Vol. 39 No. 12 Transactions of Information Processing Society of Japan Dec. 1998

Regular Paper

Augmented Encrypted Key Exchange Using RSA Encryption and Confounder

ARI MOESRIAMI BARMAWI,† SHINGO TAKADA†† and NORIHISA DOI†

The Augmented Encrypted Key Exchange (A-EKE) proposed by Bellovin and Merritt (1993) used the hash of the sender's password as the shared secret key for encryption. However, this scheme can be broken by Simmon's Attack. To overcome this, we propose to improve the scheme by incorporating Indirect RSA Encryption and Confounder. In our scheme we use a public key which will be kept by the communicating parties and will be exchanged indirectly, i.e., instead of sending the whole public key, the two parties will send the number which determines their public key, along with the shared key. Meanwhile, the confounder is used to overcome partition attack which can be used to overcome the subtle problem of A-EKE using RSA encryption. Theoretical analysis shows that our proposed scheme is more secure than A-EKE.

1. Introduction

In computer security, authentication is one of several areas that needs to be investigated to secure the environment for legitimate use. Several methods have been proposed to accomplish this. One of them is Augmented Encrypted Key Exchange (A-EKE) proposed by Bellovin and Merritt [2]. It uses two one way functions to authenticate the sender. However this scheme can be broken by Simmon's Attack [8], when the session key and the number which builds the hash of the sender's password are relatively prime. It has been shown that the probability of these numbers being relatively prime is $6/\pi^2$ (about 0.61), which means that the probability to break the scheme is also 0.61.

We propose a method to improve the scheme by using *indirect RSA* for user authentication, instead of using two one way functions. It differs from the conventional RSA in that the communicating parties exchange the public key indirectly. In other words, instead of the public key itself, they will exchange a number which determines their public key. This number will be encrypted by a shared secret key. To obtain each others public key, a hash function is applied to that number. The hash function is determined by a number which previously has been agreed upon by the communicating parties.

Their public keys are used to encrypt the exponential random numbers which will be exchanged between the communicating parties. Since these numbers will be used to determine the session key, we need to protect these numbers from attackers by keeping the public keys only known to both parties.

Thus, instead of using only one shared secret key as has been proposed by Bellovin and Merritt, we also use an integer number which will build the hash of the communicating parties' number. This number will be used later to determine the public key. Both of them are kept secret between the communicating parties and have to be agreed upon in advance.

Since the number which determines the public key is always odd, if no special precautions are taken, an attacker can rule out half of the candidate values of the shared secret key if the encrypted message was an even number. Since each session will use a different number, independent of all others previously used, trial decryptions resulting in illegal values of the number which determines the public key will exclude a different shared secret key each time. Put another way, each session will partition the remaining candidate key space into two approximately equal halves. The decrease of the key space is logarithmic; comparatively few intercepted conversations will suffice to reject all invalid guesses of the shared secret key [1),2)]. This attack is usually called *partition attack.*

We overcome this weakness by further incorporating a confounder, which was originally proposed by Gong, et al [7]. A confounder is a random number whose value can be ignored by the recipient, with the assumption that its

† Department of Computer Science, Graduate School of Science and Technology, Keio University
†† Graduate School of Information Science, Nara Institute of Science and Technology

Message 1. Alice → Bob: Alice,$\{\alpha^{R_A} \bmod \beta\}_{H(P)}$
Message 2. Bob → Alice:Bob,$\{\alpha^{R_B} \bmod \beta\}_{H(P)}, \{challenge_B\}_K$
Message 3: Alice → Bob: $\{challenge_A, challenge_B\}_K$
Message 4. Bob → Alice:$\{challenge_A\}_K$
Message 5. Alice → Bob: $\{F(P,K)\}_K$

Fig. 1 Augmented Encrypted Key Exchange (A-EKE).

length has been predetermined by the communicating parties. Since a confounder is a random number which could be odd or even, the attacker can not identify the real number that we encrypted. Hence, the attacker can not exclude different shared secret key for each trial decryption, which means that the attacker can not apply partition attack on our scheme. Detailed analysis of how the confounder can overcome the partition attack will be described in Section 4.

Section 2 first presents the Augmented-EKE, and Section 3 proposes our scheme. Section 4 analyzes our proposed scheme based on the knowledge of the hash of sender's password and also from a mathematical viewpoint. Section 5 makes concluding remarks.

2. Augmented Encrypted Key Exchange (A-EKE)

Augmented-EKE uses two one way hash functions. The first one $H(P)$ (the hash of the sender's password P) is used as the communicating parties' shared secret key:

$$H(P) = P^z \bmod n$$

where z is a random number which is chosen by the sender and n is a product of 2 prime numbers which are public. The second function $F(P,K)$ is used to verify the sender:

$$\begin{aligned} F(P,K) &= H_K(P) \\ &= P^K \bmod n \end{aligned}$$

where K is the communicating parties' session key.

The basic idea is to exchange both parties' exponential random numbers encrypted by $H(P)$. These random numbers are $\alpha^{(R_A \text{ or } R_B)} \bmod \beta$, where α is a primitive root of Galois Field of β (GF(β)), β is a prime number and both of them are public. The result of this exchange is a session key K which depends on both parties' exponential random numbers.

This session key can in practice only be known by someone who knows $H(P)$. If the intruder can guess $H(P)$, he/she will be able to impersonate the sender, the receiver or both. To reduce the risk, the sender must supply $F(P,K)$ to prove the receiver of the sender's identity. The receiver verifies the sender by checking $H_K(H(P))$ and $H(F(P,K))$, where

$$\begin{aligned} H_K(H(P)) &= (H(P))^K \bmod n \\ H(F(P,K)) &= (F(P,K))^z \bmod n \end{aligned}$$

If these two hash functions are equal, then the sender will be verified by the receiver as the legitimate sender. These two hash functions will be equal iff the password P which is used in $H(P)$ and $F(P,K)$ are equal. The complete scheme of A-EKE is shown in **Fig. 1** with Alice as the sender and Bob as the receiver.

The attacker can still break the scheme if he/she can guess the sender's password. If the session key K and z are relatively prime, then by using Simmon's attack, the probability of breaking the password is approximately 0.61. Since the probability to break the scheme is equal to the probability to break the password, the probability of breaking the protocol is also 0.61, which is quite large.

3. A-EKE Using Indirect RSA (A-EKE/RSA)

To reduce the probability of breaking A-EKE, we propose a scheme which applies Indirect RSA public key. We first briefly describe the RSA Cryptosystem and then incorporate it to A-EKE along with a confounder.

3.1 RSA Cryptosystem

RSA cryptosystem is made up of a collection of users, each having his own enciphering and deciphering keys. The enciphering key (non-secret) consists of integers n and K_p, while the deciphering key (secret) is an integer K_s. In this scheme, n is an integer which is the product of two carefully selected large primes p and q, i.e., $n = pq$. K_p denotes the public key and K_s the private key. The keys K_s and K_p must be selected in such a way that each is relatively prime to $\phi(n)$, where $\phi(n) = (p-1)(q-1)$ is the Euler Totient function and the product of the two keys should be equal to $(1 \bmod(\phi(n)))$.

In RSA, encryption of message M (where

Message 1. Alice → Bob: Alice,$\{n_A, K_A, c_A\}_{H(P)}$
Message 2. Bob → Alice:Bob,$\{n_B, K_B, \alpha^{R_B} \bmod \beta, c_B\}_{K_{p_{Alice}}}$
Message 3: Alice → Bob: $\{\alpha^{R_A} \bmod \beta\}_{K_{p_{Bob}}}$,$\{challenge_A\}_K$
Message 4. Bob → Alice:$\{challenge_A, challenge_B\}_K$
Message 5. Alice → Bob: $\{\{challenge_B\}_{K_{s_{Alice}}}\}_K$

Fig. 2 A-EKE using RSA and confounder.

$0 \leq M \leq n-1$) is as follows:

$$Y = M^{K_p} \bmod n$$

while decryption of Y is as follows:

$$Y^{K_s} \bmod n = M^{K_p K_s} \bmod n = M \bmod n$$

where $K_p \times K_s = 1 \bmod \phi(n)$.

3.2 A-EKE using Indirect RSA

Before using A-EKE with indirect RSA, we must first set the value of h_0 (a number used to build the public key which needs to be agreed in advance by the communicating parties) and assume that both parties know the hash of the sender's password P. We set the length of the confounder c_A and c_B equal to the length of h_0, because if both parties exchange the value of the length of the confounders, it can cause leaked information, while using another component to determine the length of h_0 will waste memory. In the rest of this paper, we will call the sender as Alice and the receiver as Bob. The complete scheme of our method is as follows (see **Fig. 2**):

(1) Alice sends Bob her modulus number (n_A), a number (K_A) which determines her public key and a confounder c_A, encrypted with their shared secret key ($H(P)$).

(2) Bob decrypts the message from Alice using their shared secret key, and calculates Alice's public key ($K_{P_{Alice}}$) which is equal to $(K_A)^{h_0}$. Then Bob chooses his modulus number n_B and a number (K_B) which determines his public key. Also, he chooses his random number R_B and calculates $\alpha^{R_B} \bmod \beta$. He sends Alice all of these numbers concatenated with confounder c_B, and encrypts all of them with $K_{p_{Alice}}$.

(3) Alice decrypts the message from Bob using her secret key ($K_{s_{Alice}}$) which is equal to $((K_A)^{h_0})^{-1} \bmod \phi(n)$. She picks her random number R_A, calculates ($\alpha^{R_A} \bmod \beta$) and also ($\alpha^{R_A R_B} \bmod \beta$) as their session key K. She also calculates Bob's public key ($K_{p_{Bob}}$) which is equal to $(K_B)^{h_0}$. After that, she sends Bob ($\alpha^{R_A} \bmod \beta$) encrypted with $K_{p_{Bob}}$ and also her challenge ($challenge_A$) encrypted with their session key K. A challenge is a message which is exchanged between the communicating parties to verify each other.

(4) Bob decrypts Alice's first message using his secret key ($K_{s_{Bob}}$) to obtain ($\alpha^{R_A} \bmod \beta$). Using this value, Bob calculates K which is equal to ($\alpha^{R_A R_B} \bmod \beta$). Then he decrypts Alice's second message using K. After he obtained Alice's challenge ($challenge_A$), he picks his challenge ($challenge_B$) and sends both challenges encrypted with their session key K to Alice.

(5) Alice decrypts the message from Bob to obtain ($challenge_A$ and $challenge_B$), and compares ($challenge_A$) which comes from Bob with her legitimate ($challenge_A$). If they match, then she sends ($challenge_B$) back to Bob signed with $K_{s_{Alice}}$ and encrypted with K.

(6) Finally, Bob decrypts the message from Alice using the session key K and $K_{p_{Alice}}$ to obtain ($challenge_B$). If Bob verifies that ($challenge_B$) echoed correctly, then the scheme is concluded.

This proposed scheme uses $((K_A)^{h_0})$ as Alice's public key and $(((K_A)^{h_0})^{-1} \bmod \phi(n_A))$ as Alice's secret key. The value of $(K_A)^{h_0}$ should be congruent to $((K_A)^{h_0} \bmod \phi(n_A))$.

An intruder may try to assign a modulus number n_A for Alice's modulus number, choose a number K_A as the number that determines Alice's public key, guess the value of confounder c_A and send these numbers encrypted with the guessed shared secret key $H(P)'$ to Bob. Bob will decrypt the message using $H(P)$. In this case, Bob will obtain K'_A and n'_A. Using K'_A Bob will calculate Alice's public key which is equal to $(K'_A)^{h_0}$. Then he will send his modulus number, the number which determines his public key K_B, his exponential random num-

ber ($\alpha^{R_B} \bmod \beta$) and the confounder c_B encrypted with Alice's public key to the intruder. The intruder will guess the value of h_0 (i.e., h'_0), compute her secret key, and decrypt the message from Bob using the guessed secret key. The intruder can not verify his guess until Message 4, because until then the intruder can not match his legitimate challenge ($challenge_A$) and ($challenge'_A$) sent by Bob. If both values match, then $H(P)'$ is equal to $H(P)$ and h'_0 is equal to h_0. Thus the scheme will conclude if both $H(P)$ and h_0 can be guessed, but as long as these are kept secret by the communicating parties and never sent in clear, it is very difficult to guess their values.

Suppose the intruder is able to guess one of these two elements, for example $H(P)$. With the knowledge of $H(P)$, the intruder could send K_A, n_A and confounder c_A whose length is the guessed length of h_0, encrypted by $H(P)$ which will be received by Bob as K'_A and n'_A. According to the above method the intruder can not verify his guess until Message 4, when he can match his legitimate challenge ($challenge_A$) with the challenge sent by Bob ($challenge'_A$). If both values match, then the guessed length of h_0 is true. Even though the guessed length of h_0 is true, this does not mean that the intruder can directly break the protocol, because the intruder still has to guess the value of h_0. So, it is still difficult for the intruder to calculate his public key, because he does not know the value of h_0. Thus, the strength of this scheme depends on the difficulty to guess h_0. The difficulty to guess h_0 will be discussed in Section 4.

3.3 Finding the Public Key

In this section we will discuss how Bob can calculate Alice's public key, and vice versa. After Bob receives the first message, he will know the value which determines Alice's public key K_A and he can directly calculate Alice's public key by using:

$$K_{p_{Alice}} = (K_A)^{h_0} \tag{1}$$

Bob can encrypt his message (Message 2) using Alice's public key and send it to Alice.

According to RSA, Alice's public key should be:

$$\begin{aligned} K_{p_{Alice}} = (K_A)^{h_0} &\equiv (K_A)^{h_0} \bmod \phi(n_A) \\ &\equiv X_A \bmod \phi(n_A) \end{aligned}$$

where X_A is a positive integer which is relatively prime to $\phi(n_A)$. Thus, $(K_A)^{h_0}$ should be relatively prime to $\phi(n_A)$.

Since the value of $\phi(n_A)$ has to be kept secret, the public key of Alice which is calculated by Bob will be $(K_A)^{h_0}$ instead of $(K_A)^{h_0} \bmod \phi(n_A)$. To prove that this method is plausible, there are two points which has to be observed:

- The secret key for which the public key is $(K_A)^{h_0}$ (i.e., $(K_A)^{h_0} \bmod \phi(n_A)$) is the same as that for which the public key is X_A ($X_A \bmod \phi(n_A)$). According to RSA, the secret key of Alice is the multiplicative inverse of her public key.
- The message which is encrypted with $(K_A)^{h_0}$ should be decrypted using the secret key for which the public key is $X_A \bmod \phi(n_A)$ (that is $(X_A \bmod \phi(n_A))^{-1}$).

For observing the first point, we have to prove Theorem 1.

Theorem 1 Since $(K_A)^{h_0}$ and $(X_A \bmod \phi(n_A))$ are congruent, the multiplicative inverse of both numbers are the same. The proof of this theorem is shown in Appendix A.2.

Next, let us observe the second point using Theorem 2.

Theorem 2 We will encrypt a message M using public key $(K_A)^{h_0}$, and obtain $(M)^{(K_A)^{h_0}} \bmod n_A$. The communicating party can decrypt the message using the secret key $(X_A \bmod \phi(n_A))^{-1}$ (where $(K_A)^{h_0} \bmod \phi(n_A) \equiv X_A \bmod \phi(n_A)$). The proof of this theorem is shown in Appendix A.3

Since these two points can be proven, we can say that our method is plausible.

3.4 Choosing a Number which Determines the Public Key (K_A/K_B) and the Secret Key

Since $(K_A)^{h_0}$ should be relatively prime to $\phi(n_A)$, we have to choose a number K_A which should be relatively prime to $\phi(n_A)$ (see Appendix A.5), to construct the public key. The algorithm to choose a number which determines the public key is as follows:

(1) Choose a positive integer X which is not equal to p, q or n, where $n = p * q$, and p and q are prime numbers.

(2) Calculate $gcd(X, \phi(n))$. If the value of $gcd(X, \phi(n))$ is equal to 1, go to the next step. Otherwise, go back to step 1.

(3) Compute the multiplicative inverse of the public key (i.e., $X^{h_0} \bmod \phi(n)$) to obtain the secret key. If the multiplicative inverse has no unique value or the value

is equal to 1, then find another number as the value of X and repeat the procedures from the beginning until the multiplicative inverse has a unique value and not equal to 1. Otherwise, use X as the number which determines the public key. Unique value means a value whose multiplicative inverse is not equal to the value itself.

This algorithm can be applied for choosing numbers for K_B and K_A.

The inverse value is actually the secret key. If this value is negative, the sender will get a wrong message. To overcome this problem, the value should be replaced by $\phi(n)$ added by the inverse value[6].

4. Security Analysis

In this chapter we will analyze the security of A-EKE using RSA without and with confounder.

4.1 Analysis of A-EKE/RSA without Confounder

First, we will calculate the probability of finding the public keys under the assumption that the intruder has already obtained $H(P)$. Suppose h_0 is an l-digit positive integer, then there are 10^l numbers which could be the value of h_0. For example, Alice's public key which is used by Bob to encrypt his message is $(K_A)^{h_0}$ instead of $((K_A)^{h_0} \bmod \phi(n_A))$. Thus, $(K_A)^{h_0}$ itself could be greater than n_A, but the value of $((K_A)^{h_0} \bmod \phi(n_A))$ will always be smaller than $\phi(n_A)$. According to the Carmichael Function (see Appendix A.1), if K_A and $\phi(n_A)$ are relatively prime, there are $\lambda(\phi(n_A)) - 1$ numbers which satisfy $((K_A)^{h_0} \bmod \phi(n_A) \neq 1)$ or in other words there is only one public key value which is equal to 1 (i.e., for h_0 equal to $\lambda(\phi(n_A))$) within $\lambda(\phi(n_A))$ numbers (where λ is the Carmichael Function). Since this value will be repeated in the order of $\lambda(\phi(n_A))$, there are at least γ values of h_0 which could have the same value of public key where $\gamma.\lambda(\phi(n_A)) \geq 10^l$. This means that there are $\gamma(\lambda(\phi(n_A)) - 1) + \delta$ (for $\delta < \lambda(\phi(n_A))$)value of h_0 which satisfy the requirements, where γ is a positive integer and $\delta \geq 0$. These numbers have to satisfy the following equation:

$$\gamma\{(\lambda(\phi(n_A)) - 1)\} + \delta + \gamma = 10^l$$
$$\gamma(\lambda(\phi(n_A))) + \delta = 10^l \qquad (2)$$

Thus, the probability that a guess on $((K_A)^{h_0} \bmod \phi(n_A))$ will yield the correct value of Alice's public key is

$$Pr\ (K_{p_{Alice}} \text{ is correct}) \approx \frac{\gamma}{(\gamma(\lambda(\phi(n_A)) - 1)) + \delta}$$

If $\delta \ll \gamma(\lambda(\phi(n_A)) - 1))$ then

$$Pr\ (K_{p_{Alice}} \text{ is correct}) \approx \frac{1}{(\lambda(\phi(n_A)) - 1)} \qquad (3)$$

From the above equation, we can conclude that the larger the value of $\lambda(\phi(n_A)) - 1$ becomes, the smaller the probability of obtaining Alice's public key becomes. In this case, although the intruder has already obtained the public key, it does not mean that they can break the scheme, because there are more than one value of h_0 which could yield the same value of public key.

As mentioned above, there are approximately γ numbers which satisfy the value of the public key. Thus, the probability to obtain the correct h_0 is

$$Pr\ (h_0 \text{ is correct}) \approx \frac{1}{(\lambda(\phi(n_A)) - 1)} \times \frac{1}{\gamma} \qquad (4)$$

Theorem 3 For an attacker **with** the knowledge of $H(P)$, the probability of breaking the scheme is $\frac{1}{(\lambda(\phi(n_A))-1)} \times \frac{1}{\gamma}$, where $(\lambda(\phi(n_A)) - 1)\gamma$ is greater than 2. Since the minimum value of $(\lambda(\phi(n_A)) - 1)\gamma > 2$, the maximum value of Eq. (4) is smaller than 1/2 (for n_A consists of 2 digit prime numbers or more). The proof of this theorem will be shown in Appendix A.4.

As an example, we choose the smallest 2 digit prime numbers, i.e., $p_1 = 11$ and $q_1 = 13$. Then,

$$\begin{aligned}&\lambda(\phi(n_A))\\ &= lcm(lcm(\lambda(2), \lambda(5)), lcm(\lambda(4), \lambda(3)))\\ &= lcm(lcm(\phi(2), \phi(5)), lcm(\phi(4), \phi(3)))\\ &= 4\end{aligned}$$

Since $\gamma = 1$,

$$\gamma(\lambda(\phi(n_A)) - 1)) = 1(4 - 1) = 3$$

It is already proven that the minimum value of $\lambda(\phi(n_A))$ for 2 digit prime numbers is greater than 2.

Furthermore, the maximum value of Eq. (4) for $\gamma = 1$ is

$$Pr\ (h_0 \text{ is correct}) \approx \frac{1}{(\lambda(\phi(n_A)) - 1)(\gamma)} \approx \frac{1}{(4-1)(1)} \approx \frac{1}{3}$$

Thus the maximum value of Eq. (4) for 2 digit prime numbers is 1/3 which is smaller than 1/2.

The probability will decrease with the increase of the prime numbers' length. Since the length of the prime numbers used in our proposed scheme is greater than 2 digits, the probability to break the scheme is smaller than 1/3. This means that although the attacker **has already obtained** $H(P)$ it is still harder to break A-EKE/RSA than to break A-EKE.

Next, we will discuss the probability to break the scheme for an attacker **without** the knowledge of $H(P)$. As an example, we will use IDEA [4] to encrypt the first message of A-EKE/RSA which uses 128 bit key to encrypt the message. By using brute force attack, the probability to obtain a message encrypted by IDEA is $1/2^{128}$. Using this fact, the probability of obtaining h_0 for an attacker **without** the knowledge of $H(P)$ is

$$Pr\ (h_0 \text{ is correct}) \approx \frac{1}{(\lambda(\phi(n_A)) - 1)(\gamma)} \times \frac{1}{2^{128}} \quad (5)$$

So, the probability of obtaining h_0 without the knowledge of $H(P)$ is smaller than the probability of obtaining h_0 with the knowledge of $H(P)$.

4.2 Analysis of A-EKE/RSA Using Confounder

We incorporated confounders in Messages 1 and 2 of our proposed scheme to prevent our scheme against partition attack. Let us refer to Messages 1 and 2 of Fig. 2. Suppose an attacker that intercepts Message 1, that is $\{n_A, K_A, c_A\}_{H(P)}$, inserts $H(P)'$ as the value of $H(P)$, and he obtains an even number for the value of $\{\{n_A, K_A, c_A\}_{H(P)}\}_{(H(P)')^{-1}}$. This condition does not mean that the number is an illegal value for $\{n_A, K_A, c_A\}$. Since we used a confounder c_A which is random (c_A can be odd or even), the value of $\{n_A, K_A, c_A\}$ can also be odd or even, and it means that an attacker can not exclude $H(P)'$ although the values of $\{\{n_A, K_A, c_A\}_{H(P)}\}_{(H(P)')^{-1}}$ were even. Thus, we can prevent our proposed scheme against partition attack by using the confounder c_A.

We now turn to the probability of breaking the scheme or to find h_0. Let us refer to Message 1 of Fig. 2. If we use IDEA encryption to encrypt $\{n_A, K_A, c_A\}$, the probability of obtaining $\{n_A, K_A, c_A\}$ is $1/2^{128}$. Therefore, the probability of obtaining $\{n_A, K_A\}$ is smaller than $1/2^{128}$, because to obtain $\{n_A, K_A\}$ the attacker has to guess c_A. Next, let us calculate the probability of obtaining c_A. Assume that the length of c_A is l-digits, which means that to guess c_A the attacker has to do l guesses or in other words the probability of obtaining c_A from each intercepted message is $1/l$. Thus the probability of obtaining $\{n_A, K_A\}$ from each intercepted message of Message 1 is

$$Pr(\{n_A, K_A\}) = \frac{1}{2^{128}} \times \frac{1}{l} \quad (6)$$

So, it is harder to break Message 1 which uses a confounder than to break a message not using a confounder. Certainly, with the use of a confounder, the probability of obtaining h_0 will also be smaller.

Finally, the probability to obtain h_0 for A-EKE/RSA using confounder **with** the knowledge of $H(P)$ is the product of the probability in Eq. (4) and the probability to obtain c_A, which is described as follows:

$$Pr\ (h_0 \text{ is correct}) \approx \frac{1}{(\lambda(\phi(n_A)) - 1)\gamma} \times \frac{1}{l} \quad (7)$$

and the probability to obtain h_0 for A-EKE/RSA using confounder **without** the knowledge of $H(P)$ is the product of the probability in Eq. (7) and the probability to obtain $H(P)$, which is described as follows:

$$Pr\ (h_0 \text{ is correct}) \approx \frac{1}{(\lambda(\phi(n_A)) - 1)\gamma} \times \frac{1}{l} \times \frac{1}{2^{128}} \quad (8)$$

5. Conclusion

We proposed a method incorporating indirect RSA encryption and confounder to A-EKE.

Indirect RSA Encryption was introduced into A-EKE to decrease the possibility to break the protocol for an adversary. In A-EKE, the probability was approximately 0.61. But, with our scheme, the probability depends on the chosen prime numbers. Even if the prime number is small (e.g., 2 digits), the probability is still lower than 1/3 (0.33). Larger digits will lower the probability even more.

To anticipate the problems caused by implementation of RSA, we used a confounder which

was originally proposed by Gong, et al[7]. By using a confounder, an attacker can not guess the message which includes the number that determines the public key and the modulus number. Thus, he can not exclude the illegal values of these numbers by using just a few messages, or in other words to obtain these numbers he has to do numerous trials.

Finally our analysis has shown that our method has a low probability of being broken.

References

1) Bellovin, S.M. and Merritt, M.: Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks, *Proc. IEEE Computer Society Symposium and Research in Security and Privacy*, pp.72–84 (1992).
2) Bellovin, S.M. and Merritt, M.: Augmented Encrypted Key Exchange: A Password-Based Protocol Secure against Dictionary Attacks and Password File Compromise, *First ACM Conference on Computer and Communication Security* (1993).
3) Rivest, R.L., Shamir, A. and Adleman, L.: Method for Obtaining Signatures and Public-Key Cryptosystems, *Comm. ACM*, pp.120–126 (1978).
4) Schneier, B.: *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley and Sons (1994).
5) Rhee, M.Y.: *Cryptography and Secure Communications*, McGraw-Hill Series on Computer Communications (1994).
6) Adler, A. and Coury, J.E.: *The Theory of Numbers: A Text and Source Book of Problems*, Jones and Barlett (1995).
7) Gong, L., Lomas, M.A., Needham, R.M. and Saltzer, J.H.: Protecting Poorly Chosen Secrets from Guessing Attacks, *IEEE Journal on Selected Area in Communications*, Vol.11. No.5, pp.648–656 (1993).
8) Gustavus, J.S.: *Contemporary Cryptology: The Science of Information Integrity*, IEEE Press (1992).
9) Diffie, W. and Hellman, M.: Privacy and Authentication: An Introduction to Cryptography, *Proc. IEEE*, Vol.67, No.3, pp.397–427 (1979).
10) Barmawi, A.M., Takada, S. and Doi, N.: Augmented Encrypted Key Exchange Using RSA Encryption, *8th IEEE International Symposium on Personal Indoor and Mobile Radio Communications*, pp.490–494 (1997).

Appendix

A.1 Carmichael Function

The Carmichael function of n can be calculated with the following algorithm:

(1) Find the divisor of n which is a power of distinct primes.
(2) Calculate the Carmichael function ($\lambda(n)$) which denotes the period of the sequence of the power as follows:
- If n is 1, 2 or 4, then $\lambda(n) = \phi(n)$ ($\phi(n)$ is Euler Totient Function).
- If $n > 4$ and is a power of 2 (ex. $2^3, 2^4, \ldots$), then $\lambda(n) = \phi(n)/2$.
- If n is a power of an odd prime, $\lambda(n) = \phi(n)$.
- If n is a product of $P_1, P_2, \ldots, P_k$ of powers of distinct primes, then $\lambda(n) = lcm(\lambda(P_1), \lambda(P_2), \ldots, \lambda(P_k))$

A.2 Proof of Theorem 1

Refering to Euler's Generalization on Fermat's Little Theorem, the multiplicative inverse of $x \bmod n$ is equal to $x^{\phi(n)-1} \bmod n$. The inverse of $(K_A)^{h_0} \bmod \phi(n_A)$ can be obtained as follows:

$$\begin{aligned}
&((K_A)^{h_0} \bmod \phi(n_A))^{-1} \\
&\equiv ((K_A)^{h_0})^{\phi(\phi(n_A))-1} \bmod \phi(n_A) \\
&\equiv ((K_A)^{h_0} \bmod \phi(n_A))^{\phi(\phi(n_A))-1} \\
&\equiv (X_A \bmod \phi(n_A))^{\phi(\phi(n_A))-1} \\
&\equiv (X_A \bmod \phi(n_A))^{-1} \qquad (9)
\end{aligned}$$

Thus,

$$\begin{aligned}
&((K_A)^{h_0})^{-1} \bmod \phi(n_A) \\
&\equiv (X_A)^{-1} \bmod \phi(n_A)
\end{aligned}$$

This means that the secret key for which the public key is $(K_A)^{h_0}$ is equal to the one for which the public key is X_A. ■

A.3 Proof of Theorem 2

Proof: According to RSA the receiver will obtain $(M)^{(K_A)^{h_0}(X_A)^{-1}} \bmod n_A$. Since $(X_A \bmod \phi(n_A))^{-1}$ is equal to $((K_A)^{h_0} \bmod \phi(n_A))^{-1}$, then

$$\begin{aligned}
&(M)^{(K_A)^{h_0}(X_A)^{-1}} \bmod n_A \\
&\equiv (M)^{(K_A)^{h_0}((K_A)^{h_0})^{-1}} \bmod n_A \\
&\equiv M \bmod n_A \qquad (10)
\end{aligned}$$

According to Eq. (10), the receiver will obtain the message M. This means that we can use $(X_A \bmod \phi(n_A))^{-1}$ to decrypt a message encrypted with $(K_A)^{h_0}$. ■

A.4 Proof of Theorem 3

Proof: According to the Carmichael Function,

$$\lambda(\phi(n_A) = lcm((p_1 - 1), (q_1 - 1))$$

for $n_A = p_1 \times q_1$. Let $(p_1 - 1) = 2^{k_1} \times x_1^{a_1} \times x_2^{a_2} \ldots \times x_s^{a_s}$ where $x_1, x_2 \ldots, x_s$ are prime numbers which divide $p_1 - 1$, and $a_1, a_2, \ldots, a_s$, s and k_1 are positive integers. Let $(q_1 - 1) = 2^{k_2} \times y_1^{b_1} \times y_2^{b_2} \ldots \times y_t^{b_t}$ where $y_1, y_2 \ldots, y_t$ are prime numbers which divide $q_1 - 1$, and $b_1, b_2, \ldots b_t$, t and k_2 are positive integers. Then

$$\begin{aligned}
&\lambda(p_1 - 1) \\
&= lcm(\lambda\,(x_1^{a_1}), \lambda\,(x_2^{a_2}), \ldots, \lambda\,(x_s^{a_s}), \lambda\,2^{k_1}) \\
&= lcm(\phi\,(x_1^{a_1}), (\phi\,(x_2^{a_2}), \ldots, (\phi\,(x_s^{a_s}), \\
&\quad (\phi\,(2^{k_1}))/2) \\
&= lcm((x_1^{a_1-1}(x_1 - 1)), (x_2^{a_2-1}(x_2 - 1)), \\
&\quad \ldots, (x_s^{a_s-1}(x_s - 1)), 2^{k_1-2}) \qquad (11)
\end{aligned}$$

Similarly:

$$\begin{aligned}
\lambda(q_1 - 1) \;=\; & (y_1^{b_1-1} - 1)(y_2^{b_2-1} - 1) \ldots \\
& (y_t^{b_t-1} - 1)(2^{k_2-2}) \qquad (12)
\end{aligned}$$

Using Eqs. (11) and (12):

$$\begin{aligned}
\lambda(\phi(n_A)) &= lcm(\lambda(p_1 - 1), \lambda(q_1 - 1)) \\
&= lcm(lcm((x_1^{a_1-1}(x_1 - 1)) \ldots \\
&\quad (x_s^{a_s-1}(x_s - 1))(2^{k_1-2})), \\
&\quad lcm((y_1^{b_1-1}(y_1 - 1)) \ldots \\
&\quad (y_t^{b_t-1}(y_t - 1))(2^{k_2-2}))) \qquad (13)
\end{aligned}$$

The minimum value of $\lambda(\phi(n_A))$ will be obtained if:

- All values of a_i, b_j for $i = 1, 2, \ldots, s$; $j = 1, 2, \ldots, t$ are equal to 1.
- 2^{k_1-2} and 2^{k_2-2} are similar and equal to 2.

Since $x_1, \ldots, x_s, y_1, \ldots, y_s$ are primes, then $x_1 - 1, \ldots, x_s - 1, y_1 - 1, \ldots, y_s - 1$ can be divided by 2, and the maximum value of Eq. (4) for 2 digit prime numbers of p_1 and q_1 is smaller than $1/2$. ■

A.5 Theorem 4

Theorem 4 An integer number x with the power of any positive number s (x^s) will be relatively prime to y, iff $gcd(x, y) = 1$ (x is relatively prime to y).

Proof: Let $x = r_1^{l_1} \times r_2^{l_2} \ldots \times r_z^{l_z}$ and $y = r_1^{t_1} \times r_2^{t_2} \ldots \times r_z^{t_z}$; where r_i, t_i, l_i and z are nonnegative numbers, for $i = 0, 1, 2, \ldots, z$. Then:

$$\begin{aligned}
gcd(x, y) = {} & r_1^{min(l_1,t_1)} \times r_2^{min(l_2,t_2)} \ldots \\
& \times r_z^{min(l_z,t_z)} \qquad (14)
\end{aligned}$$

$$\begin{aligned}
x^s &= (r_1^{l_1} \times r_2^{l_2} \ldots \times r_z^{l_z})^s \\
&= r_1^{l_1 s} \times r_2^{l_2 s} \ldots \times r_z^{l_z s} \qquad (15)
\end{aligned}$$

From Eqs. (14) and (15):

$$\begin{aligned}
gcd(x^s, y) = {} & r_1^{min(l_1 s,t_1)} \times r_2^{min(l_2 s,t_2)} \ldots \\
& \times r_z^{min(l_z s,t_z)} \qquad (16)
\end{aligned}$$

where $min(l_i s, t_i)$ is the minimum value of both numbers $l_i s$ and t_i. $gcd(x^s, y) = 1$ implies

$$\begin{aligned}
&min(l_1 s, t_1) = 0 \\
&min(l_2 s, t_2) = 0 \\
&\qquad\vdots \\
&min(l_z s, t_z) = 0
\end{aligned}$$

This means that $l_i s$ or t_i should be equal to 0, or in other words, there is no common divisor between x^s and y. This condition will happen if l_i or t_i are 0, because s is not zero. Now referring to Eq. (16), $gcd(x^s, y)$ will be equal to 1 if l_i or t_i is equal to 0. According to Eq. (14), if l_i or t_i is equal to 0, then $gcd(x, y)$ will be equal to 1. Thus, $gcd(x^s, y) = 1$ iff $gcd(x, y) = 1$. ■

(Received April 27, 1998)
(Accepted September 7, 1998)

Ari Moesriami Barmawi received the B.E. in Electronic Engineering from Bandung Institute of Technology (Indonesia), in 1985. She was a system engineer in Computer Centre of Indonesian Aircraft Industry during 1986–1992. Since 1993 she has been a lecturer of the Bandung Institute of Technology. She received M.E. in Computer Science from Keio University in 1997. She is currently a Ph.D. Candidate in Computer Science at Keio University. Her interest is in computer security.

Shingo Takada received the B.E. in Electrical Engineering, and M.E. and Ph.D. in Computer Science from Keio University, Yokohama, Japan in 1990, 1992, and 1995, respectively. He is currently a Research Associate at Nara Institute of Science and Technology. His interests include information search, cognitive science and software engineering. He is a member of the Information Processing Society of Japan, Japan Society for Software Science and Technology, Japanese Society for Artificial Intelligence, Association for Computational Linguistics, and ACM.

Norihisa Doi has been a professor in Department of Computer Science at Keio University since 1986, and has been the director of the Institute of Computer Education at university since 1991. He received a B.E., M.E., and Ph.D. in computer science from Keio University in 1964, 1966, and 1975, respectively. During 1975–1976, he was a visiting faculty member in Department of Computer Science at Carnegie-Mellon University. In 1976, he was a visiting professor in Computer Communication Networks Group at University of Waterloo. Currently, he is a member of Science Council of Japan, the chairman of National Committee for Informatics, a chairman of OMG (Object Management Group) Japan-SIG, the chairman of Technical Committee of Information Technology Promotion Agency (IPA). He has written many research papers and books, including "Introduction to the Language C", "Functions and Organization of Operating Systems", "How to Program" (Iwanami Shoten); "Introduction to PASCAL", "Introduction to FORTRAN 77" (Baifukan); and "Computer Security" (Kyouritu Shuppan).