Scientific Bulletin of the Petru Maior University of Tirgu Mures
Vol. 6 (XXIII), 2009
ISSN 1841-9267

A Novel Method for Generating Encryption Keys

Dascalescu Ana Cristina
Nidelea Marinela
dascalescu_anacri@yahoo.com

Abstract

The development of the informational society, which has led to an impressive growth of the information volume circulating in the computer networks, has accelerated the evolution and especially the use of modern cryptography instruments. Today, the commercial products use standard cryptographic libraries that implement certified and tested cryptographic algorithms. Instead, the fragility of encryption algorithms is given by compositional operations like key handling or key generation. In this sense, the article proposes an innovative method to generate pseudorandom numbers which can be used for the construction of secure stream encryption keys. The proposed method is based on the mathematical complements based on the algebra of the finite fields and uses a particularized structure of the linear feedback shift registers.

1. Introduction

To achieve a safe system of protection of information is necessary an analysis of all possible directions of attack on it. It is unnecessary to secure one side of the system when the attack can be easily done on one side insufficiently secured. The vulnerability of the encryption systems can be significantly accelerated if the key generation process is predictable.

The stream ciphers encrypt individual characters (usually binary digits) of a plaintext message one at a time, using an encryption transformation which varies with time. Thus, the plaintext $x = x_1x_2x_3...$ is combined character by character with a string $k = k_1k_2k_3...$ named fluid key. The main problem is to generate such an encryption key that can be obtained either randomly or based on an algorithm which starts from a small sequence of encryption keys. The article presents an effective way to generate a sequence of pseudorandom numbers, with extended period used for the building of the necessary keys of a stream encryption system. The proposed algorithm is based on the motion technique with linear feedback and mathematical complements based on finite field algebra.

2. About the generation of pseudo-randomly numbers

Almost all the cryptographic systems and protocols used in cryptography are based on the choice of unpredictable arbitrary numbers; known under the standard name of randomly numbers or numbers randomly generated. A perfect randomly number is a number that the attacker cannot guess but through gross strength. A very important part of the cryptanalysis is based on exploiting the imperfections of some functions that generates random numbers. A pseudo random numbers generator (PRNG) is an algorithm for generating a string of numbers that are relatively independent to each other and that approximates some properties of the strings of random numbers.

Since any PRNG that runs on a deterministic computer is a deterministic algorithm, the generating string will have certain characteristics that a natural random numbers string does not have. This characteristic is the frequency, guaranteed by the fact that the generator uses a fixed amount of memory, which will lead, after a sufficiently number of iterative, the algorithm returns to an interior state already crossed, and from this moment it will repeat infinitely its cyclic behavior. Another characteristic of a PRNG is highlighted by the possibility of being activated from an arbitrary starting point - the initial state.

In order to generate a sequence of numbers, are established the initial values $x_1x_2...x_k$ which form the seed of the generator.

Keywords: *pseudorandom sequences generation, backtracking method*

To this we apply the recurrent relationship, defined as:

$$x_n = g(x_{n-1}, x_{n-2}, ..., x_{n-k}), n > k$$

where $g : M \to M$ with M set of natural numbers which can be represented in computer.

The generator of pseudorandom numbers must fulfill the following conditions:

- to be simple and quick;
- to produce strings of numbers with arbitrary length which includes no repetitions;
- to generate numbers with a uniform distribution;

The fulfillment of these conditions can be ensured through a good choice of the function.

3. Generation of pseudorandom numbers using shifting registries.

The NIST has developed a set of statistical tests for random numbers that have as objective to determine binary string deviation from the quality of being random. The interpretation of these deviations should take into consideration as possible causes the fact that the generator presents designing defects as well as the fact that the tested binary string presents abnormalities which is explainable by the appearance of the randomly generated data.

It makes the following assumptions on random binary strings to be tested:

- *the uniformity*: in any moment of the bit string generation, the appearance probability of a zero or one is the same value, namely ½. The expected number of zero bytes (respectively one) is n/2, where n is the length of the string in bytes.

- *the scalability*: any test applicable to a string applies also to any string randomly extracted. So, any such substring should successfully pass any randomly test.

- *the consistency* – the behavior of a generator must be consistent relative to initial values (seeds). The generator must be tested for different initial values.[3]

In practice, the pseudorandom numbers generators used until now can be divided in the classes of generating algebraic pseudorandom numbers, classes of generating arithmetic pseudorandom numbers, classes of generating pseudorandom numbers based shifting registry (linear circuits) and generating pseudorandom numbers based on chaotic functions.[4]

Most cryptographic applications use pseudorandom numbers for the construction of encryption keys used in securing information. For example, in generating digital signatures or in the authentication process, the cryptographic protocols require such input strings with random characteristics.

4. Generation of pseudorandom numbers by shift registers.

A linear feedback shift register includes two parts: one shift register composed of one string of bits whose number determines the length of the register and a feedback function. To generate a bit, all the existing bits in the register are shifted to the right. The output of the register is the bit from the right position, which leaves the register through the shifting. The register is completed with a new bit position on the left, this being calculated as the value of a function of other bits from the register. (Fig. 1).The period of a shift register is determined by the length of the string of generated bits, before this string to be repeated.

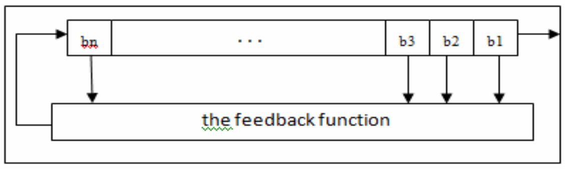


Figure 1. A linear feedback shift registers

The simplest register is LFSR (Linear Feedback Shift Register). The feedback function is XOR operation between certain bits of the register; the list of the bits is called the "tap" sequence or Fibonacci configuration.

A LFSR with L states can be also defined using a polynomial (called characteristic – fig.2) $p(x) = 1 + c_1 x + c_2 x^2 + ... + c_L x^L \in Z_2, c_n \neq 1$

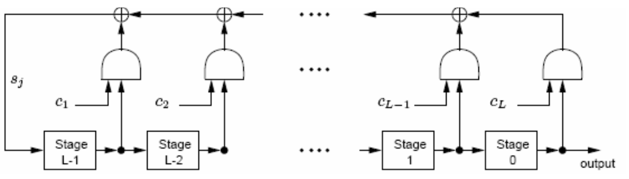


Figure 2. The tap sequence of a LFSR corresponding to a polynomial

From the figure 2, we can see that LFSR with the initial state $s = [s_{L-1}, s_{L-2}, ..., s_0]$ generate the output sequence $s = s_0, s_1, s_2, ...$ by following recurrent

$s_j = (c_1 s_{j-1} + c_2 s_{j-2} + ... + c_n s_{j-n})\ mod2$ for j>n

A shift register with linear feedback can be used for normal polynomial operations: additions, subtractions, multiplications and divisions. Therefore, the “tap" sequence may correspond to a polynomial

$g(x) \in Z_2$ and in the storage elements $f(x) \in Z_2$ the coefficients of an arbitrary polynomial can be introduced. In the Figure 3 is presented a shift register with linear feedback with a particularized structure to make the division of two polynomials.

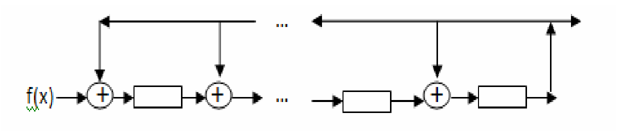


Figure.3. The division operation of two polynomials

If the polynomial $g(x) \in Z_2$ of degree n which form the "tap" sequence is irreducible over Z_2, may be defined the multitude of the rests classes polynomial modulo $g(x) \in Z_2$ such :

$$G_g = \{u = a_0 + a_1\alpha + ... + a_{n-1}\alpha^{n-1}; a_i \in Z_p\}$$

$g(\alpha) = 0$, where α can be considered a polynomial of degree n -1. [2]

In the register defined in Figure 3, if the "tap" sequence polynomial is irreducible and in the storage elements are introduced the initial polynomial coefficients $b(x) = 1$, namely the vector $(1,0,0...,0)$ at the next step α is obtain. Leaving the register to function, it is generated successively the elements of the set $\{\alpha^2, \alpha^3, ..., \alpha^{n-1}\}$, namely all the elements of the finite field G_g.

In the algebraic theory of the finite fields, applied to the G_g field defined above, have been emphasized the following results:

- the number of elements of G_g is 2^n;
- the G_g field can be built from the roots of the polynomial $P_n(x) = x^{2^n} + x$.

In conclusion, for the LFSR to be maximum period, the polynomial formed of the "tap" sequence must be an irreducible polynomial modulo 2, with degree equal to the length of the register.

5. Proposed Algorithm for the pseudorandom sequences generation.

In order to achieve the desideratum of generating pseudorandom numbers with larger periods it is proposed the elaboration of an algorithm that generates irreducible polynomials with a degree greater than 256.

The obtained results are introduced in the shifting registers with linear feedback described above, resulting sequences of pseudorandom numbers with large periods, which can be used successfully for the construction of cryptographic keys.

Having as starting point the results from the theory of finite fields listed above, the algorithm involves the selection of a polynomial such as $u_1 = x$, and for each $i = 1,2,...,256$ involves the construction of the polynomials u_i through the following steps:

1) it calculates $v_i =$ the rest of the division of $(u_i)^2$ to the polynomial $f(x) \in Z_2$

2) it calculates the $[f, (v_i + x)] = d_i$ the greatest common factor of f and polynomials ($v_i + x$) and interprets the result such result:

- if $d_i \neq 1$ then polynomial f is reducible and the testing ends.
- if $d_i = 1$, than it is built the polynomial $u_{i+1} = v_i$ and the operations 1 and 2 are repeated.

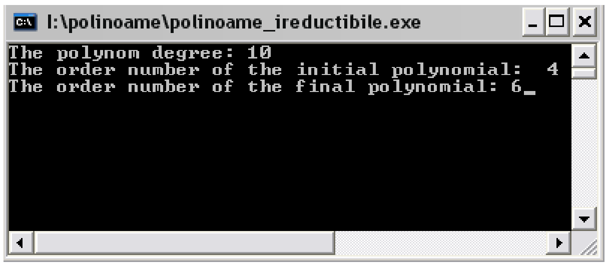
If $d_1 = d_2 = ... = d_{256} = 1$, then it concludes that the polynomial f is irreducible.

Due to the fact that the polynomial has the coefficients in Z_2 field, in the implementation of the algorithm, we optimized the management of memory in the next manner:

- we stored groups of 8 coefficients in a byte, starting from a_n to a_0. So, the coefficient a_k was stored in the byte numbered by *k div 8* on the position *k mod 8*.
- we used a string allocated dynamically because each element of it is a byte.

The generation of the polynomials was made by the Backtracking method. Because the irreducibility of a polynomial can not be tasted before the generation of all its coefficients we were able to make only a minor improvement: we testing the irreducibility of a polynomial if the number of its coefficients equal with 1 was odd.

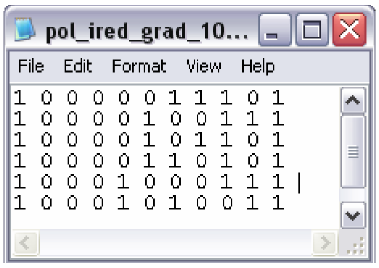
The number of irreducible polynomials with degree n in field Z2 is $N = \frac{2^n - 2}{n}$, so that after running the program, the output is corresponding to some bits sequences in number of 2^{215}. For proper management of output data is proposed that the input data to represent the order numbers of generated polynomials.



```
I:\polinoame\polinoame_ireductibile.exe
The polynom degree: 10
The order number of the initial polynomial:  4
The order number of the final polynomial: 6_
```

Figure 4. Input data

The output data are sequences of numbers 0 and 1, corresponding to coefficients of a polynomial irreducible in Z_2.



```
pol_ired_grad_10...
File  Edit  Format  View  Help
1 0 0 0 0 0 1 1 1 0 1
1 0 0 0 0 1 0 0 1 1 1
1 0 0 0 0 1 0 1 1 0 1
1 0 0 0 0 1 1 0 1 0 1
1 0 0 0 1 0 0 0 1 1 1
1 0 0 0 1 0 1 0 0 1 1
```

Figure 5. Output data

The bits sequences thus obtained are entered into the shift register with linear feedback with customized structure (Fig. 2). So, we obtain the sequence of pseudorandom numbers with large periods (2^{256}) which can be used successfully for the construction of the encryption keys.

6. CONCLUSIONS

The innovative method of generating pseudorandom numbers by the register with linear feedback, used in implementation of the presented algorithm, focuses on the following aspects:
- in the generated sequence of pseudorandom numbers is no obvious correlation between the initial values and any of the values generated by it; each element of generated string appears as the output of a random independent event with the probability ½.
- the generated sequence of pseudorandom numbers is by the large period(up to 2^{256}), due to the "tap" sequence from the shifting register linear feedback that has the structure of an irreducible polynomial in Z_2.

The data encryption using a fluid key constructed by the proposed algorithm can be achieved by the following steps:

- for each character $x_1x_2...$ of the plaintext, the appropriate ASCII code is converted into a binary string of length 7. For example:

$$x_i = A \to ord(A) = 65 \Rightarrow x_i = (1,0,0,0,0,0,1)$$

- the LFSR with initial state $s(0) = (s_0(0), s_1(0),..., s_{n-1}(0))$ generate the key stream, a sequence of 7n-bits $s_0(0), s_0(1),..., s_0(7n-1)$, which are grouped into n 7-bit blocks.
- the cipher text $y = (y_0, y_1,..., y_{n-1})$ consists of n 7-bit vectors where y_i is the XOR of the plaintext block x_i and the block of key s_i. [1]

The proposed algorithm is used to generate the irreducible polynomials with a degree higher than 256. The outputs are sequences of numbers 0 and 1, corresponding to the coefficients of an irreducible polynomial in Z_2.

Based on the bits string obtained is build the "tap sequence" for a linear feedback shift register obtaining pseudorandom numbers sequences with longer periods (2^{256}) that can be used successfully for construction of stream encryption keys.

The next phase of the research project is to develop an algorithm to generate primitive polynomials of degree higher than 256, in order to obtain pseudorandom sequences with longer period.

REFERENCES

[1] A.G. Konheim, "Computer Security and Cryptography", Wiley – Interscience Publisher, USA, 2007, pp. 244 – 261

[2] Atanasiu Adrian, "Teoria codurilor corectoare de erori" University Publisher, Bucharest 2001, pp. 76-92

[3] A. Menezes , S. Vanstome, "Handbook of Applied Cryptography", CRC Press Publisher, New York. 1997, pp.191 – 261

[4] A. Bruen, M.Forcinito, "Cryptography, Information Theory and Error-Correction", Wiley – Interscience, New Jersey 2005, pg. 56-73

[5] E.M. Gebaly, "Finit Field Multiplier Architecture for Cryptographic Applications", Waterlo Canada 2006, pp. 102-156

[6] Shaska, "Advances in Coding Theory and Cryptography", World Scientific, United Kingdom 2007, pp. 89-107