Bul. Ştiinţ. Univ. Baia Mare, Ser. B,
Matematică-Informatică, Vol. XVIII(2002), Nr. 2, 129 - 132

A METHOD FOR INCREASING THE SPEED OF CERTAIN CSPRBG FOR ONE-TIME-PAD CIPHERS

Ileana BALAZS

Abstract. One-time-pad ciphers are very important because they are cryptographicaly secure, and are easy to implement, but the drawback is that the key should be as long as the plaintext.

Such keys are hard acquired and pseudorandom bit generators are used, but they slow down the encryption algorithm, in this article a method for increasing the encryption speed is showed.

Keywords: cipher, pseudorandom bit generator, encryption algorithm
MSC 2000: 94A60, 11K45, 65C10

1. Introduction

The Vernam cipher or the one-time-pad is probably the most famous stream cipher. It is defined over a binary alphabet as:

$$c_i = m_i \oplus k_i \text{ for } i = 1, 2, 3, \ldots$$

where : $m_1, m_2, m_3, \ldots$ are the plaintext digits;

$k_1, k_2, k_3, \ldots$ are the key digits (Keystream);

$\oplus$ is the XOR functions (bitwise addition modulo 2);

$c_1, c_2, c_3, \ldots$ are the cipher text digits obtained by XOR-ing the plaintext digits and the key digits.

The decryption is defined by

$$m_1 = c_1 \oplus k_i \text{ for } i = 1, 2, 3, \ldots$$

This cipher is very famous because a one-time-pad cipher is an unconditionaly secure encryption algorithm when the key is random and it is used only once. The main issue is the key length, the key must have as many bits as the plaintext.

This is very difficult in practice so methods have been devised to construct pseudorandom sequences in a deterministic manner from a shorter sequence called seed.

A true random bit generator requires a naturally occurring source of randomness (eg. elapsed time between emission of particles during radioactive decay or thermal noise from a semiconductor diode or resistor) so it is hard to implement and the source of randomness are subject to influence by external factors and also to malfunctions.

In practice we use pseudorandom bit generators (PRBG).

2. Pseudorandom bit generators

Definition. A PRBG is a deterministic algorithm which given a truly random binary sequece of length k,outputs a binary sequece of length $l >> k$ which appears to be random.

The input to the PRBG is called the seed, while the output of the PRBG is called a pseudorandom bit sequence.

The pseudorandom sequences appear to be generated by a truly random source to anyone not knowing the method of generation Often the generation algorithm is known to all but the seed is secret .

Many algorithm has been developed to generate pseudorandom bit sequences of various types but not all suitable for cryptographic purposes and not all output sequences are random. It is hard to find good, fast pseudorandom functions that are cryptographicaly strong.

A minium security requrment for a PRBG is that the key should be sufficiently large so the search over 2^k elements (the number of all posible seeds) is completely infeasible.

Definiton. A PRBG is said to pass the next-bit test if there is no polynomial -time algorithm which , on input of the first l bits of an output sequence s, can predict the $(l+1)^{st}$ bit of s with probability significantly greater than 1/2.

Definition: A PRBG that passes the next-bit test (possibly under some plausible but unproved mathematical assuption such as the intracability of factoring integers) is called a cryptographically secure pseudorandom bit generators.(CSPRBG).

There are several generators that have been proved to be cryptographically secure, their security is based on the intractibility of an underlying theoretic number problem but the drawback is that they are relatively slow compared to generators like liniar congruential generator wich is very fast but is completely insecure for cryptographic use.

An example of CSPRBG is the RSA PRBG under the assumption that the RSA problem is intractable.

Algorithm

SUMMARY : a pseudorandom bit sequence $z_1, z_2, z_3, \ldots z_l$ of length l is generated.

```
1. Setup.Generate two secret RSA- like primes $p$ and $q$, and compute
$n = pq$ and $\phi = (p-1)(q-1)$.Select a random integer $e$, $1 < e < \phi$ such that
$gcd(e, \phi) = 1$.
2. Select a random integer $x_0$ (the seed) in the interval $[1, n-1]$.
3. for $i$ from 1 to $l$ do the following:
3.1. $x_i \longleftarrow x_{i-1}^e \bmod n$
```

3.2 $z_i \longleftarrow$the least significant bit of x_i

4. The output sequence is $z_1, z_2, \dots z_l$

The efficiency of the RSA PRBG is quite low because if $e = 3$ then generating each pseudorandom bit z_i requires one modular multiplication and one modular squaring.

The Micali-Schorr PRBG improves the efficencyof the RSA PRBG.

Algorithm

SUMMARY :a pseudorandom random bit sequence is generated.

1. Setup .Generate two secret RSA- like primes p and q, and compute
$n = pq$ and $\phi = (p-1)(q-1)$. Let $N = [\lg n] + 1$(the bitlength of n)
Select a random integer e, $1 < e < \phi$ such that $gcd(e;\phi) = 1$ and $80 \cdot e \leq N$.;
Let $k = [N(1-2/e)]$ and $r = N - k$.

2. Select a random integer x_0 (the seed) of bitlength r

3. Generate a pseudorandom sequence of length $k \cdot l$

3.1 $y_i \longleftarrow x_{i-1}^e \bmod n$;

3.2 $x_i \longleftarrow$the r most significant bits of y_i;

3.3 $z_i \longleftarrow$the k least significant bits of y_i.

4. The output sequence is $z_1||z_2||z_3||\dots||z_l$,where|| detotes concatenation.

This algorithm is more efficient then the RSA PRBG since $k = [N(1-2/e)]$ bits are generated per exponention. For example if $e = 3$ and $N = 1024$ then $k = 341$ are generated per exponentiation.

Moreover, each exponentiaton requires only one modular squaring of an $r = 683-$bit number, and one modular multiplication.

In this article we will propose a way to make the Micali-Schorr PRBG more efficient by increasing the number of output bits in each iteration from k to N, so the speed of the entire algorithm grows.

For this a clock controlled like generator could be the answere.

The modificated algorithm we'll look like this:

Algorithm

SUMMARY :a pseudorandom random bit sequence is generated.

1. Setup .Generate two secret RSA- like primes p and q , and compute
$n = pq$ and $\phi = (p-1)(q-1)$. Let $N = [\lg n] + 1$(the bitlength of n)
Select a random integer e, $1 < e < \phi$ such that $gcd(e,\phi) = 1$ and $80 \cdot e \leq N$.
Let $k = [N(1-2/e)]$ and $r = N - k$ $c_0 = 0$

2. Select a random integer x_0 (the seed) of bitlength r

3. Generate a pseudorandom sequence of length $N \cdot l$

3.1 $y_i \longleftarrow x_{i-1}^e \bmod n$;

3.2 $x_i \longleftarrow$the r most significant bits of y_i

```
3.3 $a_i \longleftarrow$expand($x_i$ from $r$ to Nbits)
3.4 $b_i \longleftarrow a_i \& y_i$, where & is bitwise and
3.5 $z_i \longleftarrow a_i \oplus y_i \oplus (b_i c_{i-1})$
3.6 $c_i \longleftarrow$the least significant bit of $a_i \oplus b_i$
4. The output sequence is $z_1||z_2||z_3||...||z_l$,where|| detotes concatenation.
```

The function in the summation generator is based on the fact that the integer addition, when viewed over $\mathbb{Z}_2$ is a nonliniar function with memory whose correlation immunity is maximum

This summation generator has high period, liniar complexity and correlation immunity.

Implemented alone like PRBG it is vulnerable to certain correlation attacks and known plaintext attack, but combined with a CSPRBG it give's increased afficecy since in every iteration the output has N bit length.

This ideea came from the LSFR (Liniar Feedback Shift Register) theory ,the LSFRs are widely used in keystream generators because they are well-suited for hardware implementation, produce sequences having large periods and good statistical properties, but they are not suitable for software implementation, so the combination beetwen the strength of a CSPRBG and the speed of a modified LFSR summation generator could be a way for increasing the speed of CSPRBG.

3. Conclusion

Finding a good, criptographycaly secure, fast generator is hard in practice but also it is a very tempting idea and it si still the area of much research.

The method described above for increasing the speed of CSPRBG it is not restricted for the Micali-Schorr algorithm but it can be applied to the Blum Blum Shub generator, the Jenkins generator and to other similar generators.

REFERENCES

[1] Handbook of Applied Cryptography

[2] A. Menzes, P. van Oorschot and S. Vanstone

Received: 12.09.2002

North University of Baia Mare
Department of Mathematics and Computer Science
E-mail: balazs@ubm.ro