Design of Almostperfectly Secure Stegosystems for Video Files

Valery Korzhik, Ekaterina Gerling
korzhikvalery11@gmail.com, gerlingeu@gmail.com

Aleksei Zhuvikin
zhuvikin@gmail.com

***Abstract*—We propose a new approach to design of stegosystems without embedding of secret information directly into cover objects. However, it is not, strongly speaking, steganography without any embedding, but rather steganography with minimal modification. The cover objects are either video films or photo sessions. Algorithms of minimized embedding procedure and secret data extraction are presented. A number of conventional and "unusual" stegosystem detection attacks are considered, and methods to mitigate them are proposed. The complexity of cover objects transforms as well as "embedding" rates for video films are estimated through simulation. The use of error-correcting codes is also implicitly addressed.**

***Keywords*—steganography, hash function, error correcting codes, vulnerability to detection, embedding rate, video films.**

I. Introduction

It is well known that steganography (SG) plays a crucial role in providing information security in the digital world. In fact, although cryptography (CR) is the primary means of ensuring information security with strong guarantees–especially when considered in a broad sense, including authentication, digital signatures and cryptographic protocols–it does have limitations. As metaphorically described as a 'mongrel' in the book [1], CR has one major drawback—it reveals the presence of additional hidden information, which should often be concealed. To close this "gap" in information security, users are nudged to implement SG. The latter technique hides even a presence of an added information in some digital cover objects (CO), such as, printed text, data, still images, movies, source codes and so on. By the way, all SGs can be divided into two classes: either those with CO given in advance or those where CO is generated freely by the source originator of stegosystems.

In the current paper we are dealing with the design of SG from the first class. In this case CO should be looking as completely innocent one. For example, it can be a short video film with children, architecture of some beautiful city, or pictures of nature and wild animals during a travelling in Africa. At a single glance, the creation of steganograms requires making small changes—either in the spatial (pixel) domain for images or in the domain of certain pixel transforms, such as after Fourier or Walsh transforms. Of course, such changes must not be visually noticeable, and even after the application of various steganalytic methods, detecting the existence of an SG should remain difficult with a high probability.

Methods of embedding, extraction and detection of SGs are presented in the well-known book [2]. The simplest embedding method is based on replacing the least significant bits (LSB) either in the pixel domain or in the transformed areas with the bits of a secret information. This makes it impossible to detect the SG presence with the naked eyes, but it remains susceptible to reliable SG detection based on histogram analysis or on the most effective method known as "Sample Pair Analysis" [3]. To prevent the compromise of SG security by the aforementioned or similar methods, many more sophisticated methods of embedding have been proposed, as presented in the book [1] and in many other papers [4].

But, in contrast to CR, for which encryption and decryption algorithms exist that provides so-called *perfect cryptographic security*–where even for the best breaking of a cipher, the information leakage about encrypted messages, given the cryptogram but unknown key, is zero. We remark, however, that such ciphers necessarily require the key length to be equal to the length of the plaintext.

Unfortunately, similar perfect SG belonging to the class of given in advance CO are known only for scenarios when steganalysis can detect SG over the noisy channels [5], which are not very common in practice. It is worth noting that a new concept, *GAN*-steganography (*Generative Adversarial Network*), was proposed in recent times. In this framework a novel technique for hiding arbitrary binary data in images using generative adversarial networks which allows to optimize the perceptual quality of the images produced by such model was proposed. But unfortunately, such SGs are far from achieving perfect security, although provide an effective payload of around 4.4 secure bits per pixel [6,7]. It may seem like an *oxymoron,* but recently, so called *coverless stegosystems* that provide perfect detection security also have been proposed [8]. Let us briefly explain this approach, as it is very close to the one proposed in the current paper.

Imagine that a very large database of images is freely accessible. If you want to hide additional data with perfect security, we can encrypt it in advance and divide it into blocks (portions) of "*m*" bit length each. Next, you take a *cryptographic hash function* with a hash length equal to "*m"* and extract images randomly from the database one by one until the hash of a chosen image matches the first portion of the ciphertext. In a positive case, we keep such an image as the first part of the stegotext. By repeating comparison procedure for all ciphertext portions, you form the full stegotext and then transmit or store it. It is clear that such SG is perfectly secure,

but it has one significant defect–it looks highly suspicious if an innocent photo session between ordinary network users consists of randomly chosen images taken from a huge database. Moreover, it is SG system that hides a secret message not in a given in advance CO, but in a randomly chosen one.

By the way, in some publications [9] it was proposed to apply so-called *"Latent Dirichlet Allocation Algorithm"*, which allows for the automatic identification of content-similar objects in a database. But of course, it does not provide a very strong content correlation between the CO chosen from the file.

In the current paper we propose to eliminate these defects. The remainder of our paper is organized as follows:

- Section II describes the proposed SG system.
- Section III presents estimates for the required number of iterations, obtained both theoretically and through simulation.
- Section IV evaluates the efficiency of different detection attacks on the proposed SG.
- The last section briefly discusses the main results and outlines directions for the future research for almostperfectly secure SG.

II. Description of the Proposed Almostperfectly Secure Stegosystem

We select video files (or films) as COs for the proposed SG. The reason for this lies in the possibility of embedding more secret data than in still images. But as a future work (see our conclusion), we believe it is important to also consider photo (image) sessions. In Fig. 1, the block scheme of secret "embedding" is presented. More accurately, it would be correct to call it a *block-scheme of CO transformation* in such a way that hidden data can be correctly extracted. This is why we sometimes put the word "embed" in the quotation marks.

The secret data intended to be "embedded" should first be encoded by an error-correction code, then encrypted with a strong stream cipher using a secret key K known only to legitimate users, and finally divided into portions, each of "m" bit length. Note that the use of stream cipher, rather than a block cipher, is recommended in order to avoid error spreading after decryption, which is typical for block cipher [10]. After that, the frames of video file should be hashed one by one to obtain m-bit hashes for each frame. Next, the hashes of the frames and the portions of the ciphertext are compared to one another. In the case of an exact match, the processed frames are stored as a part of the stegotext (stegoimages). Otherwise, a procedure of truly random selection among the frame pixels is performed, followed by the inversion of their LSBs.

These transformations are repeated until a hash match occurs or until a predefined threshold for the number of iterations is exceeded. In the latter case, the iteration algorithm should be stopped, and the process moves on to the next frame.

Extraction of the hidden information is shown in Fig. 2.

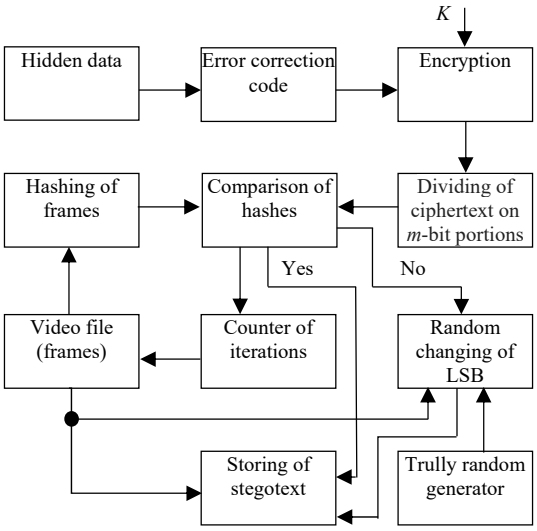

Fig. 1. Block scheme of secret data "embedding" into video file frames

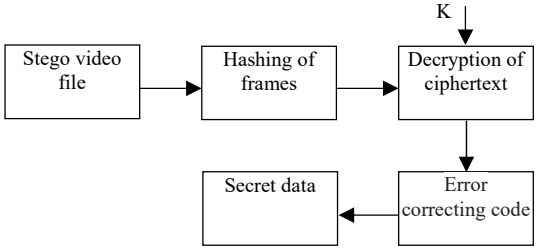

Fig. 2. Block scheme of secret data extraction.

In order to extract hidden data from the stego video file, it is necessary to first perform frame-by-frame hashing using the same hash function that was used for "embedding". Next, the extracted data should be decrypted using the secret decryption key K, known to legitimate users. Finally, any errors that may occur due to the absence of hash matches for some frames should be corrected using the chosen error-correcting code.

Let us consider some requirements for the parameters of the proposed SG. First, the number of hash comparisons (iterations) should not be too large. In fact, the number of iterations is upper bounded by the total number of pixels in each frame. Moreover, the number of iterations corresponds to the complexity of the "embedding" procedure and the required processing time for each frame.

The next restriction involves an upper bound on the probability of iteration process blocking–otherwise, the errors in frames cannot be reliably corrected. Both theoretical and simulated estimates of the required number of iterations are presented in the next section.

It is important to emphasize that in contrast to conventional LSB-based SGs, where there is a tradeoff between SG detection security and embedding rate, the proposed scheme has a tradeoff between embedding procedure complexity and embedding rate.

III. Estimates for the required number of iterations

Let us initially consider an idealized hash function that, after each iteration with LSB inversion of a truly randomly chosen pixel, produces any hash of length "m" with equal probabilities 2^{-m}.

Then, it is obvious that the probability of obtaining any given hash of length "m" after at most "l" iteration is equal to

$$P(m,l) = 1 - \left(\frac{2^m - 1}{2^m}\right)^l. \quad (1)$$

In Table I the results of calculation by (1) for different parameters "m" and "l" are given.

TABLE I. The probability of hash coincidence at "l" iterations and hash length "m" for idealized hash function

l \ m	1	5	10	20	30	40	50
1	0.500	0.031	0.001	10^{-6}	10^{-9}	10^{-12}	10^{-15}
5	0.969	0.147	0.005	$5x10^{-6}$	$5x10^{-9}$	$5x10^{-12}$	$5x10^{-15}$
10	0.999	0.272	0.009	10^{-5}	10^{-8}	10^{-11}	10^{-14}
50	1	0.796	0.048	$5x10^{-5}$	$5x10^{-8}$	$5x10^{-11}$	$5x10^{-14}$
100	1	0.958	0.093	10^{-4}	10^{-7}	10^{-10}	10^{-13}
1000	1	1	0.624	0.001	10^{-6}	10^{-9}	10^{-12}
5000	1	1	0.992	0.005	$5x10^{-6}$	$5x10^{-9}$	$5x10^{-12}$
10000	1	1	0.999	0.009	10^{-5}	10^{-8}	10^{-11}

Since the proposed scheme was implemented using the real keyless hash function CRC32, in Table II we present the same probabilities as in Table 1, but for real hash function, obtained through simulation.

TABLE II. The probabilities similar to ones presently in Table 1, but obtained through simulation for hash function CRC32.

l \ m	1	5	10
1	0.500	0.031	0.001
5	0.970	0.125	0.005
10	0.999	0.250	0.010
50	1	0.781	0.045
100	1	0.969	0.090
1000	1	1	0.603
5000	1	1	0.984
10000	1	1	1

The simulation environment consisted of an 11th Gen Intel(R) Core(TM) i7-1165G7 CPU @ 2.80 GHz, 16 GB RAM, running Windows 10 (version 22H2), with Visual Studio 2022 and Python.

Comparing the results presented in Tables I and II, we can see that they do not differ significantly from one another. Therefore, it is possible to select the parameters of the proposed SG system as follows: $m = 10$, $l_0 = 10000$. However, such computational complexity excludes SG formation in a real-time. This means that, for the chosen parameters, the probability of iteration procedure blocking, which in turn is equivalent to an error in the processed frame, is approximately $P_e = 0.01$. Since the errors spread across the hidden portion of the length "m", it is reasonable to apply a $q = 2^m$-ary shorted Reed Solomon error correcting codes (RS) to correct such errors. It is well known [11] that such a code, with parameters (n, k), $q = 2^m$, satisfies $n \le 2^m$-1 and has a minimal code distance $d = n$-k+1. It is capable of correcting all q-ary errors of multiplicity up to $d/2$. The probability of erroneous decoding for such a code is

$$P_{ed} = \sum_{i=\left[\frac{d-1}{2}\right]+1}^{n} \binom{n}{i} P_e^i \left(1 - P_e\right)^{n-i} \quad (2)$$

Substituting the previously chosen RS code parameters $n = 320$, $k = 300$, $d = 21$ and symbol error probability $P_e = 0.01$ into (2), we obtain $P_{ed} \approx 0.00034$. This value can be considered sufficiently small for each frame.

IV. Attacks on the proposed SG detectability

Let us emphasize once more the main feature of the proposed SG system: only one LSB for a randomly chosen pixel in each frame of the video file is inverted.

It is important to note that the selection of pixels for inversion should be performed randomly but not through a regular selection process, e.g., from the first pixel of the frame up to the last one. Otherwise, it may increase the efficiency of SGA.

Let us recall some SG detection attacks on conventional LSB-based SG, where the LSBs of all pixels are replaced with bits of secret information. In this case, typical histograms of CO and SG have the view shown in Fig. 3 where the histogram $V(i) = \#\{n, B(n) = i, B(n)$ is the brightness of n-th pixel.

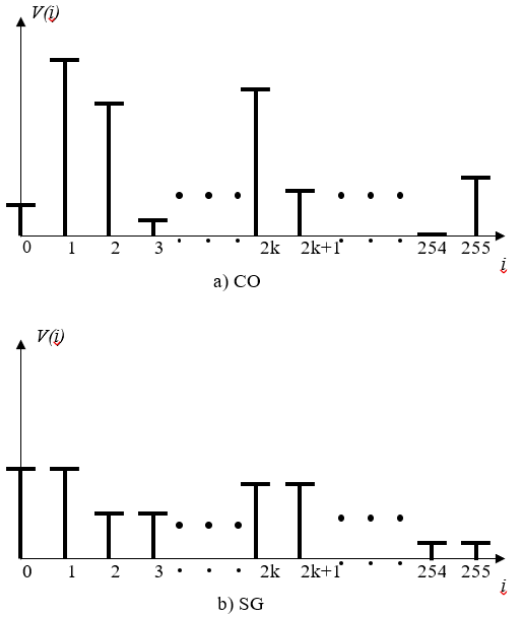

Fig. 3. Typical histograms for CO (a) and for LSB-based SG(b)

By using closely located neighboring levels of the histogram for SG, it is easy to establish a criterion for SG detection:

$\chi^2 \le \alpha$, then SG is present, (3)

$\chi^2 > \alpha$, then SG is absent,

where α is predefined threshold,

$$\chi^2 = \sum_{i=0}^{(L-1)/2} \frac{\left(V(2i) - V(2i+1)\right)^2}{2\left(V(2i) + V(2i+1)\right)}. \quad (4)$$

In fact, detection criteria (3) works satisfactory for detecting of LSB-based SGs but it is practically useless for the proposed SG because its histogram differs only slightly from the CO histogram in each frame, as only one pixel per frame is changed.

To verify this fact, a simulation was performed on typical video frames. The results are presented in Table III. From this table, we can see that even for an optimally chosen threshold, the minimal SG detection error $P_e = (P_{fa} + P_m)/2$, (where P_{fa} is the probability of false alarm, P_m is the probability of missing SG detection in the proposed system), is very close to ½, which is equivalent to random guessing.

TABLE III. THE PROBABILITIES OF ERRORS (P_{FA}, P_M, $P_E = (P_{FA}+P_M)/2$) AGAINST CHOSEN THRESHOLD FOR THE PROPOSED SG AND UNDER THE USE χ^2 CRITERIA

Probability / Threshold	P_{fa}, %	P_m, %	P_e, %
0,002	0	100	50
0,006	8	92	50
0,010	29	71	50
0,014	31	69	50
0,022	36	64	50
0,030	39	61	50
0,034	50	50	50
0,038	53	47	50
0,054	59	41	50
0,062	60	40	50
0,066	77	23	50
0,070	80	20	50
0,0546	80	20	50
0,554	90	10	50
0,558	98	2	50
0,562	100	0	50

Another detection criterion that is widely used against LSB-based SGs computes the difference between neighboring pixel byte levels as

$$\gamma = \frac{\sum_{i=0}^{L-1} \left(n(i) - n(i+1)\right)^2}{2\sigma_n^2}, \quad (5)$$

where $n(i)$ is the number of bytes with the value i among the pixels in the frame.

The results of the SG detector simulation on criterion where χ^2 is replaced by γ, are shown in Table IV.

We can see from this table that even under selection of an optimal threshold, the probability P_e is close to 1/2 and hence to the probability of a random guessing.

As for SGA using the method known as *Sample Pair Analysis* [3], which is very popular and effective for detecting conventional LSB-based SGs, it involves solving a quadratic equation to determine the probability "P" of LSB embedding in each pixel. However, such an approach is senseless for the proposed method because, we do not have any embedding with pixel probability but only the inversion of one LSB randomly chosen in each frame.

TABLE IV. THE PROBABILITIES OF ERRORS (P_{FA}, P_M, $P_E = (P_{FA}+P_M)/2$) AGAINST CHOSEN THRESHOLD α FOR THE PROPOSED SG AND UNDER THE USE OF γ - CRITERION

Probability / Threshold	P_{fa}, %	P_m, %	P_e, %
10	1	0	50
20	95	5	50
25	86	14	50
30	75	25	50
35	60	40	50
40	51	49	50
45	39	61	50
50	27	73	50
55	17	83	50
60	15	85	50
65	13	87	50
70	11	89	50
80	5	95	50
90	2	98	50
100	0	100	50

Let us return to the discussion regarding the optimal detection algorithm for the proposed SG. Since the transformation of CO to SG requires modifying only one LSB per frame, and moreover, only one randomly chosen LSB in each frame, it is reasonable to suggest that the optimal detection algorithm should analyze only LSBs of all frames. A second reasonable assumption is that the correlation between LSBs of different frames can be neglected. Hence, the decision about SG presence should be made independently for each frame, and the final decision (CO or SG) for the entire file should be determined using the majority rule.

It follows from our previous discussion that we could try to use some classifier such as *support vector machine* (SVM) [12] for SG detection after preliminary training on samples of CO and SG constructed according to our method. Of course, such a classifier could be replaced by specialized neuron network [13]. In the future, we plan to implement this approach, but we have a little hope of achieving positive results because the "invasion" by CO transform in SG is minimal. For instance, in 1920x1080 pixel frames, we modify only one LSB in ~ 2×10^6 pixels!

But let us consider an attack against the proposed SG from another (unusual) perspective–namely, the execution property of strong encryption for secret messages. It is obvious that encryption and decryption procedures using a secret key are mandatory for the proposed SG. Otherwise, a steganalyst, who we assume knows the full "embedding" and extraction algorithms, would be able to recover the secret message after hashing of the file and recognize the SG presence, since the extracted information would be meaningful.

The idea of executing such an SGA and the method to prevent it were recently proposed and published in [14,15]. On the other hand, if the "embedded" data was previously encrypted using a strong cipher, then a steganalyst would be able to extract this ciphertext and test it for pseudorandomness

using so called *NIST tests.* (See [16] and Table V). If the video file is not SG (so., no data was "embedded"), then the steganalyst would be able to extract hashes of frames, but they would not necessarily pass all NIST tests. The number of passed tests can then be compared to a predefined threshold. If it exceeds the threshold, the file is assumed to be SG; otherwise, it is considered as CO.

TABLE V. THE TITLES OF NIST TESTS ON PSEUDORANDOMNESS

N	Titles of tests
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 test
15	The random excursions variant test

In Table VI the results of NIST tests passing for SG with LSB-based embedding taken from the paper [14] are presented, where a *plus* sign (+) indicates that the corresponding test was passed and a *minus* sign (-) indicates that it was not passed. We note that the specific SG embedding method used does not affect the results. So, the embedding algorithm [15] differs from the SG proposed in the current paper because the results depend on type of cipher only. In [14], the strong cipher GOST-28147-89 was applied.

TABLE VI. RESULTS OF STEGANALYTIC TESTING FOR FULL LSB-BASED EMBEDDING WITH ENCRYPTION OF 15 SEQUENCES BY CIPHER GOST-28147-89

Test \ Sequence	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	+	+				+	+	+		+		+	+	+	

A more complete statistic is presented in Table VII, also taken from [14].

We can see from Tables VI, VII that most of the tests have been passed. In contrast to NIST passing for any SG, the results of NIST tests passing for CO as MPEG-2 file and taken from [14] are shown in Table VIII.

TABLE VII. RATIO OF NIST TESTS PASSING (IN %) CALCULATED FOR 1000 CIPHER TEXTS ENCRYPTED BY CIPHER GOST-28147-89.

NIST test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Ratio, %	98.9	99.5	99.1	98.4	99.4	99.2	99.8	98.8	98.8	992	98.4	99.0	98.6	69.7	70.0

TABLE VIII. RESULTS OF NIST TESTING AFTER EXTRACTION OF CO FOR MPEG-2 FILE.

Test \ Sequence	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			+	+	+		+	+		+	+	+			+

A more complete statistic for the extracted date from MPEG-2 covers are presented in Table IX, taken from [14].

TABLE IX. RATIOS OF PASSED TESTS EXTRACTED FROM 1000 DIFFERENT FILES MPEG-2 WITHOUT EMBEDDING

NIST test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Ratio, %	49.0	45.9	29.1	66.0	96.2	82.5	53.2	68.4	68.4	97.2	43.8	45.7	43.0	32.1	33.2

Although it is a different CO than the one investigated in the current paper, we believe that this experiment serves only as an example of the general SGA method, as we later show how it is possible to protect the detection of the proposed SG against a similar attack.

We also note that the use of *Support Vector Machine* (SVM classifier) yields significantly better results, namely the probability $P_e \sim 5.2\%$ [17].

Thus, we can see that SGA method based on the use of NIST tests occurs sufficiently effective even against the SG proposed in the current paper. But fortunately, an approach proposed in [15] effectively protects against such an attack. Let us briefly consider the proposed protection method. Then the embedding procedure consists of three steps:

1) Encryption of the secret data using any strong cipher.

2) Decompression of the encrypted binary sequence with arithmetic code (AC) given certain probabilities. Note that family of AC is error-free; however, in order to use the decompression procedure, it is necessary to define the probabilities of symbols [15], that may play the role of an additional stegokey.

3) “Embedding” of the decompressed sequences into CO according to a given SG algorithm.

In order to extract secret data from SG, it is also necessary to perform three steps:

1) Extract the “embedded” data according to the SG algorithm.

2) Compress the extracted sequence with the known probabilities of AC.

3). Decrypt the sequence obtained in the step 2 using the known crypto key.

If an attacker does not know the embedding cipher modification and the values of probabilities for AC, then they will likely obtain a sequence that does not pass NIST tests.

Table X presents the results of NIST testing after cipher modification with decompression using AC with the probabilities $P(O) = 0.49$, the encryption is by AES cipher, $P(0) = 049$, $P(1) = 0.51$, taken from the paper [15].

TABLE X. THE RESULTS OF NIST TESTING AFTER CIPHER MODIFICATION WITH AC UNDER THE PROBABILITIES $P(0)=0.49$, $P(1)=0.51$.

Test \ Sequence	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															

By comparing Table X and Table VIII, we can see that these Tables are similar; hence, it is impossible to detect SG using NIST testing after cipher modification based on AC.

Nevertheless, an attacker can apply compression to the extracted sequence if the parameters $P(0)$ and $P(1)$ for AC are known to them. In this case, they would be able to detect SG. However, if these parameters are unknown, the attacker may attempt different values $P(0)$, $P(1)$ close to 0.5. Table XI presents the results of NIST testing in terms of pass rates for each of tests and for different probabilities of AC $P(0)$, $P(1)$, chosen by the attacker close to 0.5.

It follows from the last results that the exact knowledge of the probabilities $P(0)$, $P(1)$ is not necessary to provide correct a detecting of SG based on NIST tests. Therefore, it is necessary to select a more complex secret key than a single AC parameter. In [15], a method was proposed to use the sequence of probabilities $P_i(0)$, $i = 1, 2, \ldots, N$ as an additional key, where $N = 50$ is the length of the decompressed sequence. Then, such a selection of an additional key prevents its discovery through exhaustive search.

TABLE XI. THE PASS RATES OF NIST TESTS AFTER ATTACKER'S COMPRESSION WITH DIFFERENT PARAMETERS $P(0)$, $P(1)$ CLOSE TO 0.5 GIVEN TO THE DECOMPRESSION BY LEGITIMATE USER $P(0) = 0.49$, $P(1)=0.51$.

N of 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		0	77	0	97	99	99	8	95	97	99	91	16	0	10	11
0.42	0.58		0	94	0	98	99	100	53	96	100	98	98	49	0	18	19
0.43	0.57		0	89	0	99	98	98	66	94	100	99	97	63	0	19	17
0.44	0.56		1	98	1	98	98	99	90	95	99	99	96	89	0	26	28
0.45	0.55		2	98	24	98	99	98	98	98	99	96	97	95	1	41	42
0.46	0.54		17	96	76	100	98	97	100	98	97	98	99	95	16	40	41
0.47	0.53		52	98	92	97	99	99	100	98	99	98	98	99	53	52	57
0.48	0.52		91	99	100	99	100	99	99	99	100	98	99	100	89	73	74
0.49	0.51		100	97	98	99	99	99	100	99	99	99	98	98	99	75	73
0.51	0.49		62	100	96	100	100	99	100	98	100	99	97	97	59	65	60
0.52	0.48		13	97	71	99	99	98	100	90	100	98	96	97	15	40	37
0.53	0.47		21	99	72	99	99	97	100	93	99	99	100	99	17	44	44
0.54	0.46		0	97	4	93	100	99	97	73	100	99	99	95	0	25	24
0.55	0.45		0	93	0	98	99	97	83	60	99	98	97	82	0	23	23
0.56	0.44		0	99	0	90	99	99	59	32	97	100	96	76	0	16	17
0.57	0.43		0	84	0	69	100	100	27	4	98	99	94	28	0	10	10
0.58	0.42		0	60	0	18	99	98	1	0	100	97	82	0	0	11	11

CONCLUSION

In this paper, we propose a novel stegosystem similar to the so-called embeddingless approach, specifically designed to introduce minor modifications to cover objects in the form of video files. Unlike conventional SG, which typically involves a trade-off between security and embedding rate, the proposed method emphasizes a balance between embedding rate and the computational complexity of the embedding process. On a standard PC, embedding secure information into 30 video frames requires approximately 7 minutes. Therefore, the use of higher-performance computers is recommended to reduce embedding time.

A video file was selected as the CO for the proposed SG in order to increase the total number of bits that can be embedded. Under the parameters considered in this study–including the use of RS coding and typical 10-min video files–the system can reliably and securely embed approximately 17 kbits of information. Nevertheless, the method is flexible, and COs may also include image sequences or even a single image. The title of the proposed SG “almost perfectly secure SG” is more of a metaphor, reflecting the fact that as for the moment, we do not know any conventional attacks for our SG detection. Unfortunately, however, one attack based on detection of the embedded message after its encryption by strong cipher exists. Although we refer to the paper necessity where such an attack may be removed by the use NIST tests, this modification would require a significant extension of the proposed SG algorithm.

We believe that future developments of the proposed stegosystem could follow these directions:

- Optimization of the parameters for the proposed SG, such as (m, lo).
- More careful selection of an error correction code including decoding algorithm.
- Additional simulation of cipher modification.
- Investigation of alternative CO for embedding, such as photo sessions.

REFERENCES

[1] K.G. Paterson, *Applied cryptography Knowledge Area*. Version 1.0.0, ETH, Zuruch, 2021.

[2] J. Fridrich, *Steganography in Digital Media*. Cambridge University Press, 2010.

[3] S. Dumitrescu, X. Wu, Z. Wang "Detection LSB Steganography via Sample Pair Analysis", *LNCS*, vol. 2578, 2002, pp. 355-378.

[4] M. Jain, S.K. Lenka, "A Review on Digital Leakage Prevention Using Image Steganography", *International Journal of Computer Science Engineering*, vol. 5, N 2, 2016, pp. 56-59.

[5] V. Korzhik, G.M. Luna, K. Nebaeva "Stegosystems based on noisy channels", *Proc. of the 3th Spanish Meeting on Cryptography and Information Security*, 2006, pp. 379-387.

[6] K.A. Zhang, A. Cuesta-Infante, L. Xu, K. Veeramachaneni "SteganoGAN: High Capacity Image Steganography with GANS", *arXiv*: 1901.03892v2 [cs.CV], 2019.

[7] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio "Generative Adversarial Networks", *arXiv*: 1406.2661v1 [stat.ML] 2014.

[8] Z. Zhou, H. Sun, R. Harit, X. Chen, S. Xingming "Coverless Image Steganography Without Embedding", *Lecture Notes in Computer Science*, 2015, pp.123-132.

[9] D.M. Blei, A.Y. Ng, M.I. Jordan "Latent Dirichlet Allocation", *Journal of Machine Learning Research*, N3, 2003, pp.993-1022.

[10] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[11] F.J. Macwilliams, N.J.A. Sloane, *The Theory of Error-correcting Codes*, North-Holland Publishing Company, 1979.

[12] S. Agaian, "Steganography & Steganalysis, An Overview of Research & Challenges", *Aspects of Network Security and Information Security*, NATO Science for Peace and security Series D, Information and Communication Security, vol. 17, 2008, pp. 179-210

[13] E. Herberg, "Lecture Notes: Neural Network Architectures" *arXiv*:2304.05133v2 [cs.LG], 2023.

[14] V. Korzhik, C. Nguyen, I. Fedyanin, "Side Attacks on Stegosystems Executing Message Encryption Previous Embedding", *Journal of Information Hiding and Multimedia Signal Processing*, vol.11, N1, 2020, pp.44-57.

[15] V. Korzhik, C. Nguyen, G. Morales-Luna "Cipher Modification Against Steganalysis Based on NIST Tests" *PROCEEDING OF THE 24TH CONFERENCE OF FRUCT ASSOCIATION*, 2019, pp.179-186.

[16] L.E. Bassham, A.L. Rukhin, J. Soto, J.R. Nechvatal, M.E. Smid, S.D. Leigh, M. Levenson, M. Vangel, N.A. Heckert, D.L. Banks *A Statistical Test Suite for Random and Pseudorandom Number-Generators for Cryptographic Applications,* Technical report, Gaithersburg, MD. USA, 2010.

[17] V. Korzhik, C. Nguyen, K. Akhrameeva "Detection of Video Steganography with the Use Universal Method Based on NIST-test" *Proceedings of Telecommunications Universities*, vol. 6, N1, 2020, pp.70-76 (in Russian)