The search of square m-sequences with maximum period via GPU and CPU

Paweł Augustynowicz¹ and Krzysztof Kanciak²

Abstract—This paper deals with the efficient parallel search of square m-sequences on both modern CPUs and GPUs. The key idea is based on applying particular vector processor instructions with a view to maximizing the advantage of Single Instruction Multiple Data (SIMD) and Single Instruction Multiple Threads (SIMT) execution patterns. The developed implementation was adjusted to testing for the maximum-period of m-sequences of some particular forms. Furthermore, the early abort sieving strategy based on the application of SAT-solvers were presented. With this solution, it is possible to search m-sequences up to degree 32 exhaustively.

I. INTRODUCTION

Feedback Shift Registers (FSR) are used to generate cryptographically applicable binary sequences. They have many proponents due to their simplicity, both software and hardware effectiveness and well-known properties. In particular, stream ciphers designers use them to construct invertible mappings with internal state. The strongly desirable property of stream ciphers is their long period. Therefore, the FSR used in them should also have this feature. Informally, the period of mapping is the length of the most extended cycle in its state transition graph.

In recent years, many cryptographic algorithms such as stream ciphers (for example GRAIN which is NIST standard [9], Trivium [3] or Achterbahn [2]), lightweight block ciphers and sponge-based generators [4, 10] have used NLFSR for providing both security and efficiency. In most cases, NLFSRs have much greater linear-complexity than LFSRs of the same period, which is directly connected with the security of cryptographic algorithms [12].

Computationally efficient methods for construction of cryptographically strong NLFSRs remains unknown. The most critical NLFSR related problem is finding a systematic procedure for constructing NLFSRs with a long confirmed period. Available algorithms either consider some individual cases or apply to low order NLFSRs only [7, 14, 16]. Nikolay Poluyanenko developed the most efficient method. However, it was not sufficient to obtain applicable NLFSR of degree 30 or higher [13]. Moreover, it requires the usage of special-purpose Field-Programmable Gate Arrays (FPGA) hardware, which is not commonly available.

II. BASIC NOTATIONS AND DEFINITIONS

Definition 1: Binary Feedback Shift Register of order $n$ is a mapping $\mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ of the form:

$$f(x_0, x_1, \ldots, x_{n-1}) = (x_1, x_2, \ldots, x_{n-1}, f(x_0, x_1, \ldots, x_{n-1})),$$

where:

- $f$ is a boolean function of $n$ variables;
- $x_{n-1}$ is an output bit.

Depending on the type of feedback function two main types of shift registers are concerned:

If we look at the above-mentioned subject from another point of view, NLFSRs are also known as de Bruijn sequences. In a de Bruijn series of order $n$, all $2^n$ different binary $n$-tuples appear precisely once. A modified de Bruijn sequence is obtained from a proper de Bruijn sequence by removing tuple containing zero elements only.

Another essential sequence type, which statistical and structural properties were examined, are so-called $m$-sequences. Boolean functions that generate the $m$-sequence can be constructed by introducing nonlinear disturbances into linear functions [11]. Unfortunately, complexity of this approach is extremely high for orders greater than 8. As a result in this paper we address the problem of efficient searching for $m$-sequences with a guaranteed full period by exhaustively search for the NLFSR with the following form of feedback function:

$$f(x_0, x_1, \ldots, x_{n-1}) = g(x_0, x_1, \ldots, x_{n-1}) + x_i + x_1 \cdot x_j$$

for which $i \neq j, 1 \leq i, j \leq n - 1$ and $g(x_0, x_1, \ldots, x_{n-1})$ is defined by a primitive polynomial over $\mathbb{F}_2$. Owing to the large number of candidate feedback functions, the search was conducted on GPUs and special strategy of early abort via SAT-solvers’ detection of short cycles were applied.

The aforementioned computational experiment allows obtaining an extensive, complete list of $n$-bit NLFSR ($n < 31$) with a maximum period for the considered form of feedback functions. The previous research in the investigated area has resulted in maximum period NLFSR up to degree 27 [6] on Central Processing Units (CPU) and up to degree 29 on FPGA [13]. We have enumerated all $m$-sequences up to degree 31. Obtained results suggest the dependency between the Hamming weight of feedback functions and the period of NLFSR generated by that function was observed (see Table VII).

¹² Military University of Technology Institute of Cybernetics, gen. Sylwestra Kaliskiego 2, 00-908, Warsaw, Poland.
¹ E-mail: pawel.augustynowicz@wat.edu.pl
² E-mail: krzysztof.kanciak@wat.edu.pl

DOI: 10.36244/ICJ.2019.4.3
The search of square m-sequences with maximum period via GPU and CPU

Fig. 1: A structure of Feedback Shift Register.

- linear if the feedback function is linear;
- nonlinear if the feedback functions has degree equal two or higher.

The period of an FSR is the length of the longest cyclic output sequence it generates.

Definition 2: A de Bruijn sequence of order n is a cyclic sequence of length $2^n$ of elements of $F_2$ in which all different n-tuples appear exactly once.

Definition 3: A modified de Bruijn sequence of order n is a sequence of length $2^n - 1$ obtained from a de Bruijn sequence of order n by removing one zero from the tuple of n consecutive zeros.

From the cryptographic or random number generation perspective, it is strongly desirable that NLFSR of order n should generate a de Bruijn sequence of order n. Furthermore, due to practical reasons, the following conditions should be fulfilled:
- the number of feedback function’s linear and nonlinear terms should remain as small as possible;
- the algebraic degree of feedback function should be the lowest possible;
- the feedback function should be easy to generate.

So-called square m-sequences achieve all the considered restrictions.

Definition 4: A square m-sequence is a bit sequence generated by a shift register with a feedback function with the following form:

\[ f(x_0, x_1, \ldots, x_{n-1}) = \sum_{0 \leq i \leq j \leq n-1} a_{i,j} x_i x_j. \]

Moreover, square m-sequences can be described by a very concise form of the recurrence, which can be formulated as:

\[ \forall k \geq 0 : s_{n+k} = \sum_{0 \leq i \leq j \leq n-1} a_{i,j} s_i s_j, \]

where $s_i$ denotes the i-th position in the sequence s. It is well-known, that square m-sequences can be algorithmically generated by introducing nonlinear disturbances into linear functions, for example by the following form:

\[ f(x_0, x_1, \ldots, x_{n-1}) = g(x_0, x_1, \ldots, x_{n-1}) + x_i + x_i : x_j, \]

where $i \neq j$, 0 \leq i \leq j \leq n - 1, and $g(x_0, x_1, \ldots, x_{n-1})$ is linear functions whose LFSR generates maximum-period sequence. From the theory of de Bruijn sequences [15], it can be concluded that $g(x_0, x_1, \ldots, x_{n-1})$ must be defined by a primitive polynomial in $F_2[x].$

III. MASSIVELY PARALLEL ALGORITHM

Due to overwhelming number of possible feedback functions (see Table I), we have constructed massively parallel algorithm for the search of square m-sequences. It examines the provided functions’ period completeness by enumerating the following states and checking for their uniqueness. In practice, it satisfies to prove that their initial states will be generated after exactly $2^n - 1$ steps.

<table>
<thead>
<tr>
<th>Degree</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>log2(#candidates)</td>
<td>29.05</td>
<td>30.45</td>
<td>30.75</td>
<td>32.79</td>
<td>32.85</td>
</tr>
</tbody>
</table>

TABLE I: The number of square m-sequences candidates to be examined by computational experiment.

For accurate description and outline of the feedback function examining algorithm, consider the subsequent data labels:
- LFSR – bit representation of the linear component of the feedback function;
- NLFSR – bit representation of the nonlinear component of the feedback function;
- N – the order of the shift register;

For example for the primitive polynomial of form $x_0 + x_1 + 1$ and nonlinear part of function with the form $x_1 \cdot x_2$, its bit representation of the linear component has following form in hex: 0x211 whereas the nonlinear one: 0x00c. Its length N is naturally equal 9.

Input: LFSR, NLFSR, N - length of register;

\[ i_{state} = 0x01; \]
\[ \text{for } i = 1, \ldots, 2^n - 1 \text{ do} \]
\[ b_{LFSR} = (\text{popcount}(i_{state} \text{ and LFSR})) \text{ mod 2}; \]
\[ b_{NLFSR} = (\text{popcount}(i_{state} \text{ and NLFSR})) \text{ mod 2}; \]
\[ \text{bit} = b_{LFSR} \text{ xor } b_{NLFSR}; \]
\[ i_{state} = (i_{state} \text{ rot left 1}) \text{ xor bit}; \]
\[ \text{if } i_{state} == 0x01 \text{ then} \]
\[ \text{return false; end} \]
\[ \text{end} \]
\[ \text{if } i_{state} == 0x01 \text{ then} \]
\[ \text{return true; else} \]
\[ \text{return false; end} \]

Algorithm 1: The period examination algorithm of NLFSR’s feedback function.

For the sake of completeness of the specifications considered in the algorithm 1, it should be completed that popcount indicates an operation of returning the number of ones in the given integer and mod – an instruction of a division with the remainder. The algorithm 1 considered above can be implemented on all kinds of Graphical Processing Units (GPU) resulting in efficiency advantage over modern CPUs. It is strongly recommended to take advantage of SIMD (Single Instruction Multiple Data), a parallel execution model of modern CPUs, to achieve maximum possible efficiency. With the
V. EXAMPLE APPLICATION OF SAT-SOLVER FILTERING RESULTS

Short cycle existence of polynomial can be checked during filtration phase in seconds. For instance polynomial \( x_0 + x_3 + x_{31} + x_1 + x_1 x_2 \) is being checked for consecutive cycle lengths:

1) cycle lengths equal from 2 to 5 — gives us UNSAT result in milliseconds which means that there is no 2,3,4,5-step cycle
2) cycle length equal to 6 — gives us SAT result in less than 3 seconds and bits assignment is equal to 10011010011010010100110101

Next polynomial \( x_0 + x_3 + x_{31} + x_1 + x_1 x_2 \) has 2-step cycle and SAT solver returned the assignment 1010101101101010101011010101 in less than 2 seconds. The exact distribution of cycle lengths remains unknown. Nevertheless, the vast majority of polynomials has cycles shorter than 32-steps and can be easily eliminated in seconds without using extensive computing power. It is estimated that the rejection ratio is approximately about 70% of rejected polynomials for NLFSR degree 31 and checking time less than 60 seconds. Further extension of checking time or the length of the short cycles probably will not result in a performance gain.

VI. PERFORMANCE EVALUATION

A fair comparison of the efficiency of various computing platforms is a very troublesome task due to their completely diverse characteristic. Therefore we simplify the comparison by analyzing only the most important efficiency indicators such as:

- the time of one \( n \)-bit full-cycle NLFSR enumeration \( T_{cycle} \),
- the number of simultaneously tested NLFSRs,
- the estimated time of enumeration of 1 GB pack of \( n \)-bit NLFSRs excluding memory transactions \( T_{GB} \).

The time of one \( n \)-bit full-cycle NLFSR enumeration \( T_{cycle} \) is based on the measurement of search time of \( 10^5 \) possible NLFSRs for CPU and GPU. The computations were conducted on following computing platforms:

- Intel Core i7 6700K, 4.0 GHz CPU, MSI GeForce GTX 1080 8GB GDDR5 with 32 GB of RAM,
- 2 x Xeon 2699 v3, Tesla K80 with 32 GB of RAM,
- Xeon2699 v4, Tesla P100 with 32 GB of RAM.

As it can be seen from the Tables III and IV GPUs are very efficient for small NLFSRs, but they tend to lose efficiency with the growth of NLFSR order. Consequently, as it can be concluded from Figure 2 and Tables III and IV for degrees higher than 31, it is inefficient to take advantage of GPU computing platform.

<table>
<thead>
<tr>
<th>n</th>
<th>( T_{cycle} ) (ms)</th>
<th>( T_{GB} ) (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>44</td>
<td>3584</td>
</tr>
<tr>
<td>27</td>
<td>2496</td>
<td>2550</td>
</tr>
</tbody>
</table>

TABLE II: The number of parallel computing units for different platforms.

application of SIMD vector instructions, simple calculations, such as xor, bit shifts or counting ones in a word can be performed even on eight words by one thread. For example the concurrent rotation of 8 32-bits words can be realized by the Intel processor intrinsic _mm256_mulll_epi32, whereas xor can be computed via _mm256_xor_si256.

As far as GPU implementation and its SIMT (Single Instruction Multiple Threads) parallel model are concerned, the most significant factor is to determine the number of ones in the given integer effectively. It can be realized by generating ptx code or exploiting popcntq instruction on NVIDIA graphics cards and their CUDA (Compute Unified Device Architecture) development tools. Nevertheless, it is impossible to achieve similar performance on OpenCL (Open Computing Language) implementations. Moreover we have observed that usign one thread per one i_state strategy is obviously optimal. Unluckily performing conditional instructions on GPU is exceptionally inefficient. Consequently, the inner if condition should be omitted in these kind of implementations. As a result the developed algorithm posses no early abort strategy on GPU platform, which would allow to efficient filtration of short-period NLFSRs. However, it can be realized on CPU by the usage of SAT-solvers.

IV. APPLICATION OF SAT SOLVERS

For polynomials up to degree 31, GPU exhaustive cycle verification method works well since we can examine thousands of registers at once. For polynomials of higher degrees, we found FPGA and CPU implementations much more convenient. Furthermore, before full-cycle FPGA or GPU exhaustive verification, we strongly recommend to check for short cycle existence by solving a Boolean satisfiability problem. It can be realized automatically with the help of some open source tools. Firstly, it is required to translate our for example C programming language implementation to And-Inverter Graphs (AIG), which are intermediate states only of Algebraic Normal Form generation (ANF). This step can be done by usage of ABC: System for Sequential Logic Synthesis and Formal Verification and SAW The Software Analysis Workbench. From ANF, there is the well-known path to Conjunctive Normal Form (CNF), which is finally inputed to SAT-solver (Cadical and Lingeling work well and much better than other more popular SAT-solvers in this case [1]). We do not know NLFSR cycles structure, but the majority of polynomials of degree higher than 31 can be quickly eliminated by SAT searching of cycles shorter than 16 states. FPGA or CPU based full cycle verification is being performed in case of UNSAT (no model found) result of prior SAT short cycle check. SAT-based pre-phase works entirely on the CPU, which gives us tremendous resources utilization rate of the entire computing system. The proposed approach is inspired by the work of Elena Dubrova and Maxim Teslenko [8], [5]. It is worth mentioning that the first application of SAT solvers to NLFSR was motivated by the search of short cycles in stream ciphers [8].
TABLE III: The time of one n-bit full-cycle NLFSR enumeration $T_{cycle}$ for different GPU.

<table>
<thead>
<tr>
<th>n</th>
<th>Tesla P100</th>
<th>Tesla K80</th>
<th>1080GTX</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>0.0012</td>
<td>0.0028</td>
<td>0.0010</td>
</tr>
<tr>
<td>24</td>
<td>0.0029</td>
<td>0.0074</td>
<td>0.0022</td>
</tr>
<tr>
<td>25</td>
<td>0.0068</td>
<td>0.0117</td>
<td>0.0051</td>
</tr>
<tr>
<td>26</td>
<td>0.0149</td>
<td>0.0242</td>
<td>0.0102</td>
</tr>
<tr>
<td>27</td>
<td>0.0286</td>
<td>0.0519</td>
<td>0.0207</td>
</tr>
<tr>
<td>28</td>
<td>0.0551</td>
<td>0.1399</td>
<td>0.0410</td>
</tr>
<tr>
<td>29</td>
<td>0.1087</td>
<td>0.1870</td>
<td>0.0813</td>
</tr>
<tr>
<td>30</td>
<td>0.2244</td>
<td>—</td>
<td>0.1644</td>
</tr>
<tr>
<td>31</td>
<td>0.9736</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

TABLE IV: The estimated time of enumeration of 1 GB pack of n-bit NLFSRs for GPUs.

<table>
<thead>
<tr>
<th>n</th>
<th>17+6/00</th>
<th>Xeon2699v3</th>
<th>Xeon2699v4</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0002</td>
</tr>
<tr>
<td>24</td>
<td>0.0002</td>
<td>0.0002</td>
<td>0.0003</td>
</tr>
<tr>
<td>25</td>
<td>0.0004</td>
<td>0.0004</td>
<td>0.0007</td>
</tr>
<tr>
<td>26</td>
<td>0.0008</td>
<td>0.0008</td>
<td>0.0014</td>
</tr>
<tr>
<td>27</td>
<td>0.0016</td>
<td>0.0017</td>
<td>0.0027</td>
</tr>
<tr>
<td>28</td>
<td>0.0033</td>
<td>0.0033</td>
<td>0.0055</td>
</tr>
<tr>
<td>29</td>
<td>0.0065</td>
<td>0.0066</td>
<td>0.0110</td>
</tr>
<tr>
<td>30</td>
<td>0.0130</td>
<td>0.0132</td>
<td>0.0220</td>
</tr>
<tr>
<td>31</td>
<td>0.0261</td>
<td>0.0263</td>
<td>0.0439</td>
</tr>
<tr>
<td>32</td>
<td>0.0546</td>
<td>0.0527</td>
<td>0.0877</td>
</tr>
</tbody>
</table>

TABLE V: The time of one n-bit full-cycle NLFSR enumeration $T_{cycle}$ for different CPU.

<table>
<thead>
<tr>
<th>n</th>
<th>17+6/00</th>
<th>2×Xeon2699v3</th>
<th>Xeon2699v4</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>1140</td>
<td>380</td>
<td>363</td>
</tr>
<tr>
<td>24</td>
<td>2458</td>
<td>667</td>
<td>695</td>
</tr>
<tr>
<td>25</td>
<td>4456</td>
<td>1165</td>
<td>1335</td>
</tr>
<tr>
<td>26</td>
<td>8570</td>
<td>2073</td>
<td>2566</td>
</tr>
<tr>
<td>27</td>
<td>16263</td>
<td>3884</td>
<td>4943</td>
</tr>
<tr>
<td>28</td>
<td>31364</td>
<td>7230</td>
<td>9575</td>
</tr>
<tr>
<td>29</td>
<td>60564</td>
<td>13760</td>
<td>18490</td>
</tr>
<tr>
<td>30</td>
<td>116873</td>
<td>26409</td>
<td>35707</td>
</tr>
<tr>
<td>31</td>
<td>225571</td>
<td>50972</td>
<td>69188</td>
</tr>
<tr>
<td>32</td>
<td>437658</td>
<td>98532</td>
<td>133976</td>
</tr>
</tbody>
</table>

TABLE VI: The estimated time of enumeration of 1 GB pack of n-bit NLFSRs for CPUs.

The algorithm 1 scales well on CPUs (see Tables V and VI), which means that we can search for NLFSR’s of degrees higher than 32 without any performance drop (this is a currently ongoing process).

GPU devices have a significant advantage in NLFSR’s searching up to degree 30. For higher degrees, we can see the performance drop. It is caused by switching the arithmetic of integers from 32-bit word length to 64-bit one and cannot be avoided. One of the possible solutions to the problem mentioned above is applying a bit-slicing method to the algorithm implementation, although it was not the case in our application.

VII. Most significant results

With the usage of algorithm 1 all the NLFSR with feedback function of considered form up to degree 31 were enumerated. Examples of aforementioned feedback functions are given below:

- $n = 30$:
  
  $x_0 + x_1 + x_3 + x_5 + x_7 + x_8 + x_9 + x_{12} + x_{15} + x_{16} + x_{18} + x_{21} + x_{23} + x_{27} + x_{29} + x_{41} \cdot x_{26}$;
  
  $x_0 + x_1 + x_3 + x_7 + x_9 + x_{15} + x_{16} + x_{18} + x_{20} + x_{22} + x_{23} + x_{25} + x_{29} + x_{41} \cdot x_{8}$;
  
  $x_0 + x_1 + x_3 + x_7 + x_9 + x_{11} + x_{16} + x_{18} + x_{19} + x_{22} + x_{24} + x_{27} + x_{5} \cdot x_{28}$;

- $n = 29$:
  
  $x_0 + x_6 + x_7 + x_{13} + x_{21} + x_{22} + x_{23} + x_{24} + x_{25} + x_{27} + x_{28} + x_{1} \cdot x_{17}$;
  
  $x_0 + x_2 + x_5 + x_7 + x_{10} + x_{14} + x_{17} + x_{22} + x_{25} + x_{26} + 1 \cdot x_{17} + x_{24}$;
  
  $x_0 + x_1 + x_3 + x_5 + x_7 + x_9 + x_{11} + x_{14} + x_{16} + x_{18} + x_{19} + x_{20} + x_{22} + x_{26} + x_{28} + x_{5} \cdot x_{31}$;

- $n = 28$:
  
  $x_0 + x_2 + x_5 + x_7 + x_9 + x_{10} + x_{13} + x_{14} + x_{16} + x_{17} + x_{21} + x_{22} + x_{23} + x_{25} + x_{27} + x_{41} \cdot x_{8}$;
  
  $x_0 + x_1 + x_3 + x_7 + x_8 + x_{10} + x_{12} + x_{13} + x_{14} + x_{15} + x_{22} + x_{23} + x_{24} + x_{26} + x_{28} + x_{5} \cdot x_{31}$;

- $n = 27$:
  
  $x_0 + x_2 + x_4 + x_5 + x_6 + x_{10} + x_{14} + x_{16} + x_{17} + x_{23} + x_{24} + x_{25} + x_{26} + 1 \cdot x_{5}$;
  
  $x_0 + x_4 + x_6 + x_8 + x_{10} + x_{12} + x_{13} + x_{14} + x_{15} + x_{18} + x_{22} + x_{23} + x_{24} + x_{25} + x_{31}$;
  
  $x_0 + x_4 + x_6 + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{22} + x_{23} + x_{25} + x_{31}$;

- $n = 26$:
  
  $x_0 + x_3 + x_4 + x_6 + x_8 + x_9 + x_{12} + x_{15} + x_{17} + x_{19} + x_{21} + x_{22} + x_{23} + x_{25} + x_{41} \cdot x_{15}$;
  
  $x_0 + x_2 + x_3 + x_5 + x_7 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{22} + x_{24} + x_{25} + 1 \cdot x_{5}$;
  
  $x_0 + x_2 + x_3 + x_5 + x_7 + x_9 + x_{12} + x_{14} + x_{16} + x_{18} + x_{19} + x_{20} + x_{24} + x_{25} + x_{26} + x_{28}$;

Moreover, it has been observed that the feedback functions with around half of non-zero coefficients are more likely to occur than the others. What is more, the most desirable polynomials with the low number of non-zero coefficients are extremely rare in practice or even impossible to find for high degrees. This observation has been illustrated in Table VII. Taking all of the above into consideration, it can be concluded that the construction of ideal cryptographic NLFSR with maximum-period and very concise form is still an open problem.

VIII. Conclusion and Future Work

In this article, the problem of construction of applicable NLFSR was addressed. The exhaustive search of particular
due to numerous possible of algorithm 1 via bit-slicing methodology. Moreover, it is up to degree $m$ of terms decreases with the degree of the NLFSR.

of short period NLFSR and in consequence, reduce the number of square $m$-sequences with lesser number of terms decreases with the degree of the NLFSR.

the GPU implementation should be altered for the GPU and CPU by adjusting the enumeration algorithm to SIMD and SIMT parallel execution models;

square $m$-sequences have certain cryptographic and practical properties, that are desirable, especially very concise form and lowest possible algebraic degree;

the number of square $m$-sequences with lesser number of terms decreases with the degree of the NLFSR.

the GPU implementation should be altered for the 31-degree NLFSR due to the efficiency decrease. One possible solution is to apply the bit-slicing methodology in order to avoid a costly switch to 64-bit arithmetic.

SAT solvers sieving can contribute to the fast rejection of short period NLFSR and in consequence, reduce the reasonable time of computational experiments.

It is planned to continue the search of square $m$-sequences up to degree 32 on FPGA platform and GPU after modification of algorithm 1 via bit-slicing methodology. Moreover, it is necessary to improve the sieving ratio of short period NLFSR due to numerous possible $m$-sequences of degrees 31 and 32 (approximately $2^{34.9}$ and $2^{35}$ respectively).

### TABLE VII: The number of feedback functions with certain number of non-zero coefficients.

<table>
<thead>
<tr>
<th>deg</th>
<th>$&lt;7$</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>15</th>
<th>17</th>
<th>19</th>
<th>21</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>0</td>
<td>24</td>
<td>26</td>
<td>32</td>
<td>48</td>
<td>22</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>0</td>
<td>26</td>
<td>28</td>
<td>58</td>
<td>36</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>4</td>
<td>10</td>
<td>26</td>
<td>42</td>
<td>32</td>
<td>12</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>29</td>
<td>4</td>
<td>24</td>
<td>49</td>
<td>72</td>
<td>32</td>
<td>24</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>17</td>
<td>37</td>
<td>32</td>
<td>27</td>
<td>13</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2: Comparison of the estimated enumeration time of 1 GB pack of $n$-bit NLFSRs.

### REFERENCES


The search of square m-sequences with maximum period via GPU and CPU


Pawel Augustynowicz is a PhD student at Military University of Technology in Warsaw, Poland. His research interests are related to efficient computation, parallel computing and cryptography. Nowadays he works mostly with construction and search methods of irreducible polynomials over finite fields of prime characteristic.

Krzysztof Kanciak is an Asistant Professor at Military University of Technology in Warsaw, Poland. His research interests are related to modern cryptography, cryptographic protocols and formal verification methods.

IEEE Conference Record:
#49548

2020

43rd International Conference on
Telecommunications and Signal Processing

TSP

Milan, Italy

July 7-9, 2020

SUBMIT SPECIAL SESSION
AND WORKSHOPS:
February 15, 2020
FULL PAPER SUBMISSIONS:
February 15, 2020

2020

IEEE Conference Record:
#49548

2020

43rd International Conference on
Telecommunications and Signal Processing

TSP

Milan, Italy

July 7-9, 2020

SUBMIT SPECIAL SESSION
AND WORKSHOPS:
February 15, 2020
FULL PAPER SUBMISSIONS:
February 15, 2020

2020

IEEE Conference Record:
#49548

2020

43rd International Conference on
Telecommunications and Signal Processing

TSP

Milan, Italy

July 7-9, 2020

SUBMIT SPECIAL SESSION
AND WORKSHOPS:
February 15, 2020
FULL PAPER SUBMISSIONS:
February 15, 2020

2020

IEEE Conference Record:
#49548

2020

43rd International Conference on
Telecommunications and Signal Processing

TSP

Milan, Italy

July 7-9, 2020

SUBMIT SPECIAL SESSION
AND WORKSHOPS:
February 15, 2020
FULL PAPER SUBMISSIONS:
February 15, 2020

22

DECEMBER 2019 • VOLUME XI • NUMBER 4