Compound Inversive Congruential Generator Design Algorithm

Article

2 authors, including:

Janusz Stoklosa
Poznan University of Technology
15 PUBLICATIONS **39** CITATIONS

Compound Inversive Congruential Generator Design Algorithm

Jaroslaw Bubicz
Softbank S.A., Warsaw, Poland
bubicz@post.pl

Janusz Stoklosa
Poznan University of Technology, Poznan, Poland
Janusz.Stoklosa@put.poznan.pl

ABSTRACT

In this paper we present an algorithm for the compound method of designing generators as pseudorandom stream sources. Compound method relies on a technique of combining two or more inversive congruential generators. Experiments on congruential methods showed that even if all tested generators have maximum period not all produced binary sequences have the linear complexity of the same value. We gathered prime exceptions which were moduli of inversive functions and the analysis of this fact was performed. To avoid non stable behavior of binary streams authors proposed an algorithm which can help to prepare a proper set of parameters to design compound inversive generators. They have very valuable advantages which justify expensive application for practical use. Inversive generators may be successively applied in computer local area networks.

Key words: computer networks, LAN, security, cryptography, stream ciphers, pseudorandom sequences, compound inversive congruential generator.

1. Introduction

Pseudorandom number generators are important elements used in a variety of stochastic simulation researches. This scientific activity requires reliable source of randomness. Practical implementations of pseudorandom number generation methods are interesting and give a very promising approach to create pseudorandom, uniformly distributed numbers. They are utilized in statistical and social research, in nature phenomenon analysis, and in Monte Carlo methods. We also use pseudorandom numbers as an initiation values in many cryptographic algorithms. Pseudorandom stream of bits are used as keys in stream ciphers. The method we use for pseudorandom binary sequences generation is based on mathematical algorithms. They are recurrent and tractable when implemented on different hardware platforms [1]. The resulting bit stream has properties similar to those of real random streams.

Eichenauer and Lehn [9] proposed an inversive congruential generator representing a group of nonlinear generators. It is a function of searching inversion in finite field. Compound approach with positive influence on statistic quality output sequences seems to be an important feature of these generation methods. The idea of combining generators to increase the stream's period length and improve statistical properties has its history [5, 13, 14] and today's applications justify its usefulness [1, 11, 12]. Almost three decades of practical experience has shown that the interest in inversive generators increase when computational capabilities of modern computers grow. Initially intended for simulation research the generator did not find attention because of its high complexity. Finally, it has very valuable advantages which justify expensive application for practical use. Inversive generators may be successively

applied on parallel and distributed computing platforms [1].

2. Inversive Congruential Generators

Inversive congruential generator (ICG, for short) [8, 9] is a special type of nonlinear generator. Let F_q be the finite field of order q, where q is an odd prime and let a, b be two elements of F_q. The sequence $(y_n)_{n\geq 0}$ of elements of F_q such that for a given fixed element $y_0 = seed$ and for all $n \geq 0$ $y_{n+1} = ay_n^{-1} + b$ if $y_n \neq 0$, and $y_{n+1} = b$ if $y_n = 0$, is called an inversive congruential generator over F_q. Such a generator is denoted symbolically as $ICG(q,a,b,seed)$ and is said to be ICG with parameters q, a, b and $seed$.

For the sequence $(y_n)_{n\geq 0}$, its period length T is maximum if $T = q$. As it was mentioned in [8, 15] if the polynomial $f(x) = x^2 - bx - a \in F_q[x]$, where $F_q[x]$ denotes the polynomial ring over F_q, is primitive then sequences have the period length equal to q. Such polynomials are called inversive maximal period polynomial (IMP polynomial, for short). The sufficient condition for maximum sequence period length is a proper choice of parameters a and b according to algorithm described in [7]. For the sequence $(y_n)_{n\geq 0}$ let $(x_n)_{n\geq 0}$ be such a sequence that $x_n = y_n/q$.

In this paper, the construction of a compound inversive generator relies on combining two or more congruential inversive generators, according to our method described below.

Let $p_1, p_2, ..., p_r$ be distinct prime integers, each $p_j \geq 5$. For each index j, $1 \leq j \leq r$, let $\left(x_n^{(j)}\right)_{n\geq 0}$ be of sequence of elements of F_{p_j}, that is periodic with period length p_j. In other words, $\left\{x_n^{(j)} \mid 0 \leq n < p_j\right\} = F_{p_j}$. The sequence $(x_n)_{n\geq 0}$ of compound pseudorandom numbers is defined as the sum $x_n = x_n^{(1)} + x_n^{(2)} + ... + x_n^{(r)}$. The period length of the sequence $(x_n)_{n\geq 0}$ equals $T = p_1 \cdot p_2 \cdot ... \cdot p_r$. The compound approach allows combining ICGs, provided they have full period, in parallel generation systems. This method allows obtaining very long period and modular operations may be carried out with relatively small moduli.

3. Compound Inversive Generator as a Source of Binary Streams

The compound inversive generators are accepted for practical purposes for a number of reasons.

Firstly, binary sequences produced in this way are free of undesirable statistical deviations. Inversive sequences extensively tested with variety of statistical tests remain stable under the variation of parameters [2, 4, 6].

Secondly, there exists steady and simple way of parameter choice, based on the Chou algorithm [7]. It guarantees maximum period length.

Thirdly, compound approach gives the same outstanding properties of produced sequences as single inversive generators [3, 10]. Compound inversive generators allow obtaining period length significantly greater than obtained by a single ICG. They seem to be designed for application with multiprocessor parallel hardware platforms.

Efficient application of ICGs in compound generators can be performed using the design algorithm presented in next section. The algorithm allows designing compound generator with predictable period length, predictable linear complexity level, with excellent statistical properties of produced bit streams. The procedure of designing this complex structure starts with defining finite field of p elements and ends with choosing the parameters a and b for each ICG being the component of the compound generator. It means that with each generator is associated a fixed IMP polynomial. Such a condition is sufficient for maximum period of each ICG [9] and finally for maximum period of the compound generator. The construction of IMP polynomial is the most efficient approach to find parameters for ICG with maximum period length.

An interesting conclusion which appears during our research on compound generators is the following: among prime numbers, candidates to ICG moduli, exist ones which not allow obtaining maximum linear complexity of produced bit streams. We call such prime numbers nonstable and they stand with small percentage of the set of all primes. In the set of first 1246 prime numbers (all numbers less than 10000) there exists 0.5% exceptions, i.e. nonstable primes. Excluding such primes in the design process is particularly important. Correctly prepared algorithm of designing compound generator must eliminate such primes.

3. Algorithm

In the first step of the algorithm we exclude nonstable moduli. We test candidate primes $p_1, p_2, ..., p_n$, $p_i \neq p_j$ for $i \neq j$. No one of them should belong to experimentally created table of exceptions. If n stable moduli are correctly chosen we follow the next step. The algorithm chooses the greatest prime non used up to now less than the fixed integer z, the maximum value of candidate modulus, and reject it if it is nonstable.

In the second step, for each chosen modulus p we search a polynomial $f \in F_p[x]$. In this case the algorithm systematically checks for all $c \in F_p$ to get IMP polynomial $f(x) = x^2 - cx + 1$ (according to [7]). Only such c's are considered for which neither $c+2$ nor $c^2 - 4$ are squares. If these values are not squares, then $f(x)$ is irreducible over F_p. For each chosen c, the algorithm tests the period of the sequence $(y_n)_{n \geq 0}$ that is generated by the recursion $y_{n+2} = cy_{n+1} - y_n$ for initial values $y_0 = 0$ and $y_1 = 1$. If c is such that this sequence has the period equal to $p+1$, then f is IMP polynomial. If not, the next c is tested and the step is repeated. If we find an element c such that it satisfies the test conditions, this step is finished. Finally, the algorithm fixes all elements of compound generator.

Once we have found IMP polynomial f, we start the next step with vectors (b_1, p_1), (b_2, p_2),...,(b_n, p_n), where b are freely chosen integers. We start the third step in which the algorithm completes each ICG's parameters, i.e. a_1, a_2,..., a_n, using the equation $a = b^2/(c+2)$. Finally we obtain the proper number n of basic generators $ICG(p_1, a_1, b_1, seed_1), ICG(p_2, a_2, b_2, seed_2), ..., ICG(p_n, a_n, b_n, seed_n)$, which guarantee that compound generator period equals $T = p_1 \cdot p_2 \cdot ... \cdot p_n \cdot k$, where $k \in \{1,2,\ldots, \log_2(\min(p_1, p_2, ..., p_n))\}$ denotes the number of least significant bits from the binary representation of the produced number. The linear complexity $L(s)$ of inversive stream of bits will always be limited by the inequality relations $T \geq L(s) \geq T - k$ and remain stable under the variation of parameters (a_1, b_1), (a_2, b_2), ..., (a_n, b_n).

The algorithm can be presented formally as follows:

Algorithm (Design of compound generator)

INPUT: n – number of ICG components for the compound generator, integers $b_1, b_2, ..., b_n$, z – the bound for maximum value of modulus (sufficiently great).

OUTPUT: $(p_1, a_1), (p_2, a_2), ..., (p_n, a_n)$.

METHOD:

1. for $i = 1$ to n do:
   1.1 if $b_i = 0$ or $b_i \geq z$ then goto 3
   1.2 z = *nextNearPrime* (z)
   1.3 if *testPrime* = FALSE then goto 1.2
   1.4 a = *genParameterA* (z, b_i)
   1.5 $p_i = z$, $a_i = a$
2. Return($(p_1, a_1), (p_2, a_2), ..., (p_n, a_n)$).
3. Return(„Improper parameter b_i").

Function *genParameterA*

Description: the function searches IMP polynomial and returns calculated parameter a).

INPUT: z, b where $0 < b < z$.

OUTPUT: integer a or „Change z".

METHOD:

1. *parameter* = 0
2. for *c* = *parameter* to *z*–1 do:
   2.1. $x = (c+2) \bmod z$
   2.2. $y = (c^2 - 4) \bmod z$
   2.3. if $x \neq c^2$ and $y \neq c^2$ then goto 4
3. Return („Change *z*").
4. $y_0 = 0,\ y_1 = 1$
5. for $n = 0$ to $z+1$ do:
   5.1. $y_{n+2} = cy_{n+1} - y_n$
   5.2. if $y_{n+2} = y_0$ and $n = z+1$ then goto 7
6. if $c \neq z-1$ do:
   6.1. *parameter* = $c+1$
   6.2. goto 2
7. Return ($a = -b^2/(c+2)$).

Function: *testPrime* (integer *z*)
Description: argument comparison with the exception table (experimentally determined). If the argument is not found in the table then the function returns TRUE (the argument is stable).

Function: *nextNearPrime* (integer *x*)
Description: searches and returns the greatest prime less than *x*.

4. Compound Generators

Let us consider, for instance, the compound inversive generator constructed of two ICGs: $ICG_1(q_1, a_1, b_1, seed_1)$ and $ICG_2(q_2, a_2, b_2, seed_2)$. Each of them is an independent source of pseudorandom numbers. During the generation process in each generation cycle two pseudorandom integers appeared. Each of generators demanded individual set of maximum period parameters. Parameters are created according to the Algorithm described in Section 3. Figure 1 presents how the compound inversive generator is constructed and how it works [1].

Two primes q_1 and q_2 as moduli are chosen at the beginning. Compound inversive generator works effectively when each of generators has near the same length of the generation cycle. Such situation occurs if there is the similar number of elements in F_{q1} and F_{q2}. There are no special demands for the initial values $seed_1$ and $seed_2$. In such a way the generators $ICG_1(q_1, a_1, b_1, seed_1)$ and $ICG_2(q_2, a_2, b_2, seed_2)$ are established.

Now the generation process can begin. In each generation cycle two generators working independently deliver pseudorandom numbers x_i and y_i to Addition Block. The main task of this block is to sum two numbers and consequently send the result $z_i = x_i + y_i$ to Output Block. Addition Block performs arithmetic sum of two arguments put into register of the length $\log_2(q)$, where q is the greater of two moduli. Hence the addition is performed without register overfill. Next the content of the register is send to Output Block. Its main task is the extraction of k least significant bits from the binary representation of z_i, denoted as $(x_i+y_i)_{(k)}$. Remaining bits are discarded. Finally, at the end of each generation cycle output bit stream $\{s_i\}_{i\geq 0}$ arises as the concatenation

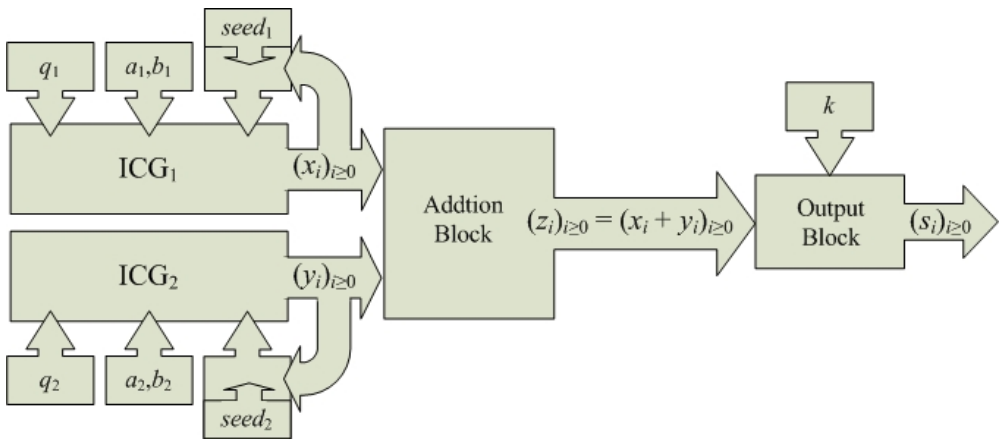

Figure 1. Compound inversive generator composed of two ICGs

(denoted below as ||) of the k-bit blocks:

$$(s_i)_{i\geq 0} = (x_0+y_0)_{(k)} \,||\, (x_1+y_1)_{(k)} || \,(x_2+y_2)_{(k)} || ...$$

5. Conclusion

There exists reliable way to associate optimal parameters to ICG generator, which allows to design compound generators without necessity of testing them.

The design algorithm proposed in the paper includes moduli testing function in the process of compound generator preparation. As a result of the use of our algorithm it is guaranteed the generation of bit streams with maximum linear complexity.

All elementary inversive generators of compound generator prepared according to the algorithm described in this paper have maximum period lengths, high level of linear complexities close to the period lengths, excellent statistical properties and the speed of generation greater that 1.5 Mbps [1].

Acknowledgment

This research carried out by Janusz Stoklosa was supported by the Polish Ministry of Education and Science as a 2005−2008 research project.

References:

[1] J. Bubicz, J. Stoklosa, "Compound inversive congruential generator as a source of keys for stream ciphers". H. R. Arabnia (Ed.), Proceedings of The 2005 International Conference on Security and Management SAM'05, CSREA Press, Las Vegas, NV, USA, 2005, 473−478.

[2] J. Eichenauer, J. Herrmann. "Inversive congruential pseudorandom numbers avoid the planes", Math. Comp., Vol. 56, 1991, pp. 297−301.

[3] J. Eichenauer, J. Herrmann, "Statistical independence of a new class of inversive congruential pseudorandom numbers", Math. Comp., Vol. 60, 1993, pp. 375−384.

[4] J. Eichenauer, J. Herrmann, H. Grothe, A. Topuzoglu, "On the lattice structure of a nonlinear generator with modulus 2^a", J. Comput. Appl. Math., Vol. 31, 1990, pp. 81−85.

[5] J. Eichenauer, J. Herrmann, "Compound nonlinear congruential pseudorandom numbers", Monatsh. Math., Vol. 117, 1994, pp. 213−222.

[6] J. Eichenauer, J. Herrmann, H. Niederreiter, "Lower bounds for the discrepancy of inversive congruential pseudorandom numbers with power of two modulus", Math. Comp., Vol. 58, 1992, pp. 775−779.

[7] W. S. Chou, "On Inversive Maximal Period Polynomials over Finite Fields", Applicable Algebra in Engineering, Communication and Computing, No. 4/5, 1995, pp. 245−250.

[8] J. Eichenauer, J. Lehn, "A non-linear congruential pseudo random number generator", Statist. Papers, Vol. 27, 1986, pp. 315−326.

[9] H. Niederreiter, "New developments in uniform pseudorandom number and vector generation", [In] Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Berlin, 1995.

[10] P. Hellekalek, "Inversive pseudorandom number generators: concepts, results and links", [In] Proceedings of the Winter Simulation Conference, 1995, pp. 255−262.

[11] J. Eichenauer, J. Herrmann, "Explicit inversive congruential pseudorandom numbers", Computing, Vol. 51, pp. 175−182.

[12] J. Eichenauer, J. Herrmann, "On generalized inversive congruential pseudorandom numbers", Math. Comput., Vol. 63, pp. 293−299.

[13] N. M. Mac Laren, G. Marsaglia, "Uniform random number generators", J. ACM, Vol. 12, pp. 83−89.

[14] G. Marsaglia, T. A. Bray, "A convenient method for generating normal variates", SIAM Review, Vol. 6, pp. 260−264.

[15] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia, USA, 1992.