VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY

MINI PROJECT

STEGANOGRAPHY

Lecturer:	Nguyễn Đức Thái
Student:	Nguyễn Duy Thuận - 1752526 Tô Hoàng Đạo - 1852309 Lâm Trịnh Long - 1811044

Ho Chi Minh City, June 2022

Content

1 Introduction of Steganography

Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination. Steganography can be used to conceal almost any type of digital content, including text, image, video or audio content; the data to be hidden can be hidden inside almost any other type of digital content. The word steganography is derived from the Greek words steganos (meaning hidden or covered) and the Greek root graph (meaning to write).

The use of steganography can be combined with encryption as an extra step for hiding or protecting data. The content to be concealed through steganography is often encrypted before being incorporated into the innocuous-seeming cover text file or data stream. If not encrypted, the hidden text is commonly processed in some way in order to increase the difficulty of detecting the secret content.

2 File Types and Methods

There are many different steganography types and methods. Here are some file types that can be use for steganography and some methods to implement each types of file:

2.1 Text Steganography

Text steganography can be achieved by altering the text formatting, or by altering certain characteristics of textual elements (e.g., characters). The goal in the design of coding methods is to develop alterations that are reliably decodable (even in the presence of noise) yet largely indiscernible to the reader. [3]
Some methods for text steganography:

- **Line-shift Coding:** This is a method of altering a document by vertically shifting the locations of text lines to encode the document uniquely.
- **Word-shift Coding:** This is a method of altering a document by horizontally shifting the locations of words within text lines to encode the document uniquely.
- **Feature Coding:** This is a method that is applied either to a format file or to a bitmap image of a document. The image is examined for chosen text features, and those features are altered, or not altered, depending on the codeword. There are many possible choices of text features; here, we choose to alter upward, vertical endlines. These endlines are altered by extending or shortening their lengths by one (or more) pixels, but otherwise not changing the endline feature.

2.2 Image Steganography

To hide a message inside an image without changing its visible properties, the cover source can be altered in ”noisy” areas with many color variations, so less attention will be drawn to the modifications. [3]
Some methods for image steganography:

- **Least Significant Bits (LSB):** This is a method embed the bits of the message directly into least significant bit plane of the cover image in a deterministic sequence. The result changes that are made to the least significant bits are too small to be recognized by the human visual system (HVS), so the message is effectively hidden.

- **Masking and filtering:** This is a method create markings in an image, usually restricted to 24 bits or grayscale images. This can be achieved for example by modifying the luminance of parts of the image. While masking does change the visible properties of an image, it can be done in such a way that the human eye will not notice the anomalies.Since masking uses visible aspects of the image, it is more robust than LSB modification with respect to compression, cropping and different kinds of image processing. The information is not hidden at the ”noise” level but is inside the visible part of the image, which makes it more suitable than LSB modifications in case a lossy compression algorithm like JPEG is being used.

- **Transformations:** This is a complex method of hiding a secret inside an image comes with the use and modifications of discrete cosine transformations. Discrete cosine transformations (DCT), are used by the JPEG compression algorithm to transform successive 8 x 8 pixel blocks of the image, into 64 DCT coefficients each.

2.3 Audio Steganography

In audio steganography, secret message is embedded into digitized audio signal which result slight altering of binary sequence of the corresponding audio file. There are several methods are available for audio steganography. We are going to have a brief introduction on some of them. [3]
Some methods for audio steganography:

- **LSB Coding:** This is a method replace LSB of binary sequence of each sample of digitized audio file with binary equivalent of secret message.

- **Phase Coding:** This is a method encodes the secret message bits as phase shifts in the phase spectrum of a digital signal, achieving an inaudible encoding in terms of signal to noise ratio.

- **Spread Spectrum:** There are two approaches are used in this technique: the direct sequence spread spectrum (DSSS) and frequency hopping spread spectrum (FHSS).
  DSSS transmissions multiply the data being transmitted by a "noise" signal. This noise signal is a pseudo random sequence of 1 and 1 values, at a frequency much higher than that of the original signal, thereby spreading the energy of the original signal into a much wider band.
  In contrast, FHSS pseudo-randomly retunes the carrier, instead of adding pseudo-random noise to the data, which results in a uniform frequency distribution whose width is determined by the output range of the pseudo-random number generator.

- **Echo Hiding:** This is a method embed the secret message into cover audio signal as an echo. Three parameters of the echo of the cover signal namely amplitude, decay rate and offset from original signal are varied to represent encoded secret binary message. They are set below to the threshold of Human Auditory System (HAS) so that echo can’t be easily resolved.

3 The Least Significant Bit Method

Digital data is computed in binary format, and similarly to numerical notation, the right digit is considered the lowest digit whereas the leftmost is considered the highest digit. Due to the positional notation, the least significant bit is also known as the rightmost bit. In a multi-bit binary number, the significance of a bit decreases as it approaches the least significant bit. Since it is binary, the most significant bit can be either 1 or 0.

A simple approach for embedding information in cover image is using Least Significant Bits (LSB). The simplest steganography techniques embed the bits of the message directly into least significant bit plane of the cover image in a deterministic sequence. Modulating the least significant bit does not result in human-perceptible difference because the amplitude of the change is small. To hide a secret message inside an image, a proper cover image is needed. Because this method uses bits of each pixel in the image, it is necessary to use a lossless compression format, otherwise the hidden information will get lost in the transformations of a lossy compression algorithm. When using a 24-bit color image, a bit of each of the red, green and blue color components can be used, so a total of 3 bits can be stored in each pixel. For example, the following grid can be considered as 3 pixels of a 24-bit color image, using 9 bytes of memory.

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

Initial pixel

When the character A, which binary value equals 10000001, is inserted, the following grid results:

(0010011$\underline{\mathbf{1}}$ 1110100$\underline{\mathbf{0}}$ 1100100$\underline{\mathbf{0}}$)
(0010011$\underline{\mathbf{0}}$ 1100100$\underline{\mathbf{0}}$ 1110100$\underline{\mathbf{0}}$)
(1100100$\underline{\mathbf{0}}$ 0010011$\underline{\mathbf{1}}$ 11101001)

Changed pixel

In this case, only three bits needed to be changed to insert the character successfully. On average, only half of the bits in an image will need to be modified to hide a secret message using the maximal cover size. The result changes that are made to the least significant bits are too small to be recognized by the human visual system, so the message is effectively hidden. As you see, the least significant bit of third color is remained without any changes.

4 Steganalysis

Steganalysis [2] seeks to detect the presence of hidden messages in steganography works. None of the existing practical steganographic algorithms achieves perfect security in the information-theoretic model for steganography [1] proposed by Christian Cachin. Therefore, warden can apply different attempts to discover our secret messages. In this section, we consider the various strategies of steganalysis that are available to the warden.

4.1 Steganalysis scenarios

As we know, the choice of steganalysis algorithm depends on the operational scenarios influenced by a various issues and constraints. There are three types of warden: passive, active, and malicious.

- **Passive warden:** acts as **Man-in-The-Middle**, intercepts communications between sender and receiver, then tests for the presence of a hidden messages. If no hidden message is detected, then the warden forwards the communications to receiver. Conversely, the warden blocks the transmission and forward neither the communication nor the hidden message if a hidden message is detected.
- **Active warden:** is permitted more freedom. In particular, an active warden is free to modify the communication between sender and receiver. Thus, even if the warden's steganalysis test fails, the communication may still be altered in an attempt to remove any hidden message that might evade detection. This type of attacker could make the situation worse because the majority of steganographic algorithms are not designed to withstand modification to the cover Work.
- **Malicious warden:** go one step further. they might attempt to impersonate sender and send receiver false messages. To do so requires warden to be capable of much more than the detection of a covert message. They must also know what steganographic algorithm sender and receiver are using and any associated steganographic and encryption keys.

The fundamental goal of the warden is to reliably detect the presence of a hidden message in communications between sender and receiver.

4.1.1 Detection

The detection of a covert message hidden in a cover Work is most often modeled as a classification problem. A steganalysis algorithm receives Works as input and classifies them as either cover Works or stego Works. There are two class of steganalysis:

- **Targeted steganalysis:** the warden may be certain that sender and receiver are covertly communicating and even know the steganographic algorithm being used.
- **Blind steganalysis:** the warden knows nothing and only has some level of suspicion that sender and receiver are covertly communicating.

In theory, if we knew the probability distribution of cover and stego Works in the space of all possible Works, we could build an optimal classifier using the likelihood ratio test. The classifier can make two types of error—deciding that a cover Work is a stego Work, which is a false alarm, and deciding that a stego Work is a cover, which is a missed detection.

Unfortunately, in reality we cannot construct a classifier this way because the probability distributions of cover and stego Works will never be exactly known due to the high dimensionality of the space of all Works. To reduce the dimensionality, Works are represented using one or more features and the classifier is designed in this lower dimensional space.

Usually, it is not possible to exactly recover the cover Work from the stego Work. Nevertheless, for some steganographic methods it is feasible to obtain an approximate estimate of the cover Work, which then can be utilized to derive useful features for steganalysis. The process of estimating the cover Work from the stego Work is called calibration.

1) **Targeted Steganalysis**
In targeted steganalysis, we know the steganographic algorithm that is being used. This provides a powerful clue with which to choose the feature representation. A clever warden may notice that the changes due to LSB embedding are not symmetrical. In fact, in LSB embedding, an even pixel value is never decreased (it is either increased or remains the same). Conversely, odd values are either decreased or remain the same. It is possible to exploit this asymmetry for the design of a targeted classifier. We will construct a detector of LSB embedding and an estimator of the number of embedding changes. Thus, the result of the steganalysis will not be just a binary decision as to the presence or absence of a hidden message, but a value that is strongly related to the length of the secret message.

2) **Blind Steganalysis**
Blind steganalysis has the same set of issues as targeted steganalysis, that is, the choice of feature representation and the determination of the decision boundary. However, the problem is made much more difficult by the fact that the steganographic algorithm being used by sender and receiver is unknown. Given a feature representation we must then train the classifier. Typically, during the training phase, a classifier needs two labeled sets, one of cover Works and the other of stego Works. However, for blind steganalysis we do not have the stego Works with which to train the classifier. Therefore, we have two options:

   - **First option:** we train the blind steganalysis classifier using stego Works that are generated from a wide variety of known steganalysis algorithms.
   - **Second option:** train the blind steganalysis classifier using only the cover Works.

4.1.2 Forensic Steganalysis

A malicious warden might attempt to impersonate sender and receiver false messages. To do so requires warden to be capable of much more than the detection of a covert message. They must also know what steganographic algorithm Alice and Bob are using and any associated steganographic and encryption keys. This higher level of analysis, which is more than just detection, is known as forensic steganalysis. The warden can begin by trying to recover some attributes of the embedded message and properties of the stego algorithm. If the message has been sequentially embedded, then the warden can can use the histogram attack. They may guess which stego method has been used and attempt to determine the stego key and extract the embedded message. If the message is encrypted, warden then needs to perform cryptanalysis on the extracted bitstream.

4.2 Some significant steganalysis algorithms

4.2.1 LSB Embedding and the Histogram Attack

For LSB embedding, even pixel values are either left unmodified or increased by 1, while odd pixel values are either left unmodified or decreased. Thus, the grayscale values (2i, 2i + 1) form a pair of values (PoV) that are exchanged into each other during embedding.

- **Assumption:**
  - Let $T_c[j] = 0, ..., 255$ Let Tc[j], j = 0 , ..., 255 denote the intensity histogram of the cover image
  - T_s be the corresponding histogram of the stego image after embedding qn bits.
    Where:

* n is the total number of pixels and $0 \leq q \leq 1$.
* we are embedding q bits per pixel (bpp).
* q is the relative message length.

Now we calculate T_s as a function of T_c and q. Assuming that the secret message is a random bitstream, which is a reasonable assumption if the message is encrypted or compressed, the expected number of pixels with grayscale 2i that will be modified to 2i + 1 is $\frac{q}{2}T_c$[2i]. This is because on average one-half of the message bits will already match the LSB of the pixels. Similarly, the expected number of pixels with grayscale 2i + 1 that will be changed to 2i is $\frac{q}{2}T_c$[2i+1]. Thus, we can write

$$\begin{aligned} E\{T_s[2i]\} &= (1-\frac{q}{2})T_c[2i] + \frac{q}{2}T_c[2i+1] \\ E\{T_s[2i+1]\} &= \frac{q}{2}T_c[2i] + (1-\frac{q}{2})T_c[2i+1] \end{aligned} \tag{1}$$

If the image is fully embedded, the histogram bins 2i and 2i + 1 will have approximately the same values. As the result, with (q = 1), E{Ts[2i]} = E{Ts[2i + 1]}.

We know that the sum T_s[2i] + T_s[2i + 1] = T_c[2i] + T_c[2i + 1] is invariant under LSB embedding. The theoretically expected value for T_s[2i] is therefore T_s[2i]=(T_s[2i] + T_s[2i + 1])/2. Thus, it is possible to detect LSB embedding for q = 1 by testing whether T_s[2i] = T_s[2i + 1].

Next, we apply Pearson's chi-squared test to determine whether T_s[2i] = T_s[2i]. The chi-square test starts by calculating the chi-square test statistics S,

$$S = \sum_{i=1}^{k} \frac{(T_s[2i] - \overline{T_s}[2i])^2}{\overline{T_s}[2i]}$$

with (k - 1) = 127 degrees of freedom. Assuming that even grayscale values indeed follow the probability mass function, $\overline{T_s}[2i]$, the test statistic, S, follows the chi-square distribution with k 1 degress of freedom. We note that in practice, unpopulated bins (grayscales) must be merged so that $\overline{T_s}[2i] > 4$ to ensure that S is approximately chi-square distributed. Intuitively, a small value of S indicates that the data follows the expected distribution and we can conclude that the image contains a message embedded using LSB embedding. On the other hand, large values of the test statistic imply that no message is embedded. The statistical significance of S is measured using the so-called p-value, which is the probability that a chi-square distributed random variable with k - 1 degrees of freedom would attain a value larger than or equal to S:

$$P(S) = \frac{1}{2^{\frac{k-1}{2}}\Gamma(\frac{k-1}{2})} \int_{S}^{\infty} e^{-\frac{x}{2}} x^{\frac{k-1}{2}-1} dx$$

If the image does not contain a hidden message, S is large and $p(S) \approx 0$. In practice, we calculate a threshold value S_{th} so that $p(S_{th}) = \alpha$, where is a chosen significance level. For a confidence level, α, we decide that a Work contains a secret message if p(S) $> \alpha$, or equivalently S<S_{th}.

So far, the derivations have been carried out for a fully embedded image with q = 1 bpp. If the stego image was embedded sequentially but perhaps not fully, we can scan the stego image in the same order in which the message has been embedded and evaluate p for the set of visited pixels. After a short transient initial phase, the p-value will be close to 1 and it will suddenly fall to zero when we arrive at the end of the message and it will stay at zero until we exhaust all pixels. If the message-carrying pixels in the image are selected pseudo-randomly rather than sequentially, this test is ineffective unless.

4.2.2 Sample Pairs Analysis

The histogram attack on LSB steganography is based on the observation that in a fully embedded image, the number of pixels with grayscale 2i and 2i + 1 should be approximately the same. This attack, however, cannot be applied when the image is only partially embedded along a pseudo-random path. Steganalysis methods that only use first-order statistics, such as histograms of pixels, ignore the dependency among neighboring pixels in natural images. However, Sample pair analysis is reliable and accurate detection algorithms if the spatial correlation within images is utilized.

Let P denote the set of all horizontally adjacent pixel pairs in the image. Let us partition P into three disjoint subsets, X, Y, and Z, where:

$$X = \{(u, v) \in P | (\text{v is even and u<v) or (v is odd and u>v})\}$$
$$Y = \{(u, v) \in P | (\text{v is even and u>v) or (v is odd and u<v})\}$$
$$Z = \{(u, v) \in P | u = v\}.$$

Furthermore, let partition the subset, Y, into two subsets, W and V, where V = Y - W, and

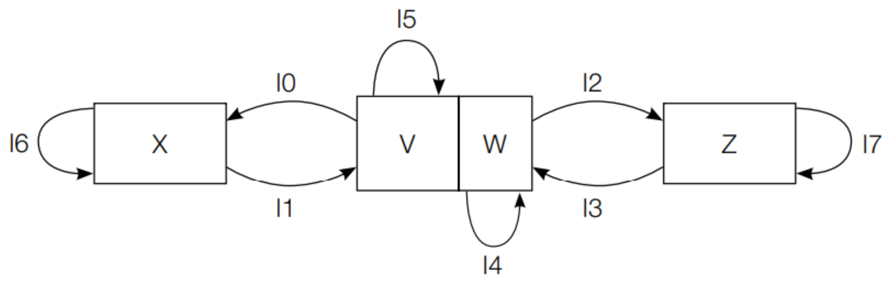
$$W = \{(u, v) \in P | \text{u = 2k, v = 2k + 1 or u = 2k + 1, v = 2k}\}$$

The sets X, Y, W, V, and Z are called primary sets. Note that $P = X \cup W \cup V \cup Z$

It will soon become clear why this partitioning is useful for detecting LSB steganography. Let us analyze what happens to a given pixel pair (u, v) under LSB embedding. There are four possibilities:

- Both u and v stay unmodified (modification pattern 00).
- Only u is modified (modification pattern 10).
- Only v is modified (modification pattern 01).
- Both u and v are modified (modification pattern 11).

LSB embedding causes a given pixel pair to change its membership to one of the four primary sets. The arrows pointing from set A to set B in the figure below are labeled by the modification pattern, meaning that a pixel pair originally in A becomes a member of B if modified by the specified pattern during LSB embedding.



For each modification pattern $\pi \in \{00, 10, 01, 11\}$ and any subset $A \subset P$, let p(π, A) be the fraction of pixel pairs in A modified with pattern π. If the message bits are randomly spread over the image, and are thus independent of the image content (nonadaptive embedding), then for each modification pattern $\pi \in \{00, 10, 01, 11\}$ and each primary set $A \subset P, A \in \{X,\ V,\ W,\ Z\}$, we must have:

$$p(\pi, A) \doteq p(\pi, P) \tag{2}$$

If q is the relative message length, then the expected relative number of modified pixels is $q/2$. Using Equation 2, we thus have

$$\pi(00, P) = (1 - \frac{q}{2})^2$$

$$\pi(01, P) = \pi(10, P) = \frac{q}{2}(1 - \frac{q}{2})$$

$$\pi(11, P) = (\frac{q}{2})^2$$

These transition probabilities and the set migration relationship fromv Figure allow us to express the cardinalities of the primary sets after embedding as functions of q and the cardinalities before the embedding. Denoting the primary sets after embedding with a prime, we obtain

$$\begin{aligned} |X^{'}| &= |X|(1 - \frac{q}{2}) + |V|\frac{q}{2}) \\ |V^{'}| &= |V|(1 - \frac{q}{2}) + |X|\frac{q}{2}) \\ |W^{'}| &= |W|(1 - q + \frac{q^2}{2}) + |Z|q(1 - \frac{q}{2})) \end{aligned} \tag{3}$$

Our goal now is to derive an equation for the unknown quantity, q, using only the cardinalities of primed sets, since they can be calculated directly from the stego image. However, to eliminate the unknown cardinalities of the primary sets of the cover image, we need an additional equation. We now note that on average.

$$|X| = |Y|$$

This assumption is true for natural images that typically contain a certain amount of noise, because it is equally likely to have u>v or u<v independent of whether u is even or odd.

The first two equalities in Equation 3 imply that

$$|X^{'}| - |V^{'}| = (|X| - |V|)(1 - q) \tag{4}$$

Thus, since |X| = |Y|, we have |X| = |V| + |W|, and from Equation 4, we have:

$$|X^{'}| - |V^{'}| = |W|(1 - q) \tag{5}$$

We see from Figure that the embedding process does not modify the union $W \cup Z$. Denoting γ = |W| + |Z| = |W'| + |Z'|, and replacing |Z| with γ - |W|, the expression of Equation 3 for |W'| becomes

$$|W^{'}| = |W|(1 - q)^2 + \gamma q(1 - \frac{q}{2}) \tag{6}$$

Eliminating |W| from Equations 5 and 6 leads to

$$|W^{'}| = (|X^{'}| - |V^{'}|)(1 - q) + \gamma q(1 - \frac{q}{2}) \tag{7}$$

Finally, since |X'| + |Y'| + |Z'| = |X'| + |V'| + |W'| + |Z'| = |P| Equation 7 is equivalent to

$$\frac{\gamma}{2}q^2 + (2|X'| - |P|)q + |Y'| - |X'| = 0 \tag{8}$$

All quantities in this equation can be calculated from the stego image (recall that $\gamma = |W'| + |Z'|$). The unknown message length, q, is obtained as the smaller root of this quadratic equation. If the equation has two complex conjugate roots, only their real parts should be taken. Also, if the smaller root is negative, one might output p = 0, as we know that p must be non-negative. Note that when $\gamma = 0$, it implies that |X| = |X'| = |Y| = |Y'| = |P|/2, and the quadratic equation becomes an identity. In this case, we cannot calculate q. However, since γ is the number of pixel pairs that only differ in their LSBs, this will only happen very rarely for natural images.

5 Application and improvement approaches

In this section, we will provide a preview of our software which is publicly available at here , which is used to demonstrate the process of steganography. Steganography has some drawbacks:

- If the warden succesfully analyze our stega work, our messages is not secret anymore.
- Some embedding techniques are publicly known, they can try various target steganalysis.

We know that the warden can get successful steganalysis hence our messages are now not secret anymore. Therefore, we should use some techniques to protect our messages, and encryption is the best choice in this circumstance. Before we embed the messages, we use single or multiple encryption methods, depending on our level of security.

Another approach to protect our stega work, is used new embedding techniques. That means we use some way to embed the bits to random position that only us know how to retrieve them. Most essential key is that never let anyone know your image resources and techniques.

In our project, we use **PNG** image format with **RSA and Base64 Encryption**

5.1 Least Significant Bit encoder for Images

This section is for the encoder application.

Upon launching this application, you would see this interface.

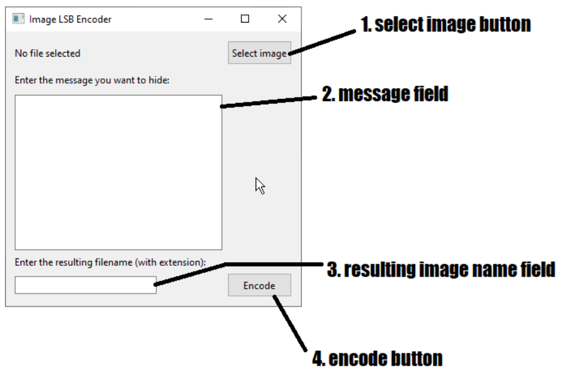


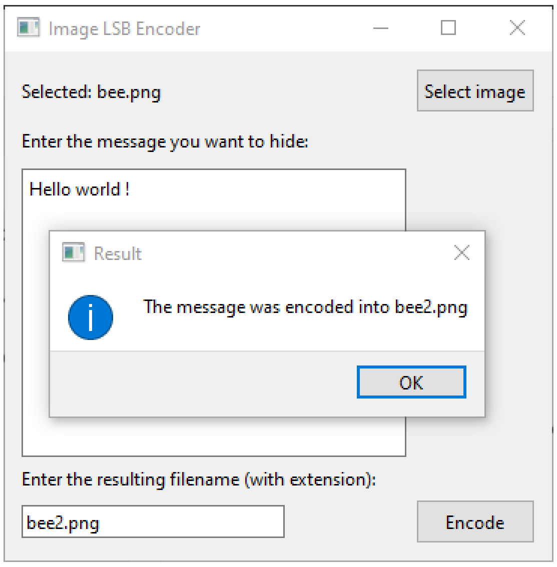
Image LSB Encoder
No file selected
Select image
Enter the message you want to hide:
Enter the resulting filename (with extension):
Encode
1. select image button
2. message field
3. resulting image name field
4. encode button

Figure 1: User interface of the LSB image encoder

And the following steps will let you operate this software:

- Press the [1.] select image button, it should show your operating system file selector, use it and select an image
- Fill in your message in the [2.] message field input box
- Enter what name you want the resulting image to have into the [3.] text field.
- Press the [4.] Encode button

If everything goes right the software should handle everything for you, and the software will display a pop-up message notifying you that the process was a success. The resulting image will be created on the same directory you ran the software at.



Selected: bee.png

Enter the message you want to hide:

Hello world !

The message was encoded into bee2.png

Enter the resulting filename (with extension):

bee2.png

Figure 2: The software after successfully encoding

5.2 Least Significant Bit Decoder for Images

This section will details about the decoder application.
Upon launching this application, you would see this interface.

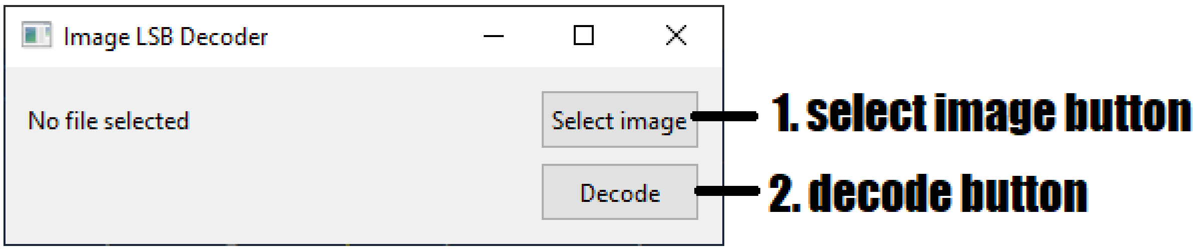
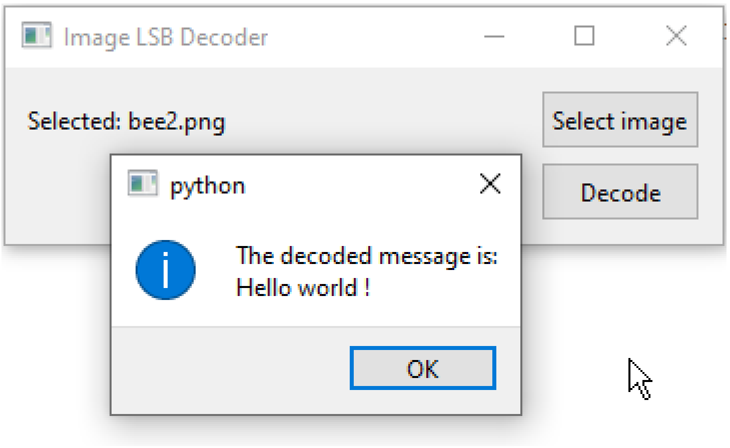


Figure 3: User interface of the LSB image decoder

And the following steps will let you operate this software:

- Press the [1.] select image button, it should show your operating system file selector, use it and select an image
- Press the [2.] Decode button

If a hidden message was discovered, it would be displayed to you via a pop-up window.



The decoded message is:
Hello world !

Figure 4: The software after successfully decoding

5.3 Improvement approaches

5.3.1 Improvement approach: RSA message encryption

LSB steganography in imagery is a fairly popular technique, and because of so, it is quite easily detected, compromising the security of this method.

To improve on that, one solution is to encrypt our message before embedding it into the image. Meaning that who ever that want to see the image needs to have the decryption key.

Here, we have implemented the RSA asymmetrical encryption as the method to hide our message before embedding it into the image. This is the software interface:

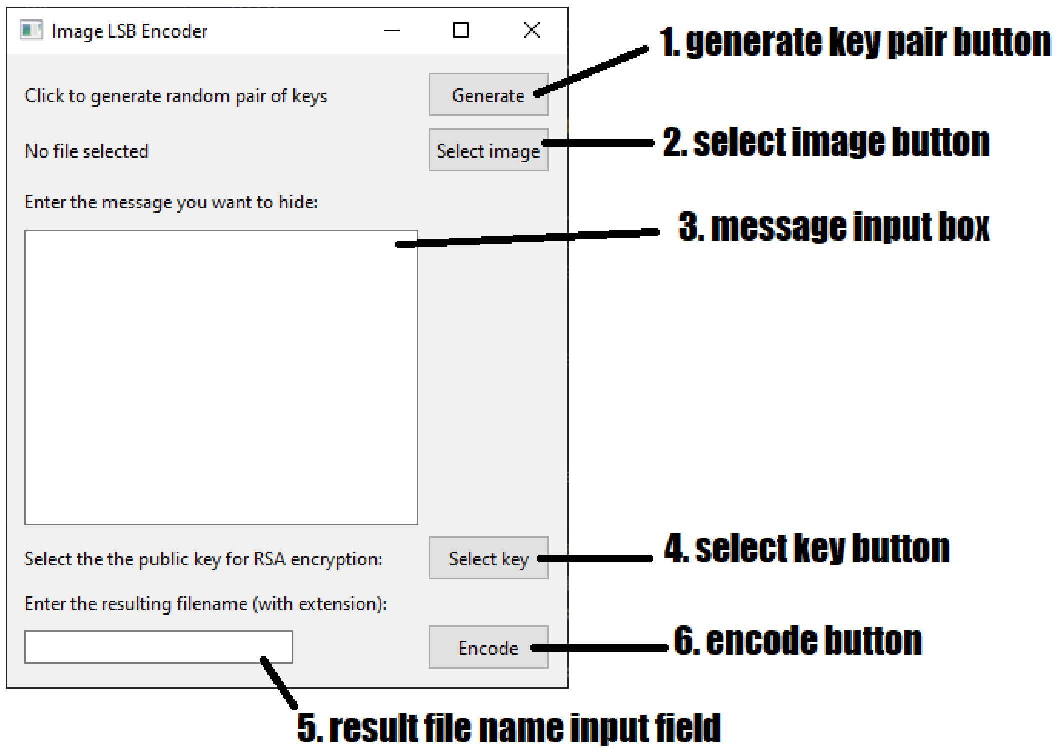


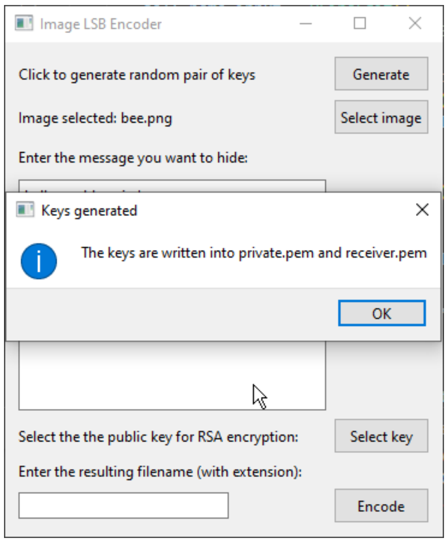
Image LSB Encoder
Click to generate random pair of keys
Generate
No file selected
Select image
Enter the message you want to hide:
Select the the public key for RSA encryption:
Select key
Enter the resulting filename (with extension):
Encode
1. generate key pair button
2. select image button
3. message input box
4. select key button
6. encode button
5. result file name input field

Figure 5: User interface of the LSB image decoder with RSA encryption extension

Here is how to use this software

- If you need a key pair, press the [1.] Generate key button, the resulting key will be stored in your folder
- Press the [2.] select image button, it should show your operating system file selector, use it and select an image
- Fill in your message in the [3.] message field input box
- Select the public key with the [4.] button, if you use our key generation function, the default key file will be receiver.pem
- Enter what name you want the resulting image to have into the [5.] text field.
- Press the [6.] Encode button

The steps to operate this software is a bit more complicated. And when the process is success, you will see a windows prompt notifying that it was a success.



Click to generate random pair of keys

Image selected: bee.png

Enter the message you want to hide:

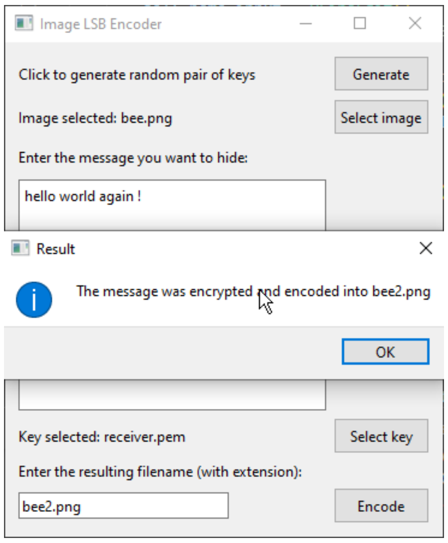
Keys generated

The keys are written into private.pem and receiver.pem

Select the the public key for RSA encryption:

Enter the resulting filename (with extension):

Figure 6: After successful key generation



Click to generate random pair of keys

Image selected: bee.png

Enter the message you want to hide:

hello world again !

Result

The message was encrypted and encoded into bee2.png

Key selected: receiver.pem

Enter the resulting filename (with extension):

bee2.png

Figure 7: After a successful LSB process with RSA encryption

And working as a pair with this RSA encryption module, we have also implemented a software which extract the LSB message from images, then decrypt it with the RSA algorithm using the same key system as the above program.
The user interface of this RSA decryption software is

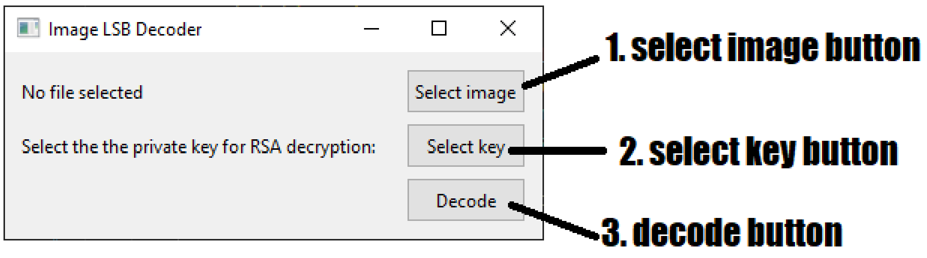
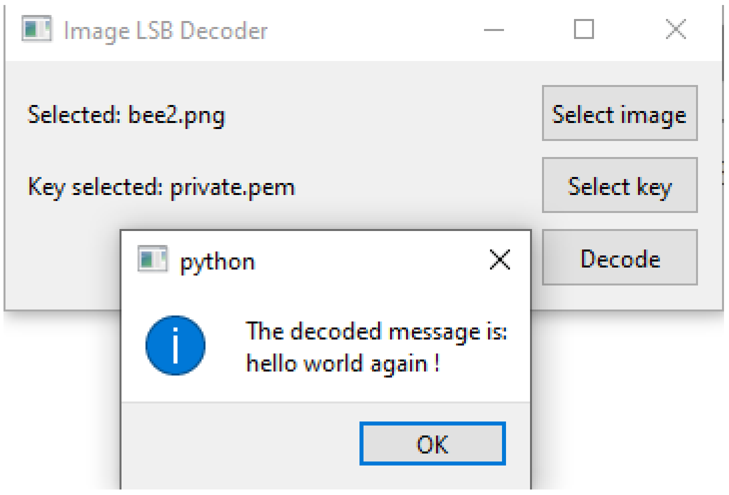


Figure 8: LSB decoder with RSA decryption integrated

Likewise, the following steps are used to operate the software

- Select your desired image with the [1.] Select image button
- Select the private key of the RSA encryption process with the [2.] Select key button
- Press the [3.] Decode button to start the progress

The resulting message will be shown to you in a pop-up window



Selected: bee2.png

Key selected: private.pem

The decoded message is:
hello world again !

Figure 9: A successful decryption

5.3.2 Simple scan and Steganalysis extension

As the result of our discovery after researching about steganalysis (section 4.), we could use those knowledge to analyse a picture, a targeted steganalysis approach to discovering whether an image has or doesn't has some hidden message embedded inside.

To do so, we use our simple scan program

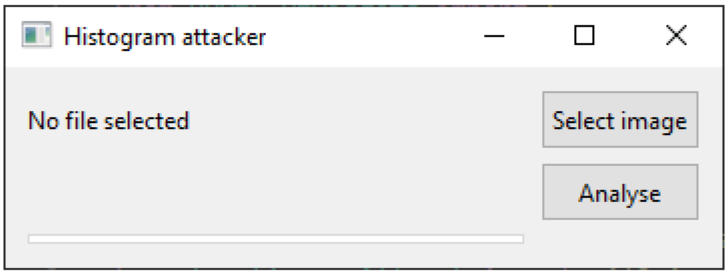


Figure 10: The user interface of our steganalysis scanner

With the program, you need only to select an image, then click analyse. Which will show the resulting image after the simple scan steganalysis algorithm.

Here I will show a small comparison between a normal image being scanned, versus an image with LSB message embedded inside being scanned.

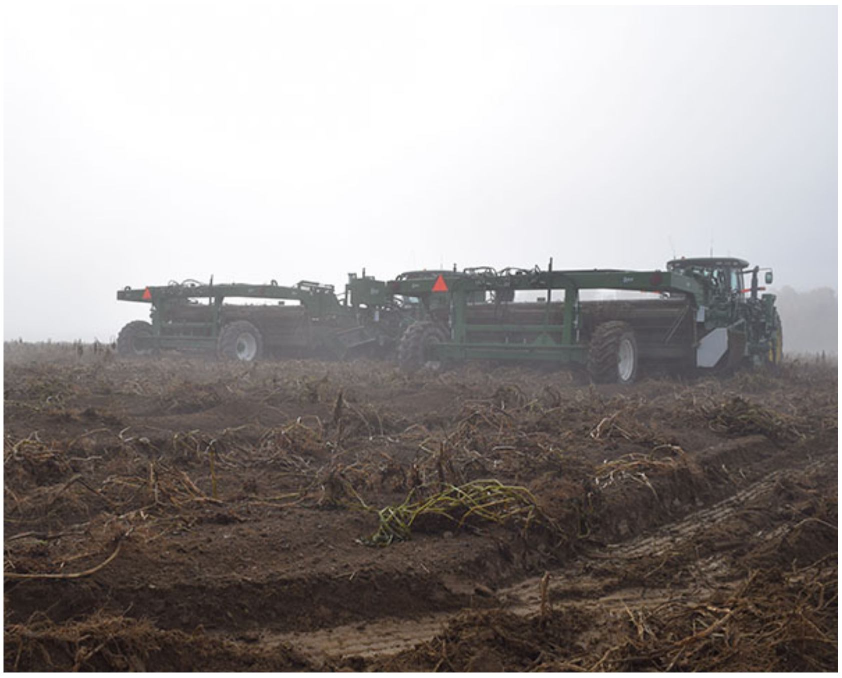


Figure 11: The image before being scanned

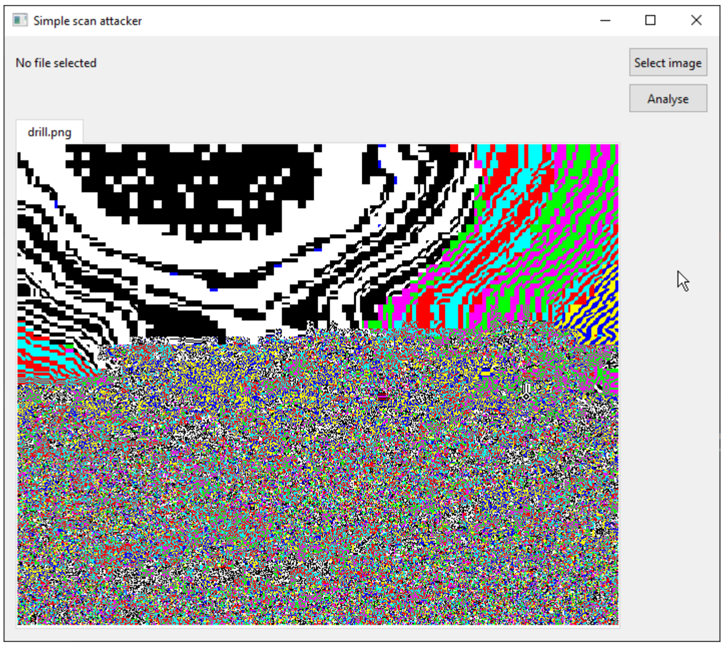


Figure 12: The normal image used to test the scanner software

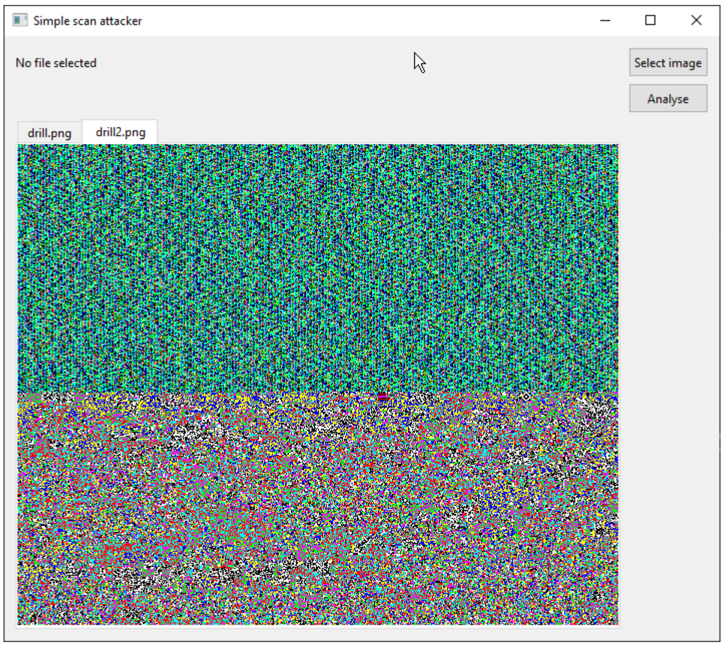


Figure 13: The image with LSB message embeded inside, being scanned

As you can see from figure 13, there is a clear indication that there is some modification to the image.

References

[1] Christian Cachin. An Information-Theoretic Model for Steganography. 2004.

[2] Ingemar J. Cox, Matthew L. Miller, Jeffrey A. Bloom, Jessica Fridrich, and Ton Kalker. *Digital Watermarking and Steganography*. The Morgan Kaufmann Series in Multimedia Information and Systems, The Morgan Kaufmann Series in Computer Security. Morgan Kaufmann, 2008.

[3] Masoud Nosrati, Ronak Karimi, Mehdi Hariri. An introduction to steganography methods. 2011.