Article

High-Capacity Imperceptible Data Hiding Using Permutation-Based Embedding Process for IoT Security

Jui-Chuan Liu [1], Ching-Chun Chang [2], Chin-Chen Chang [1,*] and Shuying Xu [1]

[1] Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan; p1200318@o365.fcu.edu.tw (J.-C.L.); p0968875@o365.fcu.edu.tw (S.X.)
[2] Information and Communication Security Research Center, Feng Chia University, Taichung 40724, Taiwan; ccc@fcu.edu.tw
* Correspondence: ccc@o365.fcu.edu.tw

Abstract: The internet of things (IoT) has become a popular technology in communication which utilizes the concept of connecting things together and exchanges information through various networks. Since data can be transferred through a wide range of channels, IoT systems suffer from potential data leakages. One of the common ways to reduce such risks is to engage steganography with secret information during transmission. A novel scheme proposed in this paper exploits simple pixel permutations to embed secret data. Instead of handling pixel blocks, the proposed scheme maneuvers on pixels directly. The proposed scheme simply manipulates the sequenced pixels using two coefficients, a threshold of range for pixel values, and a specified clustering count to fulfill the two major requirements of effective data hiding. The experimental results indicate that the proposed scheme provides a satisfactory embedding capacity and preserves a high level of image visual quality. The overall performance of the proposed scheme demonstrates its high potential in IoT security.

Keywords: data hiding; IoT security; permutation-based embedding; pixel permutation

check for updates

Citation: Liu, J.-C.; Chang, C.-C.; Chang, C.-C.; Xu, S. High-Capacity Imperceptible Data Hiding Using Permutation-Based Embedding Process for IoT Security. *Electronics* **2023**, *12*, 4488. https://doi.org/10.3390/electronics12214488

Academic Editor: Paris Kitsos

Received: 16 September 2023
Revised: 14 October 2023
Accepted: 30 October 2023
Published: 31 October 2023

1. Introduction

Internet of things (IoT) communication systems have evolved speedily to meet the massive needs of many industries, such as automobiles, wearables, and smart cities [1]. The IoT technology can connect these widely distributed wireless devices for modern communications; thus, machine-to-machine (M2M) interaction becomes one of the fundamental requirements for an abundance of applications. With the arrival of the AI evolution era, IoT technology is advancing even more rapidly, with big data running through our daily lives behind the scenes. The revenue of IoT industries is huge and will increase tremendously more in the near future. This is exciting yet scary, because data can be easily hijacked or stolen through unsecured channels which may be designed within the infrastructure of an IoT system [2,3]. The security of information in IoT technologies becomes more and more important than ever [4,5]. Some industries entangle personal privacy data, such as licensing, medical, or finance data, in their data transfers. Other industries can involve trading secrets, such as integrated circuits and software designs. Data hiding (DH) techniques, which hide secrets in cover media, were studied as one of the solutions to resolve privacy issues during data transmission [6,7].

DH is the art of hiding secrets [8–10]. Secrets are exchanged through convert channels into digital cover media. Digital images are popular media and many data hiding studies focus on them. The fundamental principle of DH for digital images is to conceal secret data into a cover image without much distortion. In order not to raise data-thieves' attention, it is important to keep the visual quality of a steganographic image visually close to a cover image. Based on the reversibility of the cover image, the data hiding methods can be divided into two categories, namely, reversible data hiding (RDH) and irreversible data hiding (IRDH). Depending on the applications' requirements, reversible ones recover the original media, while irreversible ones either do not care about the original media or the stego media has a comparable quality to its originals. Moreover, the data hiding methods can be classified into four categories according to the domains being processed: spatial [11–18], transform [19–21], compressed [22–24], and encryption [8,25–27]. When using spatial domain methods, secret data are embedded into cover images by modifying their pixel values. Different to methods in the spatial domain, the schemes in the transform domain convert the plaintext images into frequency coefficients first and modify the frequency coefficients to realize the secret embedding. In the compressed domain, the schemes compress the original images and utilize the compressed images as the carriers to embed data. With the rapid development of cloud technology, DH schemes in the encryption domain have emerged. These schemes encrypt cover images before transmitting them through the public channel to ensure the privacy of the cover images while embedding secret information.

The traditional permutation-based data hiding methods embed secret data by rearranging the pixels in an image [28,29]. In [28], Wang et al. divided the image into blocks and then rotated the pixels within each block to embed an additional 2 bits of data. The additional data can be extracted without loss by considering the location of the maximum or minimum pixel. In [29], Xu et al. proposed a regularization operation to preprocess the divided image blocks, converting most of them into regulated blocks. The additional data was then embedded into these image blocks by permuting the pixels within. Since the permutation process only disrupts the pixels without directly modifying the pixel values, it becomes challenging for malicious attackers to detect the stego image through pixel-based analyses, such as pixel histogram analysis. However, one drawback of the existing methods is that the pixels are reordered block-wise, which may lead to a dramatic change in pixel values at the edges after permutation. To solve this problem, the proposed scheme uses pixel-based permutation with two critical coefficients instead of block-wise permutation to hide data to efficiently avoid some disadvantages of block-based permutations. The novel scheme proposed here can produce better results because there are two important control coefficients in the offered scheme. One of them controls the range of the differences allowed in a selected group and the other one is to control how many members or instances can be in a group. Cover images are scanned from left to right and from top to bottom to form a pixel stream. By using the two coefficients, a multiset is created based on the current unprocessed pixel. A permutation table is generated using a random seed according to the selected multiset. Extracting secret data uses a regenerated permutation table during the data recovery stage. It will embed secret data into the cover image by the bits determined by the distinctive permutations of the multisets. Our experiments show promising results using this proposed permutation-based embedding scheme.

The contributions of the proposed scheme are described below:

- A context owner can control the distortion of cover images by using only two coefficients.
- When the visual quality of the embedded images is in an acceptable situation, the embedded capacity is relatively high compared to the state-of-the-art methods.
- The proposed scheme demonstrates its high potential in IoT security.

As we briefly stated our problem and a rough description of the anticipated solution, the related methods are to be explained in Section 2. Section 3 will lay out the prime details of the proposed scheme, followed by Section 4, which includes the experiments conducted and the analysis completed. Lastly, the conclusion will summarize our research findings.

2. Related Work

There are a few basic concepts that are integrated in the proposed scheme. General descriptions of these concepts are described to refresh some fundamental knowledge related to multisets.

2.1. Permutations of Multisets

In mathematics, a multiset is a modified set which allows multiple instances of an element. A monomial is a multiset of indeterminates. If there is a finite set,

$$A = \{a_1,\ a_2,\ \cdots,\ a_n\}, \tag{1}$$

a multiset can be represented as

$$(A,\ m) = \left\{a_1^{m(a_1)},\ a_2^{m(a_2)},\ \cdots,\ a_n^{m(a_n)}\right\}, \tag{2}$$

where A is the base set of the multiset, m is the multiplicity function from A, and $\{a_1, a_2, \cdots, a_n\}$ are distinct elements in $(A,\ m)$; $m(a_n)$ is the multiplicity of the element a_n or the number of occurrences of the element a_n. For example, a multiset with $\{a,\ a,\ b,\ b,\ c\}$ element instances can be rewritten as monomial $\{a^2,\ b^2,\ c\}$ or a^2b^2c.

If the multiplicities of the distinct elements in $(A,\ m)$ are $m(a_1),\ m(a_2), \cdots, m(a_n)$, the size of $(A,\ m)$ is as indicated in Equation (3):

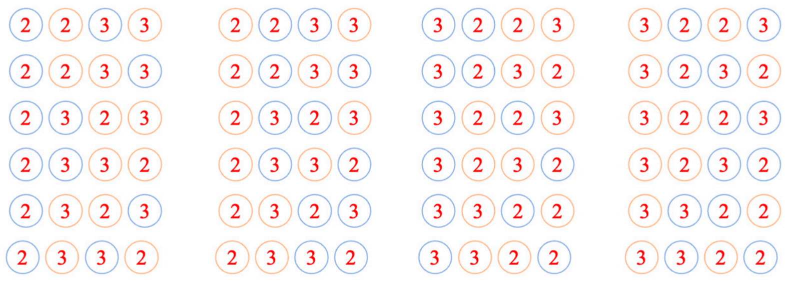
$$S = m(a_1) +\ m(a_2) + \cdots + m(a_n). \tag{3}$$

The number of distinct permutations P of the multiset $(A,\ m)$ is

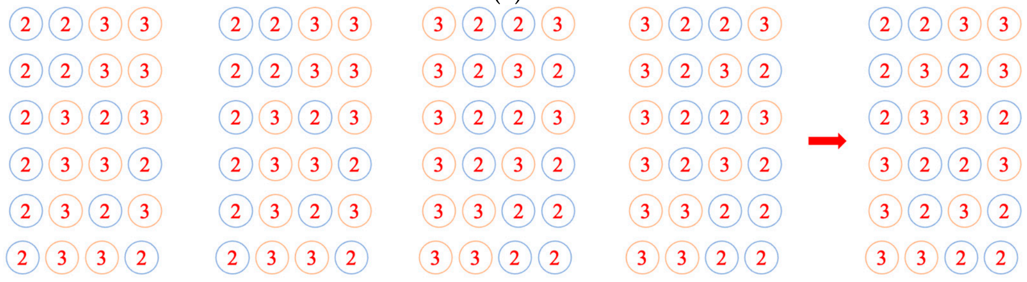
$$P = \binom{S}{m(a_1),\ m(a_2), \cdots, m(a_n)} = \frac{S!}{m(a_1)!m(a_2)! \cdots m(a_n)!} = \frac{S!}{\prod_{i=1}^{n} m(a_i)!}, \tag{4}$$

where the factorial $S!$ is the total number of possible permutations for the multiset. The factorial of $m(a_i)!$ is the amount of repeated permutations due to the multiplicity of $m(a_i)$.

Here is an example to help with understanding the permutations of a multiset: if there is a multiset $(A,\ m) = \{2, 2, 3, 3\}$, it can also be represented as $\{2^2,\ 3^2\}$. The total number of permutations of the multiset is 24, but the number of distinct permutations of the multiset is six according to $P = \frac{(2+2)!}{2! \times 2!} = \frac{4 \times 3 \times 2 \times 1}{(2 \times 1) \times (2 \times 1)} = 6$. Figure 1 displays the example's possible permutations and its distinct permutations.



(**a**)



(**b**)

Figure 1. Permutations of multiset {2, 2, 3, 3}: (**a**) all possible permutations; (**b**) distinct permutations.

2.2. Permutation Ordering

The Fisher–Yates shuffle was first introduced in 1938 and is also known as the Knuth shuffle [30]. It is an algorithm using random drawing to generate a random permutation of numbers from 1 to N. There are many algorithms that were derived to handle the duplicated numbers and implement them more efficiently to improve the performance of the original shuffle method since the controlling of the shuffling sequences is widely applied on many different applications.

3. Permutation-Based Embedding Scheme

When dealing with data hiding problems, researchers are looking for a solution with high visual media quality and high embedding quantity. The proposed scheme is straightforward and uses simple permutation operations. Figure 2 demonstrates the fundamental framework of the scheme.

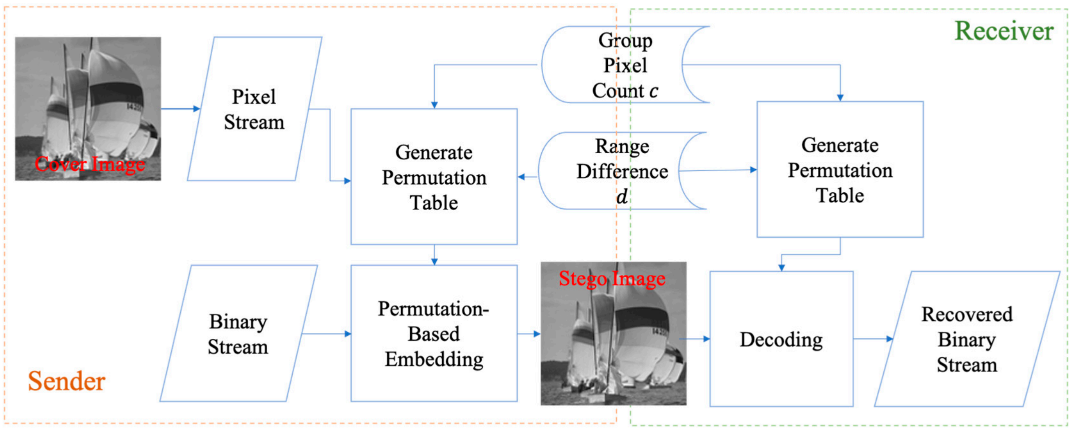


Figure 2. Framework of permutation-based embedding scheme.

A cover image I sized $W \times H$ is converted to a sequence of pixels from left to right and from top to bottom.

$$I = \{p_{00},\ p_{10},\ \cdots,\ p_{W-1,0},\ p_{01}, p_{11}, \cdots, p_{W-1,\ H-1}\}. \tag{5}$$

The p_{ij} is the pixel located at column i and row j. It is limited to greyscale images because the pixel values range from $[0,\ 255]$ for simpler number manipulation in our experiments.

There are two important coefficients for a group in the proposed scheme:

- A group pixel count c, where $c > 1$. It is used to limit the number of pixels in the multiset for fast permutation table creation.
- A difference d, where $d \geq 1$. It is to limit the range of pixel values. When a difference coefficient is large, the pixel variation among pixels in the group is large which can impact the visual quality of the cover image.

3.1. Permutation-Based Data Embedding

A list of inputs is required from a sender in order to generate permutation tables and embed secret data into a stego image to send to a receiver. The algorithm represents the flow of secret data embedding after the sender has a cover image I, a group instance count c, a difference coefficient d, and the secret data S.

$$S = \{s_0,\ s_1,\ \cdots,\ s_{r-1}\}. \tag{6}$$

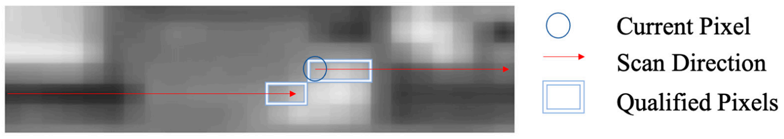
here, $s_k = \{0,\ 1\}$ and $0\ \leq k\ < r$.

After all pixels in the cover image are processed, a stego image I' is created and sent to the receivers. The detail steps are described in Algorithm 1.

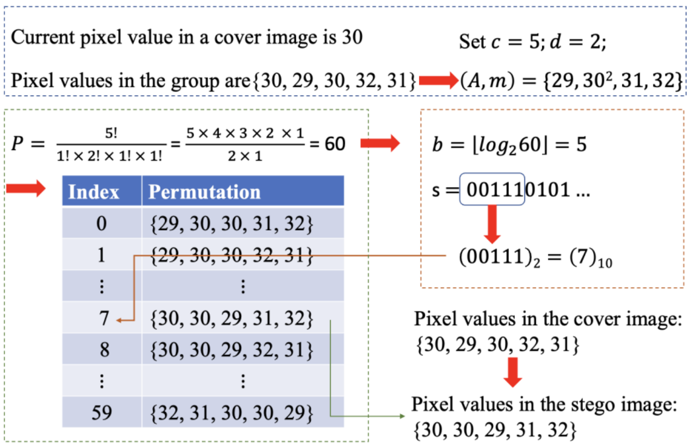
Algorithm 1: Data embedding

```
Input     A greyscale cover image $I$, a group pixel count $c$, a difference $d$, and the secret data $S$.
Output    A stego image $I'$.
Step 1    Set the first pixel from the image $I$ as the start pixel with $p_{ij} = p_{00}$.
Step 2    Set current value $v_{ij}$ as equal to the pixel value of the start pixel $p_{ij}$ and add $v_{ij}$ to a new group $G_{ij}$. Mark $p_{ij}$ as
          processed. Set $x = i,\ y = j$.
Step 3    Search pixel forward by advancing the pixel location $(x,\ y)$ until it finds a pixel value $v_{xy}$, where
          $v_{ij} - d \le v_{xy} \le v_{ij} + d$, add to the group $G_{ij}$ and mark the found pixel as processed.
Step 4    Repeat Step 3 until the number of pixels in the group $G_{ij}$ reaches the specified group pixel count $c$ or there are no
          more unprocessed pixels.
Step 5    Check if the number of pixels is less than $c$; if it is yes, continue to Step 10.
Step 6    Check if there is only one distinct pixel value in the group $G_{ij}$; if it is yes, continue to Step 10.
Step 7    Generate a permutation table according to pixel values in the group $G_{ij}$ using random seed shuffling and calculate
          the number of distinctive permutations $P_{ij}$ using Equation (4).
Step 8    Calculate the number of embeddable bits $b_{ij}$ using

              $b_{ij} = \lfloor log_2 P_{ij} \rfloor$        (7)

          where $P_{ij}$ is the number of distinctive permutations for the pixel group $G_{ij}$.
Step 9    Convert $b_{ij}$ of binary value to decimal as the index value $IDX_{ij}$ into the permutation table generated in Step 6.
          Replace the group pixel values with the ordered values pointed by the index $IDX_{ij}$.
Step 10   Check if there is an unprocessed pixel left in the pixel stream. If there is, set it as the start pixel, set the current
          pixel $p_{ij}$ and repeat Step 2. Otherwise, stop the process and send the output stego image $I'$.
```

Figure 3 details an example of permutation-based data embedding. In this example, set the permuting count in a group c to 5 and the difference coefficient to 2. In Figure 3a, assume that the circled pixel is the current processed pixel with value 30 in image I and that the proposed scheme uses a raster scan method to find five pixels with values that are within 2 in differences of the value 30. Figure 3b shows these pixel values are $\{30,\ 29,\ 30,\ 31,\ 32\} = \{29,\ 30^2,\ 31,\ 32\}$. The number of its possible distinctive permutations is 60; therefore, the number of embeddable bits is $\lfloor log_2 60 \rfloor = 5$ according to Equation (7). Assume the next 5 bits in secret data S is $(00111)_2 = 7$, index 7 is selected from permutation table and its number permutation $\{30,\ 30,\ 29,\ 32,\ 31\}$ replaces $\{30,\ 29,\ 30,\ 31,\ 32\}$ in the stego image I'.



(a)


Current pixel value in a cover image is 30
Set $c = 5; d = 2;$
Pixel values in the group are{30, 29, 30, 32, 31} → $(A, m) = \{29, 30^2, 31, 32\}$
$P = \frac{5!}{1! \times 2! \times 1! \times 1!} = \frac{5 \times 4 \times 3 \times 2 \times 1}{2 \times 1} = 60$
$b = \lfloor log_2 60 \rfloor = 5$
Index | Permutation
0 | {29, 30, 30, 31, 32}
1 | {29, 30, 30, 32, 31}
⋮ | ⋮
7 | {30, 30, 29, 31, 32}
8 | {30, 30, 29, 32, 31}
⋮ | ⋮
59 | {32, 31, 30, 30, 29}
s = 001110101 ...
$(00111)_2 = (7)_{10}$
Pixel values in the cover image: {30, 29, 30, 32, 31}
Pixel values in the stego image: {30, 30, 29, 31, 32}

(b)

Figure 3. Permutation-based embedding: (**a**) search qualified pixels in group from current pixel; (**b**) process of pixel replacement in a group.

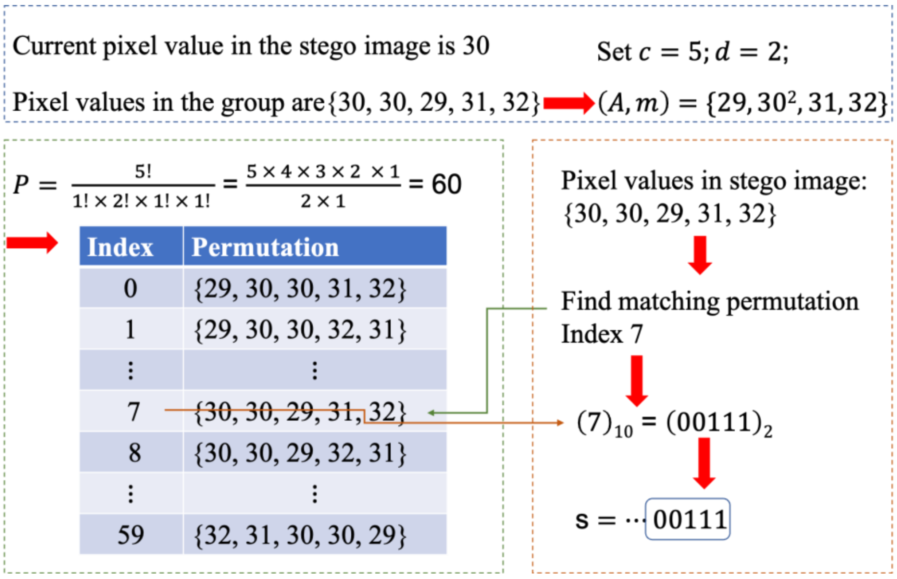
3.2. Data Extraction

When a receiver receives a stego image, the embedded secret data can be extracted by the following steps if there are known c and d. The extraction steps are described in Algorithm 2.

Algorithm 2: Data extraction

```
Input     The stego image $I'$, the group pixel count $c$, and the difference $d$.
Output    Recovered image $I''$.
Step 1    Convert stego image $I'$ to a pixel stream.
Step 2    Set the first pixel from the image $I'$ as the start pixel with $p'_{ij} = p'_{00}$
Step 3    Set current value $v'_{ij}$ as equal to the pixel value of the start pixel $p'_{ij}$ and add $v'_{ij}$ to a new group $G_{ij}$. Mark $p_{ij}$ as
          processed. Set $x = i,\ y = j$;
Step 4    Search forward by advancing the pixel location $(x,\ y)$ until it finds a pixel value $v'_{xy}$, where $v'_{ij} - d \le v'_{xy} \le v'_{ij} + d$,
          add to group and mark the pixel as processed.
Step 5    Repeat Step 4 until the number of group pixel count reach specified $c$ or there are no more pixels.
Step 6    Check if group member count is equal to $c$; continue to Step 11 if it is a yes.
Step 7    Check if there is more than one distinct member in the group; if there is only one, continue to Step 11.
Step 8    Generate a permutation table according to group members using random seed shuffling.
Step 9    Match the order of the group members with all permutation orders in the permutation table to obtain the table
          index of the order.
Step 10   Convert the order into a binary bit stream and append them to the decoded binary secret stream.
Step 11   Check if there is an unprocessed pixel left in the pixel stream. If there is, set it as the start pixel, set the current pixel
          $p'_{ij}$ and repeat Step 3. Otherwise, stop the process with a recovered image $I''$.
```

Figure 4 demonstrates the extraction of the secret data in our embedding example. After receiving c, d, and the stego image I', the pixel values extracted are $\{30,\ 30,\ 29,\ 32,\ 31\} = \{29,\ 30^2,\ 31,\ 32\}$. We can find that the index number of $\{30,\ 30,\ 29,\ 32,\ 31\}$ in the permutation table is 7, and thus the secret message $(00111)_2$ is extracted to be appended to the previously extracted secret data stream.


Current pixel value in the stego image is 30
Set $c = 5; d = 2;$
Pixel values in the group are{30, 30, 29, 31, 32} $(A, m) = \{29, 30^2, 31, 32\}$
$P = \frac{5!}{1! \times 2! \times 1! \times 1!} = \frac{5 \times 4 \times 3 \times 2 \times 1}{2 \times 1} = 60$
Pixel values in stego image: {30, 30, 29, 31, 32}
Find matching permutation Index 7
$(7)_{10} = (00111)_2$
$s = \cdots 00111$

Index	Permutation
0	{29, 30, 30, 31, 32}
1	{29, 30, 30, 32, 31}
⋮	⋮
7	{30, 30, 29, 31, 32}
8	{30, 30, 29, 32, 31}
⋮	⋮
59	{32, 31, 30, 30, 29}

Figure 4. Permutation-based secret data extraction.

4. Experimental Results and Analysis

Since the visual performance of images and data payload can judge the efficiency of the proposed scheme, experiments with the image quality, which showed desirable results, were conducted first. In order to confirm that the data payload results from our scheme can surpass the state-of-the-art schemes, related experiments were also completed for comparisons afterwards. The selected test images for experiments are displayed in Figure 5. The four images are popular greyscale images with diverse smoothness. The corresponding histograms of the images are shown in Figure 5 as well.

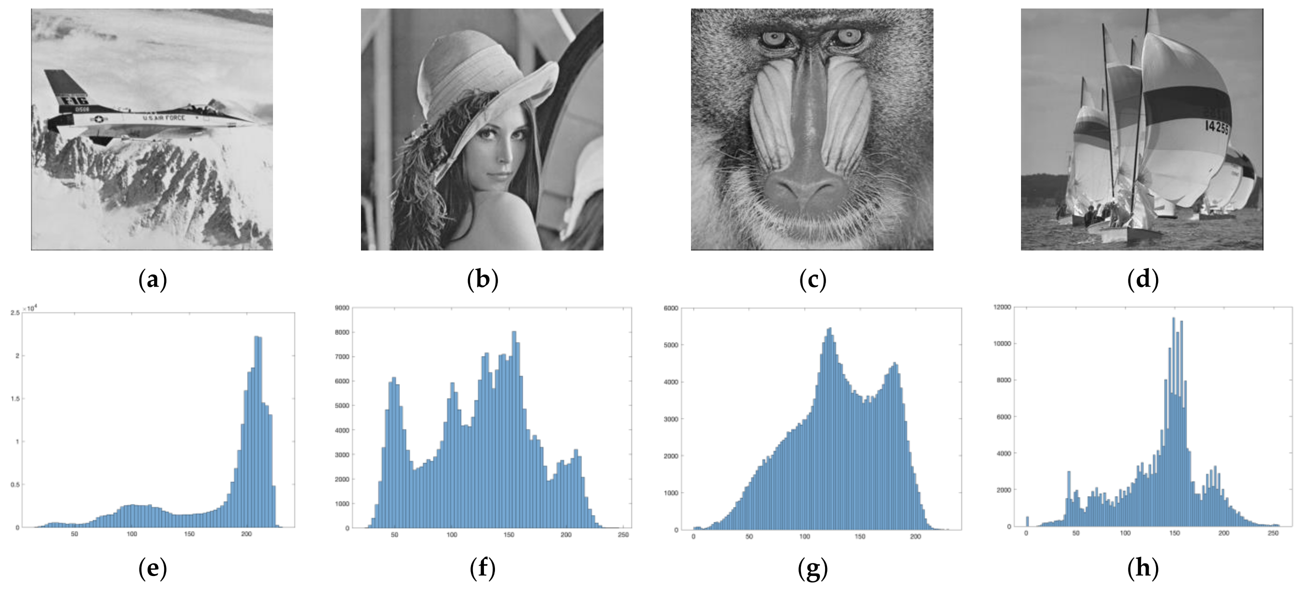


Figure 5. Four standard test images: (**a**) airplane, (**b**) Lena, (**c**) baboon, and (**d**) sailboat. (**e**–**h**) are the histograms of images (**a**–**d**).

4.1. Visual Performance

To visually demonstrate the effectiveness of the proposed scheme in terms of the visual performance, we present the stego images in Figure 6. The experiment sets the range difference to be 2 and the member count in groups to be 5. From the figure, we can see that all stego images have an outstanding visual quality. The differences in the test set are not recognizable when viewing with the naked eye. Furthermore, when examining the histogram distributions corresponding to these stego images, we observe that they are identical to those of the original images. Therefore, malicious attackers cannot detect the stego images through histogram analysis.

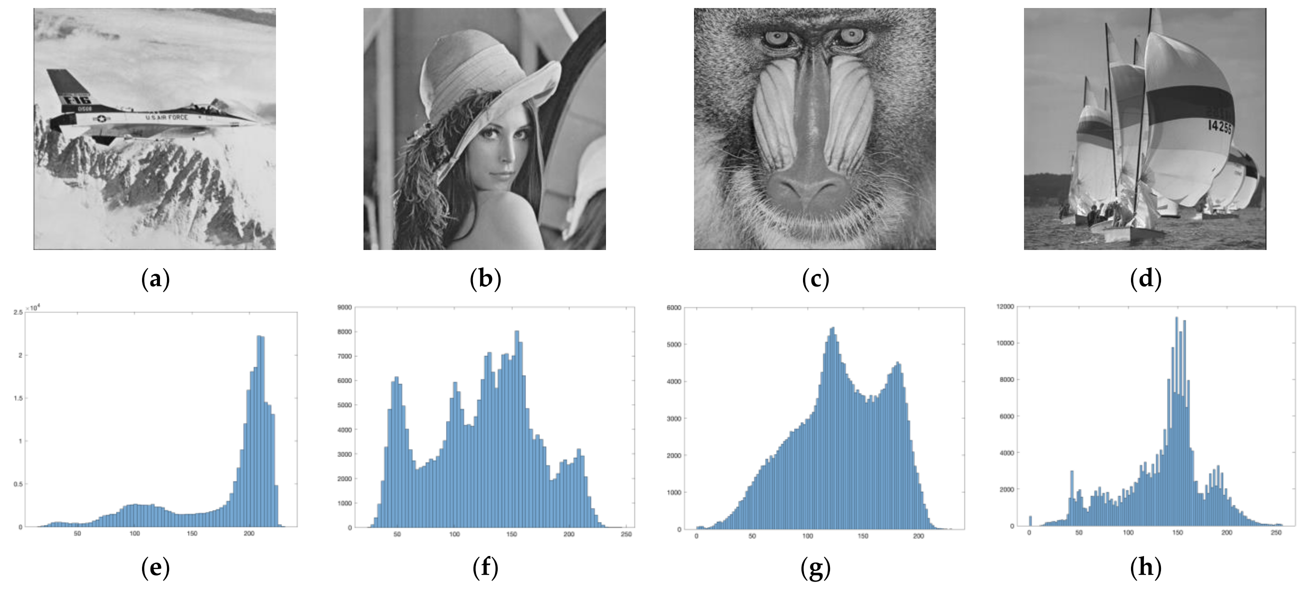


Figure 6. Visual results and their corresponding histograms when setting $\boldsymbol{d} = \mathbf{2}$, $\boldsymbol{c} = \mathbf{5}$. (**a**) airplane, (**b**) Lena, (**c**) baboon, and (**d**) sailboat. (**e**–**h**) are the histograms of images (**a**–**d**).

When evaluating the visual performance, the peak signal-to-noise ratio (PSNR) [31] is measured in the experiments to check the amount of distortion between a cover image and a steganographic image.

$$\mathrm{PSNR} = 10\left(\frac{W \times H \times 255^2}{\sum_{i=0}^{W-1}\sum_{j=0}^{H-1}(p_{ij} - p'_{ij})^2}\right), \tag{8}$$

where the size of a cover image is $W \times H$, p_{ij} is the pixel value at (i, j) of the cover image, and p'_{ij} is the pixel value at (i, j) of its stego image. The results are given in Tables 1 and 2. The experiments were conducted by using various range difference coefficient d with a fixed member count in groups; varying member counts and fixing difference coefficient d were also attempted. The PSNRs were reduced when the range difference was increased. As the differences among the pixel values become larger, it causes greater color distortion of the pixels. Nevertheless, it seems there is no impact on changing the group pixel count c. These results can be observed in the two tables.

Table 1. PSNR results when setting $c = 5$.

	Airplane	**Lena**	**Baboon**	**Sailboat**
$d = 1$	49.9138	49.6091	49.4841	50.1613
$d = 2$	45.5175	44.9660	44.7239	45.7078
$d = 3$	42.9644	42.1785	41.7483	42.7407

Table 2. PSNR results when setting $d = 2$.

	Airplane	**Lena**	**Baboon**	**Sailboat**
$c = 4$	45.9944	45.4401	45.2277	46.1493
$c = 5$	45.5175	44.9660	44.7239	45.7078
$c = 6$	45.2598	44.6983	44.4488	45.4289

We employed another two key metrics, the information entropy [32] and correlation coefficient [33], respectively, to further assess our scheme. The information entropy measures the complexity and the information content of images, while the correlation coefficient quantifies the similarity between two images. As shown in Table 3, the information entropy of the stego images matches that of the original images, indicating that our method maintains the information content of the images. The results emphasize that our approach is not only effective in hiding information but also excellent in retaining the inherent informational characteristics of the cover images. Moreover, the data in Table 4 reveals a high degree of similarity between our stego images and the original images with correlation coefficients approaching 1, further confirming the method's exceptional ability to maintain the inherent informational characteristics of the cover images. These two experiments provide additional evidence of the imperceptibility of our method.

Table 3. Image information entropy when setting $d = 2$, $c = 5$.

	Airplane	**Lena**	**Baboon**	**Sailboat**
Original image	6.7059	7.4455	7.3579	7.1914
Stego image	6.7059	7.4455	7.3579	7.1914

Table 4. Image correlation coefficient results when setting $d = 2$, $c = 5$.

	Airplane	**Lena**	**Baboon**	**Sailboat**
-	0.9995	0.9997	0.9995	0.9996

4.2. Embedding Capacity

To determine if the amount of the data embedded is satisfying or not, embedding capacity (EC) is measured. EC is the total data payload imbedded or is the total number of bits of data embedded.

The proposed scheme shows high embedding capacity according to the embedded bit counts in Tables 5 and 6. The results indicate that the ECs are improved when the range difference d is larger. It also means that embedding amounts are increased when more qualified groups can be formed. When the member count of groups is boosted, the EC is increased as well. The reason is that as the amount of distinct permutations becomes larger, more bits can be embedded.

Table 5. EC results when c is set to 5.

	Airplane	**Lena**	**Baboon**	**Sailboat**
$d = 1$	762,655	827,630	852,455	705,600
$d = 2$	1,581,345	1,777,900	1,864,670	1,431,095
$d = 3$	2,174,145	2,466,185	2,637,900	2,023,410

Table 6. EC results when d is set to 2.

	Airplane	**Lena**	**Baboon**	**Sailboat**
$c = 4$	673,780	729,240	755,710	625,768
$c = 5$	1,581,345	1,777,900	1,864,670	1,431,095
$c = 6$	4,175,618	4,827,111	5,172,407	3,684,086

Even though our scheme is designed without reversibility, we add experiments to include a simple reversible method by simply recording original permutation indices to make sure the proposed scheme can obtain reasonable results even if we add a reversible mechanism. Even though our algorithms and examples demonstrated a non-reversible scheme, it can be easily modified to support reversibility by embedding original index orders of permuted data. The embedding capacity will be less as expected because of the extra storage needed for auxiliary information to be used for image recovery. Tables 7 and 8 are the experiments with the reversible capability which showed promising results as well.

Table 7. EC results with reversibility when $c = 5$.

	Airplane	**Lena**	**Baboon**	**Sailboat**
$d = 1$	411,150	473,325	497,744	362,495
$d = 2$	1,218,577	1,413,046	1,499,844	1,071,323
$d = 3$	1,601,573	1,890,819	2,271,478	1,659,970

Table 8. EC results with reversibility when $d = 2$.

	Airplane	**Lena**	**Baboon**	**Sailboat**
$c = 4$	355,225	407,340	433,415	594,303
$c = 5$	1,218,577	1,413,046	1,499,844	1,071,323
$c = 6$	36,988,031	4,347,775	4,692,851	3,208,897

4.3. Robustness Analysis

To analyze the robustness of our scheme against various attacks, such as noise attacks and shear attaches, we used "Lena" as the cover image and the "FCU logo" as the secret data in the experiments of the robustness checks. In Figure 7, (a1) is the stego image without any attacks, (b1) encountered noise attacks and (c1,d1) is with shear attacks. (a2,b2,c2,d2) are the extracted secret data from the stego images with or without attacks. (a3,b3,c3,d3) are the recovered images after extracting the secret. However, it should be noted that the cover image could not be recovered when the additional data for original image recovery is cut. As we examine the results of our experiments related to robustness, we can not only resist both noise attacks and small-scale shear attacks but also achieve a high visual quality on both the recovered secrets and the recovered images. While the performance of our method is diminished under large-scale shear attacks, it is worth noting that the recovered secret image remains discernible to some extent.

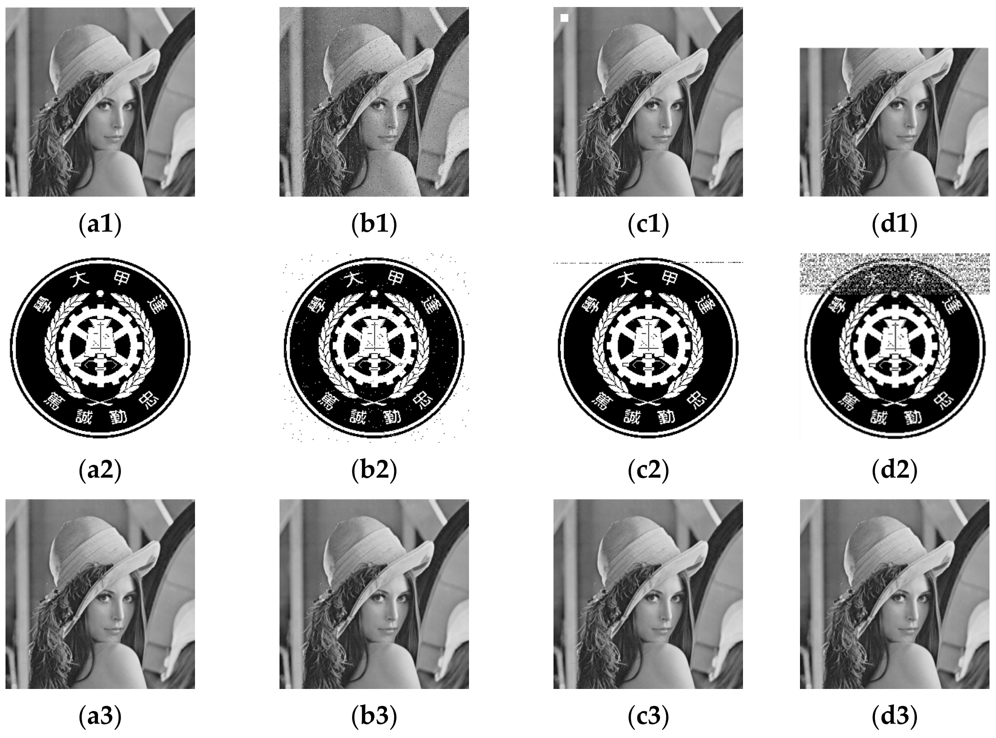


Figure 7. (**a1**) "Lena" cover image without attacks, (**b1**) "Lena" with salt and pepper noise attacks, (**c1**) "Lena" with shear attacks (20 × 20 patch), (**c1**) "Lena" with shear attacks (80 × 512 patch), (**a2**) extracted "FCU logo" from (**a1**), (**b2**) extracted "FCU logo" from (**b1**), (**c2**) extracted "FCU logo" from (**c1**), (**d2**) extracted "FCU logo" from (**d1**), (**a3**) recovered "Lena" image from (**a1**), (**b3**) recovered "Lena" image from (**b1**), (**c3**) recovered "Lena" image from (**c1**), (**d3**) recovered "Lena" image from (**d1**).

4.4. Performance Comparison with Other DH Methods

Other than some basic experiments to analyze the performances of the proposed scheme, we compare our results with some of the state-of-the-art schemes [15–18,29] to make sure our results are still solid. The results of Table 9 (Comparative performance of PSNR (dB)) use four grayscale images to compare between our method and the state-of-the-art ones by using range difference between [−1, 1] with group size 5. We can see from the numbers showed in Table 9 that the visual quality of the proposed scheme outperformed the rest of selected schemes.

Table 9. Comparative performance of PSNR (dB).

	Airplane	**Lena**	**Baboon**	**Sailboat**
[15]	35.30	35.00	31.57	33.68
[16]	36.16	35.40	33.87	33.56
[17]	40.37	40.42	40.41	40.40
[14]	44.87	46.21	43.13	43.95
[29]	46.27	45.56	44.98	45.13
[18]	46.38	46.38	46.38	46.38
Proposed scheme	49.91	49.60	49.48	50.16

Table 10 (Comparative performance of EC) also uses four grayscale images to compare between our method and the state-of-the-art schemes. The EC results are collected when setting c to 5 and d to 2. Two coefficients c and d values can be adjusted if a larger EC is desired.

Table 10. Comparative performance of EC.

	Airplane	**Lena**	**Baboon**	**Sailboat**
[29]	166,289	158,203	140,095	153,902
[16]	264,389	253,001	169,152	253,000
[14]	275,251	267,386	374,865	309,329
[18]	524,288	524,288	524,288	524,288
Proposed scheme	1,581,345	1,777,900	1,864,670	1,431,095

Table 11 (Comparative performance of EC) again uses four grayscale images to compare between our method and the state-of-the-art reversible data hiding schemes when $c = 5$ and $d = 2$. Same as in the above table, the c and d values can be adjusted if a larger EC is desired. While the proposed scheme is a non-reversible method, we reserve the maximum amount of space in order to store data for reversibility support.

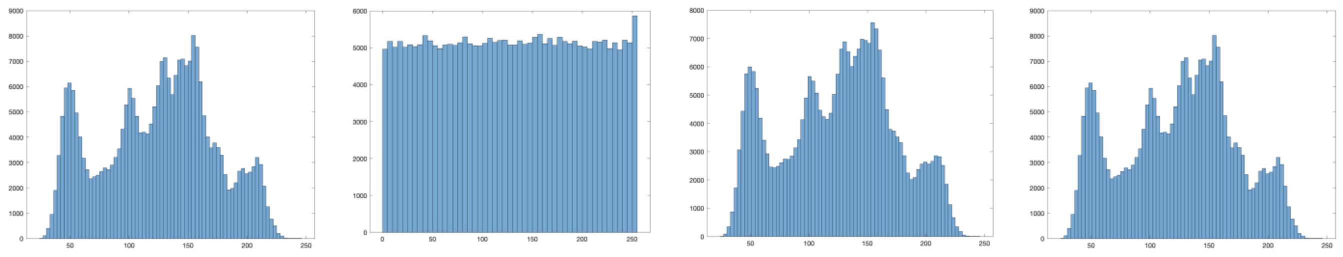
Table 11. Comparative performance of EC.

	Airplane	**Lena**	**Baboon**	**Sailboat**
[15]	404,937	537,395	715,653	602,931
[17]	655,360	655,360	655,360	655,360
Proposed scheme	1,218,577	1,413,046	1,499,844	1,071,323

We conducted a computation timing analysis comparing with other permutation-based data hiding methods [28,29] to conclude our experiments. Based on the timing results in Table 12, our scheme did not have better timing comparing with the others. The main reason for this is that our scheme required forward searches which may take time. Nevertheless, we think it is worthwhile to spend a little extra time to exchange for the outstanding embedding capacity results. In addition, we provide stego image histograms of these two permutation-based methods and our method using "Lena" as an example. From the histogram results (Figure 8), it is evident that the histogram of [28] exhibits a uniform distribution, which is due to the fact that it is conducted on the encrypted image. The histogram of [29] roughly approximates that of the original image, but its actual data distribution has also changed. In contrast, the histogram obtained by our method is identical to that of the original image. Hence, we can deduce that the other two permutation-based methods may be detected by histogram analysis, while our method will not be.

Table 12. Time complexity comparison for data embedding.

	Airplane	**Lena**	**Baboon**	**Sailboat**
[28]	0.0038	0.0057	0.0094	0.0088
[29]	0.0032	0.0059	0.0083	0.0082
Proposed scheme	0.0056	0.0085	0.0117	0.0106



(**a**) (**b**) (**c**) (**d**)

Figure 8. Histogram results of the "Lena" image. (**a**) Original image, (**b**) stego image of [28], (**c**) stego image of [29], and (**d**) stego image of the proposed scheme.

5. Conclusions

The proposed scheme offers a simple yet efficient permutation-based approach which manipulates nearby pixels in greyscale images using two coefficients to control. One of the controls is a threshold of the range for the pixel values and the other is a specified clustering count. The experiments show that the method can effectively improve the embedding capacities by using the indices of the permutation tables. The experimental results show that the proposed scheme offers an excellent embedding capacity while preserving a high level of image visual quality. Therefore, the proposed scheme underscores its significant potential in enhancing IoT security.

Because pixel changes in the scheme are limited, the visual quality is retained. Since our scheme is simple and can be easily integrated with others, it should also benefit other state-of-the-art methods.

Author Contributions: Conceptualization, C.-C.C. (Ching-Chun Chang) and C.-C.C. (Chin-Chen Chang); methodology, C.-C.C. (Ching-Chun Chang) and C.-C.C. (Chin-Chen Chang); software, S.X.; validation, S.X. and J.-C.L.; formal analysis, S.X. and J.-C.L.; investigation, J.-C.L.; resource, C.-C.C. (Ching-Chun Chang) and J.-C.L.; data curation, J.-C.L.; writing—original draft preparation, J.-C.L.; writing—review and editing, C.-C.C. (Ching-Chun Chang), S.X. and C.-C.C. (Chin-Chen Chang); visualization, J.-C.L. and S.X.; supervision, C.-C.C. (Chin-Chen Chang) and J.-C.L.; project administration, C.-C.C. (Chin-Chen Chang). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ateya, A.A.; Mahmoud, M.; Zaghloul, A.; Soliman, N.F.; Muthanna, A. Empowering the internet of things using light communication and distributed edge computing. *Electronics* **2022**, *11*, 1511. [CrossRef]
2. Ahmid, M.; Kazar, O. A comprehensive review of internet of things security. *J. Appl. Secur. Res.* **2023**, *18*, 289–305. [CrossRef]
3. Anand, A.; Singh, A.K. A hybrid optimization-based medical data hiding scheme for industrial internet of things security. *IEEE Trans. Ind. Inform.* **2022**, *19*, 1051–1058. [CrossRef]
4. Huang, C.-T.; Tsai, M.-Y.; Lin, L.-C.; Wang, W.-J.; Wang, S.-J. VQ-based data hiding in IoT networks using two-level encoding with adaptive pixel replacements. *J. Supercomput.* **2018**, *74*, 4295–4314. [CrossRef]
5. Lakshmanna, K.; Kaluri, R.; Gundluru, N.; Alzamil, Z.S.; Rajput, D.S.; Khan, A.A.; Haq, M.A.; Alhussen, A. A review on deep learning techniques for IoT data. *Electronics* **2022**, *11*, 1604. [CrossRef]
6. Bender, W.; Gruhl, D.; Morimoto, N.; Lu, A. Techniques for data hiding. *IBM Syst. J.* **1996**, *35*, 313–336. [CrossRef]
7. Ni, Z.; Shi, Y.-Q.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
8. Kim, C.; Dao, N.-N.; Jung, K.-H.; Leng, L. Dual reversible data hiding in encrypted halftone images using matrix encoding. *Electronics* **2023**, *12*, 3134. [CrossRef]
9. Kumar, S.; Gupta, A.; Walia, G.S. Reversible data hiding: A contemporary survey of state-of-the-art, opportunities and challenges. *Appl. Intell.* **2021**, *52*, 7373–7406. [CrossRef]
10. Yu, C.; Zhang, X.; Li, G.; Zhan, S.; Tang, Z. Reversible data hiding with adaptive difference recovery for encrypted images. *Inf. Sci.* **2021**, *584*, 89–110. [CrossRef]

11. Yuan, J.; Zheng, H.; Ni, J. Efficient reversible data hiding using two-dimensional pixel clustering. *Electronics* **2023**, *12*, 1645. [CrossRef]
12. Kamil Khudhair, S.; Sahu, M.; K. R., R.; Sahu, A.K. Secure reversible data hiding using block-wise histogram shifting. *Electronics* **2023**, *12*, 1222. [CrossRef]
13. Peng, F.; Li, X.; Yang, B. Improved PVO-based reversible data hiding. *Digit. Signal Process.* **2014**, *25*, 255–265. [CrossRef]
14. Xiong, X.; Chen, Y.; Fan, M.; Zhong, S. Adaptive reversible data hiding algorithm for interpolated images using sorting and coding. *J. Inf. Secur. Appl.* **2022**, *66*, 103137. [CrossRef]
15. Chi, H.-X.; Horng, J.-H.; Chang, C.-C.; Li, Y.-H. Embedding Biometric Information in Interpolated Medical Images with a Reversible and Adaptive Strategy. *Sensors* **2022**, *22*, 7942. [CrossRef] [PubMed]
16. Nguyen, T.-S.; Huynh, V.-T.; Vo, P.-H. A novel reversible data hiding algorithm based on enhanced reduced difference expansion. *Symmetry* **2022**, *14*, 1726. [CrossRef]
17. Gao, K.; Horng, J.-H.; Chang, C.-C. An authenticatable (2, 3) secret sharing scheme using meaningful share images based on hybrid fractal matrix. *IEEE Access* **2021**, *9*, 50112–50125. [CrossRef]
18. Horng, J.-H.; Xu, S.Y.; Chang, C.-C. An efficient data-hiding scheme based on multidimensional mini-SuDoKu. *Sensors* **2020**, *20*, 2739. [CrossRef]
19. Zhang, H.; Hu, L.T. A data hiding scheme based on multi-directional line encoding and integer wavelet transform. *Signal Process. Image Commun.* **2019**, *78*, 331–334. [CrossRef]
20. Gao, G.; Shi, Y.-Q. Reversible data hiding using controlled contrast enhancement and integer wavelet transform. *IEEE Signal Process. Lett.* **2015**, *22*, 2078–2082. [CrossRef]
21. Kim, C.; Yang, C.-N.; Leng, L. High-capacity data hiding for ABTC-EQ based compressed image. *Electronics* **2020**, *9*, 644. [CrossRef]
22. Huang, F.J.; Qu, X.C.; Kim, H.J.; Huang, J.W. Reversible data hiding in JPEG images. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 1610–1621. [CrossRef]
23. Hong, W.; Ma, Y.-B.; Wu, H.-C.; Chen, T.-S. An efficient reversible data hiding method for AMBTC compressed images. *Multimed. Tools Appl.* **2017**, *76*, 5441–5460. [CrossRef]
24. Kim, C. Dual reversible data hiding based on AMBTC using Hamming code and LSB replacement. *Electronics* **2022**, *11*, 3210. [CrossRef]
25. Zhang, X. Separable and error-free reversible data hiding in encrypted images. *Signal Process.* **2016**, *123*, 9–21.
26. Zhang, H.; Li, L.; Li, Q. Reversible data hiding in encrypted images based on block-wise multi-predictor. *IEEE Access* **2021**, *9*, 61943–61954. [CrossRef]
27. Zhang, W.; Ma, K.; Yu, N. Reversibility improved data hiding in encrypted images. *Signal Process.* **2014**, *94*, 118–127. [CrossRef]
28. Wang, X.; Chang, C.-C.; Lin, C.-C. Reversal of pixel rotation: A reversible data hiding system towards cybersecurity in encrypted images. *J. Vis. Commun. Image Represent.* **2022**, *82*, 103421. [CrossRef]
29. Xu, S.; Chang, C.-C.; Horng, J.-H. Image covert communication with block regulation. *IEEE Signal Process. Lett.* **2023**, *30*, 1217–1221. [CrossRef]
30. Knuth, D.E. *The Art of Computer Programming*; Pearson Education: London, UK, 1997; Volume 3.
31. Smith, S. *Digital Signal Processing: A Practical Guide for Engineers and Scientists*, 2nd ed.; Newnes: Boston, MA, USA, 2002; pp. 532–535.
32. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
33. Asuero, A.G.; Sayago, A.; González, A.G. The correlation coefficient: An overview. *Crit. Rev. Anal. Chem.* **2006**, *36*, 41–59. [CrossRef]