Cipher Modification Against Steganalysis Based on NIST Tests

Valery Korzhik, Nguyen Duy Cuong
The Bonch-Bruevich Saint-Petersburg State University of Telecommunications
Saint-Petersburg, Russia
val-korzhik@yandex.ru, cuong0111@gmail.com

Guillermo Morales-Luna
Computer Science, CINVESTAV-IPN
Mexico City, Mexico
gmorales@cs.cinvestav.mx

***Abstract*—Part of our team proposed a new steganalytic method based on NIST tests at MMM-ACNS 2017 [1], and it was encouraged to investigate some cipher modifications to prevent such types of steganalysis. In the current paper, we propose one cipher modification based on decompression by arithmetic source compression coding. The experiment shows that the current proposed method allows to protect stegosystems against steganalysis based on NIST tests, while security of the encrypted embedded messages is kept. Protection of contemporary image steganography based on edge detection and modified LSB against NIST tests steganalysis is also presented.**

I. Introduction

A new steganalytic algorithm (SGA) based on the use of NIST tests [2] has been proposed in [1]. It was assumed in that setting that the messages prior their embedding into cover object (CO) are encrypted by some strong cipher. The reasons of prior encryption were obvious. Firstly, ciphering protects the embedded message against its reading by attackers in case the embedding was somewhat detected and the stegokey determining the pseudo-random walk through the CO, where the message bits were embedded, is also known. Secondly, if the stegokey is known, the plaintext can be easily extracted from the stegotext. Hence an attacker could be able to decide that the tested object is a stego-object (SG), if the extracted "message" is meaningful, or to reject it as a CO. (It is worth to note that in [3] another method to find stegokeys using NIST tests was also presented.)

The following steganalytic algorithm (SGA) based on the use of NIST tests was proposed in [1]:

1) Extract the sequence of "message" bits from the tested object following the known algorithm.
2) Apply the NIST tests to the extracted sequence of bits.
3) If all NIST tests are satisfied then decide that the tested object is a SG, otherwise decide that it is a CO.

The reason of such procedure is the following: it is unlikely that a "clear" CO satisfies all pseudo random tests. On the other hand, it is well known that the ciphertext obtained by strong ciphers should satisfy the pseudo random properties. Therefore, in [1] a paradox-looking motto: "strong ciphers compromise stegosystems" was formulated.

A generalization of this procedure consists in comparing the number of passed tests with some chosen prior threshold.

Another approach of SGA presented also in [1] was to use Support Vector Machine (SVM) for SG detection. New SGA was applied in [1] to detect LSB based embedding (both replacing and matching (±1 LSB)) and for matrix embedding with Hamming codes [4]. Efficiency of this SGA cannot be characterized as excellent because it is inferior to some SGA known before, but its advantage is that it can be used for any SG systems with known stegokey and it is simple enough for implementation.

However, NIST tests-based SGA can be prevented by *cipher modification*. This means that the cipher used prior to embedding should be modified in such a way to be robust against breaking and simultaneously the ciphertext should pass most of NIST tests. One such cipher modification to solve this problem is presented in the Section 2. In the Section 3 we consider image SG based on edge detection and we show that the proposed method is sufficiently effective against NIST-based SGA. Section 4 concludes the paper.

II. Robust cipher modification method

Many methods to modify the block cipher aiming to robustness can be proposed, but a most natural one is a decompression algorithm using error free source compression code. We propose to use a well-known class of arithmetic coding (AC) [5]. This code family is error free, i.e. after a compression/decompression procedures the information is recovered without errors. Before applying the decompression procedure, it is necessary to set for it the probabilities of symbols. Obviously that decompression results in a stretching of sequences while compression results in reducing of sequences.

The embedding procedure consists of three steps:

1) Encryption of the messages by any strong cipher like 3DES, GOST, AES, etc [6], [7].
2) Decompression of the encrypted binary sequence with AC given some probabilities.
3) Embedding of the decompressed sequence into CO according to a given SG algorithm.

ISSN 2305-7254

In order to obtain the plaintext given the SG it is necessary to perform also three steps:

1) Extract the embedded bits following to SG algorithm.
2) Compress the ciphertext sequence with known AC probabilities.
3) Decrypt the sequence obtained in point 2 with known cryptokey and decryption algorithm.

If an attacker does not know which embedding method with the cipher modification was used in the object, then he/she faces the problem that the extracted sequence does not pass already through NIST tests. If an attacker knows about possible use of cipher modification, then he/she may try to compress initially the extracted sequence and only after that to apply NIST testing. In that case, the attacker must know the probabilities of AC compression/decompression that play here the role of stegokey.

Let us try to investigate the proposed method above by starting with the simplest case of two probabilities for symbols 0 and 1, P(0) and P(1) = 1 – P(0), respectively.

In Table I a list consisting of the standard 15 NIST tests is presented (a detailed description of these tests is in [2]).

Table II shows the NIST testing for different sequences encrypted by AES-128.

In Table III the pass rates for the same conditions as in Table II are presented, but they are calculated over the statistics of 1000 sequences for each test.

Table IV shows NIST testing for 15 different image covers.

By comparing Table II and Table IV, it is concluded that it is very easy to distinguish SG-LSB from image covers using the NIST tests.

Let us consider cipher modification where the encrypted sequence is decompressed by arithmetic coding with the parameters P(0) = 0.49; P(1) = 0.51. The results of NIST testing after cipher modification are presented in Table V.

In Table VI the pass rates are presented with conditions of Table V using 1000 sequences for each test.

TABLE I. TITLES OF NIST TESTS ON PSEUDO-RANDOMNESS

N	Title of test
1	The frequency 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 (cusums) test
14	The random excursion teat
15	The random excursions variant test

TABLE II. DEMONSTRATION OF NIST TESTING FOR 15 SEQUENCES ENCRYPTED BY CIPHER AES-128. (GREY COLOR MEANS THAT TEST HAS PASSED; WHITE COLOR MEANS THAT CORRESPONDING TEST HAS NOT PASSED)

Sequence / Test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	■	■	■	■	■	■	■	■	■	■	■	□	■	■	■
2	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
3	■	■	■	■	■	■	■	■	■	■	■	■	□	■	■
4	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
5	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
6	■	■	■	■	■	■	■	■	■	■	□	■	■	■	■
7	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
8	■	■	■	■	□	■	■	■	■	■	■	■	■	■	■
9	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
10	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
11	■	□	■	■	■	■	■	■	■	■	■	■	■	■	■
12	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
13	■	■	■	■	■	■	■	■	■	■	■	□	■	■	■
14	■	■	□	□	■	□	■	■	■	■	■	■	■	■	■
15	■	■	□	□	■	■	■	■	■	■	■	■	■	■	■

TABLE III. PASS RATES (IN PERCENT) OBTAINED FOR STATISTIC CONSTITUTING OF 1000 SEQUENCES FOR EACH TEST

NIST test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pass rate (%)	99.1	98.9	99.1	99.0	98.7	99.4	99.8	98.8	99.1	99.1	98.3	98.8	99.0	69.7	69.1

TABLE IV. DEMONSTRATION OF NIST TESTING FOR 15 DIFFERENT IMAGE COVERS

Sequence / Test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2															
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															

TABLE V. RESULTS OF NIST TESTING AFTER CIPHER MODIFICATION BY DECOMPRESSION WITH AC FOR PARAMETERS *P(0) = 0.49; P(1) = 0.51*

Sequence / Test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2															
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															

TABLE VI. RESULTS OF PASS RATES CALCULATION FOR CIPHER MODIFICATION

NIST test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pass rate (%)	0	0	0	51.8	98.9	96.7	0	0.1	86.0	99.3	1.3	0	0	0	0

By comparing Tables II, V and taking into account Tables III and VI, it can be seen that the cipher modification using AC decompression results in the impossibility to detect stegosystems with NIST testing. (It is worth to note that in Table V, the 5th and 10th tests are passed with high reliabilities but the same property is valid for covers (see Table IV). Therefore, this fact cannot be used for SG detection).

Nevertheless, an attacker can apply a prior compression of the extracted sequence if the parameters P(0) and P(1) of AC are known. However, if they are unknown, then the attacker may try all possible values close to 0.5.

In Table VII the results of NIST testing after compression with different probabilities P(0), P(1) are presented if decompression was performed by legitimate user with fixed probabilities P(0) = 0.49; P(1) = 0.51.

From Table VII it can be seen that the exact knowledge of the secret parameters P(0), P(1) is not necessary to provide a correct detection of SG with NIST testing. Therefore, it is not a problem to try all possible keys close to 0.5 in order to get the described result.

Thus, it is necessary to select a more complex secret key than one parameter of AC. We propose to take as such key the probabilities of the i-th symbol $P_i(0), i = 1,2,..,N$, where N is the length of decompressed sequence.

In order to simplify the decompression/compression procedures, we suggest to fix $N = 50$ symbol probabilities and to repeat such probabilities periodically up to the end of decompressed sequence. Each probability $P_i(0), i = 1,2,..,N$ has to be selected as i.i.d and uniformly distributed value on the interval $(0.4 \div 0.6)$.

In Table VIII the pass rates are presented for different tests and for 10 randomly selected symbol probabilities.

The results of Table VIII allows to conclude that after cipher modification an attacker is already unable to detect SG using NIST tests.

TABLE VII. THE RESULTS OF PASS RATES FOR NIST TESTING AFTER ATTACKER'S COMPRESSION BY AC WITH DIFFERENT PROBABILITIES *P(0), P(1)*

NIST test			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(0)	P(1)																
0.40	0.60	Pass rate (%)	0	48	0	93	100	100	0	67	97	97	85	0	0	2	4
0.41	0.59	Pass rate (%)	0	77	0	97	99	99	8	95	97	99	91	16	0	10	11
0.42	0.58	Pass rate (%)	0	94	0	98	99	100	53	96	100	98	98	49	0	18	19
0.43	0.57	Pass rate (%)	0	89	0	99	98	98	66	94	100	99	97	63	0	19	17
0.44	0.56	Pass rate (%)	1	98	1	98	98	99	90	95	99	99	96	89	0	26	28
0.45	0.55	Pass rate (%)	2	98	24	98	99	98	98	98	99	96	97	95	1	41	42
0.46	0.54	Pass rate (%)	17	96	76	100	98	97	100	98	97	98	99	95	16	40	41
0.47	0.53	Pass rate (%)	52	98	92	97	99	99	100	98	99	98	98	99	53	52	57
0.48	0.52	Pass rate (%)	91	99	100	99	100	99	99	99	100	98	99	100	89	73	74
0.49	**0.51**	**Pass rate (%)**	**100**	**97**	**98**	**99**	**99**	**99**	**100**	**99**	**99**	**99**	**98**	**98**	**99**	**75**	**73**
0.51	0.49	Pass rate (%)	62	100	96	100	100	99	100	98	100	99	97	97	59	65	60
0.52	0.48	Pass rate (%)	13	97	71	99	99	98	100	90	100	98	96	97	15	40	37
0.53	0.47	Pass rate (%)	21	99	72	99	99	97	100	93	99	99	100	99	17	44	44
0.54	0.46	Pass rate (%)	0	97	4	93	100	99	97	73	100	99	99	95	0	25	24
0.55	0.45	Pass rate (%)	0	93	0	98	99	97	83	60	99	98	97	82	0	23	23
0.56	0.44	Pass rate (%)	0	99	0	90	99	99	59	32	97	100	96	76	0	16	17
0.57	0.43	Pass rate (%)	0	84	0	69	100	100	27	4	98	99	94	28	0	10	10
0.58	0.42	Pass rate (%)	0	60	0	18	99	98	1	0	100	97	82	0	0	11	11

TABLE VIII. THE PASS RATES FOR 10 RANDOMLY GENERATED SYMBOL PROBABILITIES $P_1(0), P_2(0),.., P_{50}(0)$ UNIFORMLY DISTRIBUTED ON INTERVAL 0.4 ÷ 0.6

Set of probabilities	NIST test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Set 1	Pass rate (%)	0	0	0	0	98.9	8.9	0	0	0	99.0	0	0	0	0	0
Set 2	Pass rate (%)	0	0	0	0	98.9	0.8	0	0	0	99.2	0	0	0	0	0
Set 3	Pass rate (%)	0	0	0	0	99.2	1.9	0	0	0	99.2	0	0	0	0	0
Set 4	Pass rate (%)	0	0	0	0	99.3	44.6	0	0	0	99.3	0	0	0	0	0
Set 5	Pass rate (%)	0	0	0	0	99.2	18.0	0	0	0	98.8	0	0	0	0	0
Set 6	Pass rate (%)	0	0	0	0	98.8	22.9	0	0	0	98.6	0	0	0	0	0
Set 7	Pass rate (%)	0	0	0	0	99.4	12.1	0	0	0	98.5	0	0	0	0	0
Set 8	Pass rate (%)	0	0	0	0	98.4	31.7	0	0	0	99.3	0	0	0	0	0
Set 9	Pass rate (%)	0	0	0	0	99.3	5.2	0	0	0	99.2	0	0	0	0	0
Set 10	Pass rate (%)	0	0	0	0	99.3	5.3	0	0	0	99.0	0	0	0	0	0

TABLE IX. THE PASS RATES FOR 10 DIFFERENT RANDOMLY CHOSEN INCORRECT KEYS, CORRECT KEY AND COMPRESSED BY RANDOMLY CHOSEN COVER

	NIST test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Set 1	Pass rate (%)	99.0	98.6	99.1	99.3	99.1	98.9	99.8	98.6	98.6	98.5	97.9	98.7	98.5	63.9	66.2
Set 2	Pass rate (%)	99.5	98.7	98.9	99.4	99.2	99.4	99.7	98.4	98.4	99.3	99.1	99.5	99.2	64.5	63.8
Set 3	Pass rate (%)	98.1	98.8	99.2	98.7	99.2	97.8	99.8	98.6	98.6	99.1	98.6	98.6	97.7	64.2	64.6
Set 4	Pass rate (%)	99.1	99.0	98.6	98.8	99.1	99.0	99.6	99.2	99.2	99.1	98.3	99.0	99.1	63.1	63.2
Set 5	Pass rate (%)	99.0	99.0	98.9	98.9	99.3	99.0	99.9	99.1	99.1	99.0	97.7	97.8	98.5	62.2	63.5
Set 6	Pass rate (%)	98.2	98.8	99.3	98.8	99.3	98.7	99.6	99.3	99.3	98.9	97.7	98.8	98.0	65.8	66.8
Set 7	Pass rate (%)	98.7	99.1	99.4	98.7	98.8	98.9	99.7	98.7	98.7	98.4	98.8	98.8	98.3	62.3	63.0
Set 8	Pass rate (%)	99.0	98.5	99.0	99.0	99.0	99.0	99.1	98.8	98.8	98.9	98.0	99.0	98.6	62.6	64.4
Set 9	Pass rate (%)	99.1	99.3	98.5	98.4	99.3	98.9	99.9	98.9	98.9	98.2	98.2	99.0	98.8	61.1	63.0
Set 10	Pass rate (%)	99.1	99.4	99.2	98.5	99.4	98.8	99.6	99.2	99.2	98.4	98.5	98.9	99.0	63.3	65.0
For covers	Pass rate (%)	79.6	77.7	78.3	80.2	89.8	89.0	85.2	79.9	79.9	96.1	73.0	74.9	77.5	47.4	46.3
For correct key	Pass rate (%)	99.1	98.9	99.1	99.0	98.7	99.4	99.8	98.8	99.1	99.1	98.3	98.8	99.0	69.7	69.1

TABLE X. THE EMBEDDING RATES WITH THE USE OF 10 STEGO KEYS OF CIPHER FOR A MODIFICATION TECHNIQUE

Number of set	1	2	3	4	5	6	7	8	9	10
The embedding rate *R*	0.9905	0.9885	0.9892	0.9932	0.9916	0.9920	0.9908	0.9926	0.9902	0.9901

Let us consider a more sophisticated attack when an attacker compresses initially the extracted sequence with randomly chosen symbol probabilities $\widetilde{P}_1(0), \widetilde{P}_2(0),.., \widetilde{P}_{50}(0),$ and after it applies NIST tests. The results of such attack for 10 different randomly chosen sets of symbol probabilities are presented in Table IX. In the same table are shown the results of testing for compression by correct decompression key and for covers with randomly chosen incorrect key.

We can see from this table that an attacker be unable to distinguish SG from covers by NIST testing.

It is obviously that decompression procedure results in a decreasing of the embedding rate, because it is a "generic property" of AC. Although the explicit embedding rate *R* after decompression by AC is sufficiently hard to estimate theoretical for given set of symbol probabilities $P_i(0), i = 1,2,..,50$ we can use the entropy bound [8]:

$$R \geq H(P(0) = 0.4; P(1) = 0.6) = 0.971 \quad (1)$$

However, this problem can be solved experimentally. For the set of legal sequence of probabilities $P_i(0), i = 1,2,..,50$ with its periodical repetition, we get that average decreasing of the embedding after decompression by AC can be estimated as 97.1%. This value is acceptable cost for protection against NIST tests-based detection.

In Table X are shown the value of embedding rates obtained for 10 randomly generated stego keys (the probabilities of the first 50 symbols as it was described above).

We can see that these rates are even greater that the bound (1).

III. Application of Cipher Modification and NIST Testing for a Detection of Edge-Based Steganography

We considered before the simplest steganography based on LSB embedding algorithm. In this section, the method of cipher modification will be implemented to more contemporary edge-based stegosystem.

Embedding and extracting procedure of such SG were described [8] in details. The embedding procedure taken from [8] is presented in Fig. 1.

The edge preserving mechanism tries to preserve edges unchanged after embedding. (Details of this module are given in [8]). Before applying an edge detection algorithm the p LSB of each CO are set to zero. Since in gray scale images, 8 bits typically represent the luminance of each pixel, then $q = 8 - p$ MSBs remains unchanged.

In embedding process, pixels are embedded by k - *bits* substitution. The value k is different: for edge pixels (k_e) against the smooth pixels (k_s), where $k_e > k_s$. The embedding formula is:

$$y' = y \oplus_k m \quad (2)$$

where: y is a pixel value in the cover image,

m is the secret message,

y' is the stego pixel value after embedding bits,

$\oplus_k$ denotes the substitution operator.

In order to improve the SG security authors of [7] applied well known modified LSB substitution method [4].

In our experiment we select the following parameters: $p = k_e + 1$; $k_e = 3$; $k_s = 2$. The strong cipher AES 128 was chosen for message encryption and embedding into 1000 images with size 512x512 which were taken from BOSSBase 1.01 [11]. (It is worth to nothing that the main idea to use edge-base method and modified LSB embedding is to increase the embedding rate keeping practically the same security as ordinary LSB-based SG. Authors of [8] claim that their scheme embeds in average higher than two bits per pixel (bpp) in contrast to conventional LSB-based SG with the embedding rate 1 bit/pixel.)

Because for the considered SG there is nothing secret stego key it is easy to extract the encrypted message and apply to them NIST testing procedure described in Section 2.

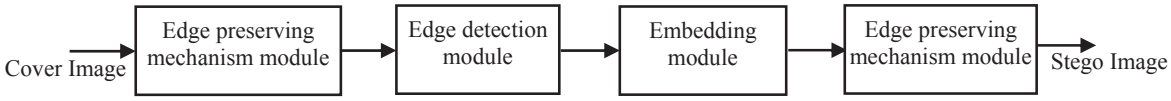

Fig. 1. Schematic view of the edge-based SG embedding

TABLE XII. Results of NIST testing for 15 different images extracted from edge-based SG

Sequence / Test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2															
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															

TABLE XIII. The pass rates on each tes0t for 1000 sequences taken from 1000 stego images

NIST test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pass rate (%)	99.5	98.7	98.8	98.7	98.9	99.1	99.9	99.1	98.7	98.8	98.4	98.2	99.1	72.8	73.8

TABLE XIV. RESULTS OF NIST TESTING FOR 15 DIFFERENT SEQUENCES EXTRACTING FROM COVER IMAGES

Sequence / Test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	■	■		■			■	■	■			■		■	
2						■									
3				■		■			■						
4									■					■	
5		■	■	■	■		■	■	■	■	■	■	■	■	
6	■	■	■		■	■	■	■		■	■	■		■	
7		■	■			■			■						
8						■			■						
9		■										■		■	
10	■	■	■	■	■	■	■	■		■	■	■	■	■	■
11						■			■			■			
12									■					■	
13	■	■		■		■	■		■			■		■	
14				■						■	■	■			
15	■		■	■			■			■	■	■		■	

TABLE XV. THE PASS RATES ON EACH OF 15 TESTS TAKEN FROM 1000 IMAGES IN SOURCE [11]

NIST test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pass rate (%)	49.2	19.1	21.0	18.4	86.2	70.1	21.5	20.0	26.8	94.3	22.8	19.0	39.3	39.1	43.4

In Table XII are presented the results of NIST testing for 15 different sequences extracted from edge-based SG.

We can see from this Table that almost all sequences have passed NIST tests.

In Table XIII are presented the pass rates for different NIST tests obtained for image base consisting of 1000 different images.

The results of Table XIII show that there are 13 of 15 tests in which their pass rates are greater than 98.4%.

In Table XIV are presented the results of NIST testing for cover images without any embedding.

We can see that this table contains more white colors cells than Table XII. This means that many sequences do not passe NIST tests. The results of pass rates for image covers taken from image source [11] are presented in Table XV.

It follows from Table XV that 9 from 15 tests have pass rates approximately 20%. This means that it is easy to separate edge-based SG from covers using such tests.

In fact, using a set of threshold it is possible to calculate the corresponding probabilities P_m (missing of SG detection) and P_{fa} (false alarm of SG detection), and the probability of error $P_e = (P_m + P_{fa})/2$, as presented in Table XVI.

Table XVI shows that the optimal threshold 12 (the number of passed tests) provides minimum value $P_e = 7.3\%$. One can use more effective method of SG detection based on nonlinear kernelized weighted SVM where the kernel function is Gaussian $K(x,x') = \exp(-\gamma\|x - x'\|^2)$ with parameter $\gamma > 0$ that is controlling the width of the kernel and $\|\cdot\|$ is the Euclidean norm of the in R^n [1]. The penalization coefficient C, which is called by box constraint, it used in order to provide a trade off between the probabilities P_m and P_{fa}. In Table XVII these probabilities are presented based on statistics consisting of 1000 images on training stage and different 1000 images on testing stage.

By comparing Table XVI and Table XVII, we can see that SVM technique is slightly better against threshold technique.

Let us consider also, as before, cipher modification. Such technique will execute decompression with AC of the encrypted messages before embedding.

As the parameters of AC were chosen the probabilities $P_1(0), P_2(0), .., P_{50}(0)$ of zeros for the first 50 symbols that are repeated periodically up to the end of all embedding symbols. These probabilities were truly random generated as i.i.d and uniformly distributed on interval (0.4÷0.6). (It is obvious that continuous values $P_1(0), P_2(0), .., P_{50}(0)$ can be quantized on small interval and transformed into bit string that form stego key for this SG).

The pass rates for different NIST tests after compression by AC with the same parameters are shown in the first row of Table XVIII. In the second row of Table XVIII are shown the pass rates for different NIST tests after compression by AC with the randomly generated by attacker parameter. In the third row of this table are shown the pass rates for cover images after their compression.

Table XVIII is similar to Table IX and hence an attacker be unable to distinguish between edge-base SG and cover images.

TABLE XVI. THE PROBABILITIES P_m, P_{fa}, AND P_e AGAINST CHOSEN THRESHOLD VALUES

Threshold	P_m (%)	P_{fa} (%)	P_e (%)
1	0.0	95.3	47.65
2	0.0	91.2	45.60
3	0.0	81.1	40.55
4	0.0	63.9	31.95
5	0.0	54.5	27.25
6	0.0	42.6	21.30
7	0.0	34.1	17.05
8	0.0	28.1	14.05
9	0.0	23.5	11.75
10	0.0	19.9	9.95
11	0.0	17.1	8.55
12	0.4	14.2	**7.30**
13	2.8	12.0	7.40
14	23.3	8.2	15.75
15	41.0	4.5	22.25

TABLE XVII. THE BEST PROBABILITY P_E, OBTAINED DUE TO SVM TECHNIQUE WITH OPTIMAL PARAMETERS γ AND C

	$\gamma = 8; C = 0.25$
P_{fa}(%)	3.5
P_m(%)	0.6
P_e(%)	**2.05**

TABLE XVIII. THE PASS RATES CORRESPONDING TO DIFFERENT NIST TESTS FOR ATTACKER WITH KNOWN AND UNKNOWN STEGO KEY EXECUTING FOR EDGE-BASED SG

Test / Condition	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Known stego key	99.5	98.7	98.8	98.7	98.9	99.1	99.9	99.1	98.7	98.8	98.4	98.2	99.1	72.8	73.8
Unknown stego key	98.7	99.1	99.2	98.8	99.3	98.1	100	98.7	98.8	98.9	98.4	98.9	98.4	74.2	74.6
For covers	74.2	66.3	73.6	78.7	89.6	89.7	86.7	71.0	74.0	94.3	57.4	61.1	72.1	57.6	58.0

IV. CONCLUSION

In the current paper was undertaken an attempt to prevent NIST tests-based SGA that was proposed in [1]. We proposed in the current paper a modification of strong cipher that could be used for encryption of messages prior their embedding into cover objects. Such cipher modification has to satisfy to two main conditions: cipher should keep strong to its breaking as well as it was before a modification but the encrypted sequence of bits should do not satisfy to all pseudo random properties after modification.

As an example of such cipher modification it was proposed a decompression procedure based on arithmetic coding. It is well known that compression/decompression by these codes is error free and therefore the embedded information can be extracted correctly. On the other hand, we can use some parameters of AC that provide "breaking of pseudo randomness" after decompression that allows excluding a possibility NIST tests-base attack on any SG.

Firstly was demonstrated that the simplest choice of parameter of AC as the probability of zeros results in good results - impossibility to execute NIST tests-base SGA. However, such cipher modification is not sufficient for a prevention of NIST tests-base attack because if this parameter is known (or somewhat found) by attacker, he (or she) be able to compress by AC with the same parameters and apply next the NIST tests. In order to prevent such sophisticated attack it is necessary to execute AC parameters as secret key that cannot be found by exhaustion procedure. Thus, we propose other set of the AC parameters: $P_i(0), i = 1,2,..,50$ - the probabilities of zeros that outputs AC on *i-th* position.

Experiment showed that such cipher modification allows a providing of two mentioned above requirements and excludes a finding of stegokey by exhaustive search.

We demonstrate that SGA using NIST tests is effective also against edge-base SG but it is useless after cipher modification with the use arithmetic coding with special stegokey.

It is worth to note that decompression of the embedded sequence results in some decreasing of the embedding rate but it was demonstrated in our paper that such reduction is negligible.

Of course, the proposed cipher modification method is only one among all possible methods satisfied to two mentioned above requirements. Eventually we motivate researchers to try to find other cipher modification methods having better properties.

In line with a remark of anonymous reviewer, it is worth to note the following:

- In fact, a proposal to use arithmetic codes (AC) in steganography (SG) is not new. So P. Salee [12] suggested to execute AC in order to agree better a statistic of covers with statistic of stego objects. Similar approach was used also in the

paper [13]. However, the aim of our approach was not any "agreement" but a distortion of a good pseudorandom property for ciphertext obtained prior embedding. Moreover, we apply also key-based AC for this purpose.

- We agree that a general idea to check a "randomness" (more precisely "pseudo randomness") for SG detection was executed before our paper, (see for the thing [14]). But the difference in our approach is: firstly, we use NIST tests (instead of single chi-squared test in [14] and secondly, our method works not only with LSB-based SG but with any SG if the extracted algorithm is known or may be easy found. So it may be recognized as "universal" one.

Nevertheless, we thank anonymous reviewer for useful remarks and enjoy by the fact that we contributed into actual direction of SG detection.

REFERENCES

[1] V. Korzhik, I. Fedyanin, A. Godlewski, and G. Morales-Luna, "Steganalysis Based on Statistical Properties of the Encrypted Messages", *In International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, LNCS 10446, Aug. 2017, pp. 288-298.

[2] L.E. Bassham III, A.L. Rukhin, J. Soto, J.R. Nechvatal, M.E. Smid, E.B. Barker, S.D. Leigh, M. Levenson, M. Vangel, D.L. Banks, N.A. Heckert, J.F. Dray, S. Vo, "Spp. 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications". Technical report, Gaithersburg, MD, United States, 2010.

[3] V. Korzhik, C. Nguyen, I. Fedyanin, and G. Morales-Luna, "Side Attacks on Stegosystems Executing Message Encryption Previous to Embedding", JIHMSP, in press.

[4] J. Fridrich, *Steganography in Digital Media: Principles, Algorithms and Applications*. Cambridge University Press, 2009.

[5] K. Sayood, *Introduction to data compression*. Morgan Kaufmann, 2012.

[6] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

[7] B. Schneier, "The GOST encryption algorithm". *DR DOBBS JOURNAL*, vol. 20(1), 1995, pp. 123-124.

[8] F. Afsari, E. Eslami, P. Eslami, "Interval-valued intuitionistic fuzzy generators: Application to edge detection", *Journal of Intelligent & Fuzzy Systems*, vol. 27(3), 2014, pp. 1309-1324.

[9] F. J. MacWilliams, N. J. A. Sloane, *The theory of error-correcting codes*. North-Holland, 1977.

[10] I. J. Kadhim, P. Premaratne, P. J. Vial, and B. Halloran, "Comprehensive Survey of Image Steganography: Techniques, Evaluations, and Trends in Future Research", Neurocomputing, in press.

[11] DDE Laboratory in the Binghamton University, BOSSBASE 1.01, Web: http://dde.binghamton.edu.

[12] P. Sallee, "Model-based steganography", *In International workshop on digital watermarking*, LNCS 2939, Oct. 2003, pp. 154-167.

[13] E. Yu. Eltysheva, A. N. Fionov, "Construction of stegosystem on the basis of raster images using least-significant bit statistics", *Vestnik SibGUTI*, vol. 1, 2009, pp. 67-84. (in Russian)

[14] I.V. Nechta, "A method of steganalysis of text data based on statistical analysis", *Vestnik SibGUTI*. vol. 3, 2011, pp. 27–34. (in Russian)