3.4 Repeated Authentication

We can extend our protocol to be accommodated in a connectionless environment. Compared with a connection oriented environment, a connectionless environment requires a repeated authentication for providing continuous services. In other words, a client should be authenticated whenever sending a request. Thus, our connectionless protocol is composed of two procedures, an initialization procedure and a repeated procedure. While the initialization procedure needs a public key encryption, the repeated procedure does a relatively light computation such as a one-time padding and hashing. The initialization procedure is as follows:

1. $S \rightarrow A : S \parallel R_S^1 \parallel CERT[S]$ (6)

2. $A \rightarrow S : [A \parallel R_A^1 \parallel R_S^1 \parallel P_A \oplus R_A^1]_{K_S}$

We assume that A is a client who requests a service and S is a server who provides it. After receiving a request from A , S replies with message 1 instead of a service reply. $CERT[S]$ denotes a certificate of S . In step 2, A replies with encrypted message 2 for authentication. After initialization procedure, S is able to provide an authenticated service. But in a connectionless environment, a succeeding authentication is necessary for next connection. In our protocol, the repeated procedure exists for succeeding services. The repeated procedure is as follows for $i = 1, 2, 3, \dots$:

1. $S \rightarrow A : R_A^i \oplus R_S^{i+1} \parallel H(P_A \oplus R_A^i \parallel R_S^{i+1} \parallel A \oplus S)$ (7)
2. $A \rightarrow S : R_A^{i+1} \oplus R_S^{i+1} \parallel H(P_A \oplus R_A^{i+1} \parallel R_S^{i+1} \parallel A \oplus S)$

After receiving the next request from A , S replies with message 1 to make A recognize R_S^{i+1} . A replies with message 2 for authentication. The repeated procedure is performed iteratively. Compared with the initialization procedure, the repeated procedure is light to compute. For key exchange, a new session key can be made as $K = f(H(R_A^i \oplus R_S^i))$ for each connection.

5 Security of Our Protocol

We examine how secure our protocol is against several kinds of attacks launched to find out K or P_A .

5.1 Off-line Guessing Attacks

Off-line guessing attacks are launched by reconstructing messages with guessed or verified elements and by comparing them with messages eavesdropped. Attackers can be classified into two kinds according to their knowledge about a new session key K . One is an inside attacker who knows K and the other is an outside attacker who does not know K [8, 10]. The inside attack corresponds to the Denning-Sacco attack [12]. Even if K is stolen or compromised later, an attacker can get nothing but OP and K . Thus, for verifying a guessed P_A off line, an inside attacker should know both R_A^1 and R_A^2 in advance. It requires $2^{|R_A^1|+|R_A^2|}$ operations in message 1 and 3. Since the inside attacker could find OP in message 2, $2^{|R_A^1|}$ operations are required for finding both R_A^1 and R_A^2 . For an outside attacker, while $2^{|R_A^1|+|R_A^2|}$ operations are required in message 1 and 2, $2^{|R_A^1|+|R_A^2|+|K|}$ operations are required in message 3. As a result, an off-line guessing attacks are infeasible in our protocol.

5.2 On-line Guessing Attacks

On-line guessing attack can be defeated by detection of its failed guess [11]. Failed guess means that an

incorrectly guessed password is tried for authentication. For detection of failed guess, authenticity of messages should be ensured in the protocol[11]. Our protocol is designed deliberately to defeat undetectable on-line guessing attacks. As we mentioned previously, on-line guessing attack is launched by means of impersonating someone with guessed password or replaying eavesdropped messages. When a impersonating attack is attempted on line, S can detect it easily by the masked P_A in step 1. Replaying attack is as expensive as off-line guessing attack because attackers should find nonce values which are unverifiable in our protocol. As a result, any on-line guessing attacks are defeated in our protocol.

5.3 Replay Attack

We examine the possibility of gaining authenticity by replaying message 1. Since two random numbers included in message 1 is used always to encrypt a new session key in message 2, replaying the same message 1 results in generating the same one-time pad. Therefore, if an adversary obtains K by some cryptanalytic methods, then (s)he can possess a corresponding one-time pad and message 1. By replaying message 1, it seems that the adversary can get K in the new message 2 because K might be encrypted under the same one-time pad that was found. However, the replay attack is infeasible. The adversary must reply with message 3 to complete the protocol but (s)he cannot construct the message in which not only the new session key but also R_A^1 , R_A^2 , and P_A should be included. As we know, it is computationally infeasible to find the contents. Thus, our protocol is secure against the message replay attack.

5.4 Attacks on K

Since K is encrypted by one-time padding in message 2, an attacker should launch a brute force attack to find it in our protocol. It means that $2^{|K|}$ operations are necessary. We assume that K is a cipher key for a famous conventional block cipher system such as DES, FEAL, or IDEA, and it is chosen well at random. Thus, K may have a sufficiently large size such as 64 bits or 128 bits to defeat a brute force attack. By obtaining OP in advance or by verifying the hash value with guessed elements, an attacker could attempt to find K . But it is more difficult for attacker to obtain OP or to verify hash value because at least $2^{|R_A^1|+|R_A^2|}$ operations are necessary. As a result, a new session key K can be distributed securely in our protocol.

5.5 Attacks on $H()$

We assume the use of strong one-way hash function in our protocol. For an attack on integrity by finding collision pairs of hash value, the probability to find the

same-sized pairs is extremely low. Even if they are found, it is infeasible to cheat participants without finding the correct random values because attackers cannot make a correct one-time pad.

6 Efficiency of Our Protocols

Most protocols, which cannot provide security in protecting weak secrets, are not the points to be considered in our comparison. We compare our protocols, in terms of efficiency, only with other related protocols which are known resistant to guessing attacks.

As shown in table 1, our protocols are more efficient than other related protocols in terms of the number of protocol steps, random values, and cryptographic operations. For two party's direct authentication and key exchange, we compare our two distinctive protocols such as protocol (1) and (4) with others. In all of protocols which are resistant to guessing attacks, a public key algorithm has been inevitable for defeating verifiable text attacks. However, our protocol (5) solves the problem at the cost of computation on discrete logarithms.

Protocols	the number of protocol steps	the number of random values		the number of cryptographic operations		
		A	S	PublicKey	Conventional	Hash
Basic Protocol	3	A	2	1	0	2
		S	0			
Protocol Using A's Public Key	3	A	0	1	1	1
		S	1			
Strengthened KE [7]	5	A	2	1	3	2
		S	2			
GLNS nonce direct [8]	5	A	1	1	5	0
		S	4			
Gong's Optimal [9]	3	A	1	1	3	0
		S	2			

Table 1: Efficiency Comparison of Protocols

7 Conclusion

In this paper, we have proposed new cryptographic protocols, which ensure security and efficiency in protecting weak secrets from guessing attacks, for authentication of communicating participants and for sharing of a session key. Compared with other related research results[7, 8, 9], our protocols are more efficient in terms of the number of random values, cryptographic operations, and protocol steps. It is due to the appropriate use of one-time pad and one-way hash function which are relatively fast and secure. Since we attempted to achieve security and efficiency in authentication protocols which are resistant to guessing attacks, our protocols can be used in a variety of area of computer communications. Protocol (1) is for common two party authentication. Protocol (3) is for generating a session key by participants' negotiation. While protocol (4) is for using A 's public key, protocol (5) is for not using a complete public key algorithm. Protocol (6)/(7) is

for repeated authentication in a connectionless environment.

References

- [1] T.Kwon, M.Kang, and J.Song, "An Adaptable and Reliable Authentication Protocol for Communication Networks," *IEEE INFOCOM 97*, pp.738-745, 1997.
- [2] W.Diffie and M.Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, Vol.22, No.6, pp.644-654, Nov. 1976.
- [3] R.Needham and M.Schroeder, "Using Encryption For Authentication in Large Networks of Computers," *Communications of the ACM*, Vol.21, No.12, pp.993-999, Dec. 1978
- [4] R.Morris and K.Thompson, "Password Security: A Case History," *Communications of the ACM*, vol.22, no.11, pp.594-597, Nov. 1979
- [5] D.Feldmeier and P.Karn "UNIX Password Security - Ten Years Later," *Crypto'89*, Vol.435 of LNCS, pp.44-63, 1989
- [6] D.Klein "Filing the Cracker: A Survey of, and Improvements to Password Security," the 2nd USENIX Unix Security Workshop, pp.5-14, Aug. 1990
- [7] S.Bellovin and M.Merritt, "Encrypted Key Exchange : Password-Based Protocols Secure Against Dictionary Attacks," *IEEE Computer Society Symposium on Research in Security and Privacy*, pp.24-29, 1995
- [8] L.Gong, M.Lomas, R.Needham, and J.Saltzer, "Protecting Poorly Chosen Secrets from Guessing Attacks," *IEEE Journal on Selected Area in Communications*, vol.11, no.5, pp.648-656, June 1993
- [9] L.Gong, "Optimal Authentication Protocols Resistant to Password Guessing Attacks," 8th IEEE Computer Security Foundation Workshop, pp.24-29, June 1995
- [10] L.Gong, "Verifiable-text Attacks in Cryptographic Protocols," *IEEE INFOCOM 90*, pp.686-693, 1990
- [11] Y.Ding and P.Hostler, "Undetectable On-line Password Guessing Attacks," *ACM Operating Systems Review*, vol.29, no.4, pp.77-86, Oct. 1995
- [12] D.Denning and G.Sacco, "Timestamps in Key Distribution Protocols," *Communications of the ACM*, Vol.24, No.8, pp.533-536, 1981
- [13] D.Otway and O.Rees, "Efficient and Timely Mutual Authentication," *ACM Operating Systems Review*, Vol.21, No.1, pp.8-10, 1987
- [14] C.Meadows, "Applying Formal Methods to the Analysis of a Key Management Protocol," *Journal of Computer Security*, Vol.1, No.1, pp.5-35, 1992
- [15] M.Burrows, M.Abadi, and R.Needham, "A Logic of Authentication," *Technical Report SRC TR 39*, Digital Equipment Corporation, Feb. 1989
- [16] L.Gong, R.Needham, and R.Yahalom, "Reasoning about Belief in Cryptographic Protocols," *IEEE Symposium on Research in Security and Privacy*, pp.234-248, 1990
- [17] B.Schneier, *Applied Cryptography*, 2nd ed., John Wiley & Sons, 1996