DOI: https://doi.org/10.36910/6775-2524-0560-2023-53-10
UDC 004.056.5
Struk Iryna, Student
https://orcid.org/0000-0003-1363-5489
Yurchak Iryna, Associate Professor at the Department of Automated Design Systems
https://orcid.org/ 0009-0005-9100-8511
National University «Lviv Polytechnic», Lviv, Ukraine

SOFTWARE IMPLEMENTATION OF IMAGE STEGANOGRAPHY BASED ON LSB ALGORITHM WITH CAESAR'S CIPHER

Struk I., Yurchak I. Software implementation of image steganography based on LSB algorithm with Caesar's cipher. The purpose of this work is to conduct a study of computer steganography methods and the implementation of image steganography using the LSD algorithm and the Caesar cipher. The development of global computer networks allows fast and efficient transfer of electronic documents, but at the same time there is a risk of illegal copying and distribution of materials. Confidentiality of important information is an extremely responsible matter. Therefore, there is a need to develop methods and software tools for hiding information in various types of files, in particular images. Steganography is a mechanism for hiding messages in cloaking objects such as images, audio, text, video, and network protocol so that the hidden message cannot be detected by the naked eye or attackers.
Keywords: steganography, image steganography, information security, algorithm.

Струк І.Т., Юрчак І.Ю. Програмна реалізація стеганографії зображення на основі алгоритму LSB з шифром Цезаря. Метою даної роботи є проведення дослідження методів комп'ютерної стеганографії та реалізації стеганографії зображень з використанням алгоритму LSD та шифру Цезаря. Розвиток глобальних комп'ютерних мереж дозволяє швидко та ефективно передавати електронні документи, але водночас існує ризик нелегального копіювання та розповсюдження матеріалів. Конфіденційність важливої інформації – справа надзвичайно відповідальна. Тому виникає потреба у розробці методів і програмних засобів для приховування інформації у файлах різних типів, зокрема зображень. Стеганографія — це механізм приховування повідомлень у маскувальних об'єктах, таких як зображення, аудіо, текст, відео та мережевий протокол, щоб приховане повідомлення не було виявлено неозброєним оком або зловмисниками.
Ключові слова: стеганографія, стеганографія зображення, інформаційна безпека, алгоритм.

Global computer networks have made it possible for electronic documents to be transferred quickly and effectively, but there is also a chance that materials will be illegally copied and distributed. Maintaining the privacy of critical information is a very serious issue. Thus, techniques and software tools for concealing data in different kinds of files—especially images—need to be developed.

Image steganography is a crucial technology for cyberspace security, and its study is strategically significant. A single algorithm is used in fewer instances of image steganography as the technology becomes more widely used; instead, more adaptive image steganography is employed.There is a place for steganography in data security. Rather than taking the place of cryptography, it enhances it.

1. Stegosystem model and its types.

Steganography techniques only conceal the fact that information is being transmitted; they do not alter the content of the message. Additionally, they add an extra layer of security if the message is encrypted beforehand.

Figure 1 shows an example of a steganographic system, which is a collection of devices and techniques for establishing a covert channel for information transmission [1].

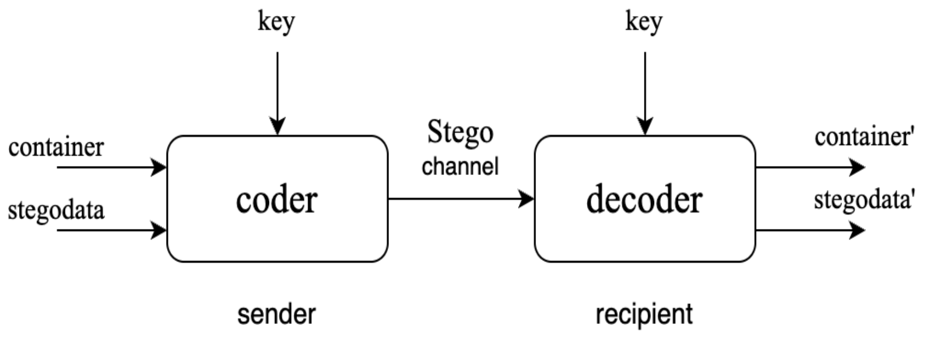


Fig. 1. Stegosystem model

Any information such as text, messages, images, etc. can be used as input data. In general, the term "container" should be used, since it can cover both text and images, as well as, for example, audio data. The term "container" is used to indicate hidden information in this master's qualification thesis.

The described model consists of the following objects:

− stegodata – data of any type embedded in the container;

− container – any information suitable for hiding in it
messages;

− key – secret key needed for encryption and decryption of messages. It is used to strengthen protection;

− a steganographic channel or simply a steganochannel - a transmission channel of a steganocontainer.

The sender chooses a container in which the data will be embedded and the embedding data, such as some text. In the encoder, data is encoded using a key. Next, an encoded container with data is transmitted over the stegochannel, which the recipient must also decode using a key. As a result, the recipient has decrypted information.

According to the type of steganokey (secret key needed to hide information), the steganosystem model can be:

- with a secret key;
- with an open key.

In the case of a secret key, a single key is used, transmitted over a secure channel. But in the case of a public key - different keys that can be transmitted freely over an unprotected channel. Such a system works effectively even when there is mutual distrust between the sender and the recipient [2].

2. The LSB method

Using the least significant bit method is a simple and common approach to hiding information in a container. You can consider the structure of such a container using the example of an image file in BMP format (BitMap image — bitmap image). Here, the image is stored without loss of quality, so the size of these files is quite large.

The BMP file can be conditionally divided into 4 parts: file header, image title palette, image.

The first two bytes in the file header are defined as the BM signature (this is the BitMap signature or identifier of the bitmap file), by which the computer program recognizes that this is an image file in the BMP format. Next, the next four bytes record the size of the file, and the next four bytes are reserved and must contain zeros. The next sequence of four bytes specifies the offset from the beginning of the file to the bytes that represent the image.

In a BMP format file with a dimension of 24 bits per pixel, information about each pixel is encoded by three RGB bytes (Red, Green, Blue). In the LSB (Least Significant Bit) steganography method, the bits of the secret message are embedded instead of the lower bits (the last, smallest, highlighted in blue in Table 2) in the bytes responsible for color coding. Changing the shade of a pixel's color when the last bit of the palette is replaced by "0" or "1" is almost imperceptible to the human eye [5].

Representation of the bytes responsible for color coding before input steganograms are given in table 1.

Table 1 - Representation of bytes before entering a steganogram

1		0	0	0	0	0	0	1	R = 65	🟩
1		0	1	1	0	0	0	0	G = 176	🟩
0		1	1	0	1	0	1	0	B = 106	🟩

The representation of the bytes responsible for color coding after entering the steganogram into the last significant bits is given in the table 2.

Table 2 - Presentation of bytes after entering the steganogram

1	0	0	0	0	0	0	0	R’ = 64	🟩
1	0	1	1	0	0	0	1	G’ = 177	🟩
0	1	1	0	1	0	1	1	B’ = 107	🟩

The capacity of a 24-bit BMP file as a stegocontainer is maximized if the last bits of all bytes encoding the duration of the three colors are used. In this case, this capacity is one eighth of the volume of the image file. For example, a 1080 by 1080 pixel image file can hide a steganogram with very large text. Calculations indicate that up to 437,400 (1080 x 1080 x 3 x 1/8 = 437400) bytes of textual information can fit in an image.

If you use only the last bits of the bytes that are responsible for the blue color transfer, this will reduce the capacity of the image file as a stegocontainer by three. Nevertheless, it will be almost impossible to visually detect the presence of a steganogram in such an image, since the human eye poorly distinguishes shades of blue.

3. Caesar cipher with key

The Caesar Cipher is a simple encryption technique used by Julius Caesar to communicate confidentially with his allies. The basic idea is to shift the letters in a text message by a certain number of positions, called an "offset" or "key."

The Caesar cipher is one of the oldest and simplest encryption methods. This is a kind of substitution cipher, where each letter of the text is replaced by a letter located a fixed number of positions in the alphabet. For example, when shifting one position, the letter A becomes B, B becomes C, and so on[6].

To encrypt, you need to know an integer known as an offset, which indicates the number of positions that each letter of the text is shifted down. Encryption can be represented using modular arithmetic by turning letters into numbers where A = 0, B = 1,..., Z = 25. For example, at shift 5, A is replaced by F, B becomes G, C becomes H, and so on. The alphabet rotates, that is, after Z, it starts again with A (Fig. 2).

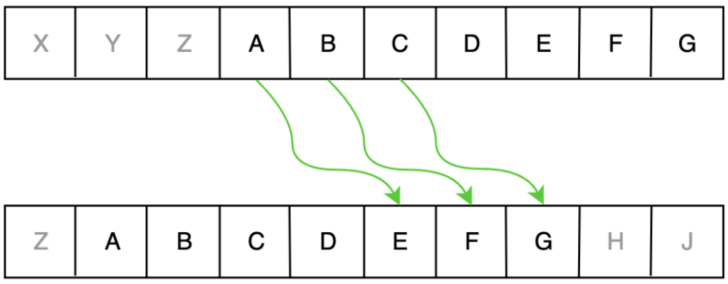


Fig. 2. Alphabet if offset is 4

An example of using the Caesar cipher to encrypt the message "WORLD" with an offset of 4:

- write the message in plain text: WORLD;
- select the offset value (in this case, offset 4 is used);
- replace each letter in the text message with the letter that occupies four positions to the right in the alphabet (W becomes A (offset 4 from W), O becomes S, R becomes V, L becomes P, D becomes H);
- the encrypted message is now "ASVPH".

To decipher a message, you simply need to move each letter back the same number of positions. In this case, you need to shift each letter in "ASVPH" back 4 positions to get the original "WORLD" message.

4. Information system design

To design the system, a block diagram of the system algorithm, a class diagram and a use case diagram were developed.

This system will have two main functions: image encoding and decoding.

Thus, the above functions, with a detailed description of their operation, should cover almost all the functionality that is required to have a complete algorithm of the information system.

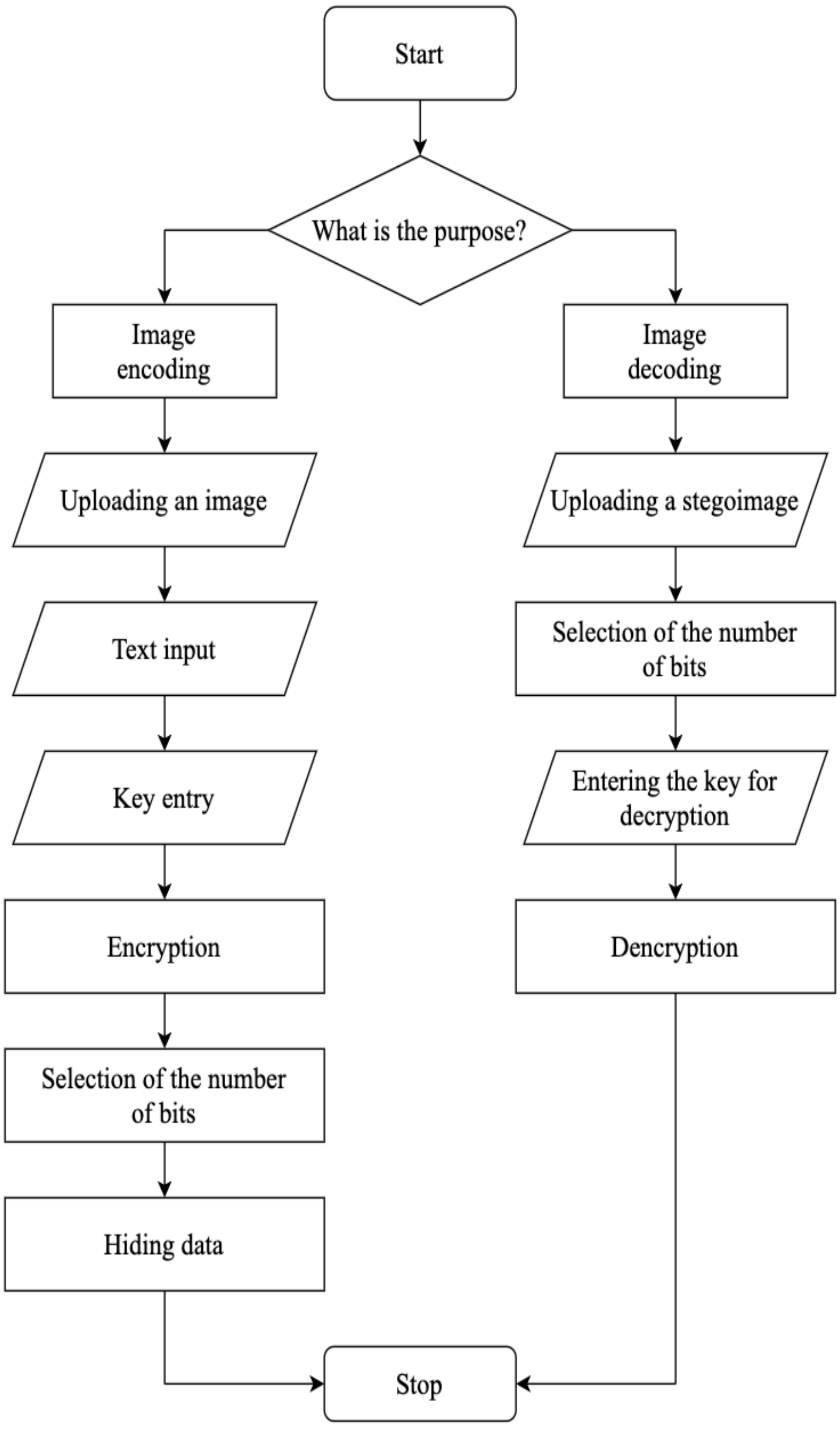


Fig. 3. Block diagram of the system algorithm

For a better understanding of how the system works, below is a class diagram with their detailed description.

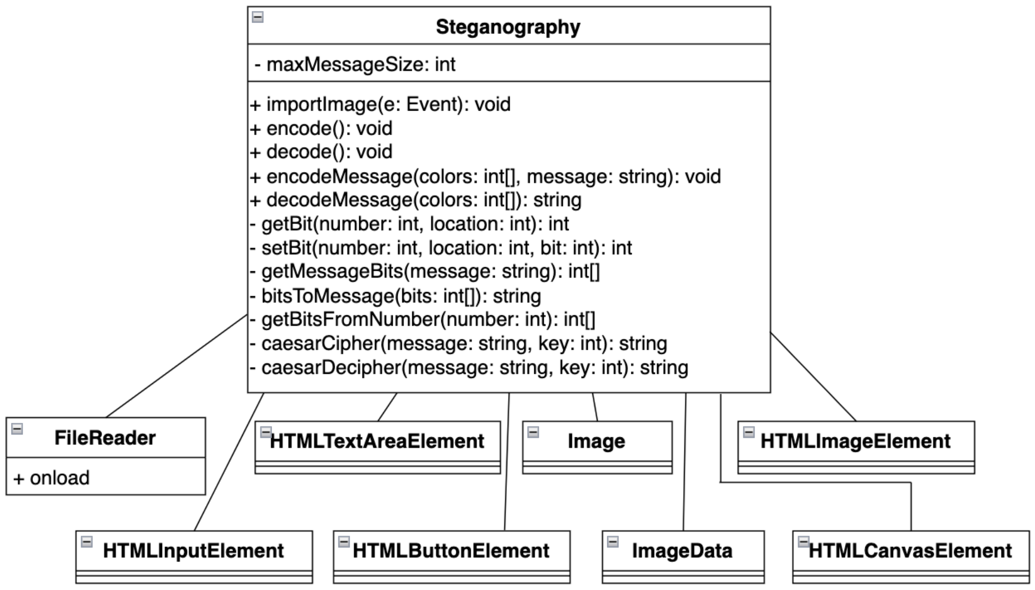

Steganography
- maxMessageSize: int
+ importImage(e: Event): void
+ encode(): void
+ decode(): void
+ encodeMessage(colors: int[], message: string): void
+ decodeMessage(colors: int[]): string
- getBit(number: int, location: int): int
- setBit(number: int, location: int, bit: int): int
- getMessageBits(message: string): int[]
- bitsToMessage(bits: int[]): string
- getBitsFromNumber(number: int): int[]
- caesarCipher(message: string, key: int): string
- caesarDecipher(message: string, key: int): string
FileReader
+ onload
HTMLTextAreaElement
Image
HTMLImageElement
HTMLInputElement
HTMLButtonElement
ImageData
HTMLCanvasElement

Fig. 4. Class diagram of the system

- ImportImage(e: Event): void: This method is called when an image file is selected. It reads the selected image file, displays a preview, and sets up the canvas for further processing. After loading the image, it calls the decode method.
- Encode(): void: This method is called when the Hide Message button is clicked. It receives the message and password from the user's input fields, encrypts the message using a Caesar cipher, and then encodes the encrypted message into an image using residual bit (LSB) encoding. The resulting image is displayed and a message appears.
- Decode(): void: This method is called when the Open Message button is clicked. It retrieves the password from the user's input field, decodes the hidden message from the image using residual bit (LSB) decoding, decrypts the message using a Caesar cipher, and displays the decrypted message on the page.
- caesarCipher(message: String, key: int): String: This method takes a message and a key as input and performs caesar cipher encryption. It shifts each letter in the message by the specified number of keys.
- caesarDecipher(message: String, key: int): String: This method takes an encrypted message and a key as input and performs Caesar cipher decryption. It shifts each letter in the message back by the specified number of keys to recover the original message.
- encodeMessage(colors: Array, message: String): void: This method takes an array of color values (pixel data) and a message as input. It encodes the message bits into the least significant bits (LSB) of the color values to hide the message in the image.
- decodeMessage(colors: Array): String: This method takes an array of color values (pixel data) as input and decodes the hidden message from the least significant bits (LSB) of the color values.
- getBit(number: int, location: int): int: The given number and location of the bit, this method gets the value of the specified bit in the number.
- setBit(number: int, location: int, bit: int): int: Given a number, bit location, and bit value, this method sets the specified bit in number to the specified value.
- getMessageBits(message: String): Array: Given a message, this method converts each character of the message into an array of bits.
- bitsToMessage(bits: Array): String: Given an array of bits, this method converts the bits to a string.
- getBitsFromNumber(number: int): Array: Given a number, this method converts the number into an array of bits.

Use case diagrams represent the elements of the use case model.

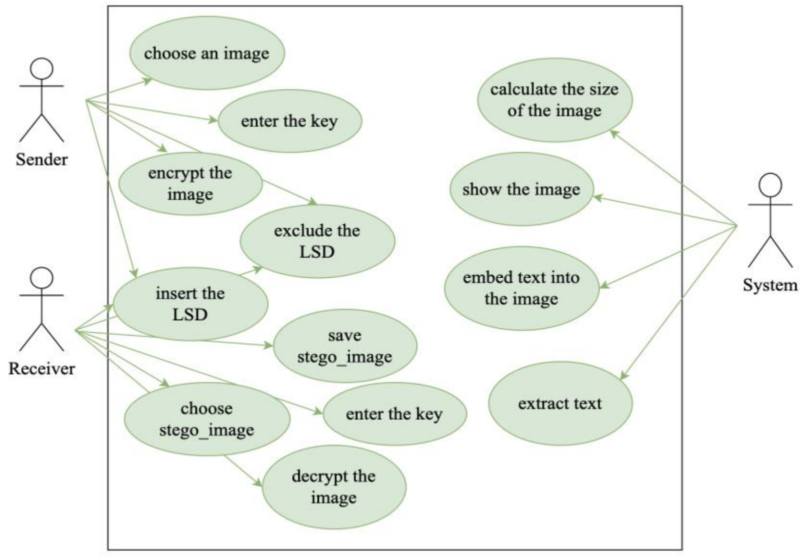


Fig. 5. Use case diagram

Use cases are described below.

Entity: Sender

Use cases: select image, enter key, encrypt image, remove LSD, insert LSD.

Description: The user initiates the image import process by selecting a file. This action triggers the system to read the selected image file and display a preview. The sender enters the message to be encoded. After entering the message, the user initiates the encoding process. The system checks whether the size of the message meets the limit. If so, it encrypts the message using a Caesar cipher and embeds it in the image using LSB steganography. The encoded image will then be displayed and a notification will appear.

Acting person: Recipient

Use cases: save stegimage, select stegimage, enter key, decrypt image, extract LSD, insert LSD.

Description: the recipient can save the received stegoimage. This action triggers the system to allow the user to save the image. The recipient also initiates the process of decoding the message from the current image. The system extracts the hidden message using LSB steganography and then decrypts it using a Caesar cipher. A decrypted message will be displayed.

Actor: System

Use cases: calculate image size, display image, embed text in image, extract text.

This use case diagram provides an overview of the main interactions between users and the steganography system. Actors are represented as figures, and usage options are shown as ovals within the system. Lines with arrows indicate the direction of interaction.

5. Software implementation

The least significant bit replacement method of hiding information in different image formats is not very resistant to compression and other types of container distortions because these operations have the potential to reveal hidden information. Furthermore, this technique is susceptible to steganoanalytic attacks, which can disclose the existence and type of hidden data.

Cryptographic techniques can be used to increase the level of protection of hidden information in an image that uses a replacement. A cryptographic algorithm like the Caesar cipher can be used to encrypt the message before it is embedded into the picture.

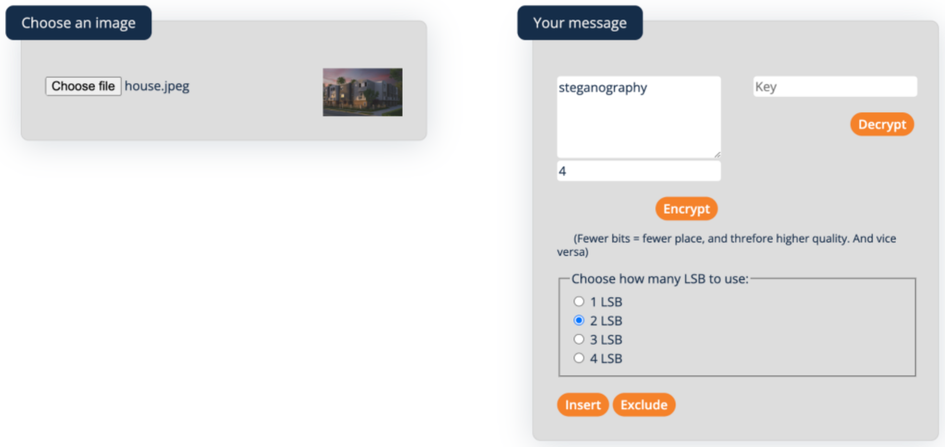
The user can choose how many message bits in this study are to be replaced by the pixel's least important bits.

The image is first loaded, and then its useful capacity is ascertained. UTF-8 encoding is used to encrypt user-entered text and convert it to binary format. Every character in this encoding has a different number of bytes to represent it; for instance, Ukrainian letters take up two bytes, some special characters take up three bytes, and Latin letters, numbers, and punctuation need one byte.

The user can then choose the order in which the R, G, and B components of the image are replaced. The signal-to-noise ratio is computed using altered image components. An image with embedded data is made by making a copy of it and making the required adjustments to it. The image with embedded data is stored separately from the original, which stays unaltered.

Click on the Choose file button and select the desired image. After that, an additional window appears on the right with the main functions(see Fig. 6). Next, you can enter the key to encrypt with the Caesar cipher. Select the number of bits in which the information will be hidden. Also, it is possible to extract information by doing all the actions in the reverse order.

First, you need to download the container file by clicking on the "Choose file" button. After that, enter the text in the field under the inscription "Your message". Enter the key, click "Encrypt". The text is now presented in encrypted form. Next, select the number of the smallest bits and the "Insert" button. A watermark appears along with the hidden message.


Choose an image

Choose file house.jpeg

Your message

steganography

Key

Decrypt

4

Encrypt

(Fewer bits = fewer place, and threfore higher quality. And vice versa)

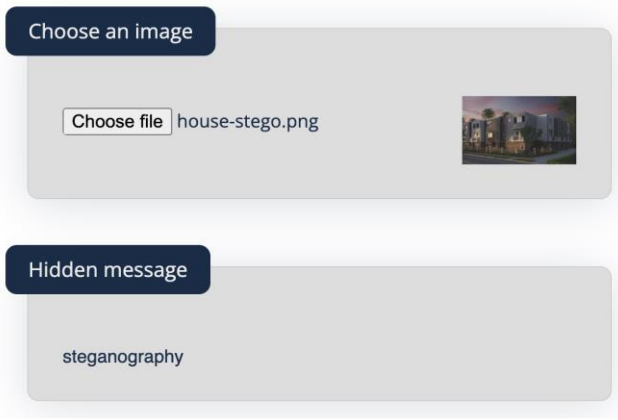
Choose how many LSB to use:
- ○ 1 LSB
- ● 2 LSB
- ○ 3 LSB
- ○ 4 LSB

Insert Exclude

Fig. 6. The main functions of the application

To save this image, just click "Save" and sign the newly created file. Data extraction can be done immediately. If you need to do this from another image, just open it by clicking "Choose file".

For extraction, everything is done in the opposite way. First, select the number of bits and click "Remove" as shown. Next, enter the key and "Decrypt". Extraction was successful (see Fig. 7).


Choose an image

Choose file house-stego.png

Hidden message

steganography

Fig. 7. Decryption of data

6. Application testing

To evaluate the results of the program operation, two methods of steganoanalysis were used - chi-square[4] attack and RS-attack[3]. These methods are statistical in nature and are used to detect the possible presence of a hidden message in areas of pixels. In particular, they are effectively applied when using the standard sequential method of making changes based on the least significant bit.

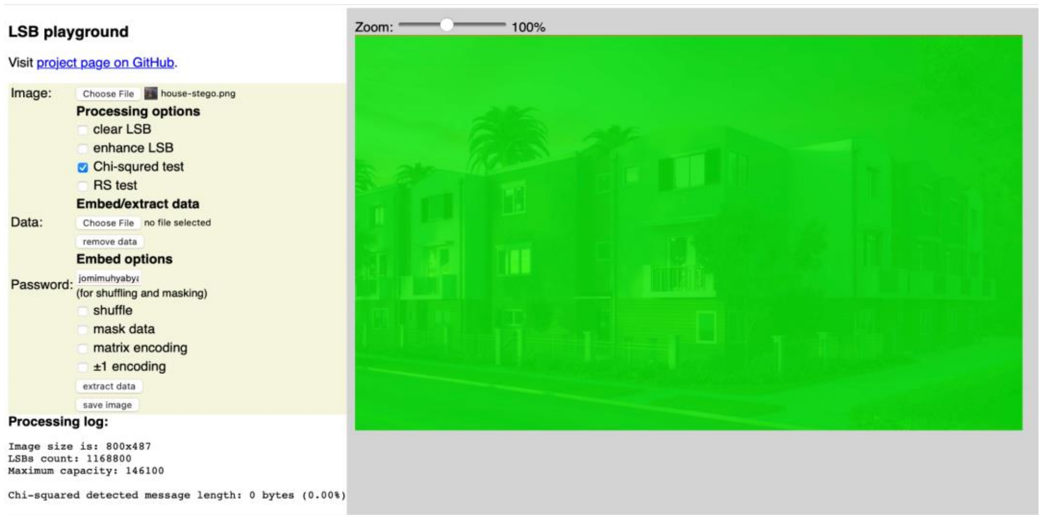
To conduct the experiment, a steganocontainer is used, in which an encrypted message was embedded:

loremipsumdolorsitametconsecteturadipiscingelitduisetleoeuturpislobortisa-liquamvelatsemvestibulum-necaliquameratvitaedictumnullasedfeugiatultricieslacus-necdictumsapien-gelitduisemetconsecteremipsumdolor.

The size of this distribution segment is 204 bytes, since its length is 204 characters, and the encoding is in the UTF-8 system, hence the size of 13 bytes is obtained.

The size of the container is 8,128 bytes. That is, the message is 2.5% of the size of the container.

To reveal the volume of hidden information in the container, which is subject to steganoanalysis, two attacks were used - chi-square attack and RS-attack. The results of these attacks are presented in fig. 8 and 9.



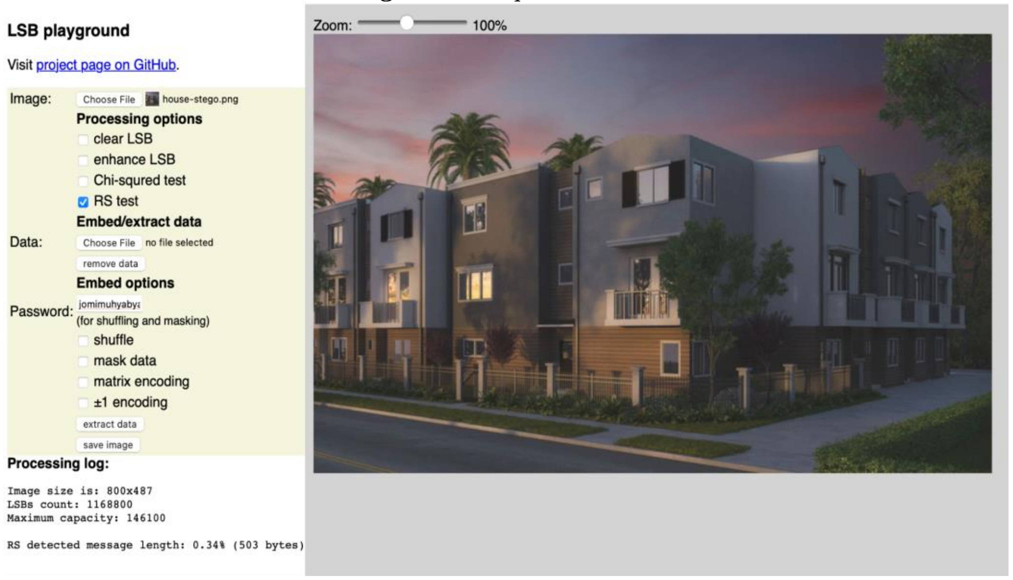
Chi-squred test [✓]

Processing log:

```
Image size is: 800x487
LSBs count: 1168800
Maximum capacity: 146100

Chi-squared detected message length: 0 bytes (0.00%)
```

Fig. 8. Chi-Square Attack



RS test [✓]

Processing log:

```
Image size is: 800x487
LSBs count: 1168800
Maximum capacity: 146100

RS detected message length: 0.34% (503 bytes)
```

Fig. 9. RS-attack of steganocontainer

Based on the results of the attacks, it can be seen that their results are inaccurate, which indicates the high stability of the used concealment method for the given length of the embedded message. A chi-square attack failed to detect hidden information because this attack aims to detect statistical differences, but did not detect unusual patterns or embedded data. In turn, the RS attack was able to recognize only 0.34% of the data from the total volume of the container, although 2.5% of the container size was embedded.

On the other hand, the RS attack successfully recognized certain features of the embedded information that are different from the normal image. However, this percentage still falls short of the full recovery of the embedded message. Even if the image is opened, extracting information from it will remain a difficult task.

The results of test attacks are given in table 3.

Table 3 - Tests results

Type of attack	% of embedded text in the container	% of detection
Xi	2,5%	0%
RS	2,5%	0,34%

Conclusion

In this work, the theoretical content of computer steganography methods was investigated and a software product based on the LSB algorithm was implemented and the Caesar cipher was used to ensure the highest level of confidentiality. The next stage involved modeling and designing an information system that uses steganography methods to protect data in images. To ensure a higher level of protection of hidden information in the image, cryptographic methods were used, and the message itself was encrypted using such a cryptographic algorithm as the Caesar cipher.

The process of testing the developed solution using RS-attack and Chi-square attack is considered. Tests were conducted, the results of which are presented in the appropriate tables, and the result is demonstrated on the screenshots. A chi-square attack failed to detect hidden information because this attack aims to detect statistical differences, but did not detect unusual patterns or embedded data. In turn, the RS attack was able to recognize only 0.34% of the data from the total volume of the container, although 2.5% of the container size was embedded.

References

1. Hashim, M.M.; Rahim MS, M.; Johi, F.A.; Taha, M.S.; Hamad, H.S. Performance evaluation measurement of image steganography techniques with analysis of LSB based on variation image formats. *Int. J. Eng. Technol.* 2018, *7*, 3505–3514. [Google Scholar]

2. Sahu, M.; Padhy, N.; Gantayat, S.S.; Sahu, A.K. Local binary pattern-based reversible data hiding. *CAAI Trans. Intell. Technol.* 2022, *7*, 695–709. [Google Scholar] [CrossRef]

3. Mansour RF, Abdelrahim EM (2019) An evolutionary computing enriched RS attack resilient medical image steganography model for telemedicine applications. Multidimens Syst Signal Process 30(2):791–814. https://doi.org/10.1007/s11045-018-0575-3 (date of access: 21.11.2023).

4. Chi-square [Electronic resource] – Resource access mode: https://www.investopedia.com/terms/c/chi-square-statistic.asp (date of access: 26.11.2023).

5. Pelosi, M.; Easttom, C. Identification of LSB image steganography using cover image comparisons. *J. Digit. Forensics Secur. Law* 2021, *15*, 6. [Google Scholar]

6. Gaurav Shrivastava, Using Letters Frequency Analysis in Caesar Cipher with Double Columnar Transposition Technique, Interna tional Journal of Engineering Sciences & Research Technology. Vol. 2 Issue 6 ,Page No. 1475-1478, 01 June, 2013, ISSN: 2277-9655.