GENERATING SECURE PARAMETERS IN DIFFIE-HELLMAN PROTOCOLS USING PSEUDORANDOM SEQUENCES

Dr. Du Dinh Vien, Dr. Tran Canh Duong
Hoa Binh University
Corresponding Author: ddvien@daihochoabinh.edu.vn

Ngày nhận: 20/10/2023
Ngày nhận bản sửa: 12/11/2023
Ngày duyệt đăng: 21/12/2023

Abstract

When devices connecting to the Internet, one of the most problem occurred is security. There are many researchers focusing on this problem so far. Diffie-Hellman type key exchange protocols have a certain role in information security applications. Generating secure parameter sets ensures that the Diffie-Hellman type key exchange protocols can resist some attacks to find the user's secret key. This article proposes to perform encryption using pseudo-random strings to prevent unwanted attacks.

Keywords: *Random number generation, Asymmetric encryption/decryption, Linear Feedback Shift registers, secret key, encryption method.*

Tạo các tham số an toàn trong giao thức Diffie-Hellman sử dụng các chuỗi giả ngẫu nhiên

TS. Dư Đình Viên, TS. Trần Cảnh Dương
Trường Đại học Hòa Bình
Tác giả liên hệ: ddvien@daihochoabinh.edu.vn

Tóm tắt

Khi các thiết bị kết nối Internet, một trong những vấn đề xảy ra nhiều nhất đó là vấn đề bảo mật. Cho đến nay, có rất nhiều nghiên cứu tập trung vào vấn đề này. Các giao thức trao đổi khóa loại Diffie-Hellman có vai trò nhất định trong các ứng dụng bảo mật thông tin. Việc tạo các bộ tham số an toàn đảm bảo rằng các giao thức trao đổi khóa loại Diffie-Hellman có thể chống lại một số cuộc tấn công nhằm tìm ra khóa bí mật của người dùng. Bài viết này đề xuất thực hiện mã hóa bằng chuỗi giả ngẫu nhiên để ngăn chặn các cuộc tấn công không mong muốn.

Từ khóa: *Tạo số ngẫu nhiên, mã hóa/giải mã bất đối xứng, thanh ghi dịch phản hồi tuyến tính, khóa bí mật, phương thức mã hóa.*

1. Introduction

Chao Hoom Lim and Pil Joong Lee presented a method to avoid small subgroup attacks for cryptographic applications such as public key cryptosystems and digital signature schemes. These two authors proposed generating safe primes p, where all

prime factors other than 2 of p – 1 are large prime factors. A famous oracle attack is analyzed as follows. One somehow receives a certain calculation result as a function of the secret key from the owner of the secret key and tries to extract some information about that secret key. This attack scenario is well understood in the cryptographic community. However, many protocols based on the discrete logarithm problem turn out to leak many secret key bits from this oracle attack, unless appropriate checks are performed. Authors Chao Hoom Lim and Pil Joong Lee presented a key recovery attack on different discrete log-based schemes operating in a prime-order subgroup. Through the attack they can know part or all of the secret key in most Diffie-Hellman type key exchange protocols and some applications of the ElGamal encryption and signature scheme [1].

For cryptographic applications such as public key cryptosystems and digital signature schemes whose security is based on the difficulty of the logarithm problem on GF(p), the set of parameters (p, q, g) (with p is a prime number, q is a prime divisor of p-1 and ord(g) = q) is chosen to be used as long as it can resist attacks based on logarithmic problem solving methods [4], [8]. Ord is a very important concept in mathematics, helping us calculate the order of an element in a group G and find the smallest positive integer for that element to be equal to the unit element of the group. In encrypting information, ord is used to find the private and public keys to encrypt and decrypt information. For example, in the RSA algorithm, the order (ord) is used to find the private and public key parameters to encrypt and decrypt information. The parameters p and q recommended for use in cryptographic standards must all be large prime numbers. For example, in the FIPS PUB 186-3 [2] standard, p has a minimum size of 1024 bits and the bit length of q is 160 bits, respectively.

In this article, we propose an encryption method, generating the parameters p,q and g using a sequence of pseudo-random values with a large enough number of bits to prevent attacks. Digital signature schemes, when used in practice in Diffie-Hellman type key agreement protocols, give rise to an attack by one participant in the system to find the other's secret key. The International Organization for Standardization (ISO) has issued the ISO/IEC 11770-4:2006 standard as follows. Prime number p used in DH-KE key agreement protocols with generating element g ϵ GF(p) with degree q, must satisfy the following conditions:

(1) $p = 2qq_1q_{2...}q_k + 1$

where q_i are prime numbers (not necessarily different)

(2) $q_i > q$ *with* $1 \le i \le k$

Lim-Lee proposed an empirical algorithm for generating safe prime numbers [1]. This algorithm was also later used to generate secure prime numbers in cryptographic libraries [5], [6].

2. Perform encryption using a pseudo random sequence

Pseudo random sequence generated by Linear Feedback Shift Registers (LFSR) [7]. An m-stage linear feedback shift register described by an m-order polynomial can create a cyclic series of m with period 2^m-1. These polynomials are represented by a binary vector

c = [cm, cm−1, ..., c1, c0] or notated

with an octal vector. For example *f(x)*= $x1^0+x^6+x^5+x^3+x^2+x+1$ is a binary vector 10001101111 or octal which is 46F. The m-sequences satisfy the equilibrium property and the run-length property, giving them a random appearance. The balance characteristic shows that the probability of bit '0' and bit '1' are approximately equal. A run is defined as a group of bits of the same type (‘1’ or ‘0’) consecutively existing in the sequence. The run length is the number of bits in each run. The characteristic of running in a pseudo-random sequence must be satisfied that in one cycle of the general sequence there are $1/2^n$ number of steps with length n. From an existing pseudo-random sequence, if we shift each bit position to the right or left, we will obtain a new sequence m with the number of elements that coincide and do not coincide with the original sequence. The autocorrelation function is used to evaluate the similarity of the sequence with its translation steps, it is defined as follows:

$$R(\tau)=\sum_{n=0}^{N-1}\hat{a}_n\hat{a}_{n+\tau}=\begin{cases}N, & \tau\equiv 0\\ c, & \tau\neq 0\end{cases} \quad (1)$$

where, $\hat{a}_n=(-1)^{a_n}\in\{+1,-1\}$, $a_n\in\{1,0\}$, c is a small value.

Figure 1 depicts the m-string generator

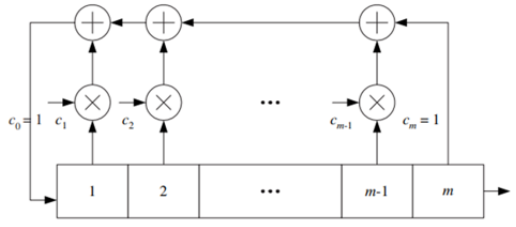

Figure 1. Diagram depicting the m-string generation circuit using the Fibonacci configuration

Autocorrelation is a measure of the degree of similarity between a sequence and its time-varying sequences. It can be used to predict the starting point of a PN sequence by peak detection. To ensure a good autocorrelation function (ACF) of the interleaving, the PN sequence is mapped directly into the interleaving set [6].

The encryption steps are as follows [9]:

- Choose a primitive polynomial

- Create a PN sequence corresponding to the selected primitive polynomial. Store all S−1 shifts of this string. Generate a PN sequence of length $S = 2^m - 1$ for an integer m using a linear feedback shift register with feedback connections determined by a primitive polynomial of degree m on the Galois field GF(2).

- Append a "0" bit at the end of each sequence moved from the original sequence to satisfy the balance between the number of "1" bits and the number of "0" bits in this sequence.

- Find clusters (runs) with at least two consecutive “1” bits.

- Stores the positions of the "1" bits of the bursts.

- Between two consecutive "1" bits, insert a "0" bit from the nearest position.

- Save swap positions.

- Continue steps 6 until the end of the sequence.

- Storing the swap positions, we have an orthogonal permutation sequence.

3. Results and discussion

During the code creation process, the author chooses a nonlinear sequence with large length, high complexity and good correlation function. The complexity of some sequences is achieved by nonlinear nesting of linear sequences created by a set of linear feedback shift registers. To perform

encryption, in this article the author analyzes, proposes, and simulates the application of an interleaved structure using a linear feedback shift register. To creates a sequence m with length

$N = 2^m - 1$, we represent a prime polynomial h(d) of degree m as follows:

$$h(d)=h_0+h_1d+h_2d^2+\ldots+h_{m-1}d^{m-1}+h_md^m=\sum_{i=0}^{m} h_id^i \quad (2)$$

This polynomial corresponds to a linear feedback shift register (LFSR) as shown in Figure 2. The diagram consists of m blocks representing memory elements or flip-flops. A memory element can remember the value '0' or '1'.

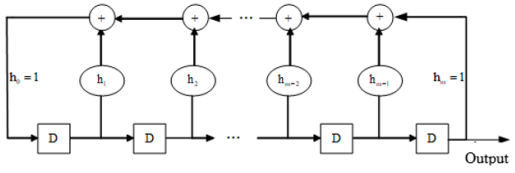

Figure 2. Equivalent feedback shift register h(d)

At each transmission time, the value in the memory elements is shifted to the right element, and the memory elements respond in the form of polynomial h(d) which is additive and feedback to the left element. The sum operation is calculated modulus 2, which is equivalent to the XOR operation in electronic circuits.

Statistics on the number of prime polynomials of degree m are shown in the following table 1.

Some polynomials can be used to generate pseudorandom codes of the following form:

$x^6+x^5+x^4+x^3+x+1$, x^7+x^6+1,
x^7+x+1, x^7+x^3+1, $x^7+x^3+x^2+x+1$,
$x^7+x^5+x^4+x^3+1$, $x^8+x^4+x^3+x^2+1$,
$x^9+x^6+x^4+x^3+1$, $x^{10}+x^6+x^5+x^3+x^2+x+1$,
$x^{12}+x^{11}+x^{10}+x^8+x^7+x^2+1$,
$x^{15}+x^{14}+1$, $x^{23}+x^{18}+1$, $x^{29}+x^2+1$

Table 1. Statistics on the number of prime polynomials of degree m

Number of degrees of the generating polynomial (m)	Length of sequence (N = 2m – 1)	Number of polynomials
……	……	……
6	63	6
7	127	18
8	255	16
9	511	48
10	1023	60
11	2047	176
12	4095	144
13	8191	630
14	16383	756
15	32767	1800
16	65535	2048
17	131071	7710
18	262143	7776
……	……	……

The security of information depends on two factors: encryption key and encryption algorithm. In e-commerce, encryption algorithms are often made public, so the security of information depends only on the security of the encryption key. In a cryptosystem that uses pseudo-random keys, the system will provide an initial random number (key seed) to the key generation algorithm to create an encryption key for each communication session. During the encryption process, the encryption algorithm will generate a translation key for that communication session. The security of the system will depend on the key seed and key generation algorithm. To resist brute force attacks to find the correct key, the key seed space must be large enough and the selection of key seeds to generate keys must be completely random. Assuming the seed length is n then the seed space cardinality will be 2n seed keys. The pseudorandom codes in this paper meet the requirement for the length of the key seed by increasing the degree of the key generation polynomial.

The simulation program is performed for polynomials of degree m = 12. The polynomial simulation program $x^{12}+x^{11}+x^{10}+x^8+x^7+x^2+1$ is as follows:

```
clear all; close all;
x1=[1 1 1 1 1 1 1 1 1 1 1 1];
n1 = length(x1); len1 = 2^n1-1;
p1(1,1) = x1(1,1); z1 = x1;
for y1 = 2 : len1 x1 = z1;
for i = 1 : n1 if (i==1) z1(1,i) = xor
(x1(1,2), xor (x1(1,7), xor (x1(1,8), xor
(x1(1,10), xor (x1(1,11), x1(1,12))))));
else z1(1,i) = x1(1,i-1); end end p1(1,y1)
= z1(1,12); end subplot 211; stem (p1);
```

Figure 3. shows the simulation results, which are the pseudorandom bits at the output.

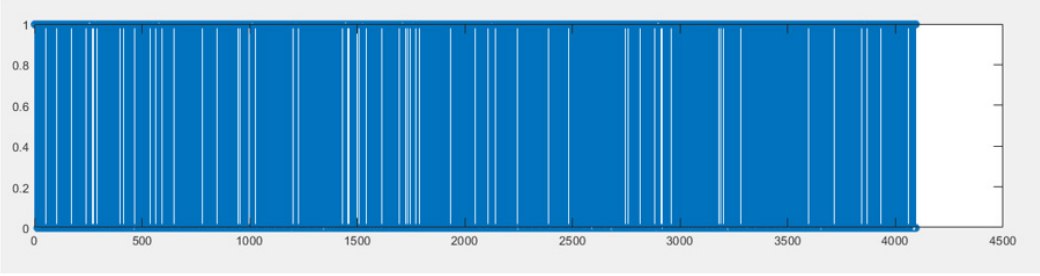

Figure 3. Schematic representation of pseudorandom bits at the output

We add a zero at the end to balance the number of 0 and 1 bits, so this string has 4096 bits. Due to the large number of bits, we just write it more briefly. The sequence represents the exact first and last bits of the sequence.

```
1111111111100110010000011001100011010001101111000000001110101111101110
0101000111001011111101101000011000010110101100011010100111001100101010011
1110011000010100000111011000001101011011100001011101001100101101100010001
0111101001110101100010000001011000110000011000011100000000010000001110010
0000000110010111001110010100000101110011001111110001110110010110001110101
1010001011010100100110010001001100000010000100100100000101100110010111100
0100100100101111010110111101000001010110011111111101101001110010011111000
1101001100010010001010110100000011110001011001110000100001011100011001000
1100110111000000010001011011110001010011011100101011110101000101100000111
1110000100011110111011011100110100000001011010011011011010110110010111011
```

0110011000100001000011101101010011001100100101100101000011110111110001010
1100010101011101000011010111101101110110000110000000111101000011111101011
1010111001000111100100011011000111101001001010001010111110101111111011110
000101100001101010100011011 0…………………………………………………………
………………………………………..01011111111000011011010111100111011010001
1001010011000011011111101010111110000101001011011010001001111111100100111
0110110110010010001101001001000100001111100000101101100111011101110001101
1100111110101101010000110010000110011010011111011001010010010110111110111
1111101011000001010100110100110100011110101110000100100000101001110001100 0
0110010101100101010110101111100100000100110011110110110001101111001010110
1111111010100101001101011000011111111110110011110001001100101000110100001 1
1010000000101001001110001001011010101110011111000001100100100110000100010 1
0001011101011001110101110111011001001101100010100000100010001000001010011 0
0100111001110000000101110110011001110100100010111000001100010100100010010 01
0101000000001110000101011110000010011100101101011101100011001011001101111 0
1010100001110001010100100001101100001011111100111101111001000011100111111 1
1110001100011001110011 00

The sequence is written in hexadecimal form as follows:

FFF320CC68DF00EBEEA397ED0C2D63539953E6141D835B85D32D88BD3AC40B1830E
081C80CB9CA0B99F8ECB1D68B52644C08495997924BD6F4159FF69C9F1A6245681E2
CE10B8C99B808B78A6E57A8B07E11EEDCD0169B6B65DB310876A664B287BE2B1574
35EDD8603D0FD75C8F2363D2515F5FDE161AA36 ……………………………………….
…………………………………………………………….5FE1B5E768CA61BF57C296D13F
C9DB648D2443E0B67771B9F5A864334FB292DF7FAC1534D1EB848538C32B2AD7C82
67B63795BFA94D61FFB3C4CA343A052712D5CF06498451759D7764D8A111053273817
66748B86291254070AF04E5AEC659BD50E2A43617E7BC873FE319CC.

4. Conclusion

Diffie-Hellman type key exchange protocols have a certain role in information security applications. Generating secure parameter sets ensures that the Diffie-Hellman type key exchange protocols can resist some attacks to find the user's secret key. The article proposes to perform encryption using pseudo-random strings to prevent unwanted attacks. Based on pseudo-random sequences of degree m, we can increase the complexity of the code by increasing degree m. On the other hand, we can create combinations to create the Gold sequence, thereby increasing the number of codes according to actual needs. The simulation program with a generating polynomial $x^{12}+x^{11}+x^{10}+x^8+x^7+x^2+1$ has created 4095 pseudorandom sequences, each string has 4095 bits. Depending on the actual need for security and information security, we can increase the degree m of the generating polynomial to increase the number of pseudo-random bit strings and at the same time increase the number of bits per sequence. In general, a generating polynomial corresponding to a linear feedback shift recording circuit can generate 2^m pseudo-random strings, each string consisting of 2^m bits, from which the parameter p of the key with complexity can be created. The parameters q and g are also created in the same way as parameter p. Thus, we can create parameters p, q and g large enough to withstand attacks.

References

[1] C. Lim and P. Lee (1997), *A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup,* EUROCRYPT 1997.

[2] FIPS PUB 186-3(2009), *Digital Signature Standard (DSS),* https://csrc.nist.gov/csrc/media/publications/fips/186/3/archive/2009-06-25/documents/fips_186-3.pdf,Accessed on 10/9/2020.

[3] H. Yoshida (1999), *LUN Security Considerations for Storage Area Networks.* Technical Report, Hitachi Data Systems.

[4] J. M. Pollard (1978), Monte Carlo methods for index computation (rood p), *Math. Comp*., 32(143), pp.918-924.

[5] MIRACL Cryptographic SDK, https://github.com/miracl/MIRACL, Accessed on 10/9/2020.

[6] PGP.I, OpenPGP, https://www.openpgp.org/, Accessed on 10/9/2020.

[7] Santit Traithavil (2006). *Simulation of PN Code Sequences for Cellular Systems.* Department of Engineering, The Australian National University - February 17, 2006.

[8] S. C. Pohlig and M. E. Hellman (1978), *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance,* IEEE Trans. Inform. Theory, IT-24 (1), pp.106-110.

[9] Tran Canh Duong et al (2021). *Generating the interleave division multiple access (IDMA) used in 5G mobile communication system.* Journal of Xidian University - ISSN No:1001-2400 - Volume-15-Issue-12-December-2021 pp. 546-534.