Journal of Engineering and Applied Sciences 1 (2) : 98-102, 2006
© Medwell Online, 2006

Private-Key Generation for Secured Electronic Transactions

M. Ismail Jabiullah and M. Lutfar Rahman
Department of Computer Science and Engineering University of Dhaka, Dhaka, Bangladesh

Abstract: A new private-key generation technique has been designed, developed and implemented. The technique is based on three-level composite function. A randomly taken and uniquely permuted vector has been taken as input of the first function and produced output as an integer form. The first function's output and a matrix form are taken as the input of the secured function and produced output an integer that forms another matrix. Taking the above two functions as input parameter, the third function produced the final matrix as the output. The third matrix is the produced private-key and has been used to generate ciphertext from the communicating plaintext. The complexity of the generating private-key has also been measured and found very high for the intruder.

Key words: Session-keys, cryptography, encryption, decryption, composite function and complexity

INTRODUCTION

Encryption is the key principal for secure transmission of electronic message. Two fundamental approaches for encryption are in use: conventional encryption and public-key encryption[1]. The present work concerns generation of a new method for generation of private keys for secure transmission of electronic messages. With conventional encryption two parties share a single encryption/decryption key. The principal challenge with conventional encryption is the distribution and protection of the keys. Conventional encryption is used to encrypt transmitted data, using a one-time or short-term session-key. The session-key can be distributed by a trusted key distribution center or transmitted in encrypted form using public-key encryption. A conventional encryption scheme has five ingredients. The original message or data called the plaintext is encrypted, using an algorithm and a key. The encryption algorithm performs various substitutions and transformations on the plaintext by using the key. The exact substitutions and transformations performed by the algorithm depend on the key. Ciphertext, the scrambled message produced out of the encryption process, is then transmitted. The ciphertext it depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. Decryption algorithm is essentially the encryption algorithm turn in reverse. It takes the ciphertext and the secret key and produces the original plaintext. An enemy, or intruder, may accurately copy down the complete ciphertext. But, unlike the intended recipient, the intruder does not know what the decryption key is and so cannot decrypt the ciphertext easily. Sometimes the intruder can not only listen to the communication channel but can also record messages and play them back later, inject new messages, or modify legitimate messages before they get to the receiver. It will often be useful to have a notation for relating plaintext, ciphertext and key. Here $C = E_K(P)$ has been used to mean that the encryption of the plaintext P using key K gives the ciphertext C. Similarly $P = D_K(C)$ represents decryption of C to get the plaintext again. It then follows that $D_K (E_K (P)) = P$.E. and D are just mathematical functions. Here, both the functions are of two parameters and one of the parameters has been written as a subscript, rather then as an argument, to distinguish it from the message. There are two requirements for secure use of conventional encryption. They are (a) the algorithm should be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. (b) Sender and receiver must obtain copies of the secret key or private-key in a secure fashion and must keep the key secure. If someone can discover the private-key and knows the algorithm, all communication using this key is readable[2].

Many techniques of private-key generation can be used for secure communications. The two most popular conventional encryption techniques are DES and triple DES, but the ultimate goal is to increase the computational complexity for intruder to match plain text from ciphertext. In this study, the mathematical formula has been used for generating the short-term session key. For this purpose, one of the composite-type functions has been used. Emphasizing on the matrix complexity, we have taken vector and matrix as inputs. If the level of the function is increased and the function takes the input in the matrix form then the complexity will be increased. In this study, a key generation process on the basis of composite function has been formalized. The hardness, i.e., computational complexity is measured using the size of the key.

KEY GENERATION METHODOLOGY

Outline: Three levels of function have been used. First function takes the input as a vector, which is randomly unique, permuted and it gives the output in an integer form, which is also the input of the second function. The second function also takes another input, which is in the matrix form. Taking these two inputs second function gives an output in the integer from which generates the matrix. Taking the above two functions as parameters, third function gives the final matrix. This is the key. This third matrix has been used to generate the ciphertext. Here the complexity is dependent on both the level of composition and on the size of the matrix. The first vector is randomly permutated, the second matrix is also randomly permutated and also each row of this matrix is unique from each other. The third matrix is also randomly permuted and each column is different from each other. So the number of comparisons increases with this property for matching ciphertext with plaintext. Considering the table lookup technique, some functions have been used to generate the table or matrix. And every element of the matrix is used to forward or backward substitution.

Mathematical analysis: Consider the functions $f_1(x)= z_1$, $f_2(x_2,y_2) = z_2$, $f_3(x_3,y_3) = z_3$, where all the values are integers with $1<=x_i, y_i, z_i <=N$. Function f_1 can be represented by a vector M_1 of length N. That is $|M_1| = N$ in which kth entry is the value $f_1(k)$. Similarly, functions f_2 and f_3 can be represented by $N\times N$ square matrices M_2 and M_3, respectively.

Construction scheme: Construct M_1 with a random permutation of all integers between 1 and N. That is, each integer appears exactly once in M_1, construct M_2 so that each row contains a random permutation of the first N integers. Finally, M_3 satisfy the following condition $f_3(f_2(f_1(k),p)k) = p$, for all k, p with $1< = k, p< = N$.

Algorithms for key generation: Three algorithms named RPVector, RPMatrix and ResultPMatrix have been defined that are used for vector generation, matrix generation and key generation, respectively. They are given as follows:

```
RPVector (M1, N):
M1 is a vector defined by the size N and both are
global. This algorithm takes N and gives M1 vector,
which is randomly unique permuted and the
elements of M1 are in 1<= element <=N. [3]
1:{Flag $\leftarrow$ 0, count $\leftarrow$ 1
2: do {
3:    Generate a random number "temp"   where
   $1\le$ temp $\le$ N
4:    if tem $\ne$ 0 then
5:    {for i $\leftarrow$ 1 to count{
6: if $M_1$[i] = temp then
7: flag $\leftarrow$ 0 and break
8: else flag $\leftarrow$ 1}}
9:if (flag = 1) then{
10: M1[count] $\leftarrow$ temp
11:  count $\leftarrow$ count + 1
12: flag $\leftarrow$ 0}}
13: while count $\le$ N
14: return}
```

```
ResultPMatrix (M1, M2, M3, N)
M1 is an N size vector, $M_2$ and $M_3$ are N x N
matrices. This algorithm takes $M_1$, $M_2$ and N as input
and produces N x N square matrix $M_3$.
1:{call RPVector($M_1$, N):
2: call RPMatrix($M_2$, N): {
3:  for k $\leftarrow$ 1 to N{
4: for p $\leftarrow$ 1 to N {
5: t1 $\leftarrow$ M1[k]
6: t2 $\leftarrow$ M2[t1, p]
7:    M3[t2,k] $\leftarrow$ p }}
8:    exit}
```

Implementation of the algorithm: To implement this algorithm C language has been used, because it has several advantages over the other languages. The program has taken as integer as input and produced: (I) a randomly permuted N size M_1 vector whose elements are between 1 and N, (ii) an N by N matrix M_2 whose rows are randomly unique permuted and (iii) using these two, it will also produce M_3 whose columns are also randomly unique permuted and also different from each other. The finally produced matrix M_3 is used as the private-key.

RPMatrix (M_2, N):
M_2 is an N x N matrix and both M_2 and N are global. This algorithm takes N as input and produces N x N matrix whose rows are uniquely random permuted and are different from each other.

```
1: {Flag ← 0, count1 ← 1{
2: do{
3: count ← 1
4: Generate a random number temp where 1 ≤ temp ≤ N
5: for i ← 1 to count1{
6: if $M_2$[i, count] = temp then
7: flag1 ← 0 and break
8: else {
9: flag1 ← 1
10: flag2 ← 0}}
11: do{
12: if flag1 = 0 then
13:  break
14: if flag2 = 1 then
15: Generate a random number temp where 1 ≤ temp ≤ N
16: if temp ≠ 0 then{
17: for i ← 1 to count{
18: if $M_2$[count1,i] = temp then
19: flag ← 0 and break
20: else
21: flag ← 1}}
22: if (flag = 1) and (flag1 =1) then {
23: $M_2$[count 1, count] ← temp
24: count ← count +1
25: flag ← 0
26: flag2 ← 1}}
27: while count ≤ N
28: if (flag1 = 1) then
29: count1 ← count1 + 1}
30: while count1 ≤ N
31: return}}
```

Each time the program will execute it will produce different keys on the basis of its input integer. This is the session-key. The program developed in C using the above algorithms, is executed in an IBM compatible PC having Pentium II processor, 64MB RAM, SVGA monitor, 8 GB HDD.

ENCRYPTION AND DECRYPTION USING THE KEY

On each character of the input plaintext, forward and backward substitution occurred according to the element of the key matrix by the row and column basis. So, if an intruder wants to match the original plaintext, he/she has to generate the original key that is very time consuming on the basis of key size and message.

Encryption Algorithm:
This algorithm takes previously generated M_3 key matrix, N the size of the matrix and an input plaintext file in. It produces ciphertext file out. The steps for the algorithm are as follows:

```
1: {Open a file  in as input and out as output
2: count ← 0
3: while not(eof(in)
4: {ch ← getchar(in)
5: if count = n
6: count ← 1
7: else
8: count ← count+1
9: for i ← 1 to n{
10: if i mod 2=1
11: ch ← ch + $M_3$[i][count]
12: else
13: ch ← ch-$M_3$[i][count]; }
14: for i ← 1 to n{
15: if i mod 2 = 1
16: ch ← ch + $M_3$[count][i]
17: else
18: ch ← ch-$M_3$[count][i] }
19: write ch to the output file out}
20: exit}
```

Decryption algorithm:
This algorithm takes previously generated M_3 key matrix, N the size of the matrix and an input ciphertext file in. It produces original plaintext file out. The steps for the algorithm are as follows:

```
1: { Open a file in as input and out as output
2: count ? 0;
3: while not(eof(in)) {
4: ch ← getchar(in);
5:    if(count=n)
6:    count ← 1;
7:    else
8:    count ← count+1;
9:    for i ← 1 to n{
10:   if i mod 2=1
11:   ch ← ch-m3[i][count];
12:   else
13:   ch ← ch+m3[i][count]; }
14:   for i ← 1 to n{
15:   if i mod 2=1
16:    ch ← ch-m3[count][i];
17:   else
18: ch ← ch+m3[count][i]; }
19: write ch to the output file
```

Implementation: This program implemented in C language takes a plaintext as input from the file. The implemented encryption algorithm taking M_3 matrix as parameter produces the ciphertext, on the basis of plaintext, which is stored in a file. The ciphertext generated for a plaintext,

Implementation of the algorithm: To implement this algorithm C language has been used, because it has several advantages over the other languages. The program has taken an integer as input and produced: (i) a by using the above encryption algorithm is given below:
Plaintext: This is a test message for cipher text generation.
Cipher Text: aZjhìY;ÜZÝ`p□]d|It□utW![;bèb-Tp□,t\u={r`fg-d1+gëö

This program takes a ciphertext as input from the file. The implemented decryption algorithm taking M_3 matrix as parameter it produces the plaintext, on the basis of the ciphertext, which is stored in a file. The plaintext generated for the ciphertext by using the above decryption algorithm is given below:
Ciphertext: aZjhìY;ÜZÝ`p□]d|It□utW![;bèb-Tp□,t\u={r`fg-d1+gëö

Plaintext: This is a test message for cipher text generation.

Complexity: Any arrangement of a set of N objects in a given order is called a permutation of the objects (taken all at a time)[4]. Any arrangement of r< = N of these objects in a given rder is called an rpermutation or a permutation of the N objects taken r at a time and is defined by P (N, r) = N!/(N-r)! If all objects are taken at a time then P(N, N) = N*(N-1)*(N-2)* ...*3*2*1 = N! So, there are N! permutations of N objects (taken all at a time). The key has been generated on the basis of unique permutation vector M_1 and matrix M_2 whose rows are different from each other independently and uniquely permuted. The key matrix is generated whose columns are uniquely permuted and different from each other. If there is an N×N matrix, then there are N! permutations on each column independently. As the columns are uniquely permuted and different from each other, there are N! permutations in the first column, (N!-1) permutations in the second column, (N!-2) permutations in the third column, continuing up to N times there are {N!-(N-1)} permutations in the Nth column. In brute-force attack[1], if an intruder wants to match the key, he/she has to try every possible key on a piece of ciphertext until an intelligible translation is obtained. The counting product rule defines as if there is an event E which can occur in m ways and independent of event, there is a second event F which can occur in n ways. Then the combination of E and F can occur in m * n ways. So, by production rule there is N! * (N!-1) * (N!-2) *... * {N!-(N-1)} = O $(N!)^N$ possible keys for an intruder to match the key[4]. On average on the basis of brute-force attack, half of all possible keys must be tried to achieve success.

Time calculation table: A machine has been taken that can perform 10^{12} comparison per second. The following Table can be calculated on the basis of the key size N.

Key size (N)	No of alternative keys	Time
5	24883200000	0.0248832 sec
6	$10^{17.14399497858761076387634944 1023}$	1.61243136 days
10	$10^{65.5976303287679375117476123996006}$	$10^{46.09882}$ years
50	$10^{3224.1537436236017678157401267 2281}$	$10^{3204.65493}$ years
90	$10^{12435.47422110097199321959370051 72}$	$10^{12415.97541}$ years
100	$10^{15797.0003654715788373914924802 99}$	$10^{15777.50155}$ years
12	$10^{27595.0284919733756672362939811134}$	$10^{27575.52968}$ years
15	$10^{43036.2476535174762133795852215454}$	$10^{43016.74884}$ years
15	Infinite	Infinite

CONCLUSION

Private-key plays a fundamental role in cryptography for secure electronic communications. Three level composite mathematical functions are used for generating private key. For this a randomly taken and uniquely permuted vector has been taken as input of the first function and produced output as integer. The first function's output and matrix form are taken as the input of the second function and produced output as an integer that forms an another matrix. Taking the above two functions as input parameter for the third function and produced the final matrix that is used as the private-key. The generated private-key has been used to produce ciphertext from the communicating plaintext. The complexity of the generating private-key has also been measured and found very high for the intruder. Here, emphasis is given on for increase the complexity to transfer from ciphertext to plaintext. In sequence of stages, given size vector and square matrices are generated for making the key. Simulation works has been accomplished with the generated private-key on the plaintext and also on the cipher text. This private-key generation technique can be used for cryptology lab session as a teaching tool and the generated session-key can be used for secure communication on-line or off-line.

REFERENCES

1. Stalling, W., Data and Computer Communication, 6th Edn., Prentice Hall International, Inc., pp: 649-685.
2. Tanenbaum, A.S., 1999. Computer Networks, 3rd Edn., Prentice Hall of India Private Limited, pp: 577-618.

3. Lipschutz, S. and M.L. Lipson. Theory and Problems of Discrete mathematics, 2nd Edn, Schaum's Outline Series, McGraw-Hill, pp: 133-151.
4. Horowitz, E. and S. Sahni, 1999. Sanguthevar, Computer Algorithms, Galgotia Publications Pvt. Ltd., pp: 1-68.
5. William, S., 2003. Cryptography and Network Security, Principles and Practice, 3rd Edn., Pearson and Education.
6. Charlie, K., P. Radia and S. Mike, 2003. Network Security, Private Communication in a Public World, 2nd Edn., Pearson and Education.
7. Menezes, P., O. Van and S. Vanstone, 1997. Handbook of Applied Cryptography, CRC Press.
8. Bruce, S., 1996. Applied Cryptography, 2nd Edn., John Wiley.
9. Lutfar, R.M., 2004. A review on Cryptography and Cryptographic Applications, Magazine, Department of Computer Science and Engineering, 1st Edn., Dhaka University.
10. Behrouz, A.F., 2004. Data Communications and Networking-3rd Edn., Tata-McGraw Hill.
11. Ismail, J.M., S.K.R. Mizanur and M.R. Lutfar, 2002. Session-key Generation for Message Authentication using Conventional Encryption Techniques, Proceedings of the 3rd International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT'02, Kanazawa, Japan.
12. Ismail, J.M., S.K.R. Mizanur and M.R. Lutfar, Pseudorandom bit string for Cryptographic Applications, J. Sci. Dhaka University, pp: 1022-2502.
13. Ismail, J.M., S.K.R. Mizanur and M.R. Lutfar, Pseudorandom bit string Generation for Secure Electronic Transactions, Accepted for publication, Nuclear Science and Applications, J. Bangla. Atomic Energy Commission, Dhaka.
14. Ismail, J.M., S.K.R. Mizanur and M.R. Lutfar, Encryption with Randomly Chosen Base Conversion and Special Symbols, Accepted for publication, Nuclear Sci. and Applications, J. Bangladesh Atomic Energy Commission, Dhaka.
15. Ismail, J.M., Abdullah Al-Shamim and M.R. Lutfar, 2003. Session-key Generation using Conventional Encryption Techniques, Proceedings of the 6th International Conference on Computer, Communication and Information Technology, ICCIT'03, Jahangir Nagar University, Bangladesh.
16. Ismail, J.M., S.K.R. Mizanur and M.R. Lutfar, 2002. Session-key Generation for Message Authentication using Conventional Encryption Techniques, J. Electrical Engin., The Institute of Engineers, Bangladesh.