Software model of a highly undetectable stegosystem – ⊕HUGO model

V N Kustov¹ and A G Krasnov²

¹ Informatic and Information Security Department, St. Petersburg state Railways University of Emperor Alexander I, 9 Moskovsky Ave., Saint Petersburg, North-Western Federal district, 190031, Russia
² JSC “Nexign”, 199155, Saint Petersburg, Uralskaya street, house 4 letter b, room 19n, Russia

E-mail: kvnvika@mail.ru, aleksei.krasnov@gmail.com

Abstract. The article is devoted to the research of a software model of the hidden message transmission process using one of the new methods of steganography proposed by the authors. The presented model implements a hard-to-detect method for embedding hidden messages in digital still images used as covering objects (containers). The main principle of using the model is to minimize adequately specified distortion using a new efficient encoding algorithm. Distortion is defined as an operation ⊕ (excluding OR - XOR) between the next half-byte of the hidden message and the right half-byte of the specially selected byte of the covering object, which is selected from the condition for minimizing the penalty function calculated on some selected neighbourhood of this byte. This allows authors to keep the original image almost unchanged and make it indistinguishable from the original, even after large payloads are embedded in it. Thus, the embedded hidden message may not be visible when the steganalysis even for the experienced steganalytic. The ⊕ operation has the property of bijectivity (it is reversible), which makes quite easy to use it to extract a hidden message on the recipient's side, if the recipient has key information. The stegosystem model is called the ⊕Highly Undetectable SteGOsystem model. In short, it can be referred to as the ⊕HUGO model.

1. Introduction

To determine the stability of a stegosystem, attempts are usually made to apply an information-theoretical approach [1]. This approach uses a stegosystem model with a channel without repeated stego transmissions in the presence of a passive opponent with unlimited computing capabilities.

The most common definitions of stegosystem stability are based on the requirement that the probability distribution on a set of stegos is indistinguishable from the probability distribution on a set of empty covering objects. Statistical indistinguishability is understood in this case as indistinguishability with respect to arbitrary algorithms.

Then the authors will use the following terminology. Let's assume that the sender has a covering object called the source object, as well as a function (algorithm) G that allows you to convert the source container to a covering object with noise. For example, in the case of a covering object containing visual information, there is a still digital image. The original of the covering object is an electronic photocopy of the object. The G algorithm allows you to introduce random distortion (noise)
into this covering object. Let's use the idea of creating a steganographic channel by masking the hidden message with the noise introduced by the \( G \) algorithm into the covering object. We assume that a passive opponent is likely to pass empty (valid) covering objects with noise generated by the \( G \) algorithm over the communication channel.

2. Stegosystem model

Let \( C, K, \) and \( M \) be some finite sets. In the model under consideration, the containers (both the source and the noise container), the secret key, and the hidden message are random variables on the sets \( C, K, \) and \( M, \) respectively. We will denote by \( C, K, \) and \( M \) random variables that are the source container, secret key, and hidden message, respectively.

Denote by \( R \) a random variable on a finite set \( \hat{R}. \) This random variable is used by the sender to add additional randomness to the stego.

In our model sender and receiver have the secret key \( k, \) which is the value of a random variable \( K. \) In order to provide the recipient with a message \( m \) (which is the value of a random variable \( M \)), the sender computes stego \( s=\text{Emb}(c,m,k,r), \) where \( c \) and \( r \) — values of the random variables \( C \) and \( R, \) respectively, and sends it to the recipient. Without limiting generality, we can assume that \( s \in C. \) After receiving stego \( s, \) the recipient can restore \( m \) by calculating \( \text{Ext}(s,k). \) The sender can also send the recipient the value of a random variable \( G(c,d), \) where \( d \) is a random variable on some finite set \( D, \) and \( G \) is some function from \( C \times D \) in \( C. \) Thus, \( D \) plays the role of noise.

The opponent in the model under consideration is a family of functions \( \{A_c | c \in C \}, \) where \( A_c : C \rightarrow \{0,1\}. \) The goal of the opponent is to know the value of \( C \) and distinguish the stego (value \( S=\text{Emb}(C,M,K,R) \)) from the container with noise (values \( C'=G(C,D) \)). If \( c \) is the source container known to the opponent, then the equality \( A_c(x)=1 \) \((A_c(x)=0)\) means that, according to the opponent, the container \( x \in C \) is a stego (respectively, it is empty).

Thus, the stability of the stegosystem against the threat of detection of a steganographic channel based on an attack with a known container is considered.

The above-mentioned opponent can make two types of errors, called errors of the first and second kind in mathematical statistics. The first type of error is considered to be a situation when the opponent mistook an empty container for a stego. If the opponent considers the stego to be an empty container, then there is a second kind of error. Denote by \( \alpha \) and \( \beta \) the average (in \( C \)) probabilities of errors of the first and second kind, respectively.

Two attempts to determine the information-theoretical stability of stegosystems are known from the literature. The definition of Kashen [2] is based on the following requirement: the entropy of an empty container (a container with noise) relative to the stego must be small. We emphasize that we are talking about relative entropy. Thus, Cashin considers the opponent's task of distinguishing an empty container from a stego as a task of testing statistical hypotheses. Another approach is described in the work of Zöllner et al. [3]. It is based on this requirement: knowing the container and its corresponding stego does not reduce the entropy of the hidden message. Note that here the opponent's task is essentially meant to extract some information about a hidden message (obviously, the detection of a steganographic channel is a special case of this task).

Also known is the work of Anderson and Petiolas [4] and its early version [5], in which some mathematical statements are formulated (for example, the estimation of the capacity of steganographic channels from above via the entropy difference). However, these works are reviews and do not give mathematically strict definitions of the concepts under consideration.

However, it should be noted that all the above estimates of the stability of the stegosystem are based on the entropy approach and require precise determination of the distribution laws of random variables \( C' \) and \( S. \) This requirement is the main obstacle that makes it difficult, and in some cases impossible, to apply this approach.

In this case, simpler relations can be used for statistical evaluation of the effectiveness of the developed stegosystem implementing the \( \oplus \) HUGO algorithm.
The main widely used metric for displaying the difference between empty and filled covering objects is the peak signal-to-noise ratio (PSNR), calculated using the formula:

$$\text{PSNR} = 10 \log_{10} \left( \frac{255^2}{\text{MSE}} \right).$$

This is the ratio between the maximum possible signal value and the power of noise that distorts the signal value [6].

The mean square error (MSE) determines the difference between the pixel intensities of the stego and the covering object. MSE (denoted by the symbol $\bar{G}$) is calculated from the following relation [7]:

$$\bar{G} = \frac{1}{MN} \sum_{i=1}^{N} \sum_{j=1}^{M} (f(i,j) - f'(i,j))^2,$$

where $f(i,j)$ is the brightness of the covering object pixel, and $f'(i,j)$ is the brightness of the corresponding stego pixel, $N$ is the length of the digital image in pixels, $M$ is the width of the digital image in pixels.

A high value of $\bar{G}$ indicates poor quality of the original image, and vice versa.

Capacity is a percentage of the size of the original covering object $V_C$ and the secret message $V_m$, calculated using the formula:

$$\text{Capacity} = \frac{V_m}{V_C}.$$

Correlation ($r_{cs}$) is used to display the degree of tightness of the paired linear relationship between the covering object $C_i$ and the stego $S_i$:

Usually $r_{cs}$ is calculated from the ratio [8]:

$$r_{cs} = \frac{\text{cov}_{cs}}{(n-1)\bar{G}_C \bar{G}_S},$$

where $\text{cov}_{cs}$ is called covariance and is calculated from the ratio:

$$\text{cov}_{cs} = \sum_{j=1}^{K} (c_j - \bar{c}) (s_j - \bar{s}).$$

If we expand the product of $6_c G_s$, we get the formula for calculating them:

$$6_c G_s = \sum_{j=1}^{K} (c_j - \bar{c})^2 \sum_{j=1}^{K} (s_j - \bar{s})^2$$

Assuming $K = N*M$, $n = 2$, we obtain the following relation for the correlation coefficient [9]:

$$r_{cs} = \frac{\sum_{j=1}^{K} (c_j - \bar{c})(s_j - \bar{s})}{\bar{G}_C \bar{G}_S},$$

and given the ratio for $6_c G_s$, we get the final expression for calculating the $r_{cs}$ correlation coefficient in the following form:

$$r_{cs} = \frac{\sum_{j=1}^{K} (c_j - \bar{c})(s_j - \bar{s})}{\sqrt{\sum_{j=1}^{K} (c_j - \bar{c})^2 \sum_{j=1}^{K} (s_j - \bar{s})^2}}$$

where: $c_j$ is the byte $j$ value of covering object $C_i$; $s_j$ is the byte $j$ value of stego $S_i$; $\bar{c}$ and $\bar{s}$ are average values of the bytes $C_i$ and $S_i$, respectively; $6_c$ is MSE for $C_i$; $6_s$ is MSE for $S_i$; $n$ is the number of compared observations (in this case, $n = 2$); $K$ - is the number of bytes in $C_i$ and $S_i$.

3. ⊕HUGO Algorithm

Performing an operation to embed a secret message in a covering object, you can use the ⊕ operation for the selected byte of the covering object. Selecting a suitable byte for the ⊕ operation is performed by analyzing the 3x3 neighborhood for each byte of the covering object as shown in the figure 1.

A Bi byte that meets the condition is considered suitable for performing an embedding operation [10]:

$$B_i \neq \frac{\sum_{j=1}^{8} A_j}{8}.$$ (1)
The implementation of this provision makes it possible to exclude the monotonous area of the covering object from the embedding operation, thereby increasing the stability of the stegoalgorithm to steganalysis. As a result, only parts of the image with a high frequency of changes in the brightness level of pixels and the embedded message become indistinguishable from noise and will be used for embedding.

![Figure 1](image1.png)

**Figure 1.** Determining of the appropriate byte for embedding.

We should note also that you can use other simpler types of neighborhoods, such as three consecutive bytes or a 3x3 cross.

The scheme for performing operations for embedding and extracting a secret message in accordance with the \( \oplus \) HUGO algorithm is illustrated in figure 2.

![Figure 2](image2.png)

**Figure 2.** The scheme for performing operations in accordance with the \( \oplus \) HUGO algorithm.

To describe the \( \oplus \) HUGO algorithm, assume that the secret message \( M \), the subset of bytes of the covering object \( C \) selected for the embedding operation and satisfying the relation (1), and the subset
of corresponding m bytes of stego S are the final rows of bytes. We describe the execution of the ⊕HUGO algorithm by the following sequence of actions.

For the embedding operation (Emb), perform:
1. the Next half-byte $m_i$ of the hidden message is added to the right half-byte of the next byte $c_i$ from the subset $C$. The result of the operation is written to the right half-byte of the stego $s_i$. Formally, this operation can be written as the ratio: $s_i = m_i \oplus c_i$.
2. Point 1 is executed for all half-bytes of the hidden message.

In this example, for the embedding operation of the first half-byte $m_1$ of a hidden message, the embedding process can be represented as the following relations: $s_1 = m_1 \oplus c_1$. Representation of this operation in hexadecimal code is as follows: $1 = B \oplus A$. In binary code, it looks like this: $0001 = 1011 \oplus 1010$.

Let's look at the operation of extracting a hidden message (Ext).
1. The Next right half-byte $s_i$ stego with the operation ⊕ is added to the right half-byte of the next byte $c_i$. The result of the operation is written to the next half-byte $m_i$ of the hidden message. Formally, this operation can be written as the ratio: $m_i = s_i \oplus c_i$.
2. Point 1 is executed for all half-bytes of the stego.

In this example, for the operation of extracting the first half-byte $m_1$ of a hidden message, the process can be represented as the following relations: $m_1 = s_1 \oplus c_1$. Representation of this operation in hexadecimal code looks like this: $B = 1 \oplus A$. In binary code, it looks like this: $1011 = 0001 \oplus 1010$.

It should be noted that in this embodiment, the implementation of ⊕HUGO algorithm, for processes of embedding and extraction use the same operation ⊕ are used. This is due to the remarkable property of this operation, called bijectivity (reversibility). The process of embedding and extracting a hidden message is similar to the gamut encryption process. The gamma role in this case is played by the right half-bytes of the bytes selected according to condition (1) of the covering object. They also act as a secret key.

4. Software model
The goals of creating a software model are:
1. Confirmation of theoretical assumptions in practice;
2. Calculation of algorithm performance indicators for different input data:
   • Pearson correlation coefficient between the container before embedding and the container after embedding the secret message,
   • Capacity;
3. Creating a prototype of a software package that implements the proposed algorithm.

The software model is described in the unified modeling language (UML). The static structure of system objects is described using a class diagram. User interaction with the program is described using the Use Case Diagram and Activity Diagram.

Figure 3 shows a class diagram.

Figure 3. Class Diagram.
Figure 4 shows a use case diagram.

![Use case diagram](image)

**Figure 4.** Use case Diagram.

The prototype of the software package is implemented in the high-level programming language Python. This programming language is chosen because it is intended for scientific computing, including digital image processing. The relative laconism of Python allows you to create a program that is much shorter than its counterpart, which makes it faster to create prototypes. In addition, Python is suitable for scientific research and prototyping mainly due to a large and actively developing ecosystem of libraries created by third-party developers.

The program uses the following third-party libraries:

- **cv2** library for converting images to a three-dimensional data array and back when embedding a secret message in a container;
- **numpy** library for working with multidimensional arrays when embedding a secret message in a container;
- **binascii** library for converting data to binary when converting a secret message to a bit string;
- **math** and **scipy** libraries for performing mathematical and statistical calculations when calculating the Pearson correlation coefficient.

The software package consists of two independent packages. The first package inserts a secret message into the container and calculates the algorithm's performance indicators. The second package extracts the secret message from the container.

The following data must be submitted for the first software package:

- file with a secret message;
- container file;
- the number of embedded container bits (the number of bits of each RGB channel of each pixel of the container image to embed the secret message in);
• value of the sensitivity (the minimum allowed standard deviation). Embedding when executing the `encode_img` function will be performed if the standard deviation is greater than the specified sensitivity;
• value of the sensitivity reduction step.
The output of the first software package is:
• calculated values of the algorithm's performance indicators (MSE, PSNR, Pearson correlation coefficient, capacity);

![Figure 5. Activity Diagram.](image)

• container with a built-in secret message in a PNG file;
• two-dimensional graphs of the dependence of MSE, PSNR and Pearson correlation coefficient on sensitivity;
• two-dimensional graph of the dependence of the Pearson correlation coefficient on the number of embedded bits.

The following data must be submitted for the second software package:
• a container file with a built-in secret message;
• the number of embedded container bits (the number of bits of each RGB channel of each pixel of the container image to embed the secret message in);
• value of the sensitivity.

The output of the second software package is a file with a secret message.

The list of functions implemented in the functions of the first software package and their purpose are shown in the table 1.
Table 1. The functions of the first software package and their purpose.

| Function name      | The purpose of the function                                                                 |
|--------------------|---------------------------------------------------------------------------------------------|
| input_secret       | Enter the name of the file with the secret message                                           |
| input_container    | Entering the name of the container file                                                     |
| input_number_of_lsb| Enter the number of embedded bits of the container                                           |
| input_sensitivity  | Entering sensitivity                                                                        |
| input_step         | The input step of reduction of sensitivity. If the previously specified sensitivity is high for embedding a secret message in the selected container, the sensitivity will decrease by the specified step and the embedding will be attempted again. And so each time until the secret message is fully embedded in the container or the sensitivity is reduced to the first negative value (it is impossible to embed. the container is too small for a secret message) |
| encode_img         | Embedding a secret message in a container                                                   |
| calculation_psnr   | PSNR calculation                                                                            |
| calculation_capacity| Capacity calculation (the ratio of the number of bits of a secret message to the number of pixels in the container) |
| save_output        | Saving a container with a built-in secret message in a PNG file                            |
| kpi_output         | The output of the calculated indicators of the effectiveness of the algorithm:            |
|                    | • MSE                                                                                      |
|                    | • PSNR                                                                                    |
|                    | • Pearson correlation coefficient                                                         |
|                    | • capacity                                                                                 |
| chart_sensitivity  | Construction of two-dimensional graphs of the dependence of MSE, PSNR and Pearson correlation coefficient on sensitivity |
| chart_number_of_lsb_pearsons_r| Building a two-dimensional graph of the dependence of the Pearson correlation coefficient on the number of embedded bits |

The list of functions implemented in the second software package and their purpose are shown in the table 2.

Table 2. The functions of the second software package and their purpose.

| Function name      | The purpose of the function                                                                 |
|--------------------|---------------------------------------------------------------------------------------------|
| input_container    | Enter the name of a container file with a built-in secret message                           |
| input_number_of_lsb| Enter the number of embedded bits of the container                                         |
| input_sensitivity  | Entering sensitivity                                                                        |
| decode_img         | Extract a secret message from a container                                                   |
| save_secret        | Saving a file with a secret message                                                        |
5. The modeling results

Dependence of the Pearson correlation coefficient on the capacity is shown in the table 3 and at the diagram, figure 6.

Table 3. Dependence of the Pearson correlation coefficient on the capacity.

| Capacity     | Pearson correlation coefficient |
|--------------|-------------------------------|
| 0.0020938271604938 | 0.999999778871843             |
| 0.0057975308641975  | 0.999999395816408             |
| 0.0181827160493827  | 0.999998071126027             |
| 0.0613234567901235  | 0.999993505057878             |
| 0.1976049382716049  | 0.999979058609949             |
| 0.5933086419753086  | 0.999937203360553             |

Figure 6. Dependence of the Pearson correlation coefficient on the capacity.

Thus, the simulation results allow us to assert the practical indistinguishability of the covering object (container) from the stego (container with a secret message) even at the capacity values of 0.5933086419753086, since the value of the Pearson correlation coefficient does not exceed the value of 0.999937203360553. This makes solving the problem of steganalysis very difficult even for a very experienced steganalytic.

6. Conclusion

Thus, the article considers a software model of the hidden message transmission process using one of the new methods of steganography proposed by the authors. The presented model implements a hard-to-detect method for embedding hidden messages in digital still images used as covering objects (containers). The main principle of using the model is to minimize adequately specified distortion using a new efficient encoding algorithm. Distortion is defined as an operation ⊕ (excluding OR - XOR) between the next half-byte of the hidden message and the right half-byte of the specially selected byte of the covering object, which is selected from the condition for minimizing the penalty.
function calculated on some selected neighborhood of this byte. This allows authors to keep the original image almost unchanged and make it indistinguishable from the original, even after large payloads are embedded in it. Thus, the embedded hidden message may not be visible when the steganalysis even for the experienced steganalytic. The $\oplus$ operation has the property of bijectivity (it is reversible), which makes quite easy to use it to extract a hidden message on the recipient's side, if the recipient has key information. The stegosystem model is called the $\oplus$Highly Undetectable SteGOsystem model. In short, it can be referred to as the $\oplus$HUGO model. This model can be implemented for embedding hidden messages in the spatial area of digital still images much more efficiently than using the Least Significant Bit (LSB) algorithm. The proposed stegosystem model provides higher performance than the LSB algorithm and allows you to transmit a longer message, while providing higher resistance to detection. The results of statistical tests of the software model developed by the authors are presented, confirming its high resistance to existing methods of steganalysis. Among other things, the novelty of the authors' solution is that the cryptographic gamming algorithm is directly used as the implementation algorithm. This allows you to opt out of pre-encryption of a secret message.

References
[1] Varnovsky N P Golubev E A and Logachev O 2014 Modern trends in steganography http://cryptography.ru/wp-content/uploads/2014/09/steganography.pdf (in Russian)
[2] Cachin C 1998 An information-theoretic model for steganography Proc. 2nd Intern. Workshop on Inform. Hiding LNCS Vol 1525 306–318
[3] Zöllner J, Federrath H, Klimant H, Pfitzmann A, Piotraschke R, Westfeld A, Wicke G and Wolf G 1998 Modeling the security of steganographic systems Proc. 2nd Intern. Workshop on Inform. Hiding LNCS Vol 1525 344–354
[4] Anderson R and Petitcolas F 1998 On the limits of steganography IEEE J. on Selected Areas in Communications Vol 16 No 4 474–481
[5] Anderson R 1996 Stretching the limits of steganography Proc. 1st Intern. Workshop on Inform. Hiding LNCS Vol 1174 39–48
[6] Kustov V and Protsko D 2018 Using a discrete wavelet transform to embed information in images Collection of scientific papers based on the materials of the XVII international scientific conference Russia Center for Scientific Conferences "International Research Federation Public science" 15-20
[7] Mean squared error https://en.wikipedia.org/wiki/Mean_squared_error
[8] Correlation https://en.wikipedia.org/wiki/Correlation
[9] Kustov V and Protsko D 2019 A steganographic model using wavelet and arnold cat transform with multi-threshold error-correcting coding hungary (Budapest) Proc. Of 42nd International Conference on Telecommunications and Signal Processing (TSP) IEEE 49-52
[10] Kustov V and Silanteva E 2020 ±1Highly Undetectable Stegosystem Model Using Digital Still Images Italy (Milan) Proc. Of 43rd International Conference on Telecommunications and Signal Processing (TSP) IEEE 6-9