Original Research Article: full paper

(2017), «EUREKA: Physics and Engineering»
Number 1

DEVELOPMENT OF THE SEARCH METHOD FOR NON-LINEAR SHIFT REGISTERS USING HARDWARE, IMPLEMENTED ON FIELD PROGRAMMABLE GATE ARRAYS

Nikolay Poluyanenko
Department of Information Systems and Technologies Security
V. N. Karazin Kharkiv National University
4 Svobody Sq., Kharkiv, Ukraine, 61022
rsnos@mail.ua

Abstract

The nonlinear feedback shift registers of the second order in GF(2) are considered, because based on them it can be developed a generator of stream ciphers with enhanced cryptographic strength.

Feasibility of nonlinear feedback shift register search is analyzed. These registers form a maximal length sequence, using programmable logic devices.

Performance evaluation of programmable logic devices in the generation of pseudo-random sequence by nonlinear feedback shift registers is given. Recommendations to increase this performance are given. The dependence of the maximum generation rate (clock frequency), programmable logic devices on the number of concurrent nonlinear registers is analyzed.

A comparison of the generation rate of the sequences that are generated by nonlinear feedback shift registers is done using hardware and software.

The author suggests, describes and explores the search method of nonlinear feedback shift registers, generating a sequence with a maximum period. As the main result are found non-linear 26, 27, 28 and 29 degrees polynomials.

Keywords: stream ciphers, random number generators, M-sequence, search of nonlinear shift registers, non-linear polynomials.

DOI: 10.21303/2461-4262.2017.00271

© *Nikolay Poluyanenko*

1. Introduction

There is currently rapid development of cryptanalytic systems. One of the main requirements to the main element of the cryptographic stream encryption system – a generator of pseudo-random sequences (PRSs) is an indiscernible of the sequence, complexity, speed and repetition period for PRSs [1]. Cryptographic primitives that meet these requirements are constructions on the basis of linear feedback shift registers (LFSRs).

Common cryptographic algorithms, which are built using LFSRs, are: stream cipher A5/1, used to ensure the privacy of telephone mobile communication of GSM standard [2]; stream cipher E0, used in the Bluetooth protocol [3], etc. LFSR main disadvantage is its linearity, which leads to a relatively simple cryptanalysis [4].

2. Overview of the problem and formulation of research problems

As an alternative for LFSRs to PRSs generation in the stream cipher nonlinear feedback shift registers (NLFSRs) are proposed. NLFSRs on the basis of stream ciphers are included in Achterbahn [5], Dragon [6], Grain [7], Trivium [8] and VEST [9]. In [10] it is shown that NLFSRs are more resistant to cryptanalytic attacks than LFSRs. Using L cells NLFSRs, a cryptanalyst can take up to $O\left(2^L\right)$ [11] or as given in [12], a sequence of $L\cdot(L+1)/2+L$ bits is necessary to determine the structure of L-bit NLFSR generating this sequence.

At the same time, large size NLFSR generation with a guaranteed period remains an unsolved problem [13]. Only some special cases were considered in [14]. It is known that LFSRs generate maximum length sequence (M-sequence) equal to 2^L-1 if and only if when its characteristic polynomial is primitive [15]. For NLFSRs such property is not found to this day. Small NLFSRs with a maximum period can be built with the help of simulation. Nevertheless, modern computing capacities allow to simulate NLFSRs with a size only to L<35 [16]. In [17], NLFSRs with a size to L<26 are shown. This is insufficient for cryptographic applications that require long periods, for example, 2^{128} [18].

Non-standard hardware solution for implementation of cryptographic algorithms is the use of field programmable gate arrays (FPGAs) [19]. FPGA allows to construct digital devices with high-level hardware description languages, which reduces the complexity of the development and allows the reuse the code through the use of IP-cores [20].

In [21], a search of NLFSRs that are implemented on FPGAs on the basis of analysis of de Bruijn sequences is carried out. Good statistical properties of NLFSR-generated sequences are confirmed and obtained non-linear 25 and 27 polynomials are given.

FPGAs have a high versatility and reliability, which ensures thorough testing [22]. Programmable logic is attractive due to possibility of providing performance close to ASIC technology, to achieve high throughput with high construction flexibility and low power consumption [23].

One of the important advantages of the implementation of cryptographic algorithms on FPGAs is the ability to construct a parallel and asynchronous architecture, superior to GPU and CPU microprocessor-based solutions in performance [24].

In this paper we study NLFSRs performance implemented on the FPGAs and problems of their optimization. Search method of NLFSRs generating M-sequence (M-NLFSRs) is given. This method is based on a practical synthesis of the results obtained previously. The aim of this research is to explore the possibility of NLFSR implementation of FPGA, as well as the development of an acceptable, for hardware and temporal parameters, the search algorithm for M-NLFSR.

3. M-NLFSR search method

The registers are used multiplication of only two cells, and such NLFSRs are called NLFSRs of the second order. Later, in NLFSR operation we understand NLFSR of the second order in GF(2). General NLFSR construction for the register, consisting of $L = 4$ cells, is shown in **Fig. 1**.

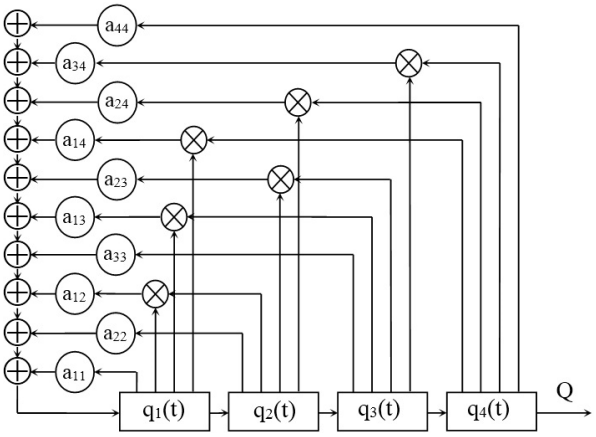


Fig. 1. General NLFSR construction

Where $a_{ij} \in \{0,1\}$ appropriate the presence or absence of feedback, $q_i(t) \in \{0,1\}$ – value of the i-th register at the time t, Q – generated sequence, bits. $\otimes$ denotes a nonlinear function of multiplication and $\oplus$ – linear function of summation.

Preliminary feedback coefficients a_{ij} are limited that equate some of the coefficients to zero. The total number of variables (initially not equal to zero) of feedback coefficient a_{ij} denoted as n_1. As has been shown in [25], the number of feedback coefficients a_{ij} for NLFSR with L size is calculated by ratio $n_L = L \cdot (L+1)/2$, therefore, n_1 can take values in the range $0 \le n_1 \le n_L$.

M-NLFSR search method consists of two stages that can be carried out simultaneously or sequentially.

Stage 1: There is a decrease in the test set by excluding NLFSRs not forming an M-sequence. The total amount of rejected polynomials is selected in the range of 90–99 % of the possible amount. The volume of rejected polynomials is limited by available memory and time resources.

Rejected set is defined by an analytical method, which is implemented in software. The analytical method consists of checking the feedback coefficients to meet certain requirements as

detailed in [25, 26, 12]. The essence of these requirements is to analyze the place, type and arrangement of non-zero coefficients of feedback a_{ij}. Mismatch of a_{ij} coefficients to specified requirements means the inability of NLFSR to form M-sequence and, therefore, its rejection.

After the first stage, we obtain the set of polynomials, acceptable (for time-consuming) to check in the second stage. Let's denote the number of such polynomials as k_0^{FPGA}.

Stage 2: Direct verification of the set of polynomials, obtained after the first stage, is carried out for the possibility of M-sequence generation by them is carried out using computer system.

As shown below, for moving the computational process from CPU on the FPGA, performance (speed) of each of NLFSRs is maintained. This fact allows to tens or hundreds of times increase the overall performance of the complex for M-NLFSR search compared to use only a PC.

4. The structure of the computer system for M-NLFSR search

Complete structure of computer system, which was used for the calculations, is shown in **Fig. 2.**

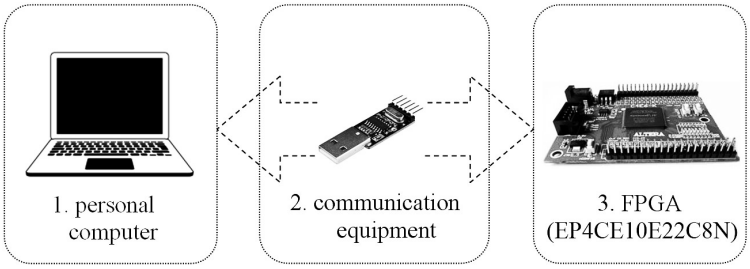


Fig. 2. The structure of the computer system for M-NLFSR search

The hardware consists of the card with integrated chip of Altera company of Cyclone IV E FPGA EP4CE10E22C8N family (China). FPGAs are made at 60 nm optimized process and includes 10 320 logic elements (LEs – Logic Elements). Project for FPGA is written in Verilog HDL, for compilation and simulation CAD CAD Altera Quartus Prime Version 16.0.0 was used.

USB-TTL converter was used as communication equipment (the maximum data transfer speed of 921 600 bps), performs the function of Universal Asynchronous Receiver/Transmitter (UART) between the FPGA and the PC. Two USB-TTL converters were used to enhance the data rate. They operate in parallel with a maximum data rate.

The software realizes: reading data from the hard disk; formation of a package to send to the FPGA; synchronous (for two USB-TTL converters) data sending to the specified COM port; receiving information through the COM port from the FPGA; decoding, visualization and storage on disk test results carried out in the FPGA.

5. FPGA-based NLFSR performance analysis

Hardware performance is a number of tested NLFSRs per time unit. Hardware performance consists of the following components:

– time for one cycle in the module that implements NLFSR operation (denoted as t_{cycle}^{FPGA}). Cycle is implementation of cyclic changes in NLFSR state in which all possible combinations are implemented for M-NLFSR, and the system returns to the initial state. Registers change their state in one clock cycle of the reference frequency, therefore amount of these clock cycles in a cycle should be 2^L;

– the number of simultaneously tested NLFSRs per cycle (denote as k_{NLFSR}^{FPGA}). The parameter depends on the number of LEs, engaged for one NLFSR and total capacity of used FPGA;

– operational clock frequency. Upon reaching a certain maximum clock frequency (f_{max}^{FPGA}) FPGA elements have no time to fully execute that leads to errors in the determination of M-RSNOS.

The increase of k_{NLFSR}^{FPGA} leads to an increase in the total number of involved LEs. As an example, the number LEs involved in FPGA to M- NLFSR search for $L = 29$ and $n_1 = 28$ depending from k_{NLFSR}^{FPGA}, are shown in **Fig. 3.**

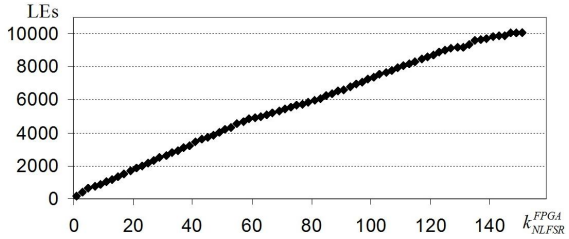
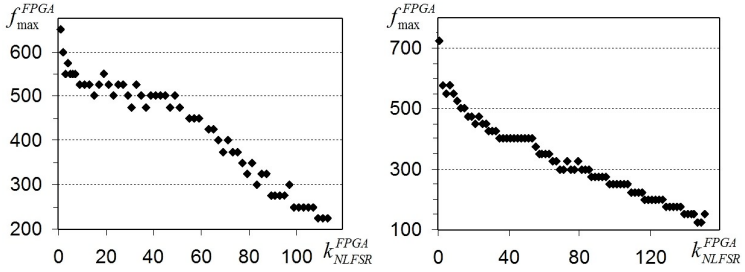


Fig. 3. Dependence of LEs involved in FPGA on k_{NLFSR}^{FPGA} for $L = 29$

Due to the increase in the number of LEs involved in FPGA, their mutual arrangement and increasing the physical distance between the interacting elements in the chip, there is a decrease of f_{max}^{FPGA}. Also f_{max}^{FPGA} significantly affect the temperature factor, which can be partly explains a small deviation observed in **Fig. 4.**

Let's consider in detail how to change the above characteristics on the number of simultaneously tested NLFSRs. **Fig. 4, *a*** shows the dependence results of observed f_{max}^{FPGA} on k_{NLFSR}^{FPGA} for NLFSRs with a size of $L = 28$ and $n_1 = 28$.



a *b*

Fig. 4. Dependence of f_{max}^{FPGA} on k_{NLFSR}^{FPGA}: *a* – for $L = 28$, *b* – for $L = 29$

Taking into account obtained results, FPGA configuration modification has been made for $L = 29$. Modification allowed to improve performance for $k_{NLFSR}^{FPGA} = 1$ to $f_{max}^{FPGA} = 725$ MHz and reduce LEs in the module. **Fig. 4, *b*** shows the results of dependence of f_{max}^{FPGA} on k_{NLFSR}^{FPGA} for NLFSRs with a size of $L = 29$ and $n_1 = 28$.The change of the clock frequency for f_{max}^{FPGA} determination was conducted discretely, in steps of 25 MHz.

Knowing f_{max}^{FPGA}, we can determine t_{cycle}^{FPGA} for different number of simultaneously calculated NLFSRs in FPGA. Based on these data, we can calculate the time t_0^{FPGA} that is necessary to spend for the analysis of all k_0^{FPGA} polynomials in the search for the M-NLFSR in hardware. Elapsed time (in hours) will be calculated by the following relationship:

$$t_0^{FPGA} = \frac{1}{3600\,\text{sec}} \cdot \frac{k_0^{FPGA}}{k_{NLFSR}^{FPGA}} \cdot t_{cycle}^{FPGA}.$$

The dependence of the estimated time for M-NLFSR search for L=29 and restrictions imposed, in which n_1=28, $k_0^{FPGA} = 17\,619\,713$, on the number of simultaneously processed polynomials in FPGA is shown in **Fig. 5**.

As can be seen in the graph, maximum system performance can't be achieved with maximum use of all FPGA elements due to the need to reduce the clock frequency. The optimal number of simultaneously tested NLFSRs in this case is in the range from 79 to 115, after which the time for search begins to increase.

In [27], there is a generation time of 1 GB by the generator, written in assembler and optimized for different NLFSRs, including LFSRs. To compile all the examples FASM was used (flat assembler version 1.71.51). The calculations were performed on a personal computer (64 bit Windows 7 SP 1, Intel Core i5-3210M CPU 2,5GHz processor). According to the research results,

the time to generate the 1 GB ranges from 16 to 76 seconds, depending on the size and form of the polynomial.

For comparison, **Table 1** shows the time for generation of 1 GB NLFSR implemented in FPGA on different clock frequencies (f^{FPGA}).

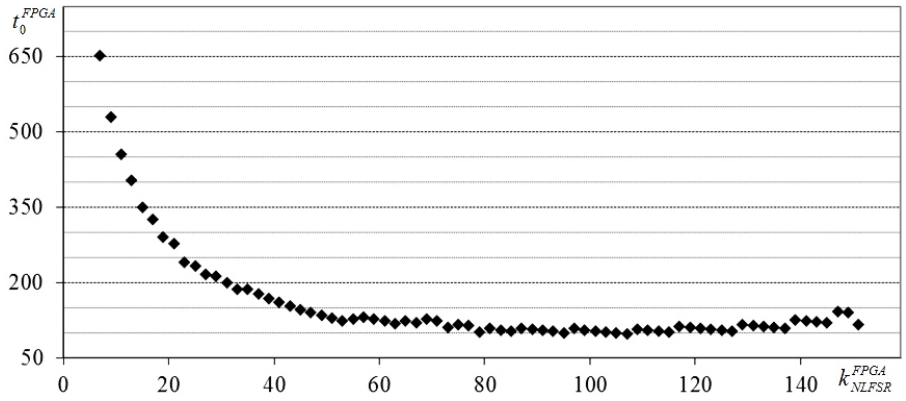


Fig. 5. Dependence of t_0^{FPGA} (in hours) on k_{NLFSR}^{FPGA} for $L = 29$

Table 1
Dependence of time for generation of 1 GB NLFSR on different f^{FPGA}

f^{FPGA}	t
50 MHz	172 s
100 MHz	86 s
200 MHz	43 s
400 MHz	21,5 s
600 MHz	14,3 s
700 MHz	12,3 s
725 MHz	11,8 s

As can be seen from **Table 1**, the use of the FPGA-based hardware components for PRSs generation can improve performance (two and more times) as compared to using only the CPU of a personal computer.

It is worth noting that these results are valid for used FPGA model. Using another FPGA model, it is likely to get higher system efficiency. Thus, [28] shows the results of the efficiency of hardware implementations of ciphers of GRACE-S family for various FPGA families. Unfortunately, investigated families didn't contain Cyclone IV, but tested FPGAs of Stratix IV and Stratix V family showed more than two times greater efficiency than FRGAs of Cyclone II or Cyclone V family.

6. Search results for M-NLFSRs

M-NLFSRs search with a size of L =26, 27, 28 and 29 with the above described method was carried out. Obtained polynomials, as well as time expenditures, are shown below.

$L = 26$:

$$x^{26} + x^{24} + x^{23} + x^{20} + x^{18} + x^{14} + x^{12} + x^6 + x^5 + x^4 + x^{23} \cdot x^{25} + x^{24} \cdot x^{25}$$

$$x^{26} + x^{25} + x^{22} + x^{21} + x^{19} + x^{17} + x^{16} + x^{15} + x^{13} + x^{12} + x^6 + x^5 + x^4 + x^{23} \cdot x^{25}$$

$$x^{26} + x^{24} + x^{22} + x^{21} + x^{20} + x^{19} + x^{17} + x^{16} + x^{15} + x^{14} + x^{12} +$$

$$+x^{11} + x^8 + x^7 + x^4 + x^{23} \cdot x^{24} + x^{23} \cdot x^{25} + x^{24} \cdot x^{25}$$

$$x^{26} + x^{22} + x^{21} + x^{20} + x^{19} + x^{17} + x^{16} + x^{15} + x^{14} + x^{12} + x^{11} + x^8 + x^7 + x^4 + x^{23} \cdot x^{24} + x^{24} \cdot x^{25}.$$

Search time – 43 hours 48 minutes.

L = 27:

$$x^{27}+x^{26}+x^{25}+x^{24}+x^{19}+x^{16}+x^{15}+x^{11}+x^{8}+x^{6}+x^{5}+x^{3}+x^{24}\cdot x^{25}+x^{24}\cdot x^{26}$$

$$x^{27}+x^{26}+x^{19}+x^{16}+x^{15}+x^{13}+x^{12}+x^{11}+x^{10}+x^{9}+$$
$$+x^{8}+x^{7}+x^{6}+x^{5}+x^{4}+x^{3}+x^{24}\cdot x^{26}+x^{25}\cdot x^{26}$$

$$x^{27}+x^{24}+x^{22}+x^{20}+x^{19}+x^{17}+x^{16}+x^{13}+x^{10}+x^{8}+x^{7}+x^{6}+x^{4}+x^{25}\cdot x^{26}$$

$$x^{27}+x^{24}+x^{23}+x^{17}+x^{14}+x^{9}+x^{8}+x^{7}+x^{5}+x^{4}+x^{3}+x^{25}\cdot x^{26}$$

Search time – 33 hours 04 minutes.

L = 28:

$$x^{28}+x^{27}+x^{25}+x^{24}+x^{22}+x^{20}+x^{19}+x^{16}+x^{14}+x^{13}+x^{10}+x^{8}+x^{5}+x^{25}\cdot x^{27}$$

$$x^{28}+x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{22}+x^{21}+x^{20}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+$$
$$+x^{7}+x^{6}+x^{25}\cdot x^{26}+x^{25}\cdot x^{27}$$

$$x^{28}+x^{25}+x^{20}+x^{19}+x^{17}+x^{16}+x^{13}+x^{11}+x^{10}+x^{9}+x^{8}+x^{7}+x^{6}+x^{3}+x^{25}\cdot x^{27}+x^{26}\cdot x^{27}$$

$$x^{28}+x^{24}+x^{23}+x^{20}+x^{18}+x^{17}+x^{16}+x^{14}+x^{13}+x^{12}+x^{5}+x^{25}\cdot x^{27}$$

$$x^{28}+x^{27}+x^{26}+x^{25}+x^{15}+x^{14}+x^{11}+x^{10}+x^{25}\cdot x^{26}+x^{25}\cdot x^{27}$$

$$x^{28}+x^{25}+x^{24}+x^{21}+x^{20}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{10}+x^{9}+x^{5}+x^{3}+$$
$$+x^{25}\cdot x^{27}+x^{26}\cdot x^{27}.$$

Search time – 73 hours 25 minutes.

L = 29:

$$x^{29}+x^{28}+x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{20}+x^{16}+x^{12}+x^{11}+x^{6}+x^{4}+x^{26}\cdot x^{27}$$

$$x^{29}+x^{28}+x^{27}+x^{26}+x^{25}+x^{21}+x^{19}+x^{15}+x^{10}+x^{9}+x^{8}+x^{6}+x^{4}+x^{26}\cdot x^{28}$$

Search time – 133 hours 26 minutes.

The use of hardware component will significantly reduce M-NLFSRs search time. As an example, for software search at L=25 was spent 18 days, and by using hardware components for L=26 – less than two days. Estimated time for which would have made a similar search for L=26, using a software method, would constitute more than 3 months. Taking into account the modifications, introduced in the search algorithm for L=27, 28 and 29, we can expect another three-time increase in performance for L=26, while using hardware components.

Thus, using this method, we get a general performance increase for search of M-NLFSRs generating M-sequences by more than 150 times as compared with the method using only PC CPU resources. The use of more expensive and productive FPGAs will further reduce the time to search of M-NLFSRs.

7. Conclusions

The method for M-NLFSRs search using hardware-software system is given. Based on the method, as an example, the result of the search of non-linear 26, 27, 28 and 29 degrees polynomials for generation of M-sequences is given.

Use of FPGA-based hardware component can significantly improve the performance of complex for search of M-NLFSR. The time for M-NLFSR search is increased by more than 150 times as compared to using only the computing power of PC.

FPGA-based NLFSR performance capabilities are given. Recommendations on optimizing the performance of the complex to search of M-NLFSR are given.

A performance comparison of NLFSR generators performed by using the software and hardware implementation is carried out. It is shown that the hardware implementation of single FPGA-based NLFSR isn't inferior to the performance of its program analogue and in the case of a generator with several parallel NLFSRs can significantly exceed its.

Stable performance of FPGA-based NLFSR is implemented with a clock frequency (generation speed in NLFSR), equal to 725 MHz.

On the basis of the results it can be developed stream ciphers with enhanced cryptographic strength. The application of obtained non-linear polynomial allows to obtain stream ciphers, which are analogues of modern developments in this field, and the use of the described search method allows to find higher degree M-NLFSR.

References

[1] Horbenko, Yu. (2015). Pobuduvannia ta analiz system, protokoliv i zasobiv kryptohrafichnoho zakhystu informatsii. Chastyna 1: Metody pobuduvannia ta analizu, standartyzatsiia ta zastosuvannia kryptohrafichnykh system. Kharkiv: Fort, 960.

[2] Biham, E., Dunkelman, O. (2000). Cryptanalysis of the A5/1 GSM Stream Cipher. Progress in Cryptology – INDOCRYPT 2000, 43–51. doi: 10.1007/3-540-44495-5_5

[3] Shaked, Y., Wool, A. (2006). Cryptanalysis of the Bluetooth E 0 Cipher Using OBDD's. Information Security, 187–202. doi: 10.1007/11836810_14

[4] Schneier, B. (2000). A self-study course in block-cipher cryptanalysis. Cryptologia, 24 (1), 18–33. doi:10.1080/0161-110091888754

[5] Gammel, B. M., Gottfert, R., Kniffler, O. (2007). Achterbahn-128/80: Design and analysis. Workshop Record of The State of the Art of Stream Ciphers – SASC 2007, 152–165.

[6] Chen, K., Henricksen, M., Millan, W., Fuller, J., Simpson, L., Dawson, E., Lee, H., Moon, S. (2005). Dragon: A Fast Word Based Stream Cipher. Information Security and Cryptology – ICISC 2004, 33–50. doi:10.1007/11496618_5

[7] Hell, M., Johansson, T., Meier, W. (2007). Grain: a stream cipher for constrained environments. International Journal of Wireless and Mobile Computing, 2 (1), 86. doi: 10.1504/ijwmc.2007.013798

[8] Canniere, C., Preneel, B. (2006). TRIVIUM specifications. eSTREAM, ECRYPT Stream Cipher Project. Available at: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.9030&rank=1

[9] Gittins, B., Landman, H., O'Neil, S., Kelson, R. (2005). A presentation on VEST hardware performance, chip area measurements, power consumption estimates and benchmarking in relation to the aes, sha-256 and sha-512. IACR Cryptology ePrint Archive, 415.

[10] Canteaut, A. (2006). Open Problems Related to Algebraic Attacks on Stream Ciphers. Lecture Notes in Computer Science, 3969, 120–134. doi: 10.1007/11779360_10

[11] Dubrova, E., Teslenko, M., Tenhunen, H. (2008). On Analysis and Synthesis of (n, k)-Non-Linear Feedback Shift Registers. 2008 Design, Automation and Test in Europe, 133–137. doi: 10.1109/date.2008.4484856

[12] Kuznetsov, O., Svatovskyi, I. (2016). Analiz ta porivnialni doslidzhennia symetrychnykh kryptohrafichnykh peretvoren na postkvantovyi period. Kharkiv: KhNU im. V. N. Karazina, 119.

[13] Dubrova, E. (2013). A Scalable Method for Constructing Galois NLFSRs With Period $2^n - 1$ Using Cross-Join Pairs. IEEE Transactions on Information Theory, 59 (1), 703–709. doi: 10.1109/tit.2012.2214204

[14] Janicka-Lipska, I., Stokłosa, J. (2004). Boolean feedback functions for full-length nonlinear shift registers. Journal of Telecommunications and Informatioin Technology, 4, 28–30.

[15] Golomb, S. W. (1982). Shift Register Sequences. Aegean Park Press, 119.

[16] Dubrova, E. (2014). Generation of full cycles by a composition of NLFSRs. Designs, Codes and Cryptography, 73 (2), 469–486. doi: 10.1007/s10623-014-9947-3

[17] Dubrova, E. (2012). A list of maximum – period NLFSRs. IACR Cryptology ePrint Archive, 166.

[18] Schneier, B. (1995). Applied cryptography (2nd ed.): protocols, algorithms, and source code in C. New York: John Wiley & Sons, Inc., 758.

[19] Kulikova, A. S., Lysenko, I. V. (2012). Realization of diverse stream data encryption with keyless hash functions on the basis of programmable logic. Information processing systems, 7 (105), 22–26.

[20] Kulanov, V., Kharchenko, V., Perepelitsyn, A. (2010). Parameterized IP Infrastructures for fault-tolerant FPGA-based systems: Development, assessment, case-study. 2010 East-West Design & Test Symposium (EWDTS), 452–455. doi: 10.1109/ewdts.2010.5742075

[21] Rachwalik, T., Szmidt, J., Wicik, R., Zabłocki, J. (2012). Generation of Nonlinear Feedback Shift Registers with special – purpose hardware. In Communications and Information Systems Conference, 1–4.

[22] Perepelitsyn, A. E. (2016). Usage of parametrizable ip infrastructures for fpga-based fault-tolerant onboard systems development. Electronic and computer systems, 5 (79), 104–112.

[23] Kolesnyk, I. N., Kulanov, V. O., Perepelitsyn, A. E. (2016). Analysis of fpga technologies application as a part of cloud infrastructure. Electronic and computer systems, 6 (80), 130–135.

[24] Perepelitsyn, A., Shulga, D. (2013). FPGA technologies in medical equipment: Electrical impedance tomography. East-West Design & Test Symposium, 1–4. doi: 10.1109/ewdts.2013.6673157

[25] Potii, A. V., Poluyanenko, N. A. (2008). Analyz svojstv reghystrov sdvygha s nelynejnoj obratnoj svjazjju vtorogho porjadka, gheneryrujushhykh posledovateljnostj s maksymaljnim peryodom. Prykladnaja radyoelektronyka, 3, 282–290.

[26] Potii, A., Poluyanenko, N. (2016). The selection of forming polynomials for shift register with nonlinear feedback second order that generates the sequence with maximum period. Computer science and cybersecurity, 2 (2), 22–30. Available at: http://periodicals.karazin.ua/cscs/article/view/6209/5747

[27] Poluyanenko, N., Potii, A. (2016). Sravnenye ob'ema ansamblia M-RSLOS y M-RSNOS, skorosty heneratsyy na ykh osnove, dlia GF(2) y v rasshyrenyiakh polia GF(2^2). Radyotekhnyka, 186 (216), 153–160.

[28] Kliucharev, P. G. (2013). Proyzvodytelnost y effektyvnost apparatnoi realyzatsyy potochnikh shyfrov, osnovannikh na obobshchennikh kletochnikh avtomatakh. Nauka i obrazovanye, 10, 299–314. Available at: http://technomag.bmstu.ru/file/669391.html?__s=1