Constructing a Secure Point-to-Point Wireless Environment by Integrating Diffie-Hellman PKDS RSA and Stream Ciphering for Users Known to Each Other

Yi-Li Huang and Fang-Yie Leu
TungHai University
Taichung, Taiwan
{yifung,leufy}@thu.edu.tw

Abstract

Recently, wireless communication has been popularly used in our everyday life. However, its security is a crucial work since messages are broadcasted via wireless channels through which hackers can intercept the massages and then know the contents of the messages. That is why security has been one of the most important issues in wireless communication studies. Encrypting messages to ciphertext is one of the methods to solve this problem. But, it also causes another problem, i.e., how to safely and confidentially encrypt and deliver security keys so that hackers have to spend a very long time before they can decrypt the ciphertext. In this paper, we proposed an authentication approach, called the secure point-to-point encryption method (SePem for short), which integrates RSA, Diffie-Hellman PKDS and a stream cipher technique to provide users with a highly secure point-to-point wireless network without requiring a CA. According to the security analysis of this study, the SePem can efficiently and securely protect a wireless environment. Our simulation results show that the performance of this method can meet users' communication needs.

Keywords: Diffie-Hellman PKDS, RSA, Stream cipher technique, Pseudo random number sequence, Pseudo random number generator, Increasing-doublingable sequence ladder

1 Introduction

In recent years, wireless communication due to rapid hardware cost reduction and providing its devices with portability has become one of the most important communication methods in our everyday life. Many people communicate with others through wireless environment almost everyday. However, from privacy viewpoint, wireless security is a crucial work since messages are delivered to their destinations through the air so hackers can maliciously intercept the massages and decrypt the messages.

In 1976, Diffie and Hellman [1] proposed the concept of a public key distribution system (PKDS for short) which is a public key cryptosystem, and with which two users could individually generate the same secret key by mutually exchanging their public keys through public network channels without directly delivering the private key.

On the other hand, a single key encryption can truly prevent a message's contexts from being known to hackers. However, once the key, very often a fixed-length key, has been decrypted by hackers, the hackers will realize what information is conveyed in the message. Stream cipher techniques [2] have been proved that they can provide ciphertext with higher security levels than those provided by using only one single key since they encrypt a message with a stream. It is more difficult for hackers to solve a steam than to solve a key.

In 1978, Rivest, Shamir and Adleman (RSA)[3], proposed an exponential function as a one-way Trapdoor Function based on factorization. Currently, the RSA is one of the most popular and famous asymmetric encryption/decryption systems, with which site A publicizes its public keys (e_A, N_A) . The

other side, e.g., site B, encrypts its key K with (e_A, N_A), as $RSA\_En(K, e_A)$ that will be sent to A. Only A with its own private key (d_A, N_A) can decrypt $RSA\_En(K, e_A)$. Therefore, even though $RSA\_En(K, e_A)$ is delivered through a wireless channel, K can be safely sent to A.

In order to provide a network with a higher level of data security, integrity and non-repudiation, in this paper, we propose a secure point-to-point encryption method (SePem for short) for a wireless communication environment. The SePem integrates the Diffie-Hellman PKDS [1] (DH-PKDS for short), RSA and a stream cipher technique without increasing required information exchange frequencies and burdens. It uses a seed as the input to trigger a pseudo random number generator (PRNG for short) which invokes the increasing-doublingable sequence ladder algorithm that we developed to generate a pseudorandom number sequence (PRNS). Messages, i.e., plaintexts, are encrypted by the PRNS. Note that this paper is an extended version of the one [4].

2 Background and Related Work

2.1 Diffie-Hellman Public Key Distribution System

DH-PKDS establishes a communication channel between two parties, A and B, in a network; it first makes public two integers: p and g where p is a big prime and g is the primitive root of p . Party A randomly selects a large number X_a which has the same number of bits as p has , and then produces Y_a which is party A 's public key generated from X_a . Similarly, Party B chooses its private key X_b and generates a public key Y_b . Then, the two parties mutually exchange Y_a and Y_b through a public communication channel without telling each other their own private keys. After that, party A and party B respectively compute K_a and K_b where

$$K_a = Y_b^{X_a} \bmod p = \left(g^{X_b}\right)^{X_a} \bmod p = g^{X_a X_b} \bmod p$$

and

$$K_b = Y_a^{X_b} \bmod p = \left(g^{X_a}\right)^{X_b} \bmod p = g^{X_a X_b} \bmod p$$

and $K_a = K_b$ which is called common secrete key (CSK for short). But what hackers can intercept in the air is only Y_a and Y_b , excluding g and p . So, it is impossible for the hackers to reversely calculate X_a and X_b . Even if both p and g are known to the hackers beforehand, it is still very difficult to calculate values of X_a from Y_a and X_b from Y_b since solving equation $Y_a = g^{X_a} \bmod p$ and $Y_b = g^{X_b} \bmod p$ is a discrete exponential logarithm problem [5].

However, today many supercomputers or Grid systems can help to decrypt possible X_a from Y_a and X_b from Y_b, even though the cost is high. When there is only one secret key, X_a or X_b will be relatively easier to be calculated from their corresponding public keys. Generally, a multiple-key encryption can raise the security level of a delivered message. That is why, in this study, we encode plaintext with a stream cipher encryption method.

2.2 RSA system

RSA is an asymmetric encryption/decryption system, also known as a two-secrete-key system, in which a public key can be publicized and a private key must not be known to others. A private key is used to decrypt ciphertext encrypted by its corresponding public key. It also cannot be solved from the corresponding public key, the operational process of RSA is as follows.

1. The generation of keys:

   (a) Calculate $N = p \times q$ where p and q are different large primes.

(b) Find public key e such that $\gcd(e,(p-1)(q-1))=1$, then (e,N) is the encryption key pair.

(c) Find a private key d such that $ed=1 \bmod ((p-1)(q-1))$, then (d,N) is the decryption key pair.

2. Encryption/decryption: Let M be the message to be delivered, and let C be the ciphertext of M after encryption, then

   (a) Encryption: $C=M^e \bmod N$

   (b) Decryption: $M=C^d \bmod N$

3. Implementation

   (a) The sender A sends its public key (e_A,N_A) to receiver B through a wireless channel.

   (b) B encrypts the key K which will be sent to A with (e_A,N_A), and the result is $RSA\_En(K,e_A)=K^{e_A} \bmod N_A$, which is then sent $RSA\_En(K,e_A)$ to A through a wireless channel.

   (c) A acquires key K by employing the decryption, $K=RSA\_En(K,e_A)^{d_A} \bmod N_A$

2.3 Stream Cipher Encryption

Stream cipher techniques have been widely used in wireless network [6][2]. However, how to design a random number generator to produce a high quality random number sequence is a challenge. In cryptography, a stream cipher method is a symmetric key cipher method since the receiver should generate the same cipher stream before the delivered ciphertext can be accurately decrypted. Its plaintext units, e.g., bits, are sequentially encrypted one at a time by using a PRNS, typically by an exclusive-or operation. That is, different plaintext units are encrypted with different elements of the cipher stream. Figure 1 shows the block diagram of a stream cipher system.

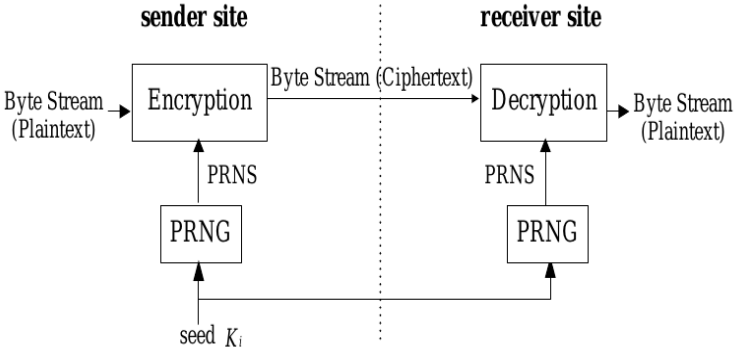


Figure 1: Block diagram of a stream cipher system.

2.4 Related Work

Several encryption mechanisms, e.g., Wireless Encryption Privacy (WEP for short) [7], WiFi Protected Access (WPA for short) [7] and Privacy Key Management Version 1 (PKMv1 for short) [8] have been proposed for wireless network.

In 1999, WEP as a part of the IEEE 802.11 standard uses RC4, one of the stream cipher algorithms, to encrypt messages. This mechanism encrypting data with a key of 64 bits was secure enough to resist

hackers' attacks before 2004. Today, its initialization vector (IV for short) is no longer long enough to protect wireless messages since now the traffic key 24 bits in length can be cracked easily.

WPA, which also uses RC4 as its encryption algorithm, was adopted by the WiFi Alliance consortium to substitute WEP. WPA uses two techniques to exchange keys. First, it employs a 48-bit IV, rather than a 24-bit IV, as its traffic key; second, it adopts Temporal Key Integrity Protocol to dynamically change the traffic key. This mechanism has a problem in that the management frames can be easily spoofed by malicious hackers since all sessions between end users and central control equipments are not robust and safe.

PKMv1, one of the secret mechanisms of IEEE 802.16 standard [9], is used to provide subscriber station (SS) and base station (BS) with a more secure environment for key distribution. Through the PKMv1, the SS and the BS can synchronize the authentication key (AK for short) and traffic encryption key (TEK for short) on both sites. However, in this mechanism only BS authenticates an SS, and the SS never verifies the identity of the BS, i.e., lacking a mechanism for mutual authentication before an AK is generated.

The DiHam [10][11] was developed based on the PKMv1 by improving the key exchange flow and providing different data security levels. Basically, its key exchange process consists of two phases, the authentication phase and TEK exchange phase. In the authentication phase, the AK is individually generated by the BS and MS after the delivery of the Authentication-Request message and Authentication-Reply message. In the TEK exchange phase, three security levels of TEK generation processes are proposed to meet different user security requirements. This phase starts when MS sends a TEK-Exchange-Request message to the BS, and ends when the BS replies with a TEK-Exchange-Reply message.

The HaKMA [12] as a three-layer authentication architecture is a new version of the DiHam by improving its key generation flow and adding a handover authentication scheme to respectively speed up the handover process and increase the security level for mobile stations (MSs). AAA server supported authentication is also enhanced by invoking an improved extensible authentication protocol (EAP).

3 System Architecture

In this study, as stated above, the SePem integrates the DH-PKDS, RSA and a stream cipher technique. Generally, applying multiple variables to a random number generation algorithm can effectively improve randomness quality of generated random numbers and then the security level of the yielded ciphertext since hackers have to calculate values for the variables one by one before they can decrypt the ciphertext. A larger number of unpredictable factors embedded in each PRNS can also raise randomness quality of the PRNS which can in turn generate a higher security-level message.

3.1 A Random Number Generator Using Increasing Doublingable Sequence Ladder

Our random number generator is developed based on the abovementioned principles. In brief, the generator treats an initial triggering sequence $X^{(1)}$, also known as a seed, as a unit before using chosen elements in the sequence to produce the first random number ψ_1 . Next, the generator follows a certain rule to convert a triggering key sequence $X^{(t-1)}$ into $X^{(t)}$, $t \geq 2$, making a generated PRNS complicated and unpredictable. The parameters of the PRNS are defined as follows.

1. PRNS $\psi = (\psi_1, \psi_2, \psi_3, ..., \psi_q)$, $0 \leq \psi_i \leq 255$, $1 \leq i \leq q$, $q \in$ N where $\psi_i \leq 255$ represents the PRNS is a byte stream. The value of q should meet the length of the plaintext since plaintext bits are encoded one by one. We suggest that n ranges between 5 and 16 (i.e., $5 \leq$ n ≤ 16) since secret keys, public keys, p and q are all 1024 bits in length, and each $x^{(t)}(k)$ is 64 bits in length where 1024 = 64 bits * 16.

2. The initial triggering sequence $X^{(1)}$ is a positive real-number sequence where

$$X^{(1)} = (x^{(1)}(1), x^{(1)}(2), \ldots, x^{(1)}(n)),\ x^{(1)}(k) \in R^{+}, 1 \le k \le n, n \in N \tag{1}$$

Algorithm 1 shows how a PRNS is produced.

Algorithm 1: Producing a PRNS {
Input: the initial triggering sequence $X^{(1)} = (x^{(1)}(1), x^{(1)}(2), \ldots, x^{(1)}(n))$ is a sequence of n elements.
Output: PRNS $\psi = (\psi_1, \psi_2, \psi_3, \ldots, \psi_q)$, a PRNS of q elements.

Step 1: Initialization

1.1 $t = 1$;
1.2 $X^{(1)} = (x^{(1)}(1), x^{(1)}(2), \ldots, x^{(1)}(n))$;
1.3

$$\psi_1 = \lceil x^{(1)} * x^{(1)}(\lfloor \frac{n}{2} \rfloor) * x^{(t)}(n) \rceil \bmod 256 \tag{2}$$

Step 2:

2.1 $t = t + 1$
2.2 $X^{(t)} = (x^{(t)}(1), x^{(t)}(2), \ldots, x^{(t)}(n)$, where

$$x^{(t)}(k) = \begin{cases} x^{(t-1)}(k) + x^{(t-1)}(n-k-1), & 1 \le k \le \lfloor \frac{n}{2} \rfloor \\ \sum_{j=1}^{k} x^{(t-1)}(j), & \lfloor \frac{n}{2} \rfloor < k \le n \end{cases} \tag{3}$$

/* producing an increasing doublingable-sequence ladder */
2.3

$$\psi_t = \lceil x^{(t)}(1) * x^{(t)}(\lfloor \frac{n}{2} \rfloor) * x^{(t)}(n) \rceil \bmod 256 \tag{4}$$

Step 3: If($\max(x^{(t)}(1), x^{(t)}(2), \ldots, x^{(t)}(n)) \ge 256$) then

$$x^{(t)}(k) = \lceil x^{(t)}(k) + \psi_t * k \rceil \bmod 26 + \frac{10 * k}{\psi_t + 3} \tag{5}$$

where $\frac{10*k}{\psi_t+3}$ rounds off at the 10 th digit below the decimal point, $1 \le k \le n$.

Step 4: If ($t < q$) then goto Step 2;}

In this algorithm, each random number is created by a mod operation (see steps 1.3 and 2.3) for the production of $x^{(t)}(1)$, $x^{(t)}(\lfloor \frac{n}{2} \rfloor)$, and $x^{(t)}(n)$ (respectively the first, middle, and last elements of the triggering sequence $X^{(t)}$, $t = 1, 2, 3, \ldots, q$) to achieve the uniqueness of each element in the PRNS, denoted by ψ . For each t, $q \ge t \ge 2$, a new triggering sequence $X^{(t)}$ is then generated by two types of summing the selected elements from its predecessor $X^{(t-1)}$ (see step 2.2), resulting in larger values for the elements of $X^{(t)}$. When $X^{(t)}$'s element values develop to a certain degree (e.g. ≥ 256), the $X^{(t)}$ elements need to be readjusted or retraced with extra unpredictable. Otherwise an unfavorable collapse of randomness may occur to shorten the period of the PRNS sequences. We solve this problem by employing Eq. (5), i.e. ,

$$\lceil x^{(t)}(k) + \psi_t * k \rceil \bmod 26 + \frac{10 * k}{\psi_t + 3}$$

for $x^{(t)}(k)$. What we would like to do is increasing the randomness and larger unpredictability between two arbitrary generated PRNS sequences.

3.2 Inegration of DH-PKDS and Stream Cipher

Generally, the integration of DH-PKDS, RSA and the stream cipher technique can also conduct more secure data transmission. Before introducing the integration algorithm, we define parameters being used first, including

1. p_i and p (both 512 bits or longer): Strong primes, respectively, for initial encryption and message encryption;
2. g_i and g (both 64 bits or longer): Primitive roots of p_i and p, respectively, for initial key and message encryption;
3. Parties A and B : End users A and B, respectively;
4. $X_{a,i}$ and X_a : Party A 's private keys;
5. $X_{b,i}$ and X_b : Party B 's private keys;
6. $Y_{a,i}$ and Y_a : Party A 's public keys;
7. $Y_{b,i}$ and Y_b : Party B 's public keys;
8. $K_{a,i}$ and $K_{b,i}$: Respectively, parties A 's and B 's seeds used to trigger the PRNG to generate PRNSs;
9. K_a and K_b : Respectively, parties A 's and B 's common secret keys;
10. e_A, d_A, N_A : Party A's RSA triple keys;
11. e_B, d_B, N_B : Party B's RSA triple keys;
12. Note that the lengths (i.e., numbers of bits) of $X_{a,i}$, X_a, $X_{b,i}$, X_b, $Y_{a,i}$, Y_a, $Y_{b,i}$, Y_b, $K_{a,i}$, K_a, $K_{b,i}$ and K_b, N_A, and N_B are all the same as those of p_i and p ;
13. Plaintext A and plaintext B : Respectively, a document that party A sends to party B, and a document that party B sends to party A ;
14. *IdA* and *IdB*: Identification of party A and identification of party B, respectively, which are used by users to recognize who is communicating with him/her.

Algorithm 2 shows the details of how to exchange keys being used.
Algorithm 2: Exchanging keys {

1. Party A sends
$$Y_{a,i} = g_i^{X_{a,i}} \bmod p_i, \tag{6}$$
(e_A, N_A) and *IdA* to party B via the wireless network;
2. Party B on receiving $Y_{a,i}$, (e_A, N_A) and *IdA* proceeds with the following calculations.
   (a)
   $$Y_{a,i} = g_i^{X_{a,i}} \bmod p_i, \tag{7}$$
   (b)
   $$K_i = K_{b,i} = Y_{a,i}^{X_{b,i}} \bmod p_i = g_i^{x_{a,i}x_{b,i}} \bmod p_i \tag{8}$$

(c) Generating new parameters, p, g, and X_b, for further communication;

(d)
$$Y_b = g^{X_b} \bmod p \tag{9}$$

(e) Plaintext $B = IdA + (p, g, Y_b) + (e_B, N_B) + IdB$; /* IdA and IdB are used to authenticate the correct communication parties */

(f) Encrypting plaintext B with PRNS generated by PRNG triggered by K_i to generate ciphertext B, where K_i is PRNG's initial triggering sequence $X^{(1)}$; /* K_i is generated in every session before party A or B wishes to communicate with each other */

(g)
$$RSA\_En(Y_{b,i}, e_A) = Y_{b,i}^{e_A} \bmod N_A \tag{10}$$

(h) Party B sends $RSA\_En(Y_{b,i}, e_A)$ and the ciphertext B to party A ;

3. Party A on receiving $RSA\_En(Y_{b,i}, e_A)$ and ciphertext B proceeds with following calculations

(a)
$$Y_{b,i} = RSA\_En(Y_{b,i}, e_A)^{d_A} \bmod N_A \tag{11}$$

(b)
$$K_i = K_{a,i} = {Y_{b,i}}^{X_{a,i}} \bmod p_i \tag{12}$$

(c) Decrypting ciphertext B with the PRNS generated by the PRNG triggered by K_i to recover plaintext B ;

(d) Generating a new parameter X_a for further communication;

(e)
$$Y_a = g^{X_a} \bmod p \tag{13}$$

(f)
$$K = K_a = {Y_b}^{X_a} \bmod p \tag{14}$$

(g)
$$RSA\_En(Y_a, e_B) = Y_a^{e_B} \bmod N_B \tag{15}$$

(h) Plaintext A $= IdA +$ Messages to be delivered $+ IdB$;

(i) Encrypting plaintext A with the PRNS generated by the PRNG triggered by K to generate ciphertext A ;

(j) Party A sends $RSA\_En(Y_a, e_B)$ and ciphertext A to party B ;

4. Party B on receiving Y_a and ciphertext A proceeds with the following calculations.

(a)
$$Y_a = RSA\_En(Y_a, e_B)^{d_B} \bmod N_B \tag{16}$$

(b)
$$K = K_b = (Y_a)^{X_b} \bmod p \tag{17}$$

(c) Decrypting ciphertext A with the PRNS generated by the PRNG triggered by K to recover plaintext A ;}

Finally, by sharing the RSA encryption/decryption keys and the common secret key, party A and party B are the only parties capable of performing the encryption and decryption of IdA and IdB, respectively. IdA and IdB are used by the two parties to recognize that the received messages are truly sent by the other party. In other words, the quality and security of the network communication have been improved by the SePem.

4 Security Analysis

The relationship between sender A and receiver B can be one of the two cases. They are well-known and not well-known to each other.

In step 1, sender A has to send its own initial key information to receiver B. Since both sides do not have any previously established keys, symmetric or asymmetric, no encryption can be done in current stage. Otherwise, B does not know how to decrypt the received message. However, the delivery of unencrypted message will be dangerous. In fact, if both users A and B are familiar with each other, A can add a secret symbol into *IdA*. Then B can make sure that the message is issued by A or not. Of course, it will be better if the secret symbol is changed frequently.

Even though hackers may try to solve $X_{a,i}$ from maliciously intercepted $Y_{a,i}, g_i$, and p_i, due to solving an exponential logarithmic algorithm, it is very difficult and time consuming for hackers to crack the DH-PKDS. The time complexity of cracking public key $Y_{a,i}$ so as to further solve private key $X_{a,i}$ is O($e^{C\sqrt{\ln(p_i)\ln(\ln(p_i))}}$) [13].

If hacker really solves $X_{a,i}$, he/she needs $B's$ public key $Y_{b,i}$ before deriving the common secret key K_i . But in step 2.8, when $Y_{b,i}$ is delivered to A by B, it is protected by RSA. Only $A's \quad (d_A, N_A)$ can decrypt encrypted K_i where the decryption formula is Eq. (11), i.e., $Y_{b,i} = RSA\_En(Y_{b,i}, e_A)^{d_A} \ mod \ N_A$. Without d_A, hackers are unable to acquire $Y_{b,i}$ and K_i . Hence, even though $X_{a,i}$ has been successfully derived by hackers, they cannot obtain $Y_{b,i}$ and K_i . On the other hand,, if hackers wish to acquire (d_A, N_A) from N_A, the problem they should face is a factorization function, which is not easy to be solved [14][15]. Further, different users often use different RSA Triple Keys (e_A, d_A, N_A) . It is inefficient if hackers choose several RSA Triple Keys and solve them one by one. Nevertheless, all users have to periodically update their RSA Triple Keys. The purpose is to raise the security level of the protected system.

When $Y_{b,i}$ has been well protected, and no hackers can easily acquire $Y_{b,i}$, the security of the common secret key K_i will be also higher. Without $Y_{b,i}$ and K_i, there are no ways for hackers to trigger the PRNG so as to generate a PRNS. The ciphertext B generated will be also safe since without d_A, hackers cannot solve $Y_{b,i}$ and K_i .

When ciphertext B is well protected, the dynamic keys (p, g, Y_b, e_B, N_B) contained in it are also safe. Without the dynamic keys, the following encrypted messages delivered will be well protected. If there is a forged receiver, i.e., hacker B', and users A and B are not very familiar with each other (i.e., case 2), the message that hacker B' sent to A contains no their secret symbol or a wrong secret key, A can realize that there is a forged B, i.e., hacker B'. In case 1, A can recognize the basic information included in *IdA* of user B, then a forged message can be identified.

5 Simulation Environment and Discussion

5.1 Simulation Model

The simulations were performed in a client-server environment. The program is developed by using java script. With our proposed scheme, the plaintext is encrypted and then sent to the server i.e., BS, via three different types of wireless channels, including 802.11b without any encryption methods, High Speed Download Packet Access (3.5G) and WiMax. We measured the timings of our key delivery in each wireless environment.

Table 1: the numbers of different operations, including modular exponential, addition/subtraction, and the invocation of PRNG of Algorithm 2.

Step	Exponential operation	Add/subtract operation	PRNG
1	1	1	0
2	4	4	1
3	5	5	2
4	2	2	1
Total	12	12	4

5.2 Performace Analysis

5.2.1 Time complexity

In this study, there are two algorithms, PRNG and key exchange.

1. PRNG
   The time complexity of the PRNG can be analyzed as follows.

   (a) In the initialization, the algorithm spends four multiplication/division operations.

   (b) In step 2.2, there are $(3n/2)-1(=n/2+(n+1))$ addition/subtraction operations, and in step 2.3, there are four multiplication/division operations.

   (c) In step 3, the number of addition/subtraction operations is three and the number of multiplication/division operations is four. But according to our experiment, Eq. (5) $x^{(t)}(k) = \lceil x^{(t)}(k) + \psi_t * k \rceil \bmod 26 + \frac{10*k}{\psi_t+3}$, is executed once about five times of iteration.

   The time of iteration between steps 2 and 3 is q-1. Therefore, the cost of the algorithm, denoted by C_{PRNG}, is

$$\begin{aligned} C_{PRNG} = (4+4(q-1)+4\lfloor\frac{q-1}{5}\rfloor)&\text{multiplication/subtraction}+ \\ ((q-1)((3n/2)-1)+3\lfloor\frac{q-1}{5}\rfloor&\text{addition/subtraction} \\ \approx \left(4\text{q+4}\left\lfloor\frac{q-1}{5}\right\rfloor\right)&\text{multiplication/subtraction} \end{aligned} \tag{18}$$

2. Key exchange

   Table 1 lists the operations invoked by the process of key exchange. From this table, we can realize that we employ the PRNG four times, and require twelve modular exponential operations and twelve addition/subtraction operations.

   Table 2 lists the numbers of exponential operations that the SePem, DiHam [10][11] and HaKMA [12][11] require. We can see the number of the SePem is between that of the DiHam and that of the HaKMA since the Diham invokes many exponential operations to generate AKs. However, the HaKMA is used in the WiMAX only.

Table 2: the comparison of the SePem, DiHam and HaKMA

Security scheme	Modular Exponential operation
DiHam with level-1 TEK	8
DiHam with level-2 TEK	13
DiHam with level-3 TEK	83
HaKMA with EAP-AKA	7
SePem	12

5.2.2 Simulation Analysis

Table 3 lists the results. We found that in each step the times required to generate the ciphertexts are short. Here, we choose IEEE 802.11b as an example. The costs of steps 1-3 are 0.509 sec, 0.179 sec and 0.523 sec, respectively. In the last step, we only spent 82.913 ms to encode a 1200-bit plaintext.

Table 3: The simulation results and the lengths of parameters used in each tested environment.

Step	1 (sec)	2 (sec)	3 (sec)	4 (ms)
802.11b with 512bits	0.509	0.179	0.523	82.913
HSDPA with 1024bits	2.544	8.594	0.303	30.938
WIMAX with 1024bits	3.754	0.468	1.967	29.838

Now, we can conclude that the SePem has many benefits. First, the system can be adapted to any wireless networks that originally need to deliver encrypted TEKs or other keys. Second, it spends very short time to generate ciphertexts on each key exchange step. At last, the proposed scheme is very secure since K_i, K, and the two PRNSs are all individually generated by parties A and B . Besides, current sizes of parameters are between 512 bits and 1024 bits in length. We can use a longer size, like 2048 bits, for an advanced use. In summary, we can conclude that our mechanism is very suitable for implementing key management and delivery in wireless environments.

6 Conclusions and Future Work

The most significant benefits of wireless communication are mobility and usage convenience, which may be accompanied with a drawback, i.e., hackers may maliciously intercept the delivered messages so that how to protect wirelessly transmitted data is one of the key issues in information security. In this study, we integrate the Diffie-Hellman PKDS, RSA and a stream cipher technique to protect Diffie-Hellman PKDS's public keys $Y_{b,i}$ and Y_a by employing the RSA, and generate a PRNS by using a PRNG triggered by K_i and K to cipher messages. On the other hand, the employment of user identities *IdA* and *IdB* can avoid a forged attack so as to achieve a high-level of security for wireless environments. Further, the cost of the SePem is not high, but it offers high security level, i.e., the SePem is worth to be popularly used.

By using the technique proposed, we have introduced a system of high complexity and unpredictability, with which positive real values are consecutively and increasingly created as the final output, i.e., random numbers. 98.6% of the random numbers produced pass 600,000 random tests provided by the Federal Information Processing Standards Publication (FIPS PUB) 140-2 [16], showing that this random

number generator is one with superb security, further guaranteeing a great security of the encryption/decryption of the SePem which uses our PRNG to produce a PRNG.

Furthermore, this security mechanism is powerful to correct some weaknesses of point-to-point mode in current wireless environments, not only let IV more safely be transmitted from user A to user B, but also achieve an authorization without Certificate Authority.

In our system, a single PRNG is used to produce a PRNS. If multiple PRNGs are employed and the generated PRNSs are of high quality and unrelated to other PRNSs, crypto-analyzers have to find out which PRNG is now being used, even if they already know the correct private keys $X_{a,i}$, $X_{b,i}$, X_a and X_b . This will further improve the system security. Also, we would like derive the formal reliability and behavior models for the SePem so users can realize the reliability and behavior of the system before using it. These constitute our future research.

References

[1] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 638–654, November 1976.

[2] Y. Xiao, H.-H. Chen, X. Du, and M. Guizani, "Stream-based cipher feedback mode in wireless error channel," *IEEE Transactions on Wireless Communications*, vol. 8, no. 2, pp. 622–626, February 2009.

[3] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, February 1978.

[4] Y.-F. Huang, K.-C. Wei, and F.-Y. Leu, "Constructing a secure point-to-point wireless environments by integrating Diffie-Hellman PKDS and stream ciphering without certificate authorities," in *Proc. of the 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'10), Krakow, Poland*. IEEE, February 2010, pp. 384–390.

[5] J. Zhang and L. Chen, "An improved algorithm for discrete logarithm problem," in *Proc. of the 2009 IEEE International Conference on Environmental Science and Information Application Technology (ESIAT'09), Chania, Crete, Greece*. IEEE, July 2009, pp. 658–661.

[6] Y.-F. Huang, C.-H. Lin, and K.-L. Wen, "A pseudo-random number generator based on grey system theory," *Far East Journal of Mathematical Sciences*, vol. 35, no. 1, pp. 1–17, September 2009.

[7] M. Shin, J. Ma, A. Mishra, and W. Arbaugh, "Wireless network security and interworking," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 455–466, February 2006.

[8] IEEE, "Ieee standard for local and metropolitan area networks part 16: Air interface for fixed broadband wireless access systems, ieee std 802.16-2004," http://standards.ieee.org/findstds/standard/802.16-2004.html, 2004.

[9] M. Barbeau, "WiMax/802.16 threat analysis," in *Proc. of the 1st ACM International Workshop on Quality of Service & Security in Wireless and Mobile Networks (Q2SWinet'05), Montreal, Canada, Canada*. ACM, October 2005, pp. 8–15.

[10] Y.-F. Huang, F.-Y. Leu, C.-H. Chiu, and I.-L. Lin, "Improving security levels of IEEE802.16e authentication by involving Diffie-Hellman PKDS," *Journal of Universal Computer Science*, vol. 17, no. 6, pp. 891–911, March 2011.

[11] F.-Y. Leu, Y.-F. Huang, and C.-H. Chiu, "Mutual authentication with dynamic keys in an IEEE802.16e PKM environment without prior authentication connection," in *Proc. of the 1st IEEE International Workshop on Cloud, Wireless and e-Commerce Security (CWECS'10), Fukuoka, Japan*. IEEE, November 2010, pp. 441–446.

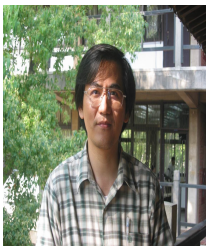
[12] Y.-F. Ciou, F.-Y. Leu, Y.-L. Huang, and K. Yim, "A handover security mechanism employing Diffie-Hellman key exchange approach for ieee802.16e wireless networks," *Mobile Information Systems*, vol. 7, no. 3, pp. 241–269, September 2011.

[13] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, July 1985.

[14] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over GF(p) and its cryptographic significance," *IEEE Transactions on Information Theory*, vol. 24, no. 1, pp. 106–111, January 1978.

[15] M. Wiener, "Cryptanalysis of short RSA secret exponents," *IEEE Transactions on Information Theory*, vol. 36, no. 3, pp. 553–558, May 1990.

[16] Information Technology Laboratory, National Institute of Standards and Technology, "Security requirements for cryptographic modules: Fips pub 140-2," hhttp://csrc.nist.gov, May 2001.



Yi-Li Huang received his master degree from National Central University of Physics, Taiwan, in 1983. His research interests include security of network and wireless communication, solar active-tracking system, pseudo random number generator design and file protection theory. He is currently a senior instructor of Tunghai University, Taiwan, and director of information security and grey theory laboratory of the University.

Fang-Yie Leu received his BS, master and Ph.D. degrees all from National Taiwan University of Science and Technology, Taiwan, in 1983, 1986 and 1991, respectively, and another master degree from Knowledge System Institute, USA, in 1990. His research interests include wireless communication, network security, Grid applications and Chinese natural language processing. He was invited to be a guest editor of Mobile Information Systems Journal, Journal of Ambient Intelligence and Humanized Computing and Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications. He is currently a workshop organizer of CWECS and MCNCS workshops, one of the program committee members of at least 10 international conferences, a full professor of Tung-Hai University, Taiwan, and a director of database and network security laboratory of the University. He is also a member of IEEE Computer Society.