Exploring Steganagraphy: From LSB to ZKP

Wu Wenhao
School of Computer Science and Technology
Huazhong University of Science and Technology
u202112001@hust.edu.cn

Wang Yibo
College of Computer science
Sichuan University
Chengdu, Sichuan 610207
wangyb0520@gmail.com

Fan Sirui
Faculty of Electronic and Information Engineering
Xi'an Jiaotong University
1714957525@stu.xjtu.edu.cn

Hu Zhuoqi
School of Software Engineering
Beijing University of Posts and Telecommunications
qzhycloud@outlook.com

Pei Haocheng
School of Cyber Science and Engineering
Sichuan University
2020141530105@stu.scu.edu.cn

ABSTRACT

The widespread adoption of images as a medium for information dissemination has given rise to the pertinent issue of image copyright. In response to this evolving scenario, image steganography has emerged as a viable and scholarly approach to address these concerns. LSB is a commonly used steganography algorithm. However, due to its simple steganography strategy, its robustness and security are insufficient. This paper studies the performance of LSB and improves its robustness against noise, cropping, resizing, rotating and grayscale. Experiments showed that the robustness improved greatly. Moreover, we combine cryptography and zero-knowledge proof to improve the security of LSB, which creates a strong steganography system. Our work may play a role in copyright protection, information transmission, etc.

Keywords

Digital Watermark, Robustness, Least Significant Bit, Cryptology, Steganography

1. INTRODUCTION

In the context of the modern internet, the widespread adoption of images as a medium for information dissemination has given rise to the pertinent issue of image copyright. In response to this evolving scenario, image steganography has emerged as a viable and scholarly approach to address these concerns. A steganographic system must provide a method to embed data imperceptibly, allow the data to be readily extracted, promote a high information rate or payload, and incorporate a certain amount of resistance to removal. The objective of digital watermarking is to embed a signature within a digital image to signify origin or ownership for the purpose of copyright protection. Once added, a watermark must be resistant to removal and reliably detected even after typical image transformations such as rotation, translation, cropping, and quantization.[6]

Least Significant Bit (LSB) is a very basic method to embed information in an image, in which least significant bit of the image is replaced with data bit.(See more details in Appendix A) As this method is vulnerable, in this study, we investigated in improving the robustness of LSB against various attacks and tried to improve its security.

2. IMPROVEMENT OF ROBUSTNESS

In the above examination, it can be observed that the LSB algorithm performs poorly in terms of robustness against image manipulation, and in practical applications, image manipulation is a very common attack behavior. Building upon this foundation, we propose improvements to the LSB algorithm to enhance its robustness against image manipulation.

2.1 Noise

Noise is always presented in digital images during image acquisition, coding, transmission, and processing steps. It happens all the time, whether intentionally or unintentionally.[3] There are all kinds of image noise, like Gaussian noise, shot noise and salt-and pepper noise. In our study, noise is added by inverting each bit of each channel of each pixel in an RGB image in a random manner with certain probability. We try to applied techniques to improve the robustness of LSB against this kind of noise attack. Note that everything can be presented in binary format, so we can encode everything into the cover image theoretically. On behalf the brevity of explanation, we apply the techniques to text.

In our study, two methods were combined to improve the robustness against noise: error correcting code and repetition with atatistic analysis.

Error Correcting Code(ECC). Noise causes the data we finally retrieved differs from the original data we encoded. However, Shannon showed that noise need not cause any

degradation in reliability.[8] Error correcting code helps to improve the robustness of the LSB. In our practice, after transforming the text to binary string of length L, we apply Reed Solomon ECC[10] encoder to get a longer binary string and record its length N. This binary string is what we actually write in the cover picture. So we have the following equation:

$$N = S + L \tag{1}$$

, where S is the binary length of all the ECC symbols of Reed Solomon ECC. We can repair $\frac{S}{2}$ errors and S erasures. In our study, S is designated to be 10. When we try to retrieve the original text, we just put the binary string extracted from the stegno image into the ECC decoder and then transform the string to the target format.

Repetition and statistic analysis. Let's say we just write the message only once into the cover image. If more than $\frac{S}{2}$ bits was inverted by the random noise, even with ECC, we would not be able to recover the original information wholly. In fact, this fails to make good use of the whole picture to store information. So, we repeat the same string over and over again, until the image is filled. Since ECC was applied, so the encoded string must be considered as a block, that is, we must record the length of the string for extraction. After extracting the very long binary string written in the image, then we separate it to blocks with length of N. Then we apply the ECC decoder to decode it and record the frequency of each block. Finally, we assume the original string to be the one with highest frequency.

Finally, we analyzed the effect of the proposed methods through experiment.

Evaluation And Results. As for original LSB without ECC and repetition, a single change of a bit can cause the damage. For LSB with ECC only, assume the length of the original binary message to be L, the length of ECC symbols to be S and the noise amount to be p, then we have the probability P_r to retrieve the original message:

$$P_r = \sum_{k=0}^{\frac{2}{S}} C_{(L+S)}^{k} p^k (1-p)^{L+S-k} \tag{2}$$

As for LSB with repetition only and LSB with both repetition and ECC, the algorithm(See Appendix B) is applied to evaluate their robustness against noise.

For each scenario, we varied the length of the message and tried to test the largest noise amount that can be withstood, i.e., can retrieve the original message. Finally, we got the relationship between the message length and the maximum noise that can bear.

Given a message with its length ranging from 8 to 29, we tested the upper bound of the noise amount. As we can see in fig **??**, being equipped with ECC, an improvement of 2%-4% was achieved. With string length ranging from 8 to 29, the upper bound of the noise it can bear ranges from 7% to 18%, this can hardly be achieved by the original LSB or LSB with ECC only. The robustness against this kind of noise was improved significantly.

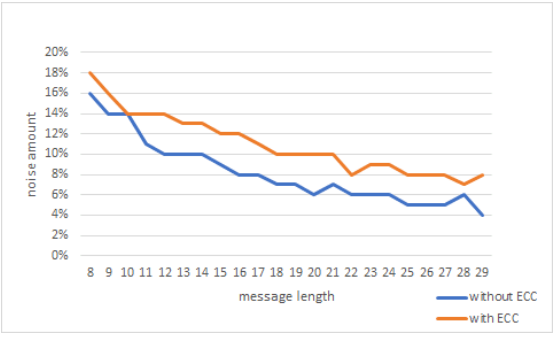

Figure 1: Conparation between repetitive LSB with/without ECC.Given a certain meaasge length, test the noise amount it can bear, the higer the better.

2.2 Cropping

In order to mitigate the potential loss of digital watermark caused by cropping operations, we have devised an algorithm aimed at countering cropping attacks. The crux of this algorithm revolves around leveraging the LSB (Least Significant Bit) technique to iteratively embed the hidden message across the entire image canvas. As a result, even if the image undergoes cropping, as long as a subset of the hidden messages is preserved, they can be extracted, effectively achieving the objective of thwarting cropping attacks.

In essence, each information subset comprises three integral components: the header and footer, the hidden message per se, and the length of the hidden message. Analogous to the concept of packets in network communication, the header and footer serve as markers to delineate the commencement and termination of each information subset during the decoding process. The length component plays the role of a checksum, providing a means to verify the veracity of the extracted hidden message. During the encoding phase, the intended hidden message is amalgamated with the header, footer, and length to form an information subset, which is recurrently inscribed into the entirety of the image utilizing the LSB technique. Careful selection of the header and footer is essential to ensure they do not conflict with the encoding information of the hidden message. For instance, if the hidden message is represented using ASCII codes, the header and footer can be judiciously designated as 0x00 and 0x11, respectively, as these values are devoid of corresponding characters in the ASCII code set. Subsequently, during the decoding process, the Least Significant Bit of the image is extracted, and the header and footer are utilized to demarcate the boundaries of each information subset. Furthermore, the length information is employed to scrutinize the accuracy of the extracted information. In more intricate scenarios, additional sophisticated validation methodologies can be incorporated.

This enhanced LSB algorithm has led to a significant advancement in robustness against cropping attacks. Empirical evidence has validated that even when the image is subjected to a reduction in size by 2%, the hidden information

header	length	hidden message	footer

Figure 2: The format of the information subset

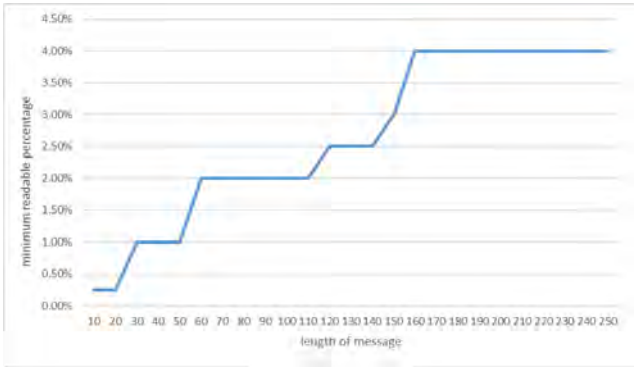

Figure 3: The length of the hidden message affects the minimum legibility percentage.

can be effectively retrieved.

By cropping the image to a certain original size, hidden messages can still be extracted. We refer to this percentage as the readable percentage, while the minimum readable percentage signifies the point where the hidden message cannot be extracted when the image is cropped to a size smaller than this percentage. During the experiment, we observed that the length of the hidden message has an impact on the minimum readable percentage.

By cropping the image to a certain original size, hidden messages can still be extracted. We refer to this percentage as the readable percentage, while the minimum readable percentage signifies the point where the hidden message cannot be extracted when the image is cropped to a size smaller than this percentage. During the experiment, we observed that the length of the hidden message has an impact on the minimum readable percentage.

An experiment was conducted to investigate this phenomenon, and the results are shown in fig 5. From the graph, it can be observed that there is a positive correlation between the length of the hidden message and the minimum readable percentage. This indicates that the longer the hidden message, the weaker its resistance to cropping. However, the experimental results also reveal that even when the length of the hidden message reaches 250 characters (equivalent to 2000 bits), the minimum readable percentage remains below 5%. This suggests that despite the growth in the length of the hidden message, the algorithm's robustness remains strong.

We also combined our improvement of robustness against noise and cropping. In this implementation the length of binary string are not needed, since the recognition of the binary string was realized by recognizing the header and footer. However if we still keep using repetition and statistic to help with decoding, it would take a long time. The code is contained in the appendices.

2.3 Resizing & Rotating

In this part, LSB doesn't seem to work very well. This is mainly due to the following reasons:

1) Changes in Pixel Values: When we scale or rotate an image, the original pixel values change. This is because these operations often involve interpolation or resampling processes, which generate new pixel values to fit the new size or direction of the image. These new pixel values may not retain the value of the original pixel's least significant bit, resulting in the loss of the hidden information.

2) Changes in Image Structure: Scaling or rotating an image changes the arrangement of its pixels. This structural change disrupts the organization of the hidden information in LSB steganography. Even in some situations where the values of the least significant bits can be retained, the hidden information cannot be correctly interpreted due to the changed pixel arrangement.

The Fourier Transform[5] is a mathematical tool that transforms one function of a real variable into another. In the context of an image, it translates the spatial domain representation into the frequency domain. The idea of applying Fourier Transform in steganography, specifically in situations involving scaling and rotation, comes from two primary factors:

1) Robustness against Transformations: Fourier Transform manipulates the frequency representation of an image, which is generally more robust to common image processing operations like scaling and rotation. When information is hidden in the frequency domain, it tends to resist such transformations better than when hidden in the spatial domain (as in the case of LSB steganography).

2) Invisibility of Changes: Changes made in the frequency domain are usually less perceptible in the spatial domain. Thus, hiding information in the frequency domain can potentially be less noticeable to human eyes than making changes directly to the pixels in the spatial domain.

Now let's explain the whole process of how to hide information in images through Fourier transform.

1. First, we perform Fourier transform on the original image to obtain its frequency domain image. Figure 4 shows the spatial and frequency domain plots of this image.

2. We add a readable watermark to the transformed frequency domain image. In this example, we set the watermark text to "Hi, DOTA", and place the watermark in the lower right corner of the frequency domain image. Figure 5 shows the comparison of the frequency domain image of the original image and the watermarked frequency domain image. We can see that there is a "Hi, DOTA" watermark in the lower right corner of the watermarked frequency domain image.

Figure 4: Original image (left) and transformed image (right)

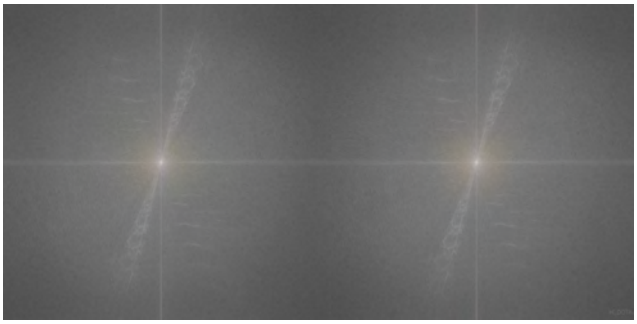

Figure 5: The frequency domain image of the original image (left) and the frequency domain image after watermarking (right)

3. We perform inverse Fourier transform on the frequency domain image with the watermark added and obtain the encrypted image. Figure 6 shows the comparison between the original image and the encrypted image. We can see that since the magnitude spectrum of the original image has been modified by adding the watermark layer, the color of the encrypted image looks darker.

4. Perform Fourier transform on the encrypted image, and we can get the frequency domain image of the encrypted image. In this image, we can see the previously added watermark in the lower right corner, which shows that we have successfully hidden the message "Hi, DOTA" in the encrypted image. Figure 7 shows the previous watermarked frequency domain image and the frequency domain image obtained after performing Fourier transforming on the encrypted image. We can find that the watermark is not as clear as before. This is because some image parameters are lost due to the addition of the watermark layer in the process of inverse Fourier transform and Fourier transform.

Figure 6: Original image (left) and encrypted image (right)

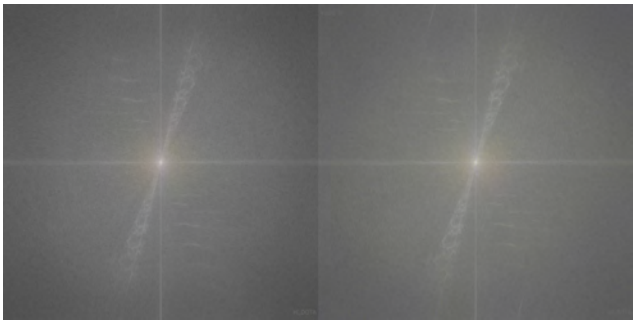

Figure 7: The watermark on the frequency domain of the original image (left) and on the frequency domain of the encrypted image (right)

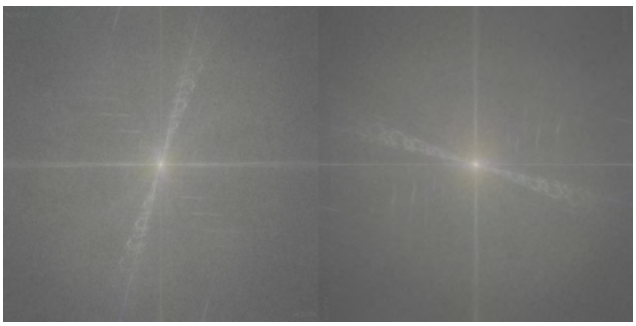

Figure 8: After FFT on the 25%-reduced encrypted image(left) and on the 90-degrees-clockwise-rotated encrypted image(right)

But the good news is that after such processing, no matter whether we resize(not too much) or rotate the encrypted image, we can get the information hidden in it by performing Fourier transform on the encrypted image. Figure 8 shows the result image of Fourier transform when we reduce the encrypted image by 25% and rotate the encrypted image 90 degrees clockwise. We can find that the watermark on the image after the Fourier transform of the encrypted image that has been reduced by 25% is not as clear as the one at the right of Figure 7 . This is because when the encrypted image is resized, we lose some image information, and the lost image information may contain watermark information. However, the clarity of the watermark after the rotation process is the same as the original watermark. This is because the information contained in the image as a whole has little change during the rotation process. We just rotated the image 90 degrees clockwise, so the frequency domain image just also rotated 90 degrees clockwise, and we can see the same watermark "Hi, DOTA" in the lower left corner.

However, the feasibility of using the Fourier Transform in steganography comes with its own challenges:

1) Complexity: The process of converting an image to the

Figure 9: 0.5 partial grayscale (left) and complete grayscale (right)

frequency domain, embedding the information, and then converting it back to the spatial domain is more complex compared to simple LSB steganography.

2) Noise Susceptibility: While the frequency domain representation is more robust against operations like resizing and rotating, it might be more susceptible to noise and compression, which can disrupt the hidden information.

3) Amount of Data: Embedding information in the frequency domain might limit the amount of data that can be hidden, compared to hiding it directly in the pixel values in the spatial domain.

In conclusion, while the Fourier Transform offers a promising avenue for steganography that can resist image resizing and rotating, it also brings its own set of challenges that need to be addressed.

2.4 Grayscale

In sRGB (standard RGB color space) , the grayscale of a original image is such a image that preserve the same perceptual luminance and RGB combination of the original one. Partial grayscale are some of the linear combination of the original image and its grayscale, which preserve the perceptual luminance as well, and are between the original image and grayscale. The image (9) above show examples of grayscale and partial grayscale.

Specifically, we assume that all images are made up of RGB pixels which take an integer value in $[0, 255]$ for every channel. For a pixel, its linear luminance is defined as: $Y = 0.2126R + 0.7152G + 0.0722B$. [1] Holding Y unchanged, make $R = G = B$, we get the grayscale pixel of the original pixel. Obviously, at this time, $R = G = B = Y$, where Y is at $[0, 255]$ (so a grayscale pixel only needs one channel to store). Let $R' = kY + (1-k)R$, $G' = kY + (1-k)G$, $B' = kY + (1-k)B$, and then the $R'G'B'$ is the k grayscale pixel of the original pixel, where k is in $[0, 1]$.

Now, the steganography strategy is divided into two parts.

[1]In fact, the definition of RGB's coefficients is varied, and we only consider this specific linear combination here, which is called Rec.709 standard[1]. But the algorithm we are about to propose will be easily extended to other linear combinations.

Basically, when $k \neq 1$, the information can be restored through a "inverse function" of the grayscale transformation relatively easily. The derivation of all the mathematical formulas are given in the appendix C.

Now, the hardest part is the case of $k = 1$, that is, a complete grayscale image. First of all, we can't restore it because it compresses the three-dimensional information into one dimension in a very rough way, and each channel can only contribute a little influence.

Therefore, we can only retain some information by modifying the original image to try to influence the final grayscale image in a predictable way somehow.

First of all, we have to prove that it is feasible from the perspective of information theory. The question now is, is there a RGB combination that cannot be distinguished after graying, no matter what modification is made to the last bit of the three channels?

If the answer is yes, then our goal of steganography can not be achieved, because once these RGB combinations, which serve as examples for the formula, appear, their performance in the grayscale image is locked and cannot embedd any message to it.

Luckily, the answer is no. We can programe to check every possible combination and see whether there's a modification that influence the grayscale result, and it turned out to be correct. The whole mathematical derivation is also in appendix C.

Therefore, we can manipulate all the pixels as needed so that each pixel can display the message we want in the last bit after it is grayed.

3. COMBINING WITH *CRYPTOGRAPHY*

The above improvements are based on the premise of considering only image manipulation. Taking this into account, we further focus on enhancing the overall security and credibility of the encryption system to withstand decryption and tampering attempts against the digital signature, thereby achieving true security of digital signatures.

3.1 Digital signature system

If we directly embed steganographic information into the carrier without any processing, attackers can eavesdrop on the transmission and potentially retrieve the hidden information by reasonably inferring the steganographic technique used by the sender. This compromises the invisibility of the hidden information, allowing the attacker to gain unauthorized access. Additionally, the attacker could employ relevant techniques to remove our digital watermark and replace it with their own information, falsely claiming that the information was published by them, leaving us powerless against such impersonation.

To address these issues, we develop a two-layer encryption method. This method involves applying different encryption techniques to the data to be hidden, providing dual protection. Firstly, we use AES encryption to ensure the confidentiality of the information. Simultaneously, we apply

digital signatures on top of AES encryption to authenticate the sender's "signature" on the information.

Our encryption scheme consists of two layers: the AES encryption layer and the digital signature layer. The entire encryption process is illustrated in the diagram below, and this part describes each module separately.

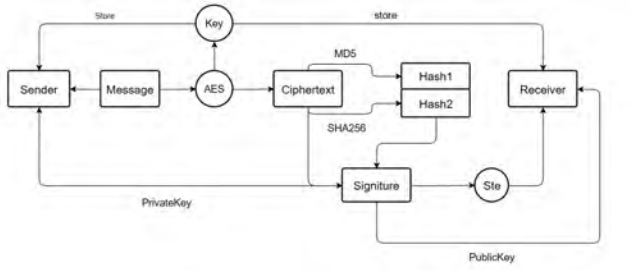

Figure 10: Stucture

In AES layer, we employ the AES-128-CBC encryption method. Both parties will first agree on a 16-character key, which will be kept secure by the information sender and receiver. When the receiver extracts the ciphertext from the image carrier, they can use the key to decrypt the ciphertext and retrieve the original message. It's worth noting that AES is currently considered a relatively secure algorithm with no efficient decryption methods other than brute force (which is infeasible). Hence, our encrypted information is relatively secure.

This implementation employs the Python Crypto.Cipher library to achieve the desired functionality. The program utilizes the AES algorithm in CBC mode to encrypt the plaintext message securely. The sender can readily provide the plaintext message for encryption, alongside the preestablished key and IV, resulting in the generation of ciphertext ready for secure transmission. Similarly, the receiver can decrypt the received ciphertext back to its original plaintext form using the identical key and IV for seamless decryption.

As for digital signature, we start with risk. Potential Risk: While the basic system employs hashing primarily for verification, the most significant risk lies in collision vulnerability, where different input data might generate identical hash values. Attackers could forge files with identical hash values to achieve the same verification effect.

Countermeasure: To mitigate this risk, we employ double hashing by applying both MD5 and SHA256 hash functions to the information, combining the resulting hash values and encrypting them with the sender's private key to form the digital signature. This combined approach significantly reduces the likelihood of collisions, further enhancing the authority of the digital signature.

This module was implemented using the 'cryptography' library to achieve its intended functionality. Upon initialization, the program automatically generates a key pair to facilitate subsequent signature operations. The chosen public exponent of 65537 and key size of 2048 were employed during key pair generation, as they are widely recognized as secure values.

When the sender inputs a message for processing, the program utilizes two distinct hash functions to create digital signatures, which are subsequently returned to the user. The receiver can then employ the message, the sender's public key, and the two signatures for the purpose of verification.

In the implemented solution, the sender can conveniently input the message, AES key, and AES initialization vector (IV). Subsequently, the system generates the ciphertext using AES-128 in CBC mode, alongside a keypair comprising a private and public key. Furthermore, the system generates two versions of a signature for the sender to conceal within a carrier medium. This approach ensures the confidentiality of the message through AES encryption and enables data integrity and authenticity through the use of digital signatures. The receiver is required to have access to the following elements for verification and decryption purposes: the ciphertext, two versions of the signature, and the public key.

Our approach offers a robust solution to protect steganography-based information by combining the strengths of cryptography and steganography. However, we acknowledge that certain aspects, such as comprehensive key management, authorized digital signature key pairs, and secure network transmission, require further research and consideration.

3.2 Zero-knowledge-proof system

Here's another method called zero-knowledge-proof to prove that you are the owner of the image.

Zero-knowledge-proof is a method by which one party (the prover) can prove to another party (the verifier) that a given statement is true, while avoiding conveying to the verifier any information beyond the mere fact of the statement's truth.[11]

We are not going to cover this area in detail, but will use it as a tool to prove the ownership of the image. In particular, we are going to use graph isomorphism as the proof basement for the time being.

Given two graphs $G_1(V_1, E_1), G_2(V_2, E_2)$, if there is a bijection m from G_1 to G_2 such that

$$\forall x, y \in V_1,\ (x, y) \in E_1 \leftrightarrow (m(x), m(y)) \in E_2, \qquad (3)$$

and then G_1 and G_2 are said to be isomorphic, written as $(G_1, G_2) \in GI$, as shown in the figure (11).

So far, no polynomial time algorithm for the graph isomorphism problem has been found yet. It is a special case of the hidden subgroup problem just like factoring and discrete logarithm, and is not proved to be NP-hard yet.

Let's assume that one day you get involved in a lawsuit over the ownership of a image. You, the owner of image, are called P. The judge is called Q.

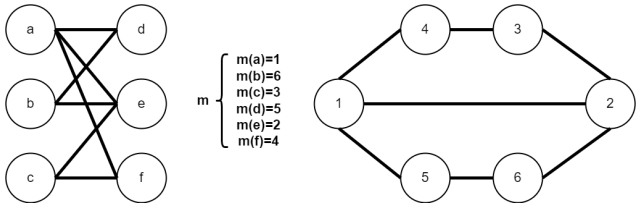

Figure 11: 0.5 partial grayscale (left) and complete grayscale (right)

P embedded two huge isomorphic graphs (G_0, G_1) in advance in the image. P knew exactly about the isomorphic bijection m from G_0 to G_1. P revealed the graphs in the court, and now needs to prove his ownership of the image in front of Q.

Here's the mechanism:

1. P: create a new graph, which is isomorphic to G_1, called H, and show H to Q. P knows the isomorphic bijection n from G_1 to H.
2. Q: pick a random $\alpha \in \{0, 1\}$ and show α to P.
3. P: show
$$\beta = \begin{cases} n & \text{if } \alpha = 0 \\ nm & \text{if } \alpha = 1 \end{cases}$$
to Q.
4. Q: reject if $H \neq \beta G_\alpha$.
5. Repeat 1∼4 until Q trust P. (count as a new round)

In this mechanism, the probability that P really knows m is $1-(\frac{1}{2})^n$ after n rounds of interaction from the perspective of Q. Therefore, Q can trust that P is the owner of the image.

Firstly, this zero-knowledge proof system does not need to publish any public keys like a digital signature system.

Secondly, the same pair of graphs can be reused for several times and it is pretty easy to create more huge isomorphic graphs if you want. Unlike limited primes used as exclusive public key, isomorphic graphs offer much more key resource.

Last but not least, problems like isomorphic graphs has the potential of good robustness. The isomorphic graphs can be directly drawn in the media (e.g., the fourier transform image of every frame of a movie). After malicious transformation, the graphs might lose some edges or vertexs. However, they're still the approximation of the original graphs and hold an approximate isomorphic relation. The image owner can still do the proof within some approximation rate, which is also convincible from the perspective of computational complexity[2].

4. CONCLUSIONS

In this article, we started with the basic LSB steganography technique and explored its performance. We conducted research and improved its robustness in various aspects, including cropping, noise,resize, and grayscale. Meanwhile, we investigated the potential of combining steganography with cryptography, presenting two effective integration schemes to enhance the security of steganography.

Although the LSB algorithm may be considered somewhat outdated, currently, the frontier research direction in digital watermarking revolves around frequency domain-based adaptive watermark embedding. However, we believe that our research can provide insights and references for new steganography algorithms. Furthermore, there is still room for improvement in our work. We envision the potential for combining robustness strategies against various attacks and comprehensively enhancing the algorithm's performance.

5. REFERENCES

[1] Bt.709 : Parameter values for the hdtv standards for production and international programme exchange. 2015. Retrieved July 22, 2023.

[2] V. Arvind, J. Köbler, S. Kuhnert, and Y. Vasudev. Approximate graph isomorphism. pages 100–111, 2012.

[3] A. K. Boyat and B. K. Joshi. A review paper: noise models in digital image processing. *arXiv preprint arXiv:1505.03489*, 2015.

[4] H. S. Fairman, M. H. Brill, and H. Hemmendinger. How the cie 1931 color-matching functions were derived from wright-guild data. *Color Research & Application*, 22(1):11–23, 1997.

[5] J. S. Lim. Two-dimensional signal and image processing. *Englewood Cliffs*, 1990.

[6] L. M. Marvel, C. G. Boncelet, and C. T. Retter. Spread spectrum image steganography. *IEEE Transactions on image processing*, 8(8):1075–1083, 1999.

[7] N. Provos and P. Honeyman. Hide and seek: An introduction to steganography. *IEEE security & privacy*, 1(3):32–44, 2003.

[8] C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[9] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[10] S. B. Wicker and V. K. Bhargava. *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

[11] Wikipedia contributors. Zero-knowledge proof — Wikipedia, the free encyclopedia. 2023. [Online; accessed 22-July-2023].

APPENDIX

A. LEAST SIGNIFICANT BIT

Least Significant Bit (LSB) is an algorithm used to embed information within an image. Its algorithmic concept is straightforward and widely adopted. In modern images, each bit comprises three channels representing the colors red, green, and blue, with each channel allocated one byte. [4]The fundamental principle of the LSB algorithm involves modifying the least significant bit of each byte to facilitate the insertion of a hidden message.[7] This approach renders

Figure 12: Images before and after embedding using the LSB algorithm.

the information imperceptible to the human eye and indistinguishable through visual inspection before and after applying steganography. It exhibits a high degree of invisibility and ensures a significant likeness between the images before and after the information is embedded using this algorithm.

A good steganography technique requires achieving invisibility, ensuring minimal differences between the images before and after the embedding process, and exhibiting robustness against image manipulation. To assess the consistency between the images before and after Steganography, Peak Signal to Noise Ratio (PSNR) and structural similarity (SSIM)[9] are introduced as evaluation metrics. To test the robustness against image manipulation, the images are subjected to operations such as blurring, rotation, resizing, cropping, and the addition of noise.

Based on the test results mentioned above, it can be observed that the LSB algorithm demonstrates excellent invisibility, as indicated by its outstanding performance in both PSNR and SSIM metrics. This signifies that the differences between the images before and after information embedding are minimal, showcasing the algorithm's remarkable capability in concealing information. However, there is still room for improvement concerning the Robustness against image manipulation. In the subsequent phase, we will propose corresponding enhancement algorithms tailored to address image manipulation challenges.

B. ALGORITHM OF LSB FOR EVALUATION OF ROBUSTNESS AGAINST NOISE

Algorithm 1: Evaluation algorithm

Input: cover image: *img*; message to write: *msg*
Output: stegno image: *s_img*; , maximum noise can withstand: *max_noise*

```
encoded_msg := encode(msg);
for len(b_msg)<len(img) do
    b_msg:=b_msg+ECC_msg
end
s_img:=Write b_msg in img;
while noise_amount:=0 to 100 do
    noised_img:=add_noise(img,noise_amount);
    decoded_msg:=decode(noised_img);
    if decoded_msg!=msg then
        max_noise:=noise_amount;
        break;
    end
end
return s_img, max_noise
```

C. MATHEMATICAL DERIVATION ABOUT GRAYSCALE

C.1 Preparation

Formally,

$$Y(R,G,B) = 0.2126R{+}0.7152G{+}0.0722B, \ abbreviated\ as\ Y, \tag{4}$$

$$gray(k,R,G,B) = (kY{+}(1{-}k)R, kY{+}(1{-}k)G, kY{+}(1{-}k)B), \tag{5}$$

therefore,

$$(R',G',B') = gray(k,R,G,B). \tag{6}$$

Especially,

$$gray(1,R,G,B) = (Y,Y,Y). \tag{7}$$

For an original image, convert all the pixels into their grayscale pixels so that we get a grayscale image, and convert all pixels into their k grayscale pixels so that we get a k grayscale image (a partial grayscale image).

It can be noted that for any pixel, it has the invariance of Y; for all pixels, it has the invariance of k. This inspires us to use this property to store information.

C.2 Derivation

When $k \neq 1$,

the k grayscale image can actually be reversed through a sort of inverse function of $gray()$. (Loosely, we still call it $gray^{-1}()$).

That is to say,

$$\begin{aligned}(R,G,B) &= gray^{-1}(k,R',G',B') \\ &= ((R'-kY)/(1-k), \\ &\quad (G'-kY)/(1-k), (B'-kY)/(1-k))\end{aligned} \tag{8}$$

Obviously we know R'G'B'. By formula (4), we can calculate Y easily. For k, we need a little trick. We introduce an "information pixel", which can be chosen from the first pixel or any other pixels that can be easily detected both before and after grayscale. The RGB of the information pixel in the original image is defined to be $(0, 255, 0)$. Therefore, after the grayscale transformation, the information will turn into $(182k, 255 - 73k, 182k)$. So we check this pixel in the grayscale, and we can calculate the k immediately. Then, we can restore the message embedded in the original image for we have already known all the values we need to calculate the formula (8).

When $k = 1$,

Table 1: LSB performance evaluation

Invisibility	Consistency		Robustness against image manipulation				
	PSNR	SSIM	blurring	rotation	resizing	cropping	noise
low	51.14dB	0.9986	low	low	low	low	low

if $\oplus$ stands for bitwise exclusive OR (which is consistent with the situation in programming languages such as C), then the question is whether

$$\begin{aligned} &\exists\ (R,G,B)\ \forall\ a,b,c \in \{0,1\}, \\ &gray(R,G,B) = gray(R \oplus a, G \oplus b, B \oplus c). \end{aligned} \tag{9}$$

And the answer is no because of a program verification for all all possible (R,G,B,a,b,c) combinations, which implies:

let $low(x) = x\&1$ (which is LSB),

$$\begin{aligned} &\forall\ (R,G,B)\ \exists\ a,b,c \in \{0,1\}, \\ &\{low(Y(R,G,B), low(Y(R \oplus a, G \oplus b, B \oplus c))\} = \{0,1\}. \end{aligned} \tag{10}$$

D. EVALUATIONS ON FOURIER TRANSFORM METHOD

D.1 The value of alpha

In this part, We will illustrate the effect of watermark strength on the watermarked frequency domain image and on the encrypted image.

In our method, we use the variable 'alpha' to control the visibility of the watermark. When alpha=0, the watermark is completely invisible; when alpha=1, the watermark will cover the original image. Figure 13-15 shows the watermarked frequency domain image and the encrypted image obtained after the inverse Fourier transform when alpha is equal to 0, 0.5, and 1. We can see that as the alpha increases, the watermark becomes more and more obvious, while the original image becomes less and less obvious. Therefore, a key point is that we need to find an alpha value that can not only generate a relatively visible watermark, but also ensure that the original image is not disrupted to the greatest extent.

In the case in our article, we use an alpha value of 0.1, a value that produces a visible watermark without disrupting the original image too much.

D.2 Evaluation of the resizing and rotating

We have mentioned in the article that when an image containing a watermark is rotated, the watermark will not change significantly, because the rotation preserves each pixel of the image intact, and only changes their position. However, when we resize an image, the pixels of the image usually increase or decrease, which may destroy the watermark to a certain extent. So we need to evaluate the maximum adjustment range that the watermark added by the Fourier transform method can bear.Take the alpha value of 0.1 in our article as an example. In this case, the image containing the watermark can be reduced to at most 50% of the original size and still show a readable watermark after Fourier

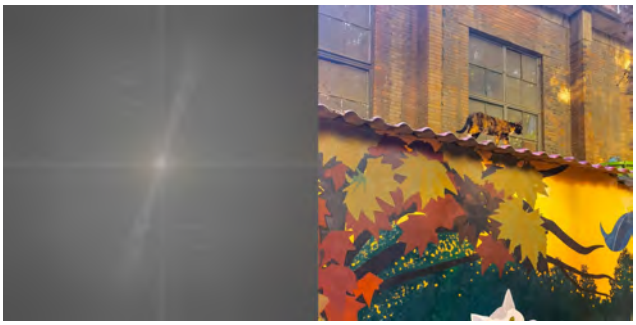

Figure 13: Watermarked frequency domain image(left) and encrypted image(right) when alpha = 0

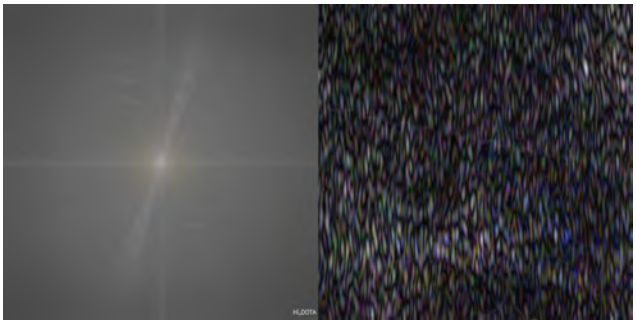

Figure 14: Watermarked frequency domain image(left) and encrypted image(right) when alpha = 0.5

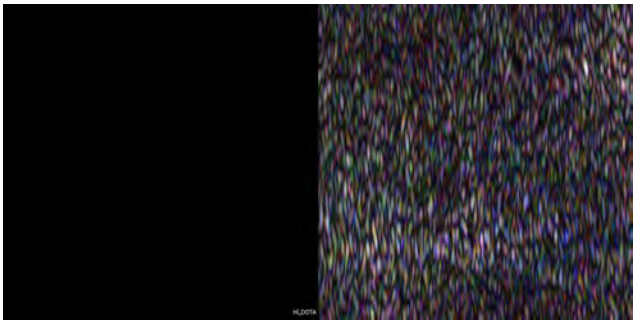

Figure 15: Watermarked frequency domain image(left) and encrypted image(right) when alpha = 1

transform. At the same time, since the pixels are increased when the image is magnified, the watermark will move from the original lower right corner to the center of the frequency domain image after the magnification, and this movement is not linear. At the same time, since the center of the frequency domain image represents low frequency, it is usually very bright, so when the watermark is moved to be covered by the brightness of the center, the watermark will not be visible any more, and the moving distance corresponding to this magnified level is different for each picture, so it is hard to be measured with a specific value. In our case, even though the image is magnified by 4 times, we can still see the watermark in its frequency domain image.

D.3 The Steps of Digital Signature

1. Signature Generation: Firstly, the ciphertext obtained from the first layer is hashed to create a data digest. Then, this digest is encrypted using a private key to generate the digital signature.

2. Signature Transmission: The ciphertext and the generated digital signature are sent together to the recipient.

3. Signature Verification: The data receiver uses the public key to decrypt the received digital signature, obtaining the digest.

4. Digest Comparison: The data receiver hashes the received original data, generating a new digest.

5. Comparison Result: The recipient compares the decrypted digest with the newly computed digest. To check if data has integrity, authenticity, and trustworthy source.