Experimental Evaluation of Certain Security Issues of Grain v1 Stream Cipher

Arsić, Aleksandra; Jelisavčić, Vladisav; and Mihaljević, Miodrag J.

Abstract—*This paper provides additional insights regarding certain Grain-v1 security evaluation and security enhancement approaches. The paper provides experimental evaluation of a recently reported the time-memory trade-off paradigm employed for cryptanalysis and experimental evaluation of statistical features of Grain-v1 output sequences which are subject of error-correction encoding and degradation by a channel with bit-deletions.*

Index Terms—*Grain-v1, lightweight encryption, stream ciphers, security evaluation, security enhancement.*

1. INTRODUCTION

Lightweight cryptographic techniques have been recognized as substantial components for providing cyber-security and particularly security within Internet of Things (IoT) and machine-to-machine communications (M2M). These techniques should support minimization of the overheads implied by employed security mechanisms. The required minimization is particularly related to minimization of the implementation complexity and the energy consumption. On the other hand, lightweight cryptographic techniques also should provide high level of the cryptographic security which implies a number of challenges regarding design and security evaluation of these techniques. Lightweight encryption techniques appear as an important class, and a number of block and stream ciphers have been proposed. One of the proposed stream ciphers is Grain-v1, [1], [2], which has received a significant attention because of it implementation requirements which can fit even into very restricted RFID implementation scenarios. Also, Grain-v1 has been subject of a number of security evaluations (see [3], [4] and [5], for example).

It has been claimed that [5] yields the most powerful method for cryptanalysis of Grain-v1 focused on a state recovery attack against Grain-v1 in the single key and IV pair setting using time-memory-data tradeoffs. The proposed cryptanalysis is based on the following. Firstly, the concept of k-normality has been extended into k-linear-normality of Boolean functions. Then, the k-linear-normality of the filter function is combined with sampling resistance under the constraints of some state bits, which makes the sampling resistance much longer, and reduces the searching space that supports wider tradeoff parameters. This kind of sampling resistance is called a conditional sampling resistance. For Grain-v1, a conditional sampling resistance has been pointed-out based on a specific guessing path that by fixing 51 bits of state constraint conditions and guessing 81 bits more of the internal state. The remaining 28 bits of the state can be recovered directly using the first 28 keystream output bits generated from the state, which is 10 bits longer than the sampling resistance given [3]. According to the conditional sampling resistance, a time-memory-data tradeoff attack against Grain-v1 has been conducted and its cryptanalytic power has been claimed by the following: The proposed cryptanalysis requires $T = 2^{61}$ table look-up operations employing a memory of $M = 2^{71}$ dimension assuming available keystream length of $D = 2^{79}$ and the preprocessing time of $P = 2^{81}$, which appear as much better than the best parameters $T = 2^{71}$, $D = 2^{53.5}$, $M = 2^{71}$ and $P = 2^{106.5}$ in the single key and IV pair setting previously reported.

Motivation for the Work. Grain-v1 is among the current candidates for lightweight stream ciphers for IoT and M2M communications. The results reported in [5] indicate additional security weaknesses of Grain-v1, but the claims regarding performance of the proposed time-memory trade-off attack have not been justified by experimental results. Accordingly, one motivation for our work was experimental evaluation of the employed time-memory trade-off approach. On the other hand, the reported weaknesses of the lightweight encryption schemes are a motivation for consideration of the generic approaches for security enhancement of the lightweight encryption techniques. One recently proposed approach has been reported in [6] where employment of the simulated channels with synchronization errors has been proposed for the security enhancement. Following the ideas of the approach proposed in [6], we have been motivated to consider statistical features of the error-correction encoded segments of Grain-v1 keystream after a channel with synchronization errors. Consequently, we address an experimental evaluation of certain error-correction encoding techniques in order to analyze the encoding implication on the statistical features of the encoded binary sequences generated by Grain-

This research is supported by the Ministry of Education, Science and Technological Development, Republic of Serbia.

v1 after a channel with synchronization errors which deletes a fraction of input bits.

Organization of the Paper. Sections II and III summarize Grain-v1 encryption and its recently reported cryptanalysis, respectively. Section IV provides an experimental analysis of the cryptanalytic approach summarized in Section III. Section V addresses impact of error-correction encoding of Grain-v1 output segments after a channel with bit deletions for different error-correction codes. Finally, main messages of this paper are given in Section VI.

2. DESCRIPTION OF GRAIN-V1

In this section a specification of Grain-v1 [1] is described. Grain-v1 is stream cipher which is a part of the eSTREAM project [7]. This cipher is based on two shift registers and filter function. One of the shift registers has linear (LFSR) and the other one has nonlinear (NFSR) feedback. The shift registers are 80 bits long. Algorithm as input accepts an 80-bit binary key and a 64-bit binary IV vector. NFSR register is initialized with key bits and denoted as $(s_0, ..., s_{79})$. LFSR was initialized with 64-bit IV and the remaining bits have value one. This register was denoted as $(b_0, ..., b_{79})$.

In every algorithm's iteration, state of registers are updated with new ones. Every bit in registers was shifted for one position in left. Bits in position i was initialized with values from bits in position $i+1$ from appropriate registers. The biggest positions in both registers are initialized with values from update functions, respectively described with functions (1) and (2). The linear shift register gets output from function (1), nonlinear register gets a value from function (2).

$$s_{t+80} = s_{t+62} \oplus s_{t+51} \oplus s_{t+38} \oplus s_{t+23} \oplus s_{t+13} \oplus s_t \quad (1)$$

$$\begin{aligned} b_{t+80} = {} & s_t \oplus b_{t+62} \oplus b_{t+60} \oplus b_{t+52} \oplus b_{t+45} \\ & \oplus b_{t+37} \oplus b_{t+33} \oplus b_{t+28} \oplus b_{t+21} \oplus b_{t+14} \\ & \oplus b_{t+9} \oplus b_t \oplus b_{t+63} b_{t+60} \oplus b_{t+37} b_{t+33} \\ & \oplus b_{t+15} b_{t+9} \oplus b_{t+60} b_{t+52} b_{t+45} \end{aligned} \quad (2)$$

The contents of shift registers represent the state of the cipher in one clock cycle t. The state of the cipher has 160 bits. At each clock cycle, filter function $h(x)$ takes five variables from both registers and as output produces a single bit. The function $h(x)$ is defined with equation (3)

$$\begin{aligned} h(x_0, x_1, x_2, x_3, x_4) = {} & x_1 \oplus x_4 \oplus x_0 x_3 \oplus x_2 x_3 \\ & \oplus x_3 x_4 \oplus x_0 x_1 x_2 \oplus x_0 x_2 x_3 \oplus x_0 x_2 x_4 \\ & \oplus x_1 x_2 x_4 \oplus x_2 x_3 x_4 \end{aligned} \quad (3)$$

where the variables x_0, x_1, x_2, x_3 and x_4 correspond to the tap positions s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64} and b_{i+63}, respectively.

The output from filter function is masked with seven more bits from NFSR register defined with equation (4). In one clock cycle, function (4) produces one bit which is also Grain-v1 algorithm's out bit. For initialization of keystream length N it is necessary that algorithm has exactly N iterations of shifting registers.

$$\begin{aligned} & z_t = h(x) \oplus \sum_{j \in A} b_{t+j} \\ & where A = \{1, 2, 4, 10, 31, 43, 56\} \end{aligned} \quad (4)$$

The cipher is clocked 160 times without producing any key, after initialization registers with key and IV vector. In the biggest position of LFSR register was putted the output bit from function (4). Described process is called initialization phase. After this phase, cipher will output 1 bit/clock. The algorithm of the Grain-v1 is described in Figure 1.

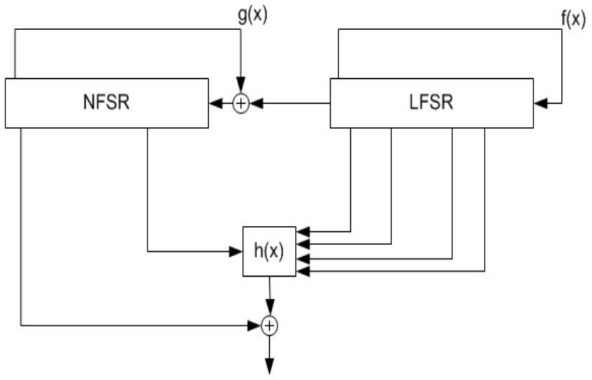

Figure 1: Grain-v1 algorithm

3. ANALYSIS AND TIME-MEMORY-DATA TRADEOFF ATTACK OF GRAIN-V1

This chapter describes mathematical analyze for Grain-v1 with an emphasis on filter function $h(x)$. It is explained how to generate a large matrix for Time-Memory-Data (TMD) Tradeoff Attack [8] according to the conditional sampling resistance.

Filter function $h(x)$ given by formula (3) is nonlinear, in general case. Therefore, it is difficult that attacker reconstructs state of registers in time t using keystream with output bits. Authors of paper [5] deduced certain bit's constraints on the registers where filter function is in linear mode. They noted that the state update function of Grain-v1 is invertible during keystream generation if the bits' conditions are complied. This has a consequence that if we can recover states of registers at some time t, we can clock it backwards to recover the used key. State recovery attack that they proposed focuses on the keystream vector generated after initialization phase. They claim that if a stream cipher has n-bit state, l bits of the internal state

can be recovered with l keystream bits, directly. This scenario is not possible in every situation. It is a special case, when keystream from Grain-v1 algorithm contains output bit from linear filter function. In this study, some bits must be on positions and have values defined with constrains so that the filter function is in linear mode. The attacker has a task to hit remaining bits. Dimension of vector that the attacker must guess is smaller than dimension n.

Authors proposed sampling resistance [3] of Grain-v1 algorithm. Specific conditional sampling resistance for Grain-v1 is presented and guess-and-determine strategy is based on it. They choose two linear modes of filter function $h(x)$ when

$$x_2 = 0, x_3 = 1, \text{ then } h(x) = x_0 \oplus x_1$$

and

$$x_0 = 1, x_1 = 0, x_2 = 1, \text{ then } h(x) = x_3.$$

Scenario when this kind of attack is possible is presented in Table 1. The first column of Table 1 consists state's constrains, positions and values of bits in registers which makes filter function $h(x)$ to be linear. The second column represents keystream bits. Using output function's bits it is possible to recover bits indicated in the last column of the Table 1. For that the attacker needs to hit values in positions denoted in fourth column in Table 1.

Given guess-and-determine strategy allows that if the attacker fixes n bits of state constraints conditions and guesses l bits more of the internal state, he can recover the remaining $(180 - n - l)$ bits of the state using the first $(180 - n - l)$ bits keystream output. Authors of mentioned paper fixed 51 bits respecting restrictions, declared in the first column of Table 1 and try to recover 28 bits, described in the fourth column, by using 28 bits from keystream and guessing 81 bits more defined in the third column. Idea was that the attacker listens output from Grain-v1 algorithm and waits a sequence on the basis of which will be able to assume that a filter function is in linear mode and content of registers on 51 position. After that attacker is able to reconstruct bits solving the system of equations using bits from keystream. For the remaining unknown bits, the attacker uses guess-and-determine strategy.

After analysis are done and known, TMD Tradeoff Attack can be realized. First phase for this attack is preprocessing phase. Goal of preprocessing phase is matrix initialization. According computation power and technical opportunities, our matrix is not the same size like in [5]. Dimensions of matrix that was generated for this study were $m \times t$ where $m = 2^{14}$ and $t = 2^{10}$. The other (81 - 24) bits are initialized by random values and stay fixed all time for guess-and-determine algorithm. The positions of these bits are also arbitrary and chosen from set cited in fourth column of Table 1. Authors of this paper suppose that attacker guessed them successfully, every time. In that way, number of all permutations is smaller, so time and memory space for computing matrix are smaller, too.

Preprocessing phase is described as:

1. Generate a fixed string $s \in \{0,1\}^{28}$ as a segment of keystream according to Table 1.
2. Form a $m \times t$ matrix that tries to cover the whole search space which is composed of all the possible permutations with guessed 24 bits of NFSR and LFSR states as follows:

(a) Randomly generate m startpoints of the chains, each point is represented like vector of 24 bits length.
(b) Under the constraint conditions of 51 bits shown in Table 1, recover the remaining 28 bits of the state using the segment of keystream s according to the guessing path. The rest 57 more bits of state must be fixed and they are some of guessed bits in Table 1. Make it the next point in the chain, and update the registers NFSR and LFSR with this point.
(c) Iterate Step (b) t times on each startpoint respectively.
(d) Store the pairs of startpoints and endpoints (SP_j, EP_j), $j = 1, ..., m$ in a matrix.

4. EXPERIMENTS AND RESULTS

Our first experiment was to generate matrix for TMD Tradeoff Attack and check if there are duplicate of states. Dimensions of our matrix were $m \times t$ where $m = 2^{14}$ and $t = 2^{10}$. First, startpoints for each row was random initialized, keeping in the mind there are no duplicate startpoints. Next step was to fill all states in matrix. Every state in chains, except first one, is result from 24 iterations of Grain-v1 algorithm with registers initialization with previous state's bits from the same row in matrix. After filling the matrix, content of matrix was analyzed.

Experiments were repeated 50 times. In every experiment, indexes and values of fixed bits, which we can not generate with guess strategy, were different. Contents of startpoints in matrix were generated every time, too. Experiments showed that some states in matrix occur more than one time. The conclusion is the matrix does not contain all the elements of search space. The numbers of repetitions are in range from 1 to 1895496 times, per one state. Repetition rate in all experiments was in range from 48.59 to 54.17 percent of number of all matrix states.

Average case for all experiments can be seen in Figure 2. All experiments had uniform results. There was no experiment with the result that had a lot of waste than the average case. Analysis of this experiment were consisted in calculating the number of occurrences of each state in the matrix. Since the number of occurrences of each state is known, states are grouped according to

TABLE 1: Guess-and-determine strategy

Step	Constraint conditions	Concerned bits	Guessed NFSR and LFSR bits	Recovered bit
0	$s_{46}=0, s_{64}=1$	z_0	$b_1, b_2, b_4, b_{31}, b_{43}, b_{56}, s_3$	b_{10}
1	$s_{47}=0, s_{65}=1$	z_1	$b_3, b_5, b_{32}, b_{44}, b_{57}, s_4$	b_{11}
2	$s_{48}=0, s_{66}=1$	z_2	$b_6, b_{33}, b_{45}, b_{58}, s_5$	b_{12}
3	$s_{49}=0, s_{67}=1$	z_3	$b_7, b_{34}, b_{46}, b_{59}, s_6$	b_{13}
4	$s_{50}=0, s_{68}=1$	z_4	$b_8, b_{35}, b_{47}, b_{60}, s_7$	b_{14}
5	$s_{51}=0, s_{69}=1$	z_5	$b_9, b_{36}, b_{48}, b_{61}, s_8$	b_{15}
6	$s_{52}=0, s_{70}=1$	z_6	$b_{37}, b_{49}, b_{62}, s_9, s_{31}$	b_{16}
7	$s_{53}=0, s_{71}=1$	z_7	$b_{38}, b_{50}, b_{63}, s_{10}, s_{32}$	b_{17}
8	$s_{54}=0, s_{72}=1$	z_8	$b_{39}, b_{51}, b_{64}, s_{11}, s_{33}$	b_{18}
9	$s_{55}=0, s_{73}=1$	z_9	$b_{40}, b_{52}, b_{65}, s_{12}, s_{34}$	b_{19}
10	$s_{56}=0, s_{74}=1$	z_{10}	$b_{41}, b_{53}, b_{66}, s_{13}, s_{35}$	b_{20}
11	$s_{57}=0, s_{75}=1$	z_{11}	$b_{42}, b_{54}, b_{67}, s_{14}, s_{36}$	b_{21}
12	$s_{58}=0, s_{76}=1$	z_{12}	$b_{55}, b_{68}, s_{15}, s_{37}$	b_{22}
13	$s_{59}=0, s_{77}=1$	z_{13}	b_{69}, s_{17}, s_{39}	b_{23}
14	$s_{60}=0, s_{78}=1$	z_{14}	b_{70}, s_{17}, s_{39}	b_{24}
15	$s_{61}=0, s_{79}=1$	z_{15}	b_{71}, s_{18}, s_{40}	b_{25}
16	$s_{22}=1, s_{44}=0, s_{65}=1$	z_{19}	b_{75}	b_{29}
17	$s_{23}=1, s_{45}=0, s_{66}=1$	z_{20}	b_{76}	b_{30}
18	$s_{24}=1, s_{46}=0, s_{67}=1$	z_{21}	-	b_{77}
19	$s_{19}=s_{20}=s_{28}=1$	$z_{16}=0$	b_{72}, b_{73}	s_0
	$s_{41}=s_{42}=s_{50}=0$	$z_{17}=0$		
	$s_{62}=s_{63}=s_{71}=1$	$z_{25}=0$		
20	$s_{20}=s_{21}=s_{29}=1$	$z_{17}=0$	b_{74}	s_1
	$s_{42}=s_{43}=s_{51}=0$	$z_{18}=0$		
	$s_{63}=s_{64}=s_{72}=1$	$z_{26}=0$		
21	$s_{30}=1, s_{52}=0, s_{73}=1$	z_{27}	-	b_{28}
22	$s_{21}=1, s_{43}=0, s_{64}=1$	z_{18}	-	s_2
23	$s_{29}=1, s_{51}=0, s_{72}=1$	z_{26}	-	b_{27}
24	$s_{28}=1, s_{50}=0, s_{71}=1$	z_{25}	-	b_{26}
25	$s_{25}=1, s_{47}=0, s_{68}=1$	z_{22}	-	b_{78}
26	$s_{26}=1, s_{48}=0, s_{69}=1$	z_{23}	-	b_{79}
27	$s_{27}=1, s_{49}=0, s_{70}=1$	z_{24}	-	b_0

the number of occurrences. In Figure 2, Y-axis represents how much states are in group with same occurrences number. States in every group have same repetition score. X-axis represents repetition number by one group.

Next experiment analyses number and length of sub-chains that appear in the other chains. Analysis are shown that if there is a state in matrix in row i and column j that appears in some other row, for example l in column k has the consequence that the rest of rows i and l have same sub-chains of states. The justification for this is a deterministic procedure for obtaining keystream from Grain-v1. In this analysis, all sub-chains with more than one appearance were grouped.

In Figure 3 is represented number and length for each sub-chains that appeared more than one time in matrix. On graph, Y-axis represents how much time sub-chains with same length were occurred in matrix. Length of sub-chains, which is unique for one group, is denoted on X-axis. For example, there are 37 sub-chains with length 804 states. Group with the longest sub-chains had 22 sub-chains length 1022.

Last analysis for this experiment was to analyze states in same row in matrix. It was tested if there are two and more equals states in same chains. Results shown that it happened to construct chain with periodically repetition sub-chains length smaller than t. In Figure 4 is described frequency of this sub-chains. On X-axis is indicated length of sub-chain in one row. On Y-axis is denoted serial number of sub-chain. From Figure 4, can be seen that these sub-chains were 12 in total. Length of sub-chains was in range from 67 to 1004 states.

5. CODING ISSUES OF SECURITY ENHANCEMENT

This section provides experimental evaluation of certain codes which could be employed for security enhancement of Grain-v1.

5.1. Codes for Binary Channels with Erasures

The considered codes belong to the family of Low Density Parity Check (LDPC) codes. LDPC software package used in experiments consists of two components: the simulator itself written in C and a parity check generator written in Octave/Matlab. The software is a modification of the software for Similac LDPC decoding the IEEE 802.11n available at [9]. Simulator used to conduct LDPC encoding was adapted from [10]. This specific implementation was chosen because of its modularity and extensibility. In order to adjust original simulator for our purposes, two additional modules were implemented: Deterministic Binary Erasure Channel (DBEC) and QC-LDPC importer (make_QCLDPC).

Simulation of QC-LDPC coded channel consists of following steps:

1) LDPC code generation,
2) encoding,
3) channel transmission,

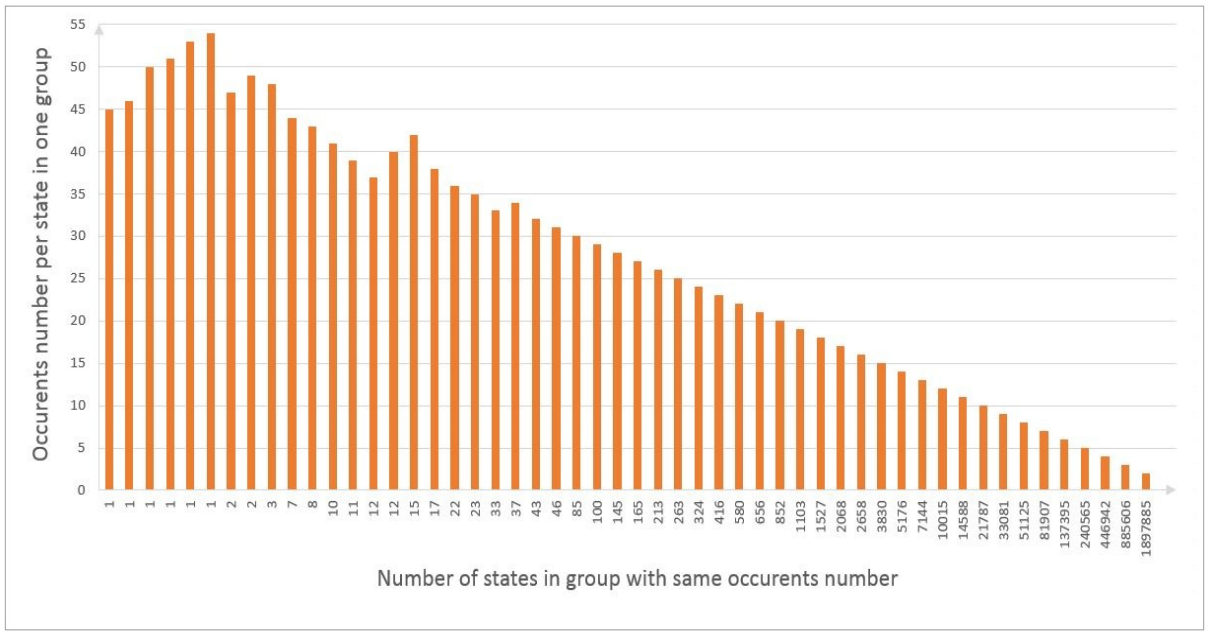

Figure 2: Classification states in groups with same occurrences number

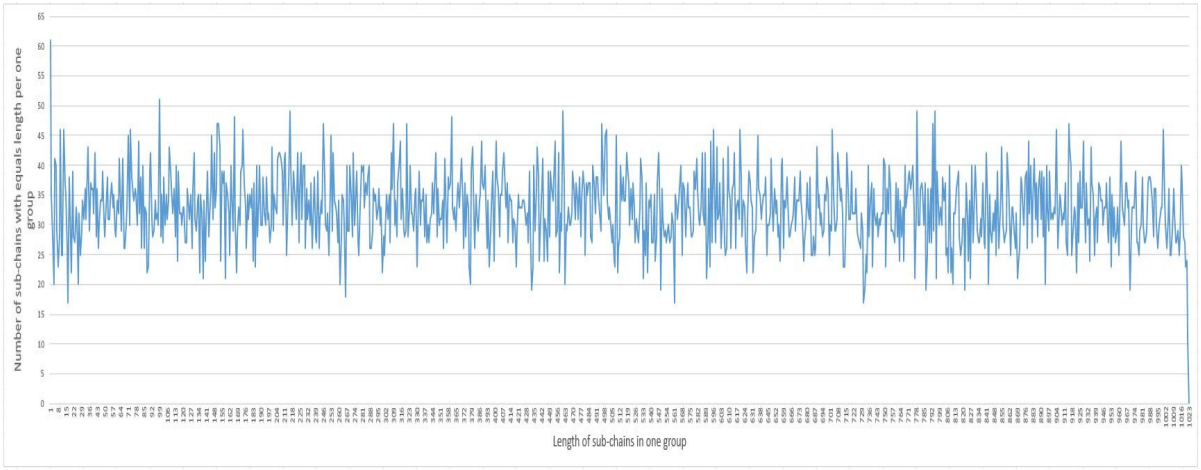

Figure 3: Classification sub-chains in groups with same length

4) decoding and
5) verification.

During the LDPC [11] code generation step parity check and generator matrices are generated based on input parameters and imported. In order to use this simulator for Grain-LDPC-encryption, modul for importing the reconstructed QC-LDPC codes had to be implemented since original package can be used only for generating only plain (non quasi-cyclic) LDPC codes. In order to go through encoding step, binary source message is needed, as well as generator matrix generated in previous step. In our experiments, input source message is obtained as an output from the Grain-v1 algorithm. Channel transmission step is used to corrupt encoded message in order to simulate erroneous channel transmission. Original package contains several erasure channel implementations, including Binary Symmetric, AWGN and AWLN channels. In our experiments, encoded message is not corrupted in order to model realistic communications channel, but to impose certain statistical properties to encrypted data. Therefore, we implemented Deterministic Binary Erasure Channel. This module behaves like an ordinary Binary Erasure Channel but with one distinction: bits are erased deterministically, based on the provided input erasure sequence. This, input to this module consists of encoded message and erasure sequence. This erasure sequence

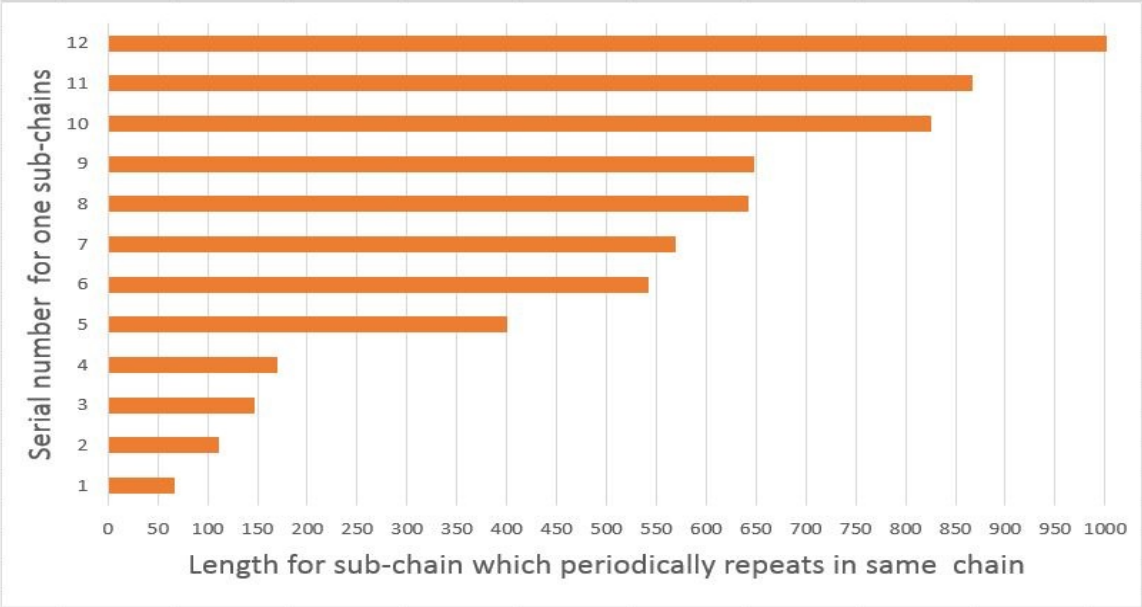

Figure 4: Length for periodically repeated sub-chains

is also prodded as an output from the Grain-v1 algorithm. Decoding step is done using the belief propagation algorithm. Input arguments to this module include transmitted message (generated in previous step), parity check matrix (generated in first step) and number of iterations. Verification step is only conducted when designing the QC-LDPC code. This step ensures that input message can be successfully restored after erasing bits in step 3. This step can be used to estimate the maximum erasure rate that can be achieved using the given code.

5.2. Tests

In this experiment is examined whether the codes are suitable for cryptographic usage. Parameters for generating the different QC-LDPC codes used in experiments are shown in Table 2. Generator matrices are represented as bitmaps in Figure 5-7. White pixels represented zero bits and bits with value one are denoted like black pixels. Goal of this experiment is to check if coded message is pseudo-random sequence and suitable for cryptographic usage. It is clear that results of experiments depend of parameters and contents generator and parity check matrices. For this experiment, set of statistical tests is used to prove that the generator is, or not, suitable for use in cryptography. Set of statistical tests is repeated on three binary streams coded with three different codes.

For detecting deviations from randomness of binary sequences, National Institute of Standards and Technology (NIST) uses a Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications [12].

The NIST Test Suite is a statistical package which contains 15 different tests that tested the randomness of binary sequences produced by cryptographic random or pseudorandom number generators. In our case, it was tested randomness of output from Grain-v1 and LDPC codes. These tests focus of a variety of different types of non-randomness that could exist in a sequence. The mentioned tests are:

1) The Frequency (Mono-bit) Test,
2) Frequency Test within a Block,
3) The Runs Test,
4) Tests for the Longest-Run-of-Ones in a Block,
5) The Binary Matrix Rank Test,
6) The Discrete Fourier Transform (Spectral) Test,
7) The Non-overlapping Template Matching Test,
8) The Overlapping Template Matching Test,
9) Maurer's "Universal Statistical" Test,
10) The Linear Complexity Test,
11) The Serial Test,
12) The Approximate Entropy Test,
13) The Cumulative Sums (Cu-sums) Test,
14) The Random Excursions Test and
15) The Random Excursions Variant Test.

The focus of The Frequency (Mono-bit) Test is to calculate the proportion of zeros and ones for the entire sequence. The purpose of this test is to determine whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. All other subsequent tests depend on the passing of this test. This test has for arguments the length of the bit-word n and the sequence of bits ϵ. For this study, n had value 10^6.

TABLE 2: Parameters for generating codes

Name	Width of matrix	Rate	Length of code-word
LDPC_2040_R18	2040	1/8	255
LDPC_4080_R116	4080	1/16	255
LDPC_972_R16	972	1/6	162

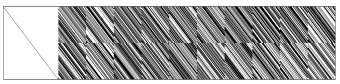

Figure 5: Generator matrix for code LDPC_972_R16

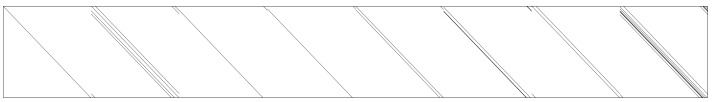

Figure 6: Generator matrix for code LDPC_2040_R18

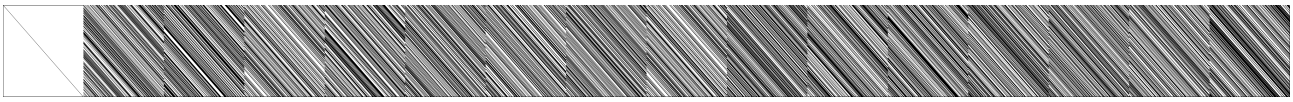

Figure 7: Generator matrix for code LDPC_4080_R116

TABLE 3: NIST tests results

STATISTICAL TEST	LDPC972R16		LDPC2040R18		LDPC4080R116	
	P-VALUE	PROPORTION	P-VALUE	PROPORTION	P-VALUE	PROPORTION
Frequency	0.921624	989/1000	0.000000*	972/1000*	0.866097	989/1000
BlockFrequency	0.834308	994/1000	0.000000*	964/1000*	0.471146	990/1000
CumulativeSums	0.647012	988/1000	0.000000*	970/1000*	0.595654	988/1000
Runs	0.125200	987/1000	0.000000*	966/1000*	0.616305	989/1000
LongestRun	0.217857	988/1000	0.336111	985/1000	0.128874	989/1000
Rank	0.733899	989/1000	0.316052	985/1000	0.084037	991/1000
FFT	0.955835	989/1000	0.000000*	2/1000*	0.859637	993/1000
NonOverlappingTemplate	0.518231	990/1000	0.000390*	979/1000*	0.511070	990/1000
OverlappingTemplate	0.620465	987/1000	0.000000*	984/1000*	0.245491	990/1000
Universal	0.024688	991/1000	0.000000*	4/1000*	0.703417	985/1000
ApproximateEntropy	0.779188	987/1000	0.000000*	963/1000*	0.469232	990/100
RandomExcursions	0.640932	599/605	0.643301	574/579	0.501618	637/643
RandomExcursionsVariant	0.312905	600/605	0.147909	575/579	0.516815	636/643
Serial	0.739424	987/1000	0.000040*	985/1000*	0.541842	989/1000
LinearComplexity	0.132640	990/1000	0.971006	989	0.522112	984/1000

The focus of Frequency Test within a Block is to determine proportion of ones within M-bit blocks. If the frequency of ones in an M-bit block is approximately $M/2$, conclusion is that entered sequence is random. Function call contains the length of the bit-word n and the length of each block, M.

Runs test has as a goal to determine a total number of an uninterrupted sequences of identical bits, called runs. In particular, this test determines whether the oscillation between such zeros and ones is too fast or too slow. Arguments for call of this test is the same like for Frequency test.

The aim of Test for the Longest Run of Ones in a Block is the longest run of ones within M-bit blocks. The purpose of this test is to determine whether the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence. Arguments for the function that tests described are length of the bit-word n, the number of blocks N and length of each block M. In this study, $M = 128$ and $N = 1000$ was used.

The focus of Binary Matrix Rank Test is the rank of disjoint sub-matrices of the entire sequence. The purpose of this test is to check for linear dependence among fixed length substrings of the original sequence. This test uses, as parameter n, M and the number of columns in each matrix Q set at 32.

The focus of Non-overlapping Template Matching Test is the number of occurrences of pre-specified target strings. The purpose of this test

is to detect generators that produce too many occurrences of a given non-periodic (aperiodic) pattern. For this test m-bit window is used to search a specific m-bit pattern. The length in bits of each template in experiments for this study had value $m = 9$.

The focus of the Overlapping Template Matching test is the number of occurrences of pre-specified target strings. Like previous test, this uses a m-bit window to search a specific m-bit pattern. Block length for this test is $m = 9$, also. Function call has as arguments the length of the bit-word n, sequence for testing ϵ and the length in bits of the template m.

The focus of Maurer's "Universal Statistical" Test is the number of bits between matching patterns (a measure that is related to the length of a compressed sequence). The purpose of the test is to detect whether or not the sequence can be significantly compressed without loss of information. A significantly compressible sequence is considered to be non-random. For Universal test it is necessary to know the length of each block L, the number of blocks in the initialization sequence Q, the length of the bit-word n and sequence of bits ϵ.

The aim of Linear Complexity Test is to determine whether or not the sequence is complex enough to be considered random. For this test, it needs to be known the value of n and the length in bits of a block M. For this study, $M = 500$ as recommended in specification of the test.

The focus of Serial test is to determine the frequency of all possible overlapping m-bit patterns across the sequence. The purpose of this test is to determine whether the number of occurrences of the 2^m m-bit overlapping patterns is approximately the same as would be expected for a random sequence. Random sequences have uniformly distribution. This test needs the length of bit-word n and the length in bits of each block m. Last parameter initialized in value 16.

The purpose of Approximate Entropy Test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and m+1) against the expected result for a random sequence. For function call is needed to know the length of the entire bit-sequence n and the length of each block m. For this experiment, it was set $m = 10$.

The focus of Cumulative Sums (Cu-sum) Test is to calculate the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the sequence. The purpose of the test is to determine whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior of that cumulative sum for random sequences.

The aim of Random Excursions Test is the number of cycles having exactly K visits in a cumulative sum random walk. The cumulative sum random walk is derived from partial sums after the (0,1) sequence is transferred to the appropriate (-1, +1) sequence. A cycle of a random walk consists of a sequence of steps of unit length taken at random that begin at and return to the origin. The purpose of this test is to determine if the number of visits to a particular state within a cycle deviates from what one would expect for a random sequence. Function call has only the length of the bit-word n and test sequence.

The focus of Random Excursions Variant Test is the total number of times that a particular state is visited (i.e. occurs) in a cumulative sum random walk. The purpose of this test is to detect deviations from the expected number of visits to various states in the random walk

For many of the tests in this test suite, the assumption has been made that the size of the sequence length, n is large (from the order 10^3 to 10^7). For each of tests in test suite, was used sequences whose length is 10^6. In every test 1000 sequences the same length are used.

After testing outputs from codes, for every test it was computed P-values. On the basis of P-value decision is made. If each P-value $< \alpha$, where α presents the significance level, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random. In our experiments α is set on 0.01.

The results obtained by testing outlined codes are given in Table 3. With $*$ are denoted unsuccessful tests. P-value is used for make decision if statistical test has hypothesis "Sequence is random" or alternative one. Column which has name "PROPORTION" denotes number of subsequences which passed test successfully. Total number of tested sub-sequences in all experiments was 1000.

After testing three different codes conclusion is that LDPC_972_R16 and LDPC_4080_R116 is suitable for cryptographic usage and LDPC_2040_R18 is not. Reason for that is structure of generator matrices for all three codes. From Figures 5-7 it can be clearly concluded that if matrix has small number of bits with value one, message coded by correspond code is not suitable for cryptographic usage.

6. CONCLUSION

This paper provides additional insights regarding certain Grain-v1 security evaluation and security enhancement approaches. The paper provides: (i) experimental evaluation of the time-memory trade-off paradigm employed for cryptanalysis in [5]: and (ii) experimental evaluation of statistical features of Grain-v1 output sequences which are subject of error-correction encoding and degradation by a channel with bit-deletions. Main messages of this paper regarding the above two issues are the following ones.

The reported power of the cryptanalysis in [5] is overestimated because it does not take into

account the repetitions into the table employed for the time-memory data trade-off based cryptanalysis. Our experimental evaluation implies that the claimed power of cryptanalysis is overestimated at least for a factor equal to two.

Our experimental evaluation of the statistical features of error-correction encoded Grain-v1 segments after a binary channel with bit deletions shows that the addressed statistics strongly depend on the selection of the error-correction code. We show that certain codes, although almost same from the error-correction prospective imply different statistical features. Accordingly, if employed for the security enhancement, the code should be evaluated regarding its impact on the statistical features, and the ones which does support indistinguishably from the statistically random sequences should be selected.

References

[1] M. Hell, T. Johansson, and W. Meier, "Grain - a stream cipher for constrained environments," *International Journal of Wireless and Mobile Computing*, pp. 86 – 93, 2007.

[2] M. Hell, T. Johansson, A. Maximov, and W. Meier, "The grain family of stream ciphers," *in M. Robshaw, O. Billet, (Eds.) New Stream Cipher Designsm, LNCS, Springer*, vol. 4986, pp. 179 – 190, 2008.

[3] T. Bjorstad, "Cryptanalysis of grain using time/memory/data tradeoffs (2008). http://www.ecrypt.eu.org/stream/grainp3.htm,"

[4] M. J. Mihaljevic, S. Gangopadhyay, G. Paul, and H. Imai, "Internal state recovery of grain-v1 employing normality order of the filter function," *IET Information Security*, vol. 6, pp. 55 – 64, June 2012.

[5] L. Jiao, B. Zhang, and M. Wang, "Two generic methods of analyzing stream ciphers," *Springer International Publishing Switzerland*, pp. 379 – 396, 2015.

[6] A. Kavcic, M. J. Mihaljevic, and K. Matsuura, "Lightweight secrecy system using channels with insertion errors: Cryptographic implications," *IEEE Information Theory Workshop, Jeju Island, Korea*, pp. 257 – 261, October 2015.

[7] "The ecrypt stream cipher project. eSTREAM portfolio revision (2008) http://www.ecrypt.eu.org/stream."

[8] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers," *6th International Conference on the Theory and Application of Cryptology and Information Security Kyoto, Japan*, vol. 1967, pp. 1 – 13, December 2010.

[9] www.csl.cornell.edu/ studer/software_ldpc.html

[10] www.cs.utoronto.ca/ radford/ldpc.software.html

[11] L. Chen, I. Djurdjevic, and S. Lin, "Near-shannon-limit quasicyclic low-density parity-check codes," *IEEE Transaction Communication*, vol. 52, pp. 1038 – 1042, April 2004.

[12] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," *National Institute of Standards and Technology*, April 2010.