A Study on Linear Complexity of *p*-ary Pseudo Random Sequences Generated by pGSSG

ANTONIYA TASHEVA*, ZHANETA SAVOVA-TASHEVA**, BOYAN PETROV*
* Faculty of Computer Systems and Technologies
Technical University of Sofia
8 Kliment Ohridski blvd., Sofia
BULGARIA
atasheva@tu-sofia.bg
** Faculty of Artillery, Air Defense and Communication and Information Systems
National Military University
1 Karel Shkorpil Str., Shumen
BULGARIA
zh.savova@mail.bg

Abstract: - Nowadays, when fast-growing areas, such as IoT, automotive systems, sensor networks, healthcare, distributed control systems, cyber-physical systems, and the smart grid, are widely used there is a need of specific device design with a better balance between security, performance, and resource requirements for resource-constrained environments. Because low linear complexity is undesirable for cryptographic applications, there is a necessity for lightweight cryptographic stream ciphers development with high linear complexity. Due to this reason a linear complexity of *p*-ary Generalized Self-Shrinking Generator (pGSSG), which has very simple design and is suitable for lightweight stream cipher, is investigated in this paper. Mathematically was shown that the extended Euclidean algorithm can be applied to find the linear complexity of *p*-ary pseudorandom sequences. The conducted tests show that the pGSSG linear complexity is close to its theoretical upper bound.

Key-Words: - Linear complexity, pseudo random sequences, pGSSG, extended Euclidean algorithm, pLFSR, lightweight cryptography

1 Introduction

The last decade of the communications development is characterized by the involvement of resource-constrained devises that are interconnected and through joint work provide advanced services to the customers. Vast are the areas of application as Internet of Things (IoT), sensor networks, healthcare systems, cyber-physical systems, RFID devices, the smart grid and more. Ensuring security in these devices is critical task because they deliver sensitive privacy information.

Lightweight cryptography is a subfield of cryptography that aims to provide solutions tailored for resource-constrained devices. [8]. Lightweight cryptographic primitives, including block ciphers, hash functions, message authentication codes and stream ciphers [2, 5, 8], are not intended for a wide range of applications and may impose limits on the power of the attacker. [8].

Stream ciphers are typically fast and work on only a few bits at a time. Because of that they are with simple design, consume low power, and have relatively low memory requirements. These features make them an attractive choice for resource-constrained devices and for applications where the amount of data is either unknown, or continuous, like network streams. Lightweight stream ciphers are expected to provide confidentiality and data integrity in such devices. Recently, International Organization for Standardization [4] standardized two stream ciphers for lightweight cryptography: Trivium and Enocoro.

Thus, important current problem is the task to find nonlinear lightweight cryptography generators that generate sequences with good distribution and statistical properties. Their security characteristics like randomness and unpredictability of the sequences must be assessed.

Recent studies on linear complexity have been conducted over almost perfect binary and ternary sequences. The authors show that the considered sequences have high linear complexity [3]. Other authors [16] research the linear complexity and minimal polynomials of some generalized cyclotomic sequences over GF(q). Randrianarisoa presents in [11] a coding theory based on the linear

complexity. He aims to design new code based cryptographic systems and thus counts how many finite sequences have linear complexity bounded by some integer.

The linear complexity is used a measure of the unpredictability of a sequence over a finite field. It is the main measurement of the complexity of a sequence and is a critical factor to investigating the non-linearity introduced to pseudo-random sequence generators due to the fact that each sequence can be recovered by the Berlekamp-Massey algorithm [1, 6] if 2λ of consecutive values of its elements are known.

Linear feedback shift registers (LFSR) have been widely used for the generation of pseudo random sequences as their output has long period, good statistical and correlation properties, needed for their practical application. Pseudorandom number generators need to apply some nonlinear rules to their LFSR in order to increase the linear complexity of their output sequences.

This paper focuses on the *p*-ary Generalized Self-Shrinking Generator (pGSSG) proposed in [13] which has very simple design and is suitable for a lightweight stream cipher.

The pGSSG architecture, given in Figure 1, consists of a pLFSR register *A*, whose length will be denoted by *L*. The feedbacks of the register are constructed according to [14] with a chosen primitive polynomial. The pGSSG selects a portion of the output *p*-ary LFSR sequence by applying a self-shrinking selection rule that chooses a single *p*-ary digit in the pLSFR output *p*-tuple or discards everything.

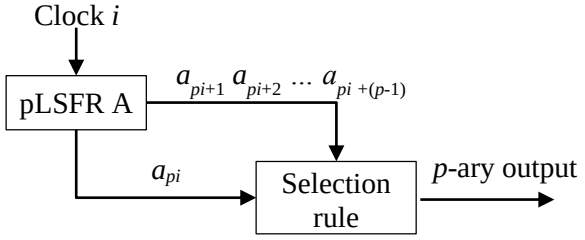

Fig. 1. *p-ary Generalized Self-Shrinking Generator*

This paper aims to evaluate the nonlinearity introduced in the pGSSG schema by calculating the linear complexity of its output sequences.

The paper is organized as follows. First, an overview of the linear complexity theory is made. Next, the extended Euclidean algorithm is explained and an algorithm for evaluating the linear complexity of *p*-ary sequences is formulated. Section 4 shows the results for the evaluated sequences and finally conclusions are made.

2 Linear Complexity Theory

The linear complexity of a sequence *S* can be defined as the length of the shortest linear recurrence relation satisfied by *S*. In engineering terms, λ_S is 0 if *S* is the zero sequence and otherwise it is the length of the shortest linear feedback shift register LFSR that can generate *S* [9].

Linear complexity of a given sequence *S* can be determined by using the Berlekamp-Massey algorithm. The algorithm is efficient for sequences with low linear complexity and hence such sequences can easily be predicted. [9, 12, 17]

If the periodic sequence *S* is collated to the power series *S*(*x*)

$$S(x) = s_0 + s_1x + \ldots + s_{T-1}x^{T-1} + s_Tx^T + \ldots \quad (1)$$

and knowing the fact that $s_{i+T} = s_i$ for every $i \geq 0$, *T* is the period of *S*, we can represent *S*(*x*) as:

$$S(x) = (s_0 + s_1x + \ldots + s_{T-1}x^{T-1})(1 + x^T + x^{2T} + \ldots). \quad (2)$$

The second multiplier in (2) is a geometric progression and its sum is

$$1 + x^T + x^{2T} + \ldots = 1/(1 - x^T). \quad (3)$$

Therefore

$$\begin{aligned} &S(x) \\ &= (s_0 + s_1x + \ldots + s_{T-1}x^{T-1})(1 - x^T)^{-1} \\ &= \frac{S^T(x)}{1-x^T} = \frac{S^T(x)/\gcd(S^T(x), 1-x^T)}{(1-x^T)/\gcd(S^T(x), 1-x^T)}, \end{aligned} \quad (4)$$

where $S^T(x) = s_0 + s_1x + \ldots + s_{T-1}x^{T-1}$ is a polynomial with power lower than the period *T*. The polynomial $S^T(x)$ is called generating function for the sequence *S*.

Comparing the upper equation (4) with the generating function of the pLFSR, (5) [14, 15]

$$S(x) = -\frac{h^0(x)}{q(x)}, \quad (5)$$

we can conclude that with accuracy to a constant the feedback polynomial *q*(*x*) has the following form

$$q(x) = \frac{1-x^T}{\gcd(S^T(x), 1-x^T)}. \quad (6)$$

Thus, the linear complexity λ_S is the power of the polynomial $\frac{1-x^T}{\gcd(S^T(x),1-x^T)}$.

In this case the following theorem is valid [7]

Theorem 1. If $\boldsymbol{S} = s_0, s_1, \ldots, s_{T-1}, \ldots.$ is a sequence with period *T* over the field *F*, the linear complexity of the sequence $\boldsymbol{S}$ is

$$\lambda_S = T - \deg(\gcd(S^T(x), 1-x^T)). \quad (7)$$

3 Algorithm for Determining Linear Complexity of a p-ary Sequence

In this paragraph, a mathematical algorithm will be justified to determine linear complexity of a sequence over $GF(p)$ using the extended Euclid algorithm.

3.1 Extended Euclidean Algorithm

The extended Euclidean algorithm is widely used in the contemporary algebraic and communication systems. It calculates the greatest common divisor gcd(a, b) of two elements of the ring R and the coefficients s and t such that

$$\gcd(a, b) = as + bt. \tag{8}$$

Let $|a| > |b|$, by consecutive execution of the division algorithm one can receive the following equation sequence

$$\begin{aligned} a &= bq_1 + r_1 \\ b &= r_1q_2 + r_2 \\ r_1 &= r_2q_3 + r_3 \\ &... \\ r_{j-2} &= r_{j-1}q_j + r_j \\ r_{j-1} &= r_jq_{j+1} + 0 \end{aligned} \tag{9}$$

According to the Euclidean algorithm the greatest common divisor of a and b is r_j ($\gcd(a, b) = r_j$).

In terms of the extended Euclidean algorithm at each division step the intermediate coefficients s_i and t_i, are calculated such that

$$as_i + bt_i = r_i \tag{10}$$

using the equations

$$\begin{aligned} s_i &= s_{i-2} - q_i\, s_{i-1},\ i = 1, 2, \ldots \\ t_i &= t_{i-2} - q_i\, t_{i-1},\ i = 1, 2, \ldots \end{aligned} \tag{11}$$

Speaking specifically about the sequences produced by a *pLFSR* and with accuracy to a constant -1 in (5) the output sequence can be represented as a division of two polynomials

$$S(x) = \frac{h(x)}{q(x)} \tag{12}$$

By equating 12 and 4 we get

$$q(x).S^T(x) + h(x).x^T = h(x) \tag{13}$$

The resulting equation (13) gives us the reason to apply the extended Euclidean algorithm for finding the gcd(x^T, $S^T(x)$) and calculating the minimal polynomial $q(x)$ generating the sequence S. Moreover, the polynomial $h(x)$ that defines the initial state of the pLFSR will be also calculated as a result of the extended Euclidean algorithm.

3.2 Linear Complexity of p-ary Sequences

Following the mathematical justification for the extended Euclidean algorithm in 3.1 the algorithm 1 for determining the linear complexity of a p-ary sequence can be synthesized.

Algorithm 1. Determining the linear complexity of a p-ary sequence.

Input: n consecutive elements of the p-ary sequence (coefficients of $S^T(x)$).

Initialization: $s_{-1}(x) = 1$, $s_0(x) = 0$, $t_{-1}(x) = 0$, $t_0(x) = 1$, $r_{-1}(x) = x^n$, $r_0 = S^T(x)$, $j = 0$

while $\deg r_j(x) \geq n/2$ **do**

$j = j + 1$

$q_j(x) = \lfloor r_{j-2}(x)/r_{j-1}(x) \rfloor$

$r_j(x) = r_{j-2}(x) - q_j(x)r_{j-1}(x)$,

$s_j(x) = s_{j-2}(x) - q_j(x)s_{j-1}(x)$,

$t_j(x) = t_{j-2}(x) - q_j(x)t_{j-1}(x)$

Output: $q(x) = t_j(x)$, $h(x) = s_j(x)$, $\lambda = \deg(q(x))$.

Next, the algorithm will be used for evaluation of the p-ary sequences generated by the pLFSRs and pGSSG.

4 Evaluation of pGSSG output sequences

In order to evaluate the work of the proposed algorithm 1 for determining the linear complexity of p-ary sequences a C# implementation is made.

First, we will show the algorithm's work by presenting an example for calculating the linear complexity of a certain p-ary sequence generated by a pLFSR. Let $GF(3^2)$ and the primitive polynomial $q(x) = 1 + x + 2x^2$ are chosen. A 3LFSR with Galois architecture is build according to [14] and the corresponding output sequence is [12202110] and the period is $T = 8$. As the length of the 3LSFR is $L = 2$ and $q(x)$ is primitive polynomial the expected linear complexity of the given sequence is $\lambda = 2$.

Next, the algorithm execution is presented.

Input: [1220], $n = 4$ consecutive elements of the p-ary sequence.

Initialization:

$s_{-1}(x) = 1$, $t_{-1}(x) = 0$,

$s_0(x) = 0$, $t_0(x) = 1$,

$r_{-1}(x) = x^4$,

$r_0 = S^n(x) = 1 + 2x + 2x^2$

Calculations:

$$\begin{aligned} q_1(x) &= \lfloor r_{-1}(x)/r_0(x) \rfloor = \\ &= \lfloor x^4/(1 + 2x + 2x^2) \rfloor = \\ &= 1 + x + 2x^2 \end{aligned}$$

$$r_1(x) = r_{-1}(x) - q_1(x)r_0(x) = 2$$

$$s_1(x) = s_{-1}(x) - q_1(x)s_0(x) = 1$$

$t_1(x) = t_{-1}(x) - q_1(x)t_0(x) = 2 + 2x + x^2$

Output: Primitive polynomial $q(x) = t_1(x) = 2 + 2x + x^2$, initial state of the register $h(x) = s_1(x) = 1$, linear complexity $\lambda = \deg(q(x)) = 2$

As one can see we found that the linear complexity $\lambda = 2$ is equal the expected value and the algorithm finished at its first step. The primitive polynomial that was found with the algorithm is equivalent to the one that was used to generate the sequence.

Tests with different sequences, i.e. different Galois Fields, primitive polynomials and lengths of pLFSRs, were made, all reconfirming the results.

As the aim of this paper is to investigate the non-linearity that is introduced to the output sequence of the pGSSG by the self-shrinking rule of the *p*-ary register's output, sets of such output sequences will be investigated with algorithm 1 in order to calculate their linear complexity.

The equation (7) sets the boundaries of the linear complexity of a *p*-ary sequence, i.e. $0 \le \lambda_S \le T$. The value $\lambda_S = 0$ is assumed to be the linear complexity of the zero sequence $S = (0, 0, \ldots, 0)$. The rule resulting from the Berlekamp-Massey algorithm says that for determining the linear complexity λ of a sequence S, $2.\lambda$ consecutive elements of S are needed. Thus, the algorithm will be tested as its input will be fed with sequenced with length $2n$, that corresponds to the maximal possible linear complexity λ.

Table 1. *Results for linear complexity of output pGSSG sequences*

GF(p^n)	Count	Period	Linear complexity	
			Min.	Max.
GF(3^2)	2	6	4	5
GF(3^3)	4	18	15	16
GF(3^4)	8	54	48	51
GF(3^5)	22	162	152	158
GF(3^6)	48	486	478	481
GF(5^2)	4	20	18	19
GF(5^3)	20	100	76	98
GF(5^4)	48	500	492	497
GF(7^2)	8	42	35	41
GF(7^3)	36	294	288	292
GF(7^4)	160	2058	2050	2055

More than 360 tests are conducted using the algorithm implementation for evaluation the linear complexity of the output sequences generated by *pGSSG* in different fields GF(p^n). Table 1 shows a summary of the test results including the number of possible output pGSSG sequences (Count), their period T, and the minimum and maximum results for the calculated linear complexities.

As it was shown in section 2, the linear complexity of the underlying pLSFR in pGSSG is equal to its length L. We can notice in Table 1 that the expected maximum value of the linear complexity of the non-linear pGSSG generator is

$$\lambda_{\max} = T - (L - 1) = (p - 1)p^{L-1} - (L - 1). \quad (51)$$

Therefore, the proposed non-linear method for self-shrinking in the pGSSG increases the linear complexity with the following coefficient

$$K_\lambda = \frac{(p - 1)p^{L-1} - (L - 1)}{L}. \quad (52)$$

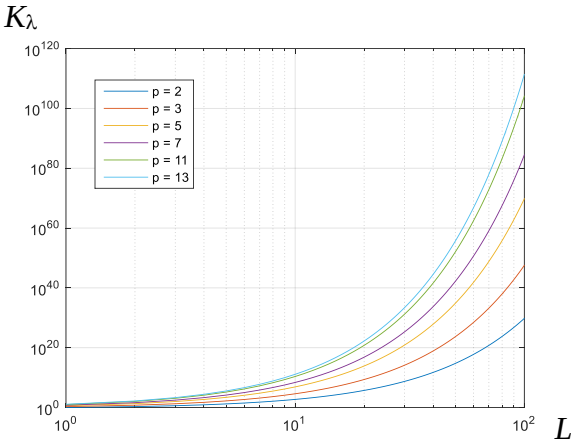

Fig. 2. *Coefficient for increasing the linear complexity for pGSSG*

As it is shown in [13] when $p = 2$ the generator *pGSSG* is converted into the classic *SSG* [10]. Having that in mind figure 2 represents the linear complexity enhancement dependency of *pGSSG* over *SSG* (shown in blue color) on the length L of the used *pLFSR* with different prime p.

5 Conclusion

In this paper we have investigated the linear complexity of *p*-ary pseudo random sequences produced by *p*-ary generalized self-shrinking generator. We have mathematically justified that the extended Euclidian algorithm can be applied to find the linear complexity of *p*-ary pseudorandom sequences. Using this algorithm the linear complexity of output sequence of the pGSSG has been calculated. The more than 360 tests that have been performed show that the linear complexity of pGSSG output sequences is close to the maximum theoretical.

Acknowledgements

This paper is a result of a project supported by the National Science Fund, Ministry of Education and Science, Bulgaria via FINANCIAL SUPPORT FOR PROJECT OF JUNIOR RESEARCHERS – 2016 [Grant Number DM07/5 – 15.12.2016]

References:

[1] Berlekamp, E. R. (1968). Algebraic coding theory. McGraw-Hill Book Co., New York-Toronto, Ont.-London.

[2] Buchanan, W. J., Li, S., & Asif, R. (2017). Lightweight cryptography methods. Journal of Cyber Security Technology, 1(3-4), 187-201.

[3] Edemskiy, V., & Minin, A. (2016). About the linear complexity of the almost perfect sequences. International Journal of Communications, 1, 223-226.

[4] ISO/IEC 29192-3:2012. International standard for lightweight cryptographic methods, ISO/IEC, 2012.

[5] Manifavas, C., Hatzivasilis, G., Fysarakis, K., & Papaefstathiou, Y. (2016). A survey of lightweight stream ciphers for embedded systems. Security and Communication Networks, 9(10), 1226-1246.

[6] Massey, J. L. (1969). Shift-register synthesis and BCH decoding. IEEE Trans. Information Theory, IT-15, 122–127.

[7] Massey, James L., and Shirlei Serconek. „Linear complexity of periodic sequences: a general theory.“ In *Advances in cryptology—CRYPTO'96*, pp. 358-371. Springer Berlin Heidelberg, 1996.

[8] McKay, K. A., Bassham, L., Turan, M. S., & Mouha, N. (2017). NISTIR 8114 report on lightweight cryptography. National Institute of Standards and Technology (NIST), Gaithersburg.

[9] Meidl, W., Winterhof, A. (2013). Linear complexity of sequences and multisequences, *Handbook of finite fields*. Chapter 10.4, pp. 318-330. Chapman and Hall/CRC.

[10] Meier, W., Staffelbach, O.: The self-shrinking generator. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 205–214. Springer, Heidelberg (1995).

[11] Randrianarisoa, T. (2018). Coding Theory using Linear Complexity of Finite Sequences. arXiv preprint arXiv:1802.10034.

[12] Rueppel R.A. (1986) Linear Complexity and Random Sequences. In: Pichler F. (eds) Advances in Cryptology — EUROCRYPT' 85. EUROCRYPT 1985. Lecture Notes in Computer Science, vol 219. Springer, Berlin, Heidelberg

[13] Tasheva, A., Tasheva, Zh., Milev, A., Generalization of the Self-Shrinking Generator in the Galois Field GF(p^n), Advances in Artificial Intelligence, vol. 2011, Article ID 464971, 10 pages, 2011.

[14] Tasheva, A., Savova-Tasheva, Zh., Petrov, B., Stoykov, K., Determining the Feedback Multipliers in a p-ary Linear Feedback Shift Registers, WSEAS Transactions on Systems and Control, Volume 13, 2018, Art. #45, pp. 420-424

[15] Venkateswarlu, A. (2007). Studies on error linear complexity measures for multisequences (Doctoral dissertation).

[16] Wang, Q., Jiang, Y., & Lin, D. (2015). Linear complexity of binary generalized cyclotomic sequences over GF (q). Journal of Complexity, 31(5), 731-740.

[17] Winterhof, A., Linear complexity and related complexity measures, in Selected Topics in Information and Coding Theory, vol. 7. Hackensack, NJ, USA: World Scientific, 2010, pp. 3–40.