Randomness analysis for multiple-recursive matrix generator

Subhrajyoti Deb[1], Bubu Bhuyan[1], and Sartaj Ul Hasan[2]

[1] Department of Information Technology, North-Eastern Hill University, Shillong, Meghalaya 793 022, India
{b.bhuyan, subhrajyotideb1}@gmail.com
[2] Department of Mathematics, Indian Institute of Technology Jammu, Jagti, Jammu 181 221, India
sartaj.hasan@iitjammu.ac.in

Abstract. Randomness testing of binary sequences generated by any keystream generator is of paramount importance to both designer and attacker. Here we consider a word-oriented keystream generator known as multiple-recursive matrix generator, which was introduced by Niederreiter (1993). Using NIST statistical test suite as well as DieHarder statistical package, we analyze randomness properties of binary sequences generated by multiple-recursive matrix generator and show that these sequences are not really adequate for any cryptographic applications. We also propose nonlinearly filtered multiple-recursive matrix generator based a special nonlinear function and establish that sequences generated by such a nonlinearly filtered multiple-recursive matrix generator provide good randomness results. Moreover, we compare our randomness test results with some of the recent lightweight stream ciphers like Lizard, Fruit, and Plantlet.

Keywords: Linear feedback shift register; stream cipher; randomness; word-oriented feedback shift register; MRMG; NIST STS; DieHarder.

1 Introduction

Linear feedback shift register (LFSR) based stream cipher is an effective security solution for various lightweight applications. Most of classical LFSR-based stream ciphers are bit-oriented and can be easily implemented in hardware. However, several emerging applications are demanding for high speed link encryption or an efficient software encryption for a large volume of data handling. In that case, bit-oriented ciphers do not provide satisfactory performance. Moreover, when we consider large amount of data, security is certainly one of the major concerns. This issue may be addressed if one uses a word-oriented feedback shift register (FSR). In fact, this idea was initially mooted by B. Preneel in [10] and he asked for designing a word-oriented FSR that uses word operations available in modern processors and the techniques parallelism. To one's surprise, even before the Preneel posed this problem, a word-oriented FSR was already there in the literature in the form of Niederreiter's Multiple-Recursive Matrix Method [9] for generating pseudo-random sequences, which we shall refer to as "Multiple-Recursive Matrix Generator" here after. In order to improve efficiency problem in MRMG, Zeng, Han and He proposed the notion of σ-LFSR [14]. In σ-LFSR design, they have restricted themselves to a special set of matrices that are compatible with word-operations. It may be noted that the notion of σ-LFSR is essentially equivalent to MRMG. Few years after the Niederreiter's MRMG, Tsaban and Vishne [12] have also proposed the idea of Transformation Shift Register (TSR) for producing pseudo-random vectors that presents yet another answer to Preneel's problem. Indeed, TSR turns out to be a special case of MRMG. However, due to the inherent linearity structure available in both MRMG and TSR, Hasan, Panario and Wang argued in [7] that instead of using them directly for any crypto application, their nonlinear counter parts should be used.

In this paper, we focus on three special kind of σ-LFSRs, namely σ-LFSR I, σ-LFSR II, and σ-LFSR III (see HHZ-1 and HHZ-2 in [14]) proposed by Zeng et al. Following the nomenclature coined by Niederreiter, it is natural to call σ-LFSR as Multiple-Recursive Matrix Generator (MRMG). Henceforth, throughout the this paper, σ-LFSR I, σ-LFSR II, and σ-LFSR III shall be referred to as MRMG I, MRMG II, and MRMG III, respectively. The sequences generated by MRMGs are tested by using NIST statistical test suite as well as DieHarder statistical package. We have observed that some of the randomness properties are not satisfied by binary sequences generated by MRMGs. In order to improve randomness results on

MRMG, it is important to employ some reasonable nonlinear function on the contents of MRMG as advocated in [7].

We introduce a new kind of framework called nonlinearly filtered MRMG, which we shall denote by NMRMG, based on a special nonlinear function. This special function is a 8-variable nonlinear function and is employed on the contents of the three kinds of MRMG (I, II, III). To the best of our knowledge, this particular type of nonlinear framework has not been yet considered in the crypto literature. The detailed randomness test analysis shows that our proposed framework provide cryptographically secure bitstream as compared to existing MRMG. In this paper, we present output data analysis, Berlekamp Massey approach, and autocorrelation of the output bitstream which establishes the effectiveness of the proposed framework. Moreover we also compare randomness test result with some of recent lightweight ciphers keystream like Lizard, Fruit, and Plantlet.

Before proceeding further, let us describe the organization of this paper. Section 2, presents basic definition of word oriented MRMG. Section 3, provides all the experimental details of the proposed framework. Section 4, provides Statistical randomness test. Section 5 provides in-depth security analysis and randomness comparison. Finally, we draw our conclusions in section 6.

2 Preliminaries

We recall some definitions and results described in [7] concerning MRMGs. As we know, in the majority of practical applications, one generally uses finite fields with characteristic 2. Henceforth, we shall restrict ourselves to fields with characteristic 2 and their extensions. We shall denote, as usual, by $\mathbb{F}_2$ the finite field with 2 elements, by $\mathbb{F}_{2^m}$ the extension field of $\mathbb{F}_2$ of degree m and by $\mathbb{F}_2[X]$ the ring of polynomials in one variable X with coefficients in $\mathbb{F}_2$.

Given any ring R and any positive integer d, let $M_d(R)$ denote the set of all $d\times d$ matrices with entries in R. Throughout this paper, we fix positive integers m and n, and a vector space basis $\{\alpha_0, \dots, \alpha_{m-1}\}$ of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$. Given any $s \in \mathbb{F}_{2^m}$, there are unique $s_0, \dots, s_{m-1} \in \mathbb{F}_2$ such that $s = s_0\alpha_0 + \cdots + s_{m-1}\alpha_{m-1}$, and we shall denote the corresponding co-ordinate vector $(s_0, \dots, s_{m-1})$ of s by $\mathbf{s}$. Evidently, the association $s \longmapsto \mathbf{s}$ gives a vector space isomorphism of $\mathbb{F}_{2^m}$ onto $\mathbb{F}_2^m$. Elements of $\mathbb{F}_2^m$ may be thought of as row vectors and so $\mathbf{s}C$ is a well-defined element of $\mathbb{F}_2^m$ for any $\mathbf{s} \in \mathbb{F}_2^m$ and $C \in M_m(\mathbb{F}_2)$.

Definition 1. *Let $C_0, C_1, \dots, C_{n-1} \in M_m(\mathbb{F}_2)$. Given any n-tuple $(\mathbf{s}_0, \dots, \mathbf{s}_{n-1})$ of elements of $\mathbb{F}_{2^m}$, let $(\mathbf{s}_i)_{i=0}^{\infty}$ denote the infinite sequence of elements of $\mathbb{F}_{2^m}$ determined by the following linear recurrence relation:*

$$\mathbf{s}_{i+n} = C_0\mathbf{s}_i + C_1\mathbf{s}_{i+1} + \cdots + C_{n-1}\mathbf{s}_{i+n-1} \quad i = 0, 1, \dots. \tag{1}$$

The system (1) *is called a* multiple-recursive matrix generator (MRMG) *of order n over $\mathbb{F}_{2^m}$, while the sequence $(\mathbf{s}_i)_{i=0}^{\infty}$ is referred to as the* sequence generated by the MRMG (1). *The n-tuple $(\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_{n-1})$ is the* initial state *of the MRMG* (1) *and the polynomial $I_mX^n - C_{n-1}X^{n-1} - \cdots - C_1X - C_0$ with matrix coefficients is the* matrix polynomial *of the MRMG* (1). *The sequence $(\mathbf{s}_i)_{i=0}^{\infty}$ is* ultimately periodic *if there are integers r, n_0 with $r \geq 1$ and $n_0 \geq 0$ such that $\mathbf{s}_{j+r} = \mathbf{s}_j$ for all $j \geq n_0$. The least positive integer r with this property is the* period *of $(\mathbf{s}_i)_{i=0}^{\infty}$ and the corresponding least nonnegative integer n_0 is the* preperiod *of $(\mathbf{s}_i)_{i=0}^{\infty}$. The sequence $(\mathbf{s}_i)_{i=0}^{\infty}$ is* periodic *if its preperiod is* 0.

The following result gives some basic facts about MRMG.

Proposition 1. *[7] For the sequence $(\mathbf{s}_i)_{i=0}^{\infty}$ generated by the MRMG* (1) *of order n over $\mathbb{F}_{2^m}$, we have*

(i) *$(\mathbf{s}_i)_{i=0}^{\infty}$ is ultimately periodic, and its period is no more than $2^{mn} - 1$;*

(ii) *if C_0 is nonsingular, then $(\mathbf{s}_i)_{i=0}^{\infty}$ is periodic; conversely, if $(\mathbf{s}_i)_{i=0}^{\infty}$ is periodic whenever the initial state is of the form $(b, 0, \dots, 0)$, where $b \in \mathbb{F}_{2^m}$ with $b \neq 0$, then C_0 is nonsingular.*

An MRMG of order n over $\mathbb{F}_{2^m}$ is *primitive* if for any choice of nonzero initial state, the sequence generated by that MRMG is periodic of period $2^{mn} - 1$.

In view of Proposition 1 if $I_mX^n - C_{n-1}X^{n-1} - \cdots - C_1X - C_0 \in M_m(\mathbb{F}_2)[X]$ is the matrix polynomial of primitive MRMG, then the matrix C_0 is necessarily nonsingular.

Corresponding to a matrix polynomial $I_m X^n - C_{n-1}X^{n-1} - \cdots - C_1 X - C_0 \in M_m(\mathbb{F}_2)[X]$, we can associate a (m, n)-block companion matrix $\mathsf{C}_{mrmg} \in M_{mn}(\mathbb{F}_2)$ of the following form

$$\mathsf{C}_{mrmg} = \begin{pmatrix} \mathbf{0} & \mathbf{0}\,\mathbf{0} & .\,. & \mathbf{0} & \mathbf{0} & C_0 \\ I_m & \mathbf{0}\,\mathbf{0} & .\,. & \mathbf{0} & \mathbf{0} & C_1 \\ . & .\,. & .\,. & . & . & . \\ . & .\,. & .\,. & . & . & . \\ \mathbf{0} & \mathbf{0}\,\mathbf{0} & .\,. & I_m & \mathbf{0} & C_{n-2} \\ \mathbf{0} & \mathbf{0}\,\mathbf{0} & .\,. & \mathbf{0} & I_m & C_{n-1} \end{pmatrix}, \tag{2}$$

where I_m denotes the $m \times m$ identity matrix over $\mathbb{F}_2$, while $\mathbf{0}$ indicates the zero matrix in $M_m(\mathbb{F}_2)$. Using a Laplace expansion or a suitable sequence of elementary column operations, it is easy to see that $\det \mathsf{C}_{mrmg} = \pm \det(C_0)$. Consequently,

$$\mathsf{C}_{mrmg} \in \mathrm{GL}_{mn}(\mathbb{F}_2) \quad \text{if and only if} \quad C_0 \in \mathrm{GL}_m(\mathbb{F}_2), \tag{3}$$

where $\mathrm{GL}_m(\mathbb{F}_2)$ is the general linear group of $m \times m$ nonsingular matrices over $\mathbb{F}_2$.

It may be noted that the block companion matrix (2) is the state transition matrix for the MRMG (1). Indeed, the k-th state $\mathbf{S}_k := (\mathbf{s}_k, \mathbf{s}_{k+1}, \ldots, \mathbf{s}_{k+n-1}) \in \mathbb{F}_{2^m}^n$ of the MRMG (1) is obtained from the initial state $\mathbf{S}_0 := (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{n-1}) \in \mathbb{F}_{2^m}^n$ by $\mathbf{S}_k = \mathbf{S}_0 \mathsf{C}_{mrmg}^k$, for any $k \geq 0$. We can identify MRMG (1) with the block companion matrix (2).

3 Experiment and result analysis

3.1 Methodology

For randomness test, we have implemented MRMG to generate 10^6 output bit using GNU Compiler Collection (GCC). In this work, we have used NIST statistical test suite (STS) and DieHarder statistical package for randomness test analysis. These packages have gained extensive popularity for large data randomness test analysis. Initially, we tested 100 independent keystream files for existing MRMGs using different keys, where each file contains 10^6 binary sequence. In the NIST and DieHarder test suites, all the test case randomness parameters or functions take the finite length of input bits and generate a real number between 0 and 1 known as the P-value. In our randomness test analysis, we have chosen the significance level of $\alpha = 0.01$, which is the default value for both DieHarder and NIST. The randomness test analysis of the existing MRMG output has exhibited a few randomness loopholes. To eliminate the randomness loopholes, we shall use an appropriate nonlinear function on MRMG scheme. The nonlinear function is described in Verilog VHDL and synthesized by Xilinx XST tool for finding its hardware implementation metrics.

Apart from that, randomness tests results of NMRMG primitives are compared with three current lightweight stream ciphers namely Plantlet, Fruit, and Lizard. We have coded these stream ciphers in GCC. For encryption/decryption, Plantlet and Fruit uses 80-bit key length and Lizard uses 120-bit key length.

Further, we have analyzed output keystream of NMRMG using different statistical parameters. We also present the cryptanalytic results of NMRMG like linear complexity analysis, autocorrelation etc. In our experiment, all the simulation was performed on Mathematica 10, SageMath Version 7.0 and GCC-4.8. The configuration of the system used for the experiment was as follows: Version: Intel(R) Xeon(R) CPU E31230 3.20 GHz, slot: XU1 PROCESSOR, size: 1794 MHz, capacity: 3800 MHz, width: 64 bits, and clock: 100 MHz. On the basis of the methodology described above, MRMG with a special nonlinear function scheme described in detail in the next subsection 3.2.

3.2 Software Implementation

In general, LFSRs are the most common primitive for communications security due to their desirable statistical properties. The most important and novel feature of MRMG is that it can generate 8-Hex output per clock. Accordingly, we run the MRMG 31,250 times to get the 10^6 binary sequence[3]. In order to compute randomness on these binary files, we used NIST STS and DieHarder package. After running the MRMG, we were able to obtain a few weak randomness test results. Note that output keystream of MRMGs weak randomness results are found out by performing several rounds of statistical tests. All the weak randomness test results are tabulated in Appendix (see Table 7). To improve randomness result the natural attention was drawn towards nonlinear function based cryptographic primitives. We recall the definition of nonlinear vectorial Boolean function for the sake of completeness.

Definition 2. *Let m and n be positive integers. An n-variable vectorial Boolean function $\boldsymbol{f}$ is a map $\boldsymbol{f} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ defined as $\boldsymbol{f}(X) := (f_1(X), \cdots, f_m(X))$, where for each $i = 1, \cdots, m$, $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$ is a usual Boolean function.*

In this section, we propose the NMRMG based on a nonlinear vectorial Boolean function as described in Table 1, which provides more secure bitstream and resist common attacks. The schematic view of our proposed framework is displayed in Fig. 1. In order to achieve specified goals from the MRMG, a simple strategy has been adopted. Let us consider the MRMG generate $(s_0, s_1, \cdots, s_n)$ sequence per clock cycle. The sequence of the subsequent states will be partitioned into parts of 4-bits each and each 4-bit parts are assigned to input variables of the function $x_0, x_1, \cdots, x_l$ as shown in Fig 1 and the corresponding output is obtained. At each iteration process framework generate 4-bit per clock. It is worth noting that, our generalized framework can be customized by modifying the number of input lines and input output word size. Assume that MRMG generate 32-bits, say $(s_0, s_1, \cdots, s_{31})$, per clock. Now, 32-output bits are separated into eight parts where each part is assigned to $x_0, x_1, \cdots, x_7$ variables and further it passes through the nonlinear function directly. A general overview of separation of output bits is shown as below.

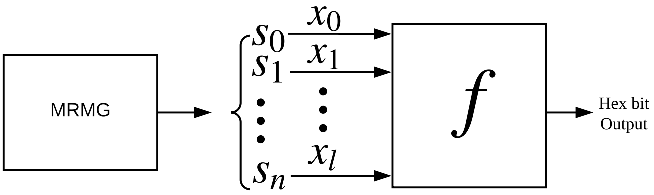

Fig. 1. Framework of NMRMG Model

$$\underbrace{\overbrace{s_0 \dots s_3}^{x_0},\ \overbrace{s_4 \dots s_7}^{x_1},\ \dots,\ \overbrace{s_{28} \dots s_{31}}^{x_7}}_{\text{per clock state bit}} \xrightarrow{f} \text{output}$$

To explain the above mentioned concepts precisely, let us review one concrete example.

Example 1. In this example we consider a word oriented LFSR generate 16-bit (4-Hex) output per clock. Let the output sequence is $8c49$. Now, we consider $x_0 = 1000(8)$, $x_1 = 1100(c)$, $x_2 = 0100(4)$, $x_3 = 1001(9)$. Now we consider one four variable nonlinear function like $f(x_0, x_1, x_2, x_3) = x_0 * x_1 \oplus x_2 * x_3 \oplus x_3$. Result of the function is $f = (1000\&1100) \oplus (0100\&1001) \oplus 1001 = 0001$. Here, we receive four bit 0001 (1 Hex) output in first clock pulse.

$\boldsymbol{f}$-function analysis

Designing a strong nonlinear function for existing MRMG with high periodicity and good cryptographic properties is a challenging task. We have studied several aspects of vectorial Boolean function which

[3] Initially we received 250000-Hex value and later it convert to 1000000(10^6) binary output for randomness test.

possess good cryptographic properties that make the structure more secure. We would like to point out here that designing a cryptographically secure Boolean function for introducing nonlinearity on the contents of FSR is a matter of research because it involves adjusting various contradicting parameters such as Balancedness, Algebraic immunity, Walsh transform, Low autocorrelation etc.

Not only the vectorial Boolean function should be cryptographically secure, but also it should be able to eliminate the randomness loopholes of the basic MRMG structure. This makes the selection of appropriate vectorial Boolean function even more difficult. After exhaustive experimentation, we could arrive at a strong vectorial Boolean function as described in Table 1 that maintains high nonlinearity and above mentioned parameters. In fact, during our experiments, we used various vectorial Boolean functions having different number of variables on the contents of MRMG. Finally, as discussed above, we zeroed in on a 8-variable strong vectorial Boolean function which maintains maximum cryptographic properties [3]. Nonlinear Boolean functions can be represented in different ways, the most commonly used are the Algebraic Normal Form (ANF) or the Truth Table. The expression of the 8-variable function(***f***) characterized in Table 1 (see ANF). Here, we compute several cryptographic properties of this function and it has been summarized in Table 1. This nonlinear function has been implemented in Very High Speed Integrated Circuit Hardware Description Language (VHDL) with structural description logic. More specifically, VHDL code synthesized using Xilinx (Virtex) FPGA, family Automotive Spartan3, device xa3s50-4-vqg100. The synthesis results and execution analysis are shown in Table 6. The schematic view of nonlinear function is displayed in Fig. 4.

Table 1. 8-variable function(***f***) with its cryptographic properties

Spectral Properties	**Results of the function**
Number of variables	8
Balanced	True
Nonlinearity	116
Algebraic immunity	3
Absolute indicator	16
Sum of square indicator()	89728
Absolute autocorrelation items (sorted)	{(0, 72), (8, 118), (16, 65), (256, 1)}
Absolute Walsh spectrum (sorted)	{(0, 4), (4, 16), (8, 28), (12, 48), (16, 76), (20, 64), (24, 20)}
Correlation immunity	0
Dimension of linear structures	0
Resiliency order	0
Set().ring()	True
Bent	False
Symmetric	False
Truth Table(format = Hex)	7eb4719b4da742a8bbe124ce18fa17fd7e6b716c4d58c2572b3e3431180d1702
Algebraic Normal Form (ANF)	$x_0x_1x_2x_3x_4x_5x_6 \oplus x_0x_1x_2x_3x_4x_5x_7 \oplus x_0x_1x_2x_3x_4x_5 \oplus x_0x_1x_2x_3x_4x_6 \oplus x_0x_1x_2x_3x_4x_7 \oplus x_0x_1x_2x_3x_5x_6x_7 \oplus x_0x_1x_2x_3x_5x_6 \oplus x_0x_1x_2x_3x_5x_7 \oplus x_0x_1x_2x_3x_5 \oplus x_0x_1x_2x_3x_6x_7 \oplus x_0x_1x_2x_3x_6 \oplus x_0x_1x_2x_3 \oplus x_0x_1x_2x_4x_5x_6 \oplus x_0x_1x_2x_4x_6x_7 \oplus x_0x_1x_2x_4x_7 \oplus x_0x_1x_2x_5x_6x_7 \oplus x_0x_1x_2x_5x_6 \oplus x_0x_1x_2 \oplus x_0x_1x_3x_4x_5x_6 \oplus x_0x_1x_3x_4x_5x_7 \oplus x_0x_1x_3x_4x_6x_7 \oplus x_0x_1x_3x_4x_7 \oplus x_0x_1x_3x_5x_6x_7 \oplus x_0x_1x_3x_5x_6 \oplus x_0x_1x_4x_5x_6 \oplus x_0x_1x_4x_5x_7 \oplus x_0x_1x_4x_6x_7 \oplus x_0x_1x_4x_7 \oplus x_0x_1x_5x_6x_7 \oplus x_0x_1x_5x_6 \oplus x_0x_1 \oplus x_0x_2x_3x_4x_5x_6 \oplus x_0x_2x_3x_4x_5x_7 \oplus x_0x_2x_3x_4x_5 \oplus x_0x_2x_3x_5x_6x_7 \oplus x_0x_2x_3x_5x_6 \oplus x_0x_2x_3x_5x_7 \oplus x_0x_2x_3x_5 \oplus x_0x_2x_4x_5x_7 \oplus x_0x_2 \oplus x_0x_3 \oplus x_0x_6 \oplus x_0 \oplus x_1x_2x_3x_4x_5x_6 \oplus x_1x_2x_3x_4x_5 \oplus x_1x_2x_3x_5x_6x_7 \oplus x_1x_2x_3x_5x_6 \oplus x_1x_2x_3x_5x_7 \oplus x_1x_2x_3x_5 \oplus x_1x_2x_3 \oplus x_1x_5 \oplus x_2x_3x_4x_5x_6 \oplus x_2x_3x_4x_5 \oplus x_2x_3x_5x_6x_7 \oplus x_2x_3x_5x_6 \oplus x_2x_3x_5x_7 \oplus x_2x_3x_5 \oplus x_2x_4 \oplus x_3x_7 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7$

We consider an MRMG that generates 32-bit (8-Hex) output per clock. If we separate out these 32-bit into 8-variables $(x_0, x_1, \ldots x_7)$, like in Example 1, where each variable initialized with 4-bit(1 Hex) and pass them through the vectorial Boolean function whose ANF is given in Table 1. As a consequence, this framework produces 4-bit per clock. In this way, a long sequence can be easily obtained. Thus, our strategy increases the level of complexity, which will eventually help in resisting the different kind of attacks.

4 Randomness Test analysis

Producing high quality random numbers from the deterministic system is a live research problem. In general, the output sequences of the Pseudorandom Generator (PRG) act like independent random variables from the uniform distribution over (0,1) [6]. A PRG can be mathematically defined as

Definition 3. *A 'pseudorandom generator (PRG)' model generates deterministic pseudorandom bitstream using a seed of lesser number of bits. A PRG is defined as the class of function* $\mathcal{G}_{\mathsf{Gen}} : \{0,1\}^m \rightarrow \{0,1\}^n$ *where* $(n > m)$*. Here, short random sequence convert into a large sequence that looks random and* m *represent as seed value or 'secret key'.*

We know, randomness of the bitstream produced by any deterministic machine is a good indicator of its security strength. Using NIST STS and DieHarder statistical package we examine randomness tests of NMRMG various output sample.

4.1 NIST Statistical Test Suite

NIST STS consists of 15 different statistical tests [4]. NIST STS is used to detect the nonrandom features of the binary sequences. Specifically, this test suite is used to decide the randomness behaviour and its essential security level. All the tests are usually grouped into test batteries which allows more complex randomness analysis. P-values are uniformly distributed and it express how the data deviate from a established statistical model. All the randomness tests are conducted to verify the correctness of our work. We compile all the fifteen tests on NMRMG pseudorandom data and achieved good randomness results. Three types of NMRMG randomness test results are elaborated in Table 4.

4.2 DieHarder statistical Test Suite

DieHarder is the modern and extensive statistical tools popularly used for randomness checking. DieHarder package has been proposed by Robert G. Brown [2]. This test suit consists of 31 fully independent statistical tests. Since, most of the generator performances are established on clever heuristics. In general, P-value of a randomness test encountered by the properties of statistical interpretation. Note that each test of the package is formulated by the number of samples, number of tuples, tsamples and psamples [2]. In our work, each test tuple value increases or decreases with respect to the interpreting P-value after the certain interval. A few tests of DieHarder, numerous interpreting P-values are available. For the evaluation of randomness, we plotted the P-value along the x-axis and the tuple values along the y-axis [4]. We consider several P-values that are generated for sts_serial, rgb lagged sum test, rgb_ bitdist, rgb_minimum_distance, and rgb_permutation tests. The DieHarder test results are plotted graphically as follows: sts_serial results of three NMRMGs are elaborates in Fig. 2(a). Fig. 2(b) illustrates P-values of rgb lagged sum test. A triple vertical bar graph Fig 2(c), compares three series of tests namely rgb_ bitdist, rgb_minimum_distance, and rgb_permutations P-values. Here, the error bar of bar graphs often represent the deviated (SEM ±) values. The remaining tests whose evaluation (assessment) resulted in one P-value are plotted in the Fig. 2(d) of each NMRMG.

Here, we introduced the notion of pseudorandomness in the context of NMRMGs. These are efficient deterministic programs that expand short, randomly selected seeds (i.e., Secret Keys) into much longer pseudorandom bit sequences that are computationally indistinguishable from truly random sequences by efficient algorithms. Hence the notion of computational indistinguishability (i.e., indistinguishability by efficient procedures) also plays a pivotal role in our discussion. Following [1], let us briefly discuss the theory of PRG for NMRMG.

Theorem 1. *A PRG must be unpredictable, we can say that* $\mathcal{G} : \mathcal{M} \rightarrow \{0,1\}^n$ *and it is assumed to be predictable if:* $\exists$ *an 'efficient' algorithm* $\mathcal{A}_{\mathsf{nmrmg}}$ *and* $\exists\ 0 < i \leq (n-1)$ *such that*

$$\underset{m \xleftarrow{\mathsf{Rand}} \mathcal{M}}{Pr}\left[\mathcal{A}_{\mathsf{nmrmg}}\big(\mathcal{G}_{\mathsf{Gen}}(m)\big)\Big|_{1,\ldots,i} = \mathcal{G}_{\mathsf{Gen}}(m)\Big|_{i+1}\right] \geq \tfrac{1}{2} + \epsilon$$

for some non-negligible ϵ *value.* $\forall_i$ *considerably no efficient adversaries can predict* $(i+1)$ *bit for some non-negligible* ϵ *value.*

[4] https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf

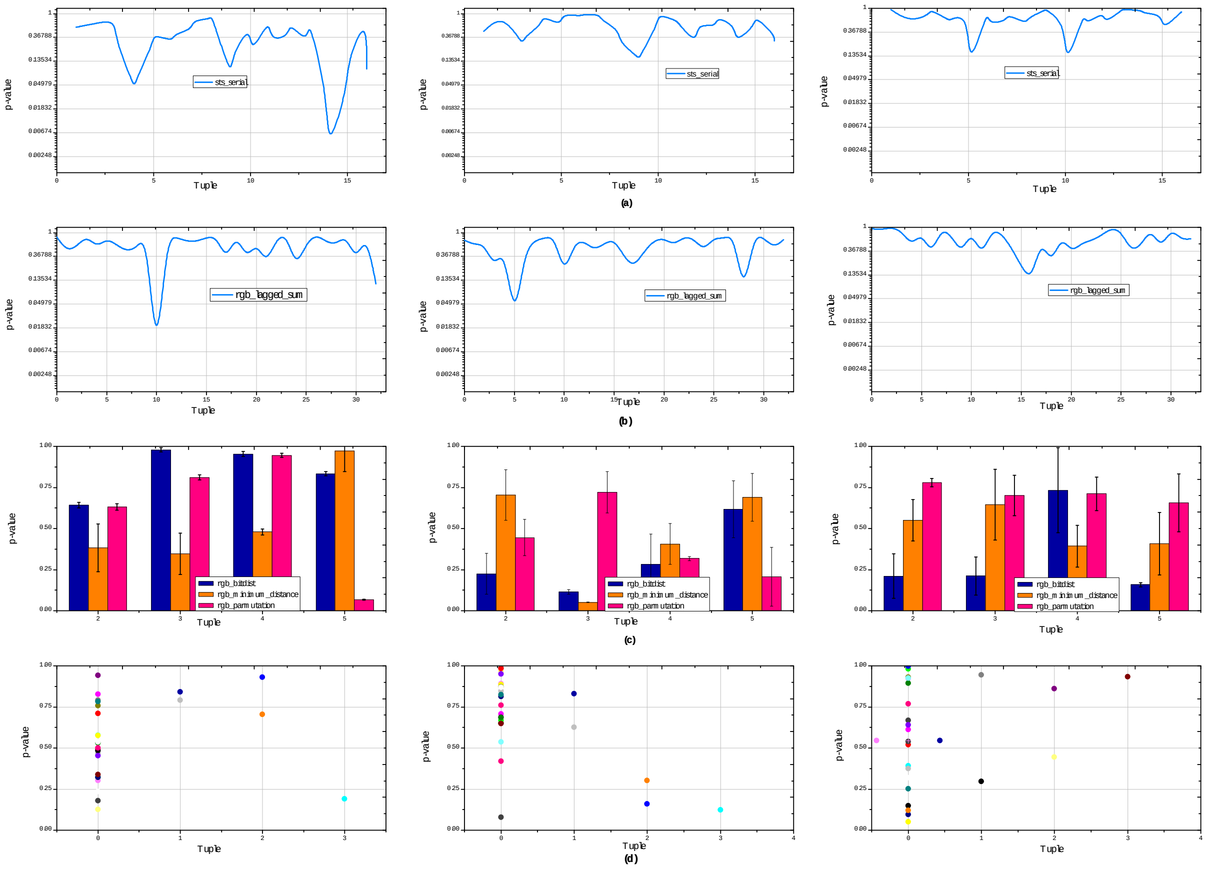

Fig. 2. DieHarder Pass Randomness results for three types of NMRMG

5 Security evaluation of NMRMG

In the previous section, we described NMRMG based framework which provides good quality random output bitstream. Here, we will explain some statistical and cryptographic properties of the keystream generated by NMRMG. Following subsection deals with various data analysis, autocorrelation and how NMRMG is potentially effective against Berlekamp Massey algorithm. These analysis will provide a good estimate of the security level provided by the proposed NMRMG.

5.1 Data Analysis

This subsection covers detailed analysis of the output data set of NMRMG. It is very important to know the probability distributions of output data set for discrete random variables. From the large (250000 Hex) output sequence, we compute a finite number of possible outcomes from $(0, 1, \ldots, 9, a, \ldots f)$. This analysis has been carried out in terms of Frequency Distribution, Coincidence Index, Entropy, Standard Deviation, Variance, and Standard Error of the Mean (SEM ±). Table 2 illustrate numerical data for the mentioned properties.

5.2 Berlekamp Massey Approach

Linear complexity (LC) is one of the most important security indicator of LFSR based stream ciphers. In cryptographic point of view, a secure stream cipher can achieve good security level when it produces a sequence with large linear complexity. In that regard, Berlekamp-Massey algorithm is used to determine the length of the LFSR. More precisely, it finds the original polynomial equation from the LFSR

Table 2. Statistical data analysis of the NMRMG framework

Properties	**NMRMG I**	**NMRMG II**	**NMRMG III**
Frequency Distribution	(0, 0.0781250000000000), (1, 0.0312500000000000), (2, 0.0625000000000000), (3, 0.0468750000000000), (4, 0.0312500000000000), (5, 0.140625000000000), (6, 0.0625000000000000), (7, 0.0468750000000000), (8, 0.0781250000000000), (9, 0.0312500000000000), (a, 0.0937500000000000), (b, 0.0312500000000000), (c, 0.0468750000000000), (d, 0.0625000000000000), (e, 0.0937500000000000), (f, 0.0625000000000000)	(0, 0.0156250000000000), (1, 0.0781250000000000), (2, 0.0156250000000000), (3, 0.0468750000000000), (4, 0.0156250000000000), (5, 0.0625000000000000), (6, 0.0781250000000000), (7, 0.0468750000000000), (8, 0.125000000000000), (9, 0.0468750000000000), (a, 0.125000000000000), (b, 0.0625000000000000), (c, 0.0625000000000000), (d, 0.0625000000000000), (e, 0.0468750000000000), (f, 0.109375000000000)	(0, 0.0781250000000000), (1, 0.0468750000000000), (2, 0.0312500000000000), (3, 0.0625000000000000), (4, 0.0625000000000000), (5, 0.0625000000000000), (6, 0.0781250000000000), (7, 0.0781250000000000), (8, 0.078125000000000), (9, 0.0625000000000000), (a, 0.0625000000000000), (b, 0.0625000000000000), (c, 0.109375000000000), (d, 0.0156250000000000), (e, 0.0625000000000000), (f, 0.0468750000000000)
Coincidence Index	0.0610119047619048	0.0659722222222222	0.0610119047619048
Entropy	2.006034264775	2.130803302135	2.080419673776
St. Deviation	0.0296463530640786	0.0347048567686618	0.0213478140957492
Variance	0.000878906250000000	0.00120442708333333	0.000455729166666667
Variance(bias = True)	0.000823974609375000	0.00112915039062500	0.000427246093750000
St. Error of the Mean (SEM ±)	0.0074115882660196	0.0086762141921655	0.0053369535239373

output sequence. According to Massey algorithm, only $2n$ consecutive output sequences are required for determining the linear complexity of the feedback polynomial of a LFSR. Let, binary sequences are $s = (s_0, s_1, \ldots, s_{n-1})$ and linear complexity is l. Therefore, connection polynomial of s can be represented as

$$C(x) = c_0 + c_1 x + c_2 x_2 + \ldots + c_l x_l. \tag{5}$$

Using Berlekamp-Massey algorithm it would be easy to identify the value of l and $C(x)$.

Example 2. For instance, an LFSR assign with seed value $\big[1, 1, 0, 1\big]$ with tap value $\big[1, 0, 0, 1\big]$. We know, $2n$ bits are enough to find connection polynomial. Initially, first 20 output bits are given by $\big[1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0\big]$. In that case, above mentioned algorithm gives the original polynomial D^4 + D^3 + 1.

In our work, we have first considered 256 output bits of the NMRMG and it is loaded into Berlekamp-Massey algorithm for the original polynomial equation. This algorithm fails to recover original feedback polynomial and the recovered polynomials are tabulated in Table 3. The tabulated results clearly show

Table 3. NMRMG Polynomial characteristics using BerlekampMassey algorithm

Length	**NMRMG I**	**NMRMG II**	**NMRMG III**
First 256-bit output sequence	D^128 + D^127 + D^126 + D^125 + D^123 + D^121 + D^118 + D^116 + D^115 + D^114 + D^113 + D^111 + D^109 + D^108 + D^107 + D^105 + D^102 + D^100 + D^99 + D^97 + D^96 + D^95 + D^90 + D^89 + D^87 + D^86 + D^85 + D^84 + D^83 + D^81 + D^80 + D^79 + D^76 + D^74 + D^72 + D^70 + D^69 + D^67 + D^66 + D^64 + D^63 + D^62 + D^60 + D^59 + D^55 + D^54 + D^53 + D^52 + D^51 + D^50 + D^49 + D^44 + D^43+ D^35 + D^34 + D^31 + D^29 + D^27 + D^25 + D^19 + D^18 + D^12 + D^11 + D^9 + D^8 + D^7 + D^5 + D^4 + D^3 + 1	D^125 + D^117 + D^112 + D^108 + D^107 + D^106 + D^103 + D^101 + D^100 + D^99 + D^95 + D^93 + D^91 + D^90 + D^88 + D^82 + D^77 + D^76 + D^75 + D^74 + D^72 + D^71 + D^69 + D^68 + D^64 + D^63 + D^60 + D^58 + D^57 + D^54 + D^53 + D^52 + D^51 + D^46 + D^45 + D^43 + D^39 + D^36 + D^35 + D^31 + D^29 + D^28 + D^25 + D^24 + D^23 + D^21 + D^20 + D^16 + D^15 + D^14 + D^13 + D^10 + D^8 + D^7 + D^5 + D^3 + D + 1	D^129 + D^128 + D^127 + D^124 + D^123 + D^120 + D^119 + D^116 + D^114 + D^113 + D^112 + D^109 + D^106 + D^105 + D^104 + D^99 + D^97 + D^96 + D^95 + D^92 + D^90 + D^89 + D^88 + D^87 + D^86 + D^84 + D^83 + D^80 + D^78 + D^77 + D^76 + D^75 + D^74 + D^73 + D^70 + D^67 + D^66 + D^63 + D^62 + D^61 + D^60 + D^58 + D^53 + D^51 + D^50 + D^46 + D^45 + D^43 + D^41 + D^39 + D^38 + D^36 + D^35 + D^33 + D^32 + D^31 + D^30 + D^29 + D^28 + D^27 + D^26 + D^25 + D^24 + D^22 + D^21 + D^16 + D^15 + D^12 + D^7 + D^6 + D^5 + D^2 + D + 1
First 64-Hexbit output sequence	D^32 + D^30 + D^26 + D^23 + D^21 + D^19 + D^18 + D^17 + D^15 + D^14 + D^13 + D^11 + D^10 + D^9 + D^8 + D^6 + D^5 + D^4 + 1	D^33 + D^32 + D^26 + D^24 + D^22 + D^21 + D^20 + D^18 + D^17 + D^11 + D^10 + D^9 + D^4 + D^3 + D^2 + D + 1	D^30 + D^27 + D^24 + D^23 + D^22 + D^21 + D^20 + D^17 + D^16 + D^15 + D^14 + D^13 + D^12 + D^11 + D^10 + D^9 + D^8 + D^7 + D^5 + D^3 + D^2 + 1

that the proposed framework defends the original feedback polynomial. Note that the adversary cannot generate the original feedback polynomial from the above mentioned algorithm in reasonable time.

5.3 Autocorrelation analysis

Binary strings with proper autocorrelation features play an essential role in secure applications. In this paradigm, the attacker attempts to establish some probabilistic relation between the Key bits and the output. Autocorrelation analysis is most relevant for LFSR based stream ciphers [11]. Let $\hat{a}$ be a periodic binary sequence of NMRMG with period p. Autocorrelation function is defined as follows

$$\mathcal{AC}_{\hat{a}}(\tau) = \sum_{\imath=1}^{p-1}(-1)^{(a_\imath + a_{\imath+\tau})} \qquad , 0 \le \tau \le p-1 \tag{6}$$

From the cryptographic overview, autocorrelation assures that the attacker can not determine correlations among shifted versions of the identical bitstream.

Example 3. Here, we consider our previous example 2. We know, seed $= \big[1,1,0,1\big]$, tap $= \big[1,0,0,1\big]$, $n = 20$, output $= \big[1,1,0,1,0,1,1,0,0,1,0,0,0,1,1,1,1,0,1,0\big]$. Now, $\mathcal{AC}(\text{output},\ 15,\ 7) = \frac{4}{15}$.

In this work, we present autocorrelation analysis for NMRMG. Here we estimate the accurate approximation of the correlation values. According to our study, we did not find any linear correlation in NMRMG output. Figure 3 has been drawn on the basis of pairs of autocorrelation indices. However, pairs of autocorrelation data is difficult to describe for large data-set, so graphically we present first 256 bit (64 Hex) random output sequence for NMRMG.

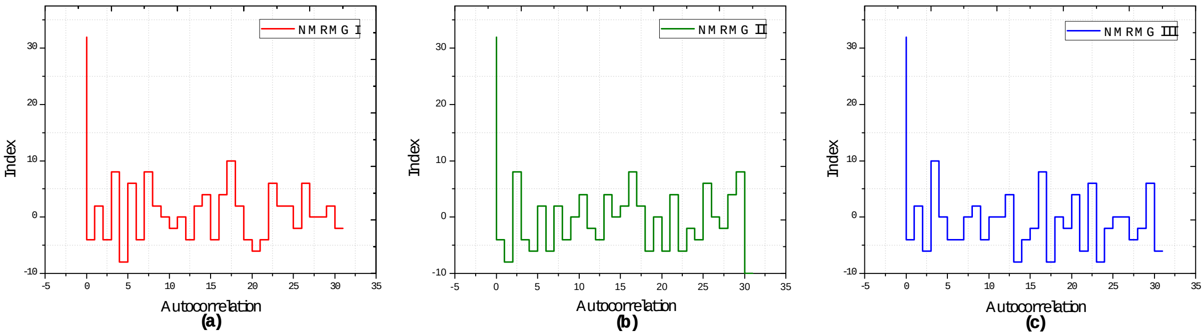

Fig. 3. Autocorrelation result analysis of (a) NMRMG I, (b) NMRMG II, (c) NMRMG III

5.4 Randomness Comparison

Very recently, Lizard [5], Fruit [13], and Plantlet [8] stream ciphers were introduced and these claimed to resist common attacks. Modern lightweight stream ciphers security rely on keystream randomness. We provide a comprehensive randomness evaluation of three current lightweight stream ciphers namely Lizard, Fruit, and Plantlet with NMRMG framework. It may be noted, designers of the Fruit have modified the original specification of their cipher and here we consider first version of the Fruit cipher. The important aspect of the randomness test analysis, we have achieved better randomness results (see Table 4) compared to three stream ciphers.

6 Conclusion

On one hand, we have shown that the binary sequences generated by MRMG do not satisfy the randomness properties. However on the other hand, we have introduced a nonlinearly filtered MRMG based on

Table 4. NIST STS randomness test results of NMRMG

Name of the test	**NMRMG I**		**NMRMG II**		**NMRMG III**	
	P-value	Evaluation	P-value	Evaluation	P-value	Evaluation
Frequency	0.304126	Pass	0.595549	Pass	0.094664	Pass
Block frequency	0.739918	Pass	0.011791	Pass	0.534146	Pass
Cumulative sums	0.739918	Pass	0.616305	Pass	0.671779	Pass
Runs	0.834308	Pass	0.419021	Pass	0.100508	Pass
Longest run	0.739918	Pass	0.574903	Pass	0.804337	Pass
Rank	0.010988	Pass	0.010255	Pass	0.010735	Pass
FFT	0.171867	Pass	0.262249	Pass	0.072663	Pass
Non Overlapping template (mean value)	0.637119	Pass	0.474986	Pass	0.407091	Pass
Overlapping template	0.108791	Pass	0.437274	Pass	0.098526	Pass
Universal	0.129620	Pass	0.137282	Pass	0.113526	Pass
Approximate entropy	0.262249	Pass	0.102526	Pass	0.127148	Pass
Serial1	0.171867	Pass	0.897763	Pass	0.949602	Pass
Serial2	0.759756	Pass	0.171867	Pass	0.050845	Pass
Linear complexity	0.045675	Pass	0.085587	Pass	0.132758	Pass
Random excursions (mean value from x = 4 . . . + 4)	0.401199	Pass	0.401199	Pass	0.534146	Pass
Random excursions variant (mean value from x = 9 . . . + 9)	0.383827	Pass	0.474986	Pass	0.449672	Pass

Table 5. NIST STS randomness test results of Lizard, Fruit, Plantlet

Name of the test	**Lizard**		**Fruit**		**Plantlet**	
	P-value	Evaluation	P-value	Evaluation	P-value	Evaluation
Frequency	0.350485	Pass	0.122325	Pass	0.924076	Pass
Block frequency	0.008879	Pass	0.350485	Pass	0.637119	Pass
Cumulative sums	0.350485	Pass	0.739918	Pass	0.514124	Pass
Runs	0.534146	Pass	0.911413	Pass	0.181557	Pass
Longest run	0.350485	Pass	0.911413	Pass	0.739918	Pass
Rank	0.534146	Pass	0.213309	Pass	0.000070	Weak
FFT	0.991468	Pass	0.739918	Pass	0.004301	Pass
Non Overlapping template (mean value)	0.550133	Pass	0.474986	Pass	0.657933	Pass
Overlapping template	0.350485	Pass	0.350485	Pass	0.851383	Pass
Universal	0.000000	Weak	0.000513	Weak	0.000000	Weak
Approximate entropy	0.350485	Pass	0.739918	Pass	0.191687	Pass
Serial1	0.911413	Pass	0.000911	Weak	0.574903	Pass
Serial2	0.137154	Pass	0.002089	Weak	0.262249	Pass
Linear complexity	0.035174	Pass	0.031254	Pass	0.383827	Pass
Random excursions (mean value from x = 4 . . . + 4)	0.410236	Pass	0.273561	Pass	0.534146	Pass
Random excursions variant (mean value from x = 9 . . . + 9)	0.216984	Pass	0.489719	Pass	0.616305	Pass

a special nonlinear vectorial Boolean function and established that randomness properties of sequences generated by such a nonlinearly filtered MRMG are significantly better than their usual linear counter parts. We have also compared our test results with some of recent lightweight stream ciphers. In future, we plan to study the componentwise linear complexity of sequences generated by nonlinearly filtered MRMG.

References

1. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Version 0.1, fr om http://cryptobook. net (2008)
2. Brown, R.G.: Dieharder: A random number test suite (version 3.31), (2013)
3. Burnett, L., Millan, W., Dawson, E., Clark, A.: Simpler methods for generating better boolean functions with good cryptographic properties. Australasian Journal of Combinatorics **29**, 231–248 (2004)
4. Deb, S., Bhuyan, B.: Performance evaluation of grain family and espresso ciphers for applications on resource constrained devices. ICT Express **4**(1), 19–23 (2018)
5. Hamann, M., Krause, M., Meier, W.: Lizard–a lightweight stream cipher for power-constrained devices. IACR Transactions on Symmetric Cryptology **2017**(1), 45–79 (2017)
6. Haramoto, H.: Automation of statistical tests on randomness to obtain clearer conclusion. In: Monte Carlo and Quasi-Monte Carlo Methods 2008, pp. 411–421 (2009)
7. Hasan, S.U., Panario, D., Wang, Q.: Nonlinear vectorial primitive recursive sequences. Cryptogr. Commun. (2017). `https://doi.org/10.1007/s12095-017-0265-2`.
8. Mikhalev, V., Armknecht, F., Müller, C.: On ciphers that continuously access the non-volatile key. IACR Transactions on Symmetric Cryptology **2016**(2), 52–79 (2017)

9. Niederreiter, H.: Factorization of polynomials and some linear-algebra problems over finite fields. Linear Algebra and its Applications **192**, 301–328 (1993)
10. Preneel, B.: Introduction to the proceedings of the second workshop on fast software encryption, vol. 1008 of lecture notes in comput. sci., 1–5 (1995)
11. Tarannikov, Y., Korolev, P., Botev, A.: Autocorrelation coefficients and correlation immunity of boolean functions. In: Boyd, C. (ed.) Advances in Cryptology — ASIACRYPT 2001. pp. 460–479. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
12. Tsaban, B., Vishne, U.: Efficient linear feedback shift registers with maximal period. Finite Fields and Their Applications **8**(2), 256–267 (2002)
13. Vahid Amin Ghafari, H.H., Chen, Y.: Fruit-v2: Ultra-lightweight stream cipher with shorter internal state. Cryptology ePrint Archive, Report 2016/355 (2016), available at: http://eprint.iacr.org/2016/355
14. Zeng, G., Han, W., He, K.: High efficiency feedback shift register: $\sigma-$lfsr. Cryptology ePrint Archive, Report 2007/114 (2007), https://eprint.iacr.org/2007/114

Appendix

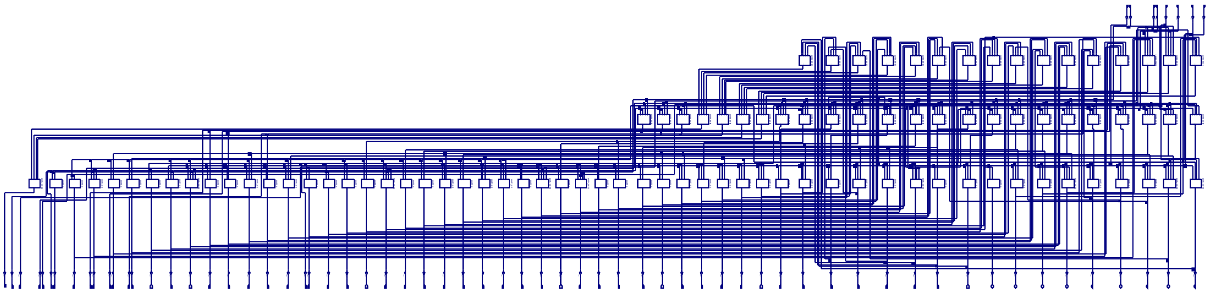

Fig. 4. Synthesized hardware structure (Schematic technology) of the function

Table 6. synthesis results of the function using xa3s50-4-vqg100

Synthesis report on xa3s50-4-vqg100			
Resources	**Use**	**Available**	**Utililization**
No. of Slices	55	768	7%
No. of 4 I/O LUTs	95	1536	6%
No. of bonded IOBs	72	63	114%
Cell in out	**fanout**	**Gate Delay (ns)**	**Net Delay (ns)**
IBUF:I->O	35	0.821	2.209
LUT4:I0->O	2	0.551	1.072
LUT2:I1->O	2	0.551	1.216
LUT4:I0->O	1	0.551	1.140
LUT4:I0->O	1	0.551	0.996
LUT4:I1->O	1	0.551	0.801
OBUF:I->O	-	5.644	-
Total- 16.654 ns (9.220 ns logic, 7.434 ns route), where 55.4% logic and 44.6% route			

Table 7. DieHarder and NIST STS weak randomness results of MRMG

Dieharder version 3.31.1, Copyright-Robert G. Brown			
Types of MRMG	**Test name**	***P*-value**	**Assessment**
MRMG I	diehard_birthdays	0.99569301	Weak
	sts_serial	0.99589102	Weak
	diehard_dna	0.00025383	Weak
	sts_monobit	0.99743159	Weak
MRMG II	sts_serial	0.99883080	Weak
	diehard_birthdays	0.99873041	Weak
	sts_monobit	0.99479301	Weak
MRMG III	rgb_lagged_sum	0.99970352	Weak
	diehard_runs	0.99671643	Weak
	sts_monobit	0.99193765	Weak
NIST STS randomness test			
MRMG I	Cumulative Sums	0.000071	Weak
	Rank	0.153763	Weak
	Universal	0.000029	Weak
	Linear Complexity	0.000174	Weak
MRMG II	FFT	0.000371	Weak
	Rank	0.000000	Weak
	Linear Complexity	0.000089	Weak
MRMG III	Runs	0.994076	Weak
	Rank	0.000079	Weak
	LinearComplexity	0.055361	Weak