Algorithm for *p*-adic Combiner Generator Synthesis

Borislav P. Stoyanov[1] and Borislav Y. Bedzhev[2]

***Abstract*— The stream ciphers have large application in the practice. As a result, they ought to satisfy great number of necessary conditions. The most significant of them are: resistance to crypto attacks, high performance velocity and cost-effective hardware implementation. It is impossible to meet all of these conditions if the stream cipher has a structure similar to the classical *Linear Feedback Shift Registers (LFSR)*. Due to this reason, recently some new stream cipher architectures have been proposed. One of them is the so-named shrinking generator, introduced in 1993. With regard to its positive features, our paper is focused on the problem of synthesis of a derivative structure, named *p*-adic combiner generator.**

***Keywords*—Programming, Cryptography, Communication system signaling.**

I. INTRODUCTION

The stream ciphers are an important class of encryption algorithms, which have large application in the practice. As a result, the stream ciphers ought to satisfy great number of necessary conditions. The most significant of them are: resistance to crypto attacks, high performance velocity and cost-effective hardware implementation. Unfortunately, these conditions are in contradiction, because if the structure of the stream cipher is simple in order to provide high performance velocity and cost-effective hardware implementation, then the crypto reliability is low. For instance, the classical fast and cheap *Linear Feedback Shift Registers (LFSR)* are vulnerable to the so - named "Berlekamp – Massey crypto attack" [4]. This attack allows finding of all bits of a *LFSR* output sequence, if *2n* its consequent bits are known. Here *n* is the number of the cells tapped in the *LFSR*. Having in mind the advantages of the stream ciphers with simple structure, recently some theoreticians [8] proposed a new approach to stream cipher design. The basic idea of this approach is building devices with high crypto reliability combining in some appropriate way crypto vulnerable, but fast and cheap elements (including *LFSR*). This meaning of stream cipher design leaded to introducing of a few new architectures. One of them is the so-named shrinking generator [2], introduced in 1993. It is a promising candidate for high-speed encryption applications due to its simplicity and provable properties.

With regard to positive features of the shrinking generator, our paper is focused on the problem of synthesis of a derivative structure, named *p*-adic combiner generator.

The paper is organized as follows. First, the basics of *p*-adic Feedback with Carry Shift Register *(FCSR)* are recalled. After then, an algorithm for automatic synthesis of *p*-adic combiner generator is presented. It is practically realized in Visual C++ environment. Finally, the advantages and possible areas of application of our algorithm are discussed.

II. BASICS OF *FCSRS*

In the shrinking generator, a control *LFSR* R_0 is used to select a portion of the output sequence of a second *LFSR* R_1. The keystream produced is, therefore, a *shrunken* version (also known as an *irregularly decimated subsequence*) of the output sequence of R_1, as depicted in Fig. 1.

The algorithm of shrinking generator consists of the following steps:

1. Registers R_0 and R_1 are clocked.
2. If the output of R_0 is 1, the output bit of R_1 forms part of the keystream.
3. If the output of R_0 is 0, the output bit of R_1 is discarded.

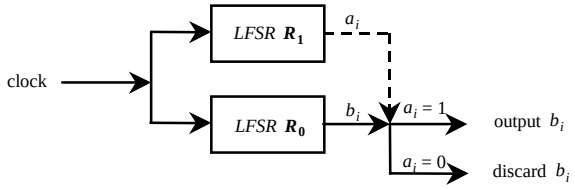

Fig. 1. The shrinking generator

Let R_0 and R_1 be maximum-length *LFSRs* of lengths L_0 and L_1, respectively, and let x be an output sequence of the shrinking generator formed by R_0 and R_1. If $\gcd(L_0, L_1) = 1$, the x has period $(2^{L_1} - 1).\ 2^{L_0-1}$ [8]. The linear complexity $L(x)$ of x satisfies Eq. (1) [8]:

$$L_1.2^{L_0-2} < L(x) \le L_1.2^{L_0-1} \qquad (1)$$

Suppose that the connection polynomials for R_0 and R_1 are chosen uniformly at random from the set of all primitive

[1]Borislav P. Stoyanov is with the Faculty of Mathematics, Informatics and Economics, Shumen University, 1 Universitetska Str., 9700 Shoumen, Bulgaria, E-mail: bpstoyanov@abv.bg

[2]Borislav Y. Bedzhev is with the Faculty of Artillery, Air Defense and Communication Information System, National Military University, 1 Karel Shkorpil Str., 9713 Shoumen, Bulgaria, E-mail: bedzhev@mail.pv-ma.bg

polynomials of degrees L_0 and L_1 over $\mathbf{Z_2}$. Then the distribution of patterns in x is almost uniform [8].

For maximum security, R_0 and R_1 should be maximum-length *LFSRs*, and their lengths should satisfy $gcd(L_0, L_1) = 1$. Moreover, secret connection should be used. Subject to these constraints, if $L_0 \approx m$ and $L_1 \approx m$, the shrinking generator has a security level approximately equal to 2^{2m}. Thus, if $L_0 \approx 64$ and $L_1 \approx 64$, the generator appears to be secure against all presently known attacks [2], [6], [8].

The above presented classical concept of shrinking generator has been extended from Klapper and Coresky in [3]. How it is shown on Fig. 2, in the derivative shrinking generator the controlling *LFSR* from Fig. 1 is replaced with a p-adic Feedback with Carry Shift Register *(FCSR)* R_0. It selects a portion of the output sequences of *LFSRs* $R_1 \div R_{p-1}$. Therefore, the produced keystream is a *shrunken and mixed* version of the output sequences of *LFSRs* R_1 to R_{p-1}. Due to this reason, the derivative structure of shrinking generator is named *p-adic combiner generator*.

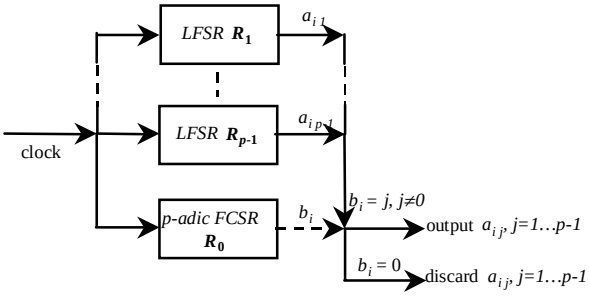

Fig 2. The p-adic combiner generator

The algorithm of p-adic combiner generator consists of the following steps:

1. All *LFSRs* from R_1 to R_{p-1} and *FCSR* R_0 are clocked.

2. If the p-adic output $b_i = j$ of control register R_0 is not equal to 0, the output bit of register R_j forms a part of the keystream. Otherwise, if the output $b_i = 0$ of control register R_0 is equal to 0, the all output bits are discarded.

In the origin papers of Klapper and Goresky only the case $p = 2$ is studied comprehensively [3]. It is shown [1], that the shrinking generator from Fig. 2 could use a generalization of *FCSRs* with stage contents and feedback coefficients in $\mathbf{Z}_p$, where p is a prime number, not necessarily 2. Due to this reason, we will explain the bases of the Feedback Shift Registers with Carry, assuming the common case $p \geq 2$ [1].

The *FCSRs* provide a simple and predictable method for the fast generation of pseudorandom sequences with good statistical properties and large periods. The *FCSRs* have an algebraic theory that parallels of *LFSRs*, in the case based on the 2-adic numbers.

Let we fix an odd positive integer $d \in \mathbf{Z}$ and let $r = \lfloor \log_p(d+1) \rfloor$ (where $\lfloor\ \rfloor$ denotes the floor or integer part). Write

$$d + 1 = d_1 p + d_2 p^2 + \ldots + d_r p^r \qquad (2)$$

for the p-ary representation of the integer $d + 1$ (so $d_r \neq 0$). The shift register uses r stages and $\lfloor \log_p(r) \rfloor$ additional p-its of memory (or less). The feedback connections are given by the p-its $\{d_1, d_2, \ldots, d_r\}$ appearing in Eq. (2).

Definition *1* **(Feedback with Carry Shift Register).** A Feedback with Carry Shift Register (*FCSR*) with connection integer d is the register depicted in Fig. 3.

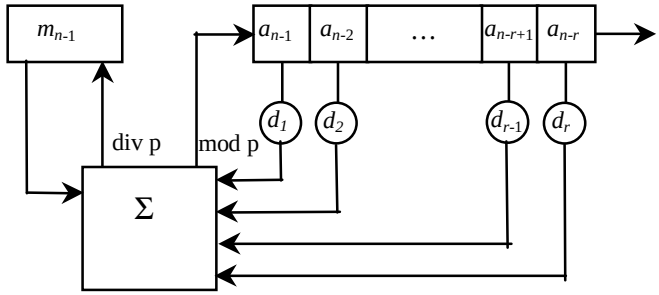

Fig. 3. Feedback with Carry Shift Register

Notice that $d_0 = -1$ does not correspond to a feedback tap, and that the coefficients of high powers of p are close to the output cell. In Fig. 3, Σ denotes integer addition. The contents of the register at any given time consists of r p-its, denoted a_{n-1}, a_{n-2}, …, a_{n-r+1}, a_{n-r}. The operation of the shift register is defined as follows:

1. Form the integer sum in Eq. (3):

$$\sigma_n = \sum_{k=1}^{r} d_k a_{n-k} + m_{n-1}. \qquad (3)$$

2. Shift the contents one step to the right, outputting the rightmost p-it a_{n-r}.

3. Place a_n from Eq. (4)

$$a_n = \sigma_n \pmod p \qquad (4)$$

into the leftmost cell of the shift register.

4. Replace the memory integer m_{n-1} with

$$m_n = (\sigma_n - a_n) / p = \lfloor \sigma_n / p \rfloor. \qquad (5)$$

We refer to d as the *connection integer* because its p-ary expansion gives the analog to the connection polynomial in the usual theory of linear feedback shift registers.

III. Algorithm for *p*-adic Combiner Generator Synthesis

From all stated in above parts of our paper (especially from Eq. (1)), it is easy to see that crypto resistance of the p-adic combiner generator depends essentially on the period of the

sequence, created from it. Due to this reason in [3] have introduced the following definition:

Definition *2* **(*l*-sequence).** An *l*-sequence is an *FCSR* sequence of maximum possible period *T*.

Klapper and Goresky have proved in [3] that the *l*-sequences are generated by *FCSR's* with connection integers *d* for which 2 is a *primitive root* of the finite Galois field *GF*(*d*). In this case, the period of the sequence is $T = d\text{-}1$ (and a single period of an *l*-sequence is a cyclic shift of the sequence formed by *reversing* a single period of the binary expansion of the fraction 1/*d*).

The results of Klapper and Goresky are generalized in [1] as follows:

Theorem *1*. If *c* and *d* are relatively prime, $-d < c \le 0$, and *d* is a prime for which *p* is *primitive root*, then the period *T* of the *p*-it sequence, presented the *p*-adic expansion of the rational number $\alpha = c / d$, is maximal.

In this case $T = \mathrm{ord}_d(p) = d\text{-}1$ (where *d* is the connection integer of the *FCSR*).

The Theorem 1 shows that the synthesis of combiner generator is reduced to the problem of proper determining of integers *p*, *c*, and *d*. With regard to this conclusion, we propose an algorithm for period calculations of *p*-adic expansion of an arbitrary rational number $\alpha = c / d$. It is realized in Visual C++ environment and is consisted of the following steps:

INPUT: The rational number in the form $\alpha = (c / d)*p^e$, where *c*, *d* and *e* are integers, *c*, *d* and *p* pairwise relatively primes and *d* positive.

OUTPUT: The period of the *p*-adic expansion of α.

1. Compute for *d* the Euler phi function $\varphi(d)$ [8].
2. Find the divisors $T_1, T_2, \ldots,$ of $\varphi(d)$.
3. The smallest T_i from the sequence $T_1, T_2, \ldots,$ such that

$$p^{T_i} \equiv 1 (\mathrm{mod}\, d) \qquad (6)$$

is the length *T* of *p*-adic expansion of $\alpha = c / d$.

In order to realize the above algorithm we are created in the class *p_adic*, described in [7], a new member function named *long period()*. It returns the period of given rational number. The member function uses an other member function, named *long powmod(long x, long n, long m)*, created in Visual C++ environment also. It executes "repeated square-and-multiply algorithm" (see [8]) to find an exponent in $\boldsymbol{Z_m}$ and returns $x^n \bmod n$.

The abilities of above mentioned member function for period calculation will be explained with following example. Let the rational number is $\alpha = (1/37)*3^0$, where the prime number is $p = 3$. Then we have:

1. $\varphi(37) = 36$.
2. The divisors of 36 are 1, 2, 3, 4, 6, 9, 12, 18 and 36.
3. The smallest divisor satisfying Eq. (6) is 36.

Analogously, the period of every rational number, satisfying conditions, showed in the Theorem 1, could be calculated. As an illustration, the periods of a few others (except $\alpha = (1/37)*3^0$) rational numbers are listed in the Table I. Our class member function *long period()* work properly if the size of *long integer* in Visual C++ is in the range $[0 \div 2^{31} - 1]$.

TABLE I
RESULTS FROM THE ALGORITHM FOR PERIOD CALCULATIONS

Rational number	Prime number	Period
-23/16	3	4
32/45312	5	58
-1/71927	13	2256

On the base of our new member function we propose an application "Period calculations"- period.exe, realized in Visual C++ environment. Its interface is shown on Fig. 4.

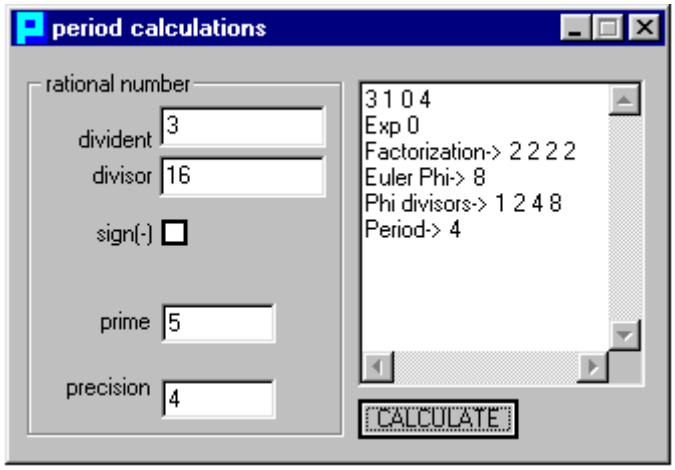

Fig. 4. Period calculations

The application calculates a Hensel code of a given rational number and corresponding period of *p*-adic expansion [5].

The program products, presented above and in [7], automate all procedures during the process of *p*-adic combiner generator synthesis. Actually, let us consider a variant of *p*-adic combiner generator, where *LFSRs* from R_1 to R_{p-1} (see Fig. 2) are replaced with analogous *FCSRs*. Then, the algorithm for *p*-adic combiner generator synthesis will consist the following steps.

1. Using our *p*-adic member function *long period()*, we can find *p* positive prime integers $d_0, d_1, \ldots, d_{p-1}$ for which *p* is a primitive root.

2. We determine the number r_j, $j = 0, 1, \ldots, p\text{-}1$ of cells tapped in every *FCSRs* $R_0, R_1, \ldots, R_{p-1}$:

$$r_j = \lfloor \log_p (d_j + 1) \rfloor. \qquad (7)$$

3. The feedback connections of every *FCSR* R_j from the sequence $R_0, R_1, \ldots, R_{p-1}$, are found in an analogous way to Eq. (2):

$$d_j + 1 = d_{1j}\, p + d_{2j}\, p^2 + \ldots + d_{rj}\, p^{r_j}\,; \qquad (8)$$

4. The proper initial loadings of every *FCSR* register are calculated applying our *p*-adic library [7] to Eq. (9):

$$a_{0j} + a_{1j}.p + \ldots + a_{r-1j}.p^{r_j-1} = \frac{c_j}{d_j} \bmod p^{r_j}\,. \qquad (9)$$

In Eq. (9) all c_j ought to satisfy the following conditions:
- c_j, d_j and p pairwise relatively primes;
- $(-d_j) < c_j \leq 0$;
- $|c_j| < d_j$.

IV. Conclusion

From all the above stated, it is easy to see that the algorithm of synthesis of *p*-adic combined generator, proposed in the paper, could be applied successfully for:

1. Developing of great number of stream ciphers, based on the *Feedback with Carry Shift Register* architecture;

2. Synthesis of pseudorandom sequences with variety of statistical properties such high linear span, low autocorrelation side-lobes and pair wise cross-correlation values, pair wise hamming distance.

It is necessary to emphasize that if the pseudorandom sequences, created with *Feedback with Carry Shift Register* architectures, are used to manipulate radio signals of some perspective communication systems, then the electromagnetic spectrum will be exploit very effectively. This conclusion follows from the fact that the low autocorrelation side-lobes and pair wise cross-correlation values of these pseudorandom sequences provide small radio interference among the consumers. Consequently, *FCSR* sequences may play a key role in the process of new generation personal communication system developing.

Acknowledgement

Authors would to thank Professor Rusin Petrov, whose observations made the drafting of the paper the best possible.

References

[1] B. Y. Bedzhev, Zh. N. Tasheva, V. A. Mutkov, "An Shrinking Data Encryption Algorithm with *p*-adic Feedback with Carry Shift Register", XII International Symposium of Theoretical Electrical Engineering ISTET 03, Warsaw, Poland, 6-9 July, 2003, Conference Proceedings, vol.II, pp. 397-400.

[2] D. Coppersmith, H. Krawczyk, Y. Mansour, "The Shrinking Generator", Proceedings of Crypto 93, Springer-Verlag, 1994, pp 22-39

[3] A. Klapper , M. Goresky, "2-adic Shift Register. Fast Software Encryption", Second International Workshop. (Lecture Notes in Computer Science, vol. 950, Springer Verlag, N. Y., 1994.) pp.174-178

[4] R. Lidl, H. Niederreiter, "Finite Fields", Addison – Wesley Publishing Company, London, England, 1983

[5] C. Limongelli, "The Integration of Symbolic and Numeric Computation by *p*-adic Construction Methods", Ph.D. thesis, University of Rome "La Sapienza", Italy, 1993.

[6] B. Schneier, "Applied Cryptography", John Wiley & Sons, New York, 1996.

[7] B. Stoyanov, B. Bedzhev, Zh. Zhekov, "Computation Model of *p*-adic Arithmetic", XXXIX International Scientific Conference on Information, Communication and Energy Systems and Technologies, ICEST 2004, 16-19 June 2004, Bitola, Macedonia, submitted for publication.

[8] P. van Oorshot, A. Menezes, S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996