mathematics

Article

A Novel LSB Matching Algorithm Based on Information Pre-Processing

Yongjin Hu [1], Xiyan Li [1,2,*] and Jun Ma [1]

[1] Department of Cryptogram Engineering, Information Engineering University, Zhengzhou 450001, China; hu_yongjin@126.com (Y.H.); sijunhan@163.com (J.M.)
[2] School of Information Science and Technology, Hainan Normal University, Haikou 571158, China
* Correspondence: xiyanli2006@163.com

Abstract: This paper analyzes random bits and scanned documents, two forms of secret data. The secret data were pre-processed by halftone, quadtree, and S-Box transformations, and the size of the scanned document was reduced by 8.11 times. A novel LSB matching algorithm with low distortion was proposed for the embedding step. The golden ratio was firstly applied to find the optimal embedding position and was used to design the matching function. Both theory and experiment have demonstrated that our study presented a good trade-off between high capacity and low distortion and is superior to other related schemes.

Keywords: data hiding; halftone; quadtree; LSB matching; golden ratio

check for updates

Citation: Hu, Y.; Li, X.; Ma, J. A Novel LSB Matching Algorithm Based on Information Pre-Processing. *Mathematics* **2022**, *10*, 8. https://doi.org/10.3390/math10010008

Academic Editor: Radi Romansky

Received: 18 November 2021
Accepted: 16 December 2021
Published: 21 December 2021

1. Introduction

With the development of the internet, the transmission and sharing of information have become increasingly convenient. However, with this convenience, criminals may tamper with or intercept information on the internet. To solve the apparently conflicting open access of the network and information security, many privacy protection methods have been studied [1–4]. Encryption can protect privacy, but the spread of encrypted files on the internet easily attracts the attention of attackers. Information hiding technology, which hides secret information in the carrier, emerged in the 1990s. After more than 20 years of research and development, the technology has gained a measure of maturity, although it is still the focus of research in network security.

According to whether an embedded image can be reconstructed, information hiding is divided into two types, reversible and irreversible. Reversible information hiding is usually divided into four categories: lossless compression [5–37], difference expansion [8–10], prediction error expansion [11–13], and histogram shifting [14–16]. All reversible information-hiding schemes can extract secret information and restore the original image; however, the hiding capacity is not high. Most of the time, we need to embed a large volume of information with low distortion, and it does not matter whether the original image can be reconstructed entirely. The least significant bit (LSB) algorithm is a classic spatial information-hiding algorithm. The secret data are embedded into the least significant bit of the pixel value. The LSB algorithm has low complexity, simple operation, and greater hiding capacity, but its robustness is poor. Today, there are many LSB matching algorithms with low distortion.

There are two major concerns when selecting a gray image as the carrier to convey information. The first relates to the high capacity of its pixel modification: if the payload of each pixel for a cover image is less than 3 bpp, human vision is not able to detect the visual artifacts of a steganographic image. An LSB++ scheme was developed to improve the power of LSB-based algorithms. Generally, all these methods have tried to use the reductant space in the cover image more fully. The second concern is the quality of the steganographic image: Digital gray images are widespread on the internet. In many cases, secret data are embedded into cover images without noticeable visual artifacts. However, a high embedding capacity can distort the image, so the transfer is not secure.

With the digital development of life and office, the secure transfer of documents and mail through the internet became necessary. Although current steganography methods send information as random bits, few people have embedded scanned documents into cover images to transmit them safely [17–20]. The scanned documents were pre-processed and embedded into the cover image, so people could share more information with the same capacity. The earliest steganography method was a simple LSB (least-significant-bit) substitution [21]. In many images, differences in the least significant bits of a pixel are imperceptible, so they seem suitable for embedding sensitive information into a cover image. To hide greater volumes of data, many improved methods have been proposed [22–26]. The authors of [22–24] improved the visual quality of the simple LSB methods with low complexity; however, they were ineffective when the embedding rate was 1 bpp [25,26] designed embedding unit consists not only one pixel, and the distortion was better than [22–24]. Refs. [36,37] proposed dual-layer LSB matching algorithms with high embedding efficiency, and the cover image can be reconstructed completely.

There are three problems associated with current methods. The embedding method of the scanned document was rarely specified; high capacity and low distortion could not be achieved together, and a completely reconstructed high-capacity image could not be achieved. The method proposed to improve the LSB matching algorithms by embedding the secret data in the optimal position based on the golden ratio. The major improvements of the proposed scheme are outlined below:

1. The secret data included random bits and scanned document, and they were pre-processed by halftone, quadtree, and S-Box transformations, and the size of the scanned document was reduced by 8.11 times.
2. The golden ratio was applied to find the optimal embedding position and design the matching function.
3. This study got a good trade-off between high capacity and low distortion.

This paper presents our solution to the three obstacles and proposes a new LSB matching algorithm based on scanned document pre-processing. Section 2 introduces related work, including data-hiding schemes based on random bit streams and scanned document images. In Section 3, we describe details of the proposed method, including secret data pre-processing, scanned-document hiding, data extraction, and image recovery. Our study investigated three candidates for pre-processing secret data: the halftone, quadtree, and simple substitution. A novel LSB matching algorithm with low distortion and based on the golden ratio is proposed for the embedding step. Pre-processing provides a steganographic image with low distortion and more transformed secret information than current methods offer. Our LSB data hiding method guarantees approximate cover image reconstruction. In Section 4, we report the experimental results and analysis. In Section 5, our conclusions are presented, and future work is proposed.

2. Related Work

The following methods were evaluated for their hiding capacity of two types of information related to this paper. The main ideas, hiding capacity, and image quality are briefly discussed.

In 2017, Soleymani et al. [17] proposed high-capacity image data hiding on a sparse message of a scanned document image. They compressed the scanned document image by halftone technology and converted the binary strings to their equivalent decimal values. Then, they embedded this information into the cover image using 3-LSB. The average payload was 5.43 bpp, and the quality of the steganographic image was 36 dB. However, this method also coded the background area of the binary image. In 2018, Soleymani et al. [18] improved [17] by using a more effective quadtree algorithm to code only the content of the binary image. The average payload was 7.98 bpp, the PSNR (peak signal to noise ratio) was

38.83 dB, and the SSIM (structural similarity index) was 0.93. Generally, for a high-quality visual image, the PSNR was greater than 50 dB.

Unlike [17,18], which tried to improve embedding capacity using the vacated room of the information and the cover image, Ref. [20] hid secret data in a gray image with the mapping method. The binary values of each pixel image and character were divided into four parts. After that, they selected two bits of the secret data, searched for a two-bit similarity in the image pixels, and saved the location of the match. This approach tried to leave the cover image unchanged and send the matches to the receiver secretly. However, when the message capacity was high, the data could not be embedded completely, and it was hard to recover the original information.

In [19], a high-capacity embedding technique and high-quality encoded image were proposed. The secret data were first converted to their equivalent decimal values then into binary strings. They hid the secret data in the edges of four similar gray images using LSB. In this approach, the PSNR of each encoded image was equal to 81.23 dB. However, the receiver needed to obtain the four images simultaneously to extract all the information.

The earliest steganography method for grayscale images was proposed in [21], which offered a simple method for embedding data in cover images. This scheme embedded information by replacing the LSB plane of the gray-level pixel value; it was invisible. The main disadvantages of this scheme were its low capacity and poor security. When the volume of secret data was high, so was the distortion of the cover image. To reduce the distortion of the LSB algorithm, in [22–24], they proposed the optimal LSB method. The optimal LSB algorithm could generate three steganographic pixel values by the remainder operator, in which one of them had the least distortion. The simple LSB method or the optimal LSB method considered one pixel as an embedding unit. The LSB matching revisited scheme [25,26] considered more than one pixel as an embedding unit. In [25], the cover image was divided into non-overlapping pixel pairs, and two bits of secret information were embedded into the first pixel and a binary function. In [26], three pixels of the cover image were considered as the embedding unit. This scheme utilized the first and second most significant bits; then, the remaining six bits were XORed. The secret data were embedded by comparing the result of XOR with three bits of the secret information. The revisited LSB matching scheme minimized the image distortion, but the embedding capacity was limited, and the original image could not be recovered completely.

3. Proposed Method

Current data-hiding methods try to provide high embedding capacity with low distortion. We propose a novel LSB matching algorithm with low distortion that embeds high-capacity data in the cover images. The constructions of this paper are as follows: (1) the scanned document was pre-processed by halftone, quadtree, decimal coding, and S-Box; (2) a novel LSB matching algorithm with the lowest distortion was applied, based on the golden ratio.

3.1. Pre-Processing Step

3.1.1. S-Box

We examined the DES [27] algorithm, a classic encryption algorithm. The S-Box is a non-linear structure and, for any S-Box, the substitution mapping listed in eight S-Boxes is such that, according to the values of rows and columns, its input is mapped to a compressed equivalent decimal value. For any S-Box, assuming $I = i_1 i_2 i_3 i_4 i_5 i_6$, let $k = i_2 i_3 i_4 i_5$ and $h = i_1 i_6$. According to the values k and h, we could look up the Box value in row h and column k: $O = o_1 o_2 o_3 o_4$, a compressed decimal value. It can be seen that the secret data were compressed from 6 to 4. For example, consider $I = 111{,}000$, and let k = 12 and h = 2. In row 2 and column 12 of the S8-Box in Table 1, the number $O = (15)_{10} = (1111)_2$ was found. The size of the secret data was reduced by a factor of 1.5. In this study, to make good use of the working principle of the S-Box, secret information in a bitstream was divided into 6-bit groups, then compressed by a substitution operator.

Table 1. An example for S8-Box.

R. \ Col.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

3.1.2. Halftone and Quadtree

When secret information was scanned in our study, it was converted to embeddable bits by halftone and quadtree techniques. The halftone method was divided into the error-diffusion [28–30] and dither types [31,32]. The halftone image generated by the dither method usually contains an artificial periodic texture; thus, we used the error-diffusion method in our study. By considering the correlation between proximate pixels, the halftone scheme converted each pixel to 0 or 1. Thus, the size of the secret information was reduced by 8 times. The halftone image of a scanned document usually includes signs and white backgrounds, shown with 0 and 1 bits, respectively. People are mainly concerned only with the document content; thus, it is necessary to separate the content from the background with a quadtree algorithm (applicable to any image dimensions). The error-diffusion method consists of three steps:

Step 1: For any scanned document, the integer matrix is converted into a real matrix B by dividing the pixel value by 255.

Step 2: Assume that the threshold t is 1/2 and real matrix B is accessed in raster scan order. If the element of the real matrix is less than t, the halftone pixel value $I(i,j)$ is 0, or 1 otherwise.

Step 3: Here, we defined one value $wc(i,j)$, and $wc = B(i,j) - I(i,j)$. The error of the current pixel is transferred in a ratio of 7:3:5:1 and superimposed on four adjacent pixels. When all the pixels were processed, we obtained the halftone image I.

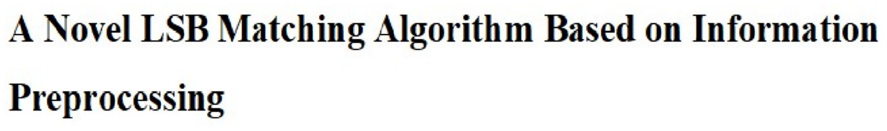
The quadtree method also consists of three steps:

Step 1: The matrix of the halftone image I was divided into four sub-rectangles. If the size of the sub-rectangle was larger than 1×1, the sub-rectangles were divided until the size of all the sub-rectangles was 1×1.

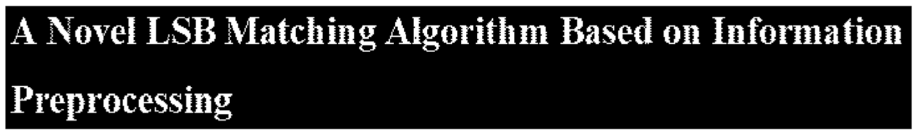
Step 2: Some sub-rectangles did not contain information, so only the content and coordinates of sub-rectangles that contain information were kept.

Step 3: All sub-rectangles that contain content are merged into larger rectangles.

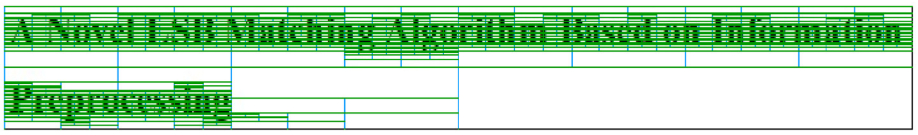
Figure 1 shows the process of scanning a document. As seen in Figure 1c, not all sub-rectangles contained a message. Figure 1d shows that it was necessary to save only the content and coordinates of the sub-rectangles that contained the message. The more sub-rectangles there were, the more content and coordinates needed to be saved. As in Figure 1e, to reduce the number of coordinates, all the sub-rectangles that contained messages were merged by scanning neighbor rectangles horizontally and vertically.


A Novel LSB Matching Algorithm Based on Information Preprocessing

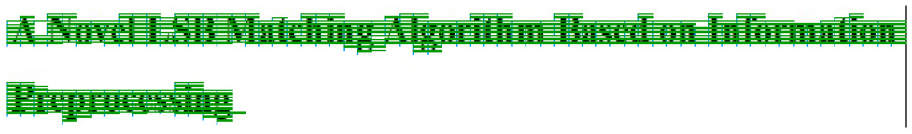
(**a**)


A Novel LSB Matching Algorithm Based on Information Preprocessing

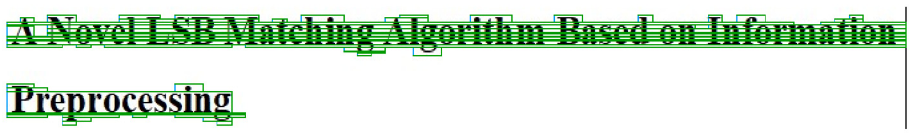
(**b**)


A Novel LSB Matching Algorithm Based on Information Preprocessing

(**c**)


A Novel LSB Matching Algorithm Based on Information Preprocessing

(**d**)


A Novel LSB Matching Algorithm Based on Information Preprocessing

(**e**)

Figure 1. (**a**) Secret document; (**b**) halftone process; (**c**) sub-rectangles; (**d**) content sub-rectangles; (**e**) content sub-rectangles merging.

3.1.3. Decimal Coding

Usually, zeros on the left side of a binary string do not affect the size of the value. In our study, the content and the merged coordinate were processed by decimal coding and S-Box substitution. In the first step, the bit string of the content and the coordinates were converted to decimal values. In the second step, the values were divided into 6-bit groups then compressed into 4-bit groups by S-Box substitution. In Figure 1, the title of the paper was tested, and the size of the original scanned document image was 17.6 KB (18,106 B). After the above steps, the size of the results was reduced to 1953 B. According to the result, we can see the secret data were compressed by 9.27 times.

3.2. Data Embedding

Mielikainen [25] proposed a simple LSB matching algorithm by modifying the pixel ± 1, and two pixels as an embedding unit. The embedding and extraction procedure of Milelikainen's scheme was illustrated as follows:

Set p and q are the cover pixels pair, and c_1 and c_2 are two bits of secret data, respectively. The embedding equation is given in Equation (1). After embedding, the stego image is obtained, and p' and q' are the modified pixels pair. The secret data c_1 can be extracted from the least significant bit of p'. The secret data c_2 can be extracted according to Equation (2).

$$(p',q') = \begin{cases} (p,q), LSB(p) = c_1 \text{ and } \mathrm{LSB}(\lfloor \frac{p}{2} \rfloor + q) = c_2 \\ (p,q+1), LSB(p) = c_1 \text{ and } \mathrm{LSB}(\lfloor \frac{p}{2} \rfloor + q) \neq c_2 \\ (p-1,q), LSB(p) \neq c_1 \text{ and } \mathrm{LSB}(\left\lfloor \frac{p-1}{2} \right\rfloor + q) = c_2 \\ (p+1,q), LSB(p) \neq c_1 \text{ and } \mathrm{LSB}(\left\lfloor \frac{p-1}{2} \right\rfloor + q) \neq c_2 \end{cases} \quad (1)$$

$$c_2 = LSB(\left\lfloor \frac{p}{2} \right\rfloor + q) \quad (2)$$

In this section, the information compressed by halftone, quadtree, decimal coding, and S-Box substitution was embedded into a cover image by a novel revisited LSB matching method. To improve the capacity of data hiding and transmission security, the secret data were compressed then embedded into the cover image by an LSB matching algorithm based on the golden ratio. For the first time, the golden ratio point was used to find the best embedding position and applied as the basic criterion to design the mapping function. First, because the output of the S-Box was 4 bits, the cover image was divided into non-overlapping pixel pairs, and every four pixels were defined as a group. Second, the optimal embedding positions were found according to the golden ratio. Finally, the XOR operation assembled the eight least-significant bits to yield four original bits from the embedding unit. Our new scheme is described below:

① In raster scan order, the cover image was divided into non-overlapping pixel pairs, each pair including four pixels. Assuming the four pixels P_i, P_{i+1}, P_{i+2} and P_{i+3} comprise a hiding unit, the four bits of secret information were $S_1S_2S_3S_4$.

② Each pixel was converted into eight binary bits, and the embedding positions were found according to the calculations $8 \times (1 - 0.618) \approx 3$. Normally, the change of the lowest three significant bits of the pixel value does not affect human vision. To get better visual quality, the optimal embedding position was found according to the calculations $3 \times (1 - 0.618) \approx 1$. The least significant bit can be used to embed information. Assuming $P_i = a_8a_7a_6a_5a_4a_3a_2a_1$, $P_{i+1} = b_8b_7b_6b_5b_4b_3b_2b_1$, $P_{i+2} = c_8c_7c_6c_5c_4c_3c_2c_1$, $P_{i+3} = d_8d_7d_6d_5d_4d_3d_2d_1$, the four bits of secret data are embedded into the exact location. Here, we defined four values A, B, C, and D, and they were obtained according to Equation (3):

$$\begin{cases} A = a_1 \oplus a_2 \oplus b_1 \\ B = b_1 \oplus b_2 \oplus c_1 \\ C = c_1 \oplus c_2 \oplus d_1 \\ D = d_1 \oplus d_2 \oplus a_1 \end{cases} \quad (3)$$

As shown in Equation (1), the values of A and D were controlled by changing a_1 of p_i. Similarly, the values of A and B were controlled by the least significant bit b_1 of p_{i+1}. B and C were controlled by the least significant bit c_1 of p_{i+2}. C and D were controlled by the least significant bit d_1 of p_{i+3}. When the pixel p_{i+1} was an odd number, A was controlled by modifying bit b_1 and b_2 by $P_{i+1} + 1$. When the pixel was an even number, A was controlled by modifying bit b_1 and b_2 by $P_{i+1} - 1$. Similarly, B, C, and D were controlled by modifying the least significant bit and the second least significant bit of p_{i+2}, p_{i+3}, and p_i.

③ We compared four secret data with four values to see whether they were the same. The four pixels did not need to be altered in the data-hiding process if they were equal. Otherwise, we needed to modify the four pixels until they were equal. We describe the scheme in detail as follows:

Step 1: If there was $(s_1 = A)\&\&(s_2 = B)\&\&(s_3 = C)\&\&(s_4 = D)$, the four pixels did not need to be altered in the data-hiding process.

Step 2: If only $(s_1 \neq A)$ or $(s_2 \neq B)$ or $(s_3 \neq C)$ or $(s_4 \neq D)$, and the pixel p_{i+1} was an odd number, we needed to control it with $P_{i+1} + 1$; otherwise, we controlled it with $P_{i+1} - 1$, so that $S_1 = A$. In the same way, if the pixel p_{i+2} was an odd number, we needed to control it with $P_{i+2}+1$; otherwise, we controlled it with $P_{i+2}-1$, so that $S_2 = B$. If the pixel p_{i+3} was an odd number, we needed to control it with $p_{i+3}+1$; otherwise, we controlled it with $p_{i+3}-1$, so that $S_3 = C$. If the pixel p_i was an odd number, we needed to control it with P_i+1; otherwise, we controlled it with P_i-1, so that $S_4 = D$.

Step 3: If only $(s_1 \neq A)\&\&(s_2 \neq B)$ or $(s_1 \neq A)\&\&(s_3 \neq C)$ or $(s_1 \neq A)\&\&(s_4 \neq D)$ or $(s_2 \neq B)\&\&(s_3 \neq C)$ or $(s_2 \neq B)\&\&(s_4 \neq D)$ or $(s_3 \neq C)\&\&(s_4 \neq D)$, if the pixel p_{i+1} was an odd number, we needed to control it with $P_{i+1}-1$, otherwise, we controlled it with $P_{i+1}+1$, so that $(s_1 = A)\&\&(s_2 = B)$. In the same manner, if the pixels p_{i+1} and p_{i+3} were odd numbers, we needed to control them with $P_{i+1}+1$, $P_{i+3}+1$, otherwise, we controlled them with $P_{i+1}-1$, $p_{i+3}-1$, so that $(s_1 = A)\&\&(s_3 = C)$. If the pixel p_i was an odd number, we needed to control it with P_i-1; otherwise, we controlled it with P_i+1, so that $(s_1 = A)\&\&(s_4 = D)$. If the pixel p_{i+2} was an odd number, we needed to control it with $P_{i+2}-1$; otherwise, we controlled it b with y $P_{i+2}+1$, so that $(s_2 = B)\&\&(s_3 = C)$. If the pixel p_i, p_{i+2} were odd numbers, we needed to control them with P_i+1, $P_{i+2}+1$; otherwise, we controlled them with P_i-1, $P_{i+2}-1$, so that $(s_2 = B)\&\&(s_4 = D)$. If the pixel p_{i+3} was an odd number, we needed to control it with $p_{i+3}-1$; otherwise, we controlled it with $p_{i+3}+1$, so that $(s3 = C)\&\&(s4 = D)$.

Step 4: If $(s_1 \neq A)\&\&(s_2 \neq B)\&\&(s_3 \neq C)$ or $(s_1 \neq A)\&\&(s_2 \neq B)\&\&(s_4 \neq D)$ or $(s_1 \neq A)\&\&(s_3 \neq C)\&\&(s_4 \neq D)$ or $(s_2 \neq B)\&\&(s_3 \neq C)\&\&(s_4 \neq D)$. If p_{i+1} was an odd number, we needed to control it with $P_{i+1}+1$; otherwise, we controlled it with $P_{i+1}-1$. If p_{i+2} was an odd number, we needed to control it with $P_{i+2}-1$; otherwise, we controlled it with $P_{i+2}+1$, so that $(s_1 = A)\&\&(s_2 = B)\&\&(s_3 = C)$. In the same manner, we modified the other pixels and obtained $(s_1 = A)\&\&(s_2 = B)\&\&(s_4 = D)$, $(s_1 = A)\&\&(s_3 = C)\&\&(s_4 = D)$, $(s_2 = B)\&\&(s_3 = C)\&\&(s_4 = D)$.

Step 5: If $(s_1 \neq A)\&\&(s_2 \neq B)\&\&(s_3 \neq C)\&\&(s_4 \neq D)$, when p_i was an odd number, we needed to control it with P_i+1; otherwise, we controlled it with P_i-1. If p_{i+1} was an odd number, we needed to control it with $P_{i+1}-1$; otherwise, we controlled it with $P_{i+1}+1$. If p_{i+3} was an odd number, we needed to control it with $P_{i+3}+1$; otherwise, we controlled it with $p_{i+3}-1$. Lastly, we obtained $(s_1 = A)\&\&(s_2 = B)\&\&(s_3 = C)\&\&(s_4 = D)$.

According to the scheme above, four bits of the secret data, s_1, s_2, s_3 and s_4, were ensured to be embedded into the pixel pairs p_i, p_{i+1}, p_{i+2} and p_{i+3} respectively.

For example, as Table 2 shows, s_1, s_2, s_3 and s_4 represent any four bits of secret information. When $p_i = (101)_{10=}(01100101)_2$, $p_{i+1} = (50)_{10=}(00110010)_2$, $p_{i+2} = (213)_{10=}(11010101)_2$, $p_{i+3} = (210)_{10=}(11010010)_2$, we obtained $A = 1$, $B = 0$, $C = 1$, and $D = 0$ according to Equation (1). We adjusted the pixel values by the above rule and let p_i, p_{i+1}, p_{i+2}, p_{i+3} denote the adjusted pixel values.

Table 2. Pixel variation using the LSB algorithm after data hiding. LSB—least significant bit.

Secret Data	q_i	q_{i+1}	q_{i+2}	q_{i+3}	Secret Data	q_i	q_{i+1}	q_{i+2}	q_{i+3}
$(0000)_2$	101 − 1	50	213 − 1	210	$(1000)_2$	101	50	213 − 1	210
$(0001)_2$	101 − 1	50	213 − 1	210 + 1	$(1001)_2$	101	50	213 − 1	210 + 1
$(0010)_2$	101 − 1	50	213	210	$(1010)_2$	101	50	213	210
$(0011)_2$	101 − 1	50	213	210 + 1	$(1011)_2$	101	50	213	210 + 1
$(0100)_2$	101 − 1	50 + 1	213 − 1	210	$(1100)_2$	101	50 + 1	213 − 1	210
$(0101)_2$	101 − 1	50 + 1	213 − 1	210 + 1	$(1101)_2$	101	50 + 1	213 − 1	210 + 1
$(0110)_2$	101 − 1	50 + 1	213	210	$(1110)_2$	101	50 + 1	213	210
$(0111)_2$	101 − 1	50 + 1	213	210 + 1	$(1111)_2$	101	50 + 11	213	210 + 1

As seen from Table 1, the probability of four pixels that needed to be modified was 1/16, the probability of three pixels that need to be modified was 4/16, the probability of two pixels that needed to be modified was 6/16, the probability of one pixel that needed to be modified was 4/16, and the probability of the preserved original pixels was 1/16. The expected value of the changed pixels of the proposed algorithm was:

$$(1/16) \times 4 + (4/16) \times 3 + (6/16) \times 2 + (4/16) \times 1 + (1/16) \times 0 = 29/16$$

The expected number of modifications per pixel was: (29/16) ÷ 4 ≈ 0.453.

As Table 3 shows, one of the most important factors of the proposed LSB matching revisited scheme was that at most, only one pixel at a time can be modified by +1 or −1 when carrying four bits of secret information. Changing four pixels at the same time does not occur. The probability of two pixels needing modification was 7/16, the probability of one pixel needing modification was 8/16, and the probability of the preserved original pixels was 1/16. The expected value of the changed pixels of the proposed algorithm was (7/16) × 2 + (8/16) × 1 + (1/16) × 0 = 22/16. The expected number of modifications per pixel was (22/16) ÷ 4 ≈ 0.344. The secret data were pre-processed: when the secret data were a bit stream, the expected number of modifications per pixel was (22/16) ÷ 6 ≈ 0.229. When the secret data in the document were scanned, the expected number of modifications per pixel was (22/16) ÷ 32 ≈ 0.0430. This result demonstrates that the proposed approach effectively prevents pixel distortion after data hiding.

Table 3. Pixel variation using the proposed LSB matching revisited after data hiding. LSB—least significant bit.

Secret Data	q_i	q_{i+1}	q_{i+2}	q_{i+3}	**Secret Data**	q_i	q_{i+1}	q_{i+2}	q_{i+3}
$(0000)_2$	101	50 − 1	213	210 − 1	$(1000)_2$	101	50	213	210 − 1
$(0001)_2$	101	50 − 1	213	210 + 1	$(1001)_2$	101	50	213	210 + 1
$(0010)_2$	101	50 − 1	213	210	$(1010)_2$	101	50	213	210
$(0011)_2$	101 − 1	50	213	210	$(1011)_2$	101 + 1	50	213	210
$(0100)_2$	101	50	213 − 1	210−1	$(1100)_2$	101	50	213 − 1	210
$(0101)_2$	101 − 1	50	213 − 1	210	$(1101)_2$	101 + 1	50	213 − 1	210
$(0110)_2$	101	50 + 1	213	210	$(1110)_2$	101	50	213 + 1	210
$(0111)_2$	101	50 + 1	213 + 1	210	$(1111)_2$	101 + 1	50	213 + 1	210

Figure 2 shows the comparison of the probability of modifying pixels of the three methods. Mielikainen [25] proposed an LSB matching revisited scheme and groups two pixels as an embedding unit. For every four bits of data embedded, the pixel modification probability of the LSB method and Mielikainen's scheme. However, the LSB matching scheme has low computation complexity. It can be seen that our proposed method modified at most two pixels every four pixels, and the magnitude of the modification was 1. The LSB scheme and Mielikainen's approach modified more pixel values. Our study set every four pixels as a unit, and the computational complexity was lower.

3.3. Extraction

During extraction, the receiver can acquire secret data without any knowledge of the cover image. There are two steps:

(1) Reading the steganographic image: the steganographic image was divided in raster scan order into non-overlapping pixel pairs, and each pair included four pixels.

(2) Extracting the secret data: The four bits of embedded information can be extracted using Equation (1) without knowing the original image information. If the secret data were in a scanned image, the coordinates and S-Box were used to recover the secret information according to content.

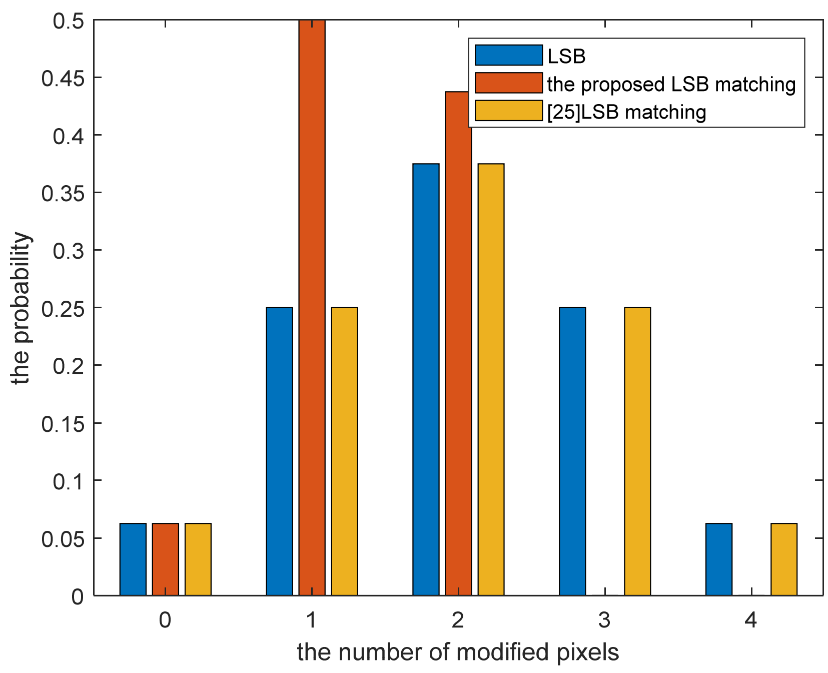


Figure 2. Relationship between the number of modified pixels and the probability.

4. Experimental Results and Comparisons

This section presents the results obtained from our study of the proposed LSB matching algorithm, using 20 standard images from the USC-SIPI image database. PSNR and SSIM were used to evaluate the image; Section 1 gives a detailed example. Our aim was to discover a general method to improve the hiding capacity of images, and we found an effective trade-off between high capacity and low distortion. In part (2), we compare the efficiency of our scheme with other schemes and discuss its implications.

4.1. A Detailed Example

Eight scanned documents of [26,33] were used as secret data. Table 4 lists eight pages of scanned documents. Figure 3 shows the relation between segment size and compression ratio. It can be seen that, for the same scanned document, the segment sizes were 1×1, 4×4, 8×8, 16×16, and 32×32, and the compression ratios for five different thresholds were 8.110523, 4.573963, 2.051653, 2.051653, and 1.665751, respectively. Figure 4 lists the relation between minimum rectangular size and the mean embedding capacity for the Lena image. The mean embedding capacities for five different thresholds were 1.66186 bpp, 2.940742 bpp, 4.494556 bpp, 6.567377 bpp, and 8.084202 bpp. Figures 3 and 4 show that when the segment size was 1×1, the compression ratio was the best, and the volume of secret data transmitted was the highest. In Figure 5, the PSNR values of the steganographic image for five different thresholds were 44.35711 dB, 44.16443 dB, 44.16444 dB, 44.16445 dB, and 44.16446 dB. When the segment size was 1×1, we determined that the visual artifacts were the best.

Table 4. Eight scanned documents and their sizes.

Page Number	Size (B)
1	308,002
2	507,020
3	524,171
4	467,128
5	419,473
6	436,539
7	492,878
8	378,691

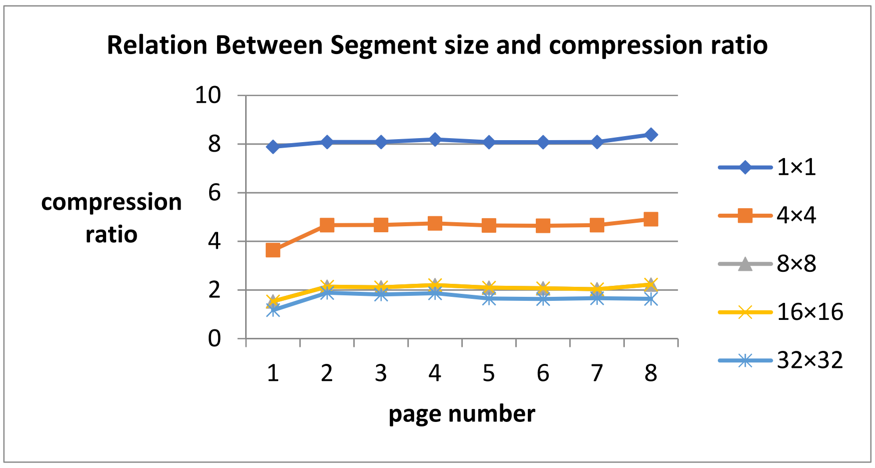


Figure 3. Relationship between segment size and compression ratio.

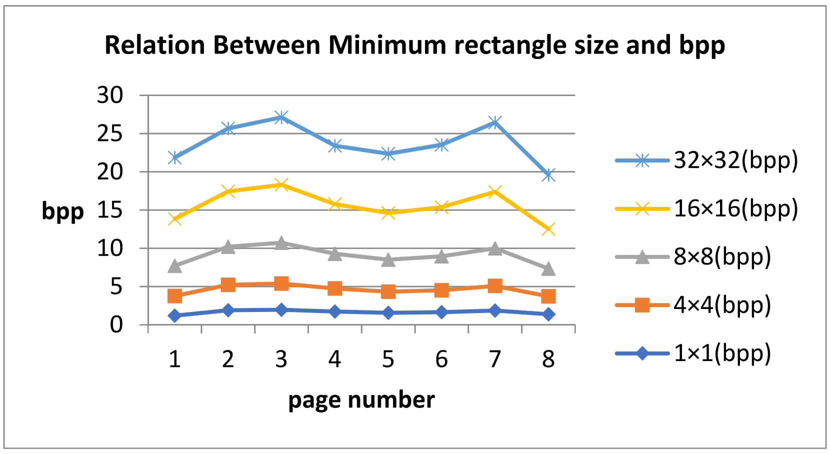


Figure 4. Relationship between segment size and bpp.

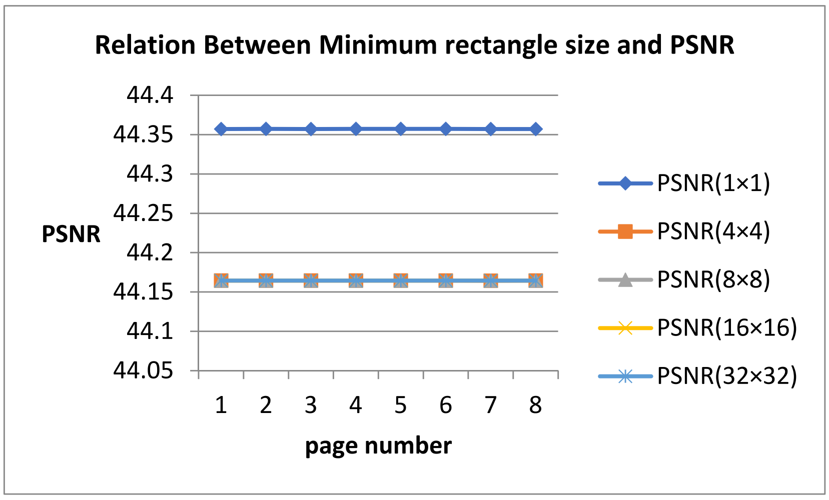
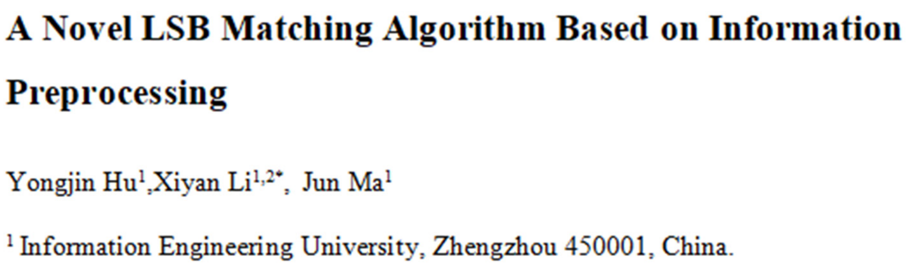


Figure 5. Relationship between segment size and PSNR. PSNR—peak signal to noise ratio.

Using the other algorithms, the 32 KB scanned document needed 262,144 bits of secret information to be embedded. In our study, we had to embed only 32,324 bits. Usually, for a small rectangle, the smaller the divided area was, the greater the time cost. It was confirmed that the larger the segmentation area, the rougher it is, and the lower the cost time. Figure 6 shows two scanned documents of 3 KB and 35 KB. Table 5 lists the actual embedding amounts and the times from embedding to complete extraction for two differently sized scanned documents. The smaller the segmentation size was, the more accurate and the smaller the time cost. The fastest processing time for the 3 KB scanned document was 1 s, and the slowest processing time was 3 s. For the 35 KB scanned document, the fastest processing time was 6 s, and the slowest processing time was 48 s. The larger the document, the longer the processing time, especially with the 32 × 32 block size, which exceeded the user's time limit.

Abstract

(**a**)


A Novel LSB Matching Algorithm Based on Information Preprocessing

Yongjin Hu[1],Xiyan Li[1,2*], Jun Ma[1]

[1] Information Engineering University, Zhengzhou 450001, China.

(**b**)

Figure 6. Scanned documents. (**a**) Scanned document in 3 KB; (**b**) Scanned document in 35 KB.

Table 5. The relationship between block size, embedding amount, and time.

Scanned Document	Norm	1 × 1	4 × 4	8 × 8	16 × 16	32 × 32
(a)	Embedding amount (bits)	1460	1972	2588	5148	5148
	time (s)	3	1	2	2	2
(b)	Embedding amount (bits)	24,828	42,068	82,036	82,036	115,780
	time (s)	6	10	27	27	48

Table 6 compares the actual embedding amount and PSNR. When the segmentation size is 1 × 1, the actual embedding amount of the two documents is the smallest, and the image quality is also the best. The values of PSNR were 69.547 and 57.4617. The distortion rate of the image increases as the amount of embedding increases. For the same document, the larger the segmentation area, the more redundant the messages and, therefore, the greater the distortion rate of the steganographic image. For the five segmentation sizes of 35 KB documents, background redundancy was eliminated to varying degrees, but the values of PSNR were all above 50 dB, which shows that the information pre-processing and matching mapping function of this algorithm is sophisticated and practical. Figure 7 shows the images and their histogram, where (a) is the cover image and its histogram, (b) is embedded in document (a) and its histogram, and (c) is embedded in the document (b) and its histogram. Visually, it is impossible to distinguish the difference between the images. The proposed algorithm has good visibility, and the PSNR values are all greater than 57 dB. Because the distortion rate is relatively low, it is not easy to attract the attention of a third party when transmitting on an open channel.

Table 6. The relationship between block size, embedding amount, and PSNR.

Scanned Document	Norm	1 × 1	4 × 4	8 × 8	16 × 16	32 × 32
(a)	Embedding amount (bits)	1460	1972	2588	5148	5148
	PSNR (dB)	69.547	68.2919	67.1127	64.1924	64.1924
(b)	Embedding amount (bits)	24,828	42,068	82,036	82,036	115,780
	PSNR (dB)	62.7946	55.168	52.254	52.254	50.7378

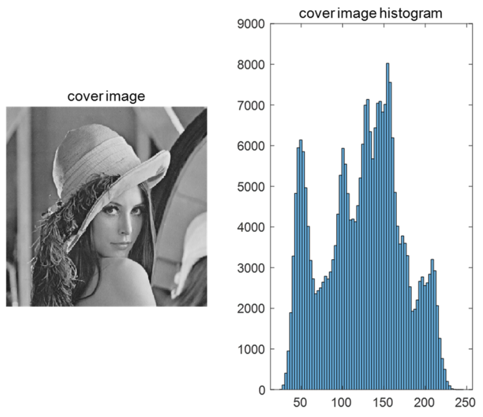
Taking Figure 6 as the secret document, we evaluated our approach against attacks like cropping, rotate, Gaussian noise, pepper, and salt noise. The results of the experiment are in Table 7, which is under extraction accuracy as well.

Table 7. PSNR and accuracy under attacks.

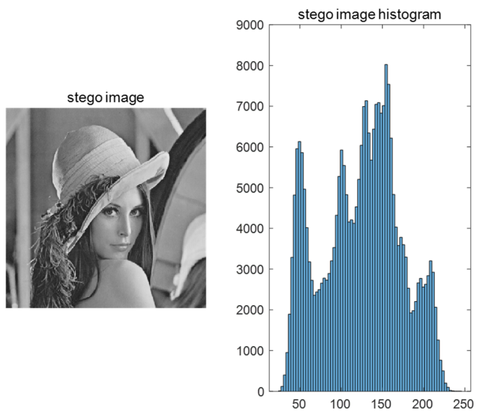
Attack	PSNR	Accuracy
No attack	62.7946	100%
Cropping (1:128,1:128)	17.5888	93.6695%
Rotate (3°)	16.3833	93.7371%
Salt&pepper (0.01)	25.3582	98.9409%
Salt&pepper (0.03)	20.6751	97.1126%
Gaussian noise (0.01)	20.0709	93.0112%
Gaussian noise (0.03)	15.5579	91.4116%

4.2. Comparisons with Related Studies and Discussion

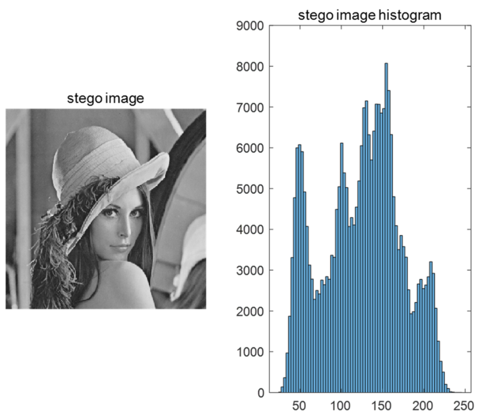
We compared our study to nine state-of-the-art schemes for hiding capacity and image distortion. Table 8 shows the PNSR comparison results for the same scanned document (262,144 bits), and the visual metric PSNR of the LSB scheme [22–24] was 51.154 dB. However, the revisited LSB matching method [25,26] can raise the PSNR to 1.247 dB and 1.763 dB, separately. Lu [36] proposed a dual image based on reversible data-hiding algorithm by improving the LSB matching scheme of [25]. Because there were two stego images, the embedding capacity was 524288 bits, and the average of PSNR was 49.24 dB. Sahu [37] improved Lu's scheme by using a dual-layer LSB matching algorithm. The secret data were embedded into four stego images, and the PSNR and embedding capacity were 46.51 dB and 1572864 bits separately. Our study was also a revisited LSB matching method, but we can embed bit stream and scanned document images into the cover image with an average PSNR of 53.025 dB and 65.55372 dB. It can be seen that Lu and Sahu's schemes with higher embedding capacity and low distortion. However, our study can embed two forms of secret data. We believe that our study demonstrates a significant improvement. Tables 9 and 10 compare our study with similar work. In the comparisons, the same cover images were processed [18] with the quadtree and LSB algorithm, which significantly improved the embedding amount and image quality over [34]. The average PSNR of our study was 44.44 dB, and the value of SSIM was closer to 1. Table 11 summarizes the proposed scheme's average quality and data hiding capacity for comparison with [17,18,34–37]. In our study, information was pre-processed, and the matching function makes the distortion rate small. This gives us information hiding with high embedding capacity and a low distortion rate.



(**a**)



(**b**)



(**c**)

Figure 7. Images and their histograms. (**a**) cover image and histogram; (**b**) stego image and histogram of Figure 6a; (**c**) stego image and histogram of Figure 6b.

Table 8. Comparison between the method proposed and [22–26,36,37].

Cover Image	[22–24]	[25]	[26]	[36]	[37]	Proposed Method	
						Bit Stream	Scanned Document
Lena	51.156	52.404	52.916	49.25	46.50	53.021	65.5538
Airplane	51.143	52.4	52.925	49.22	46.49	53.043	65.5541
Baboon	51.161	52.405	52.917	49.26	46.51	53.022	65.5538
Elaine	51.17	52.407	52.914	49.27	46.52	53.014	65.5536
Man	51.138	52.39	52.911	49.22	46.48	53.025	65.5533
Average	51.154	52.401	52.917	49.24	46.51	53.025	65.55372

Table 9. Comparison between the method proposed and [18,34].

Image	Proposed Method		[18]		[34]	
	PSNR	bpp	PSNR	bpp	PSNR	bpp
Pepper	44.29	15.90	37.48	9.41	34.93	3.95
Lena	44.36	10.73	37.71	6.35	N/A	N/A
Aerial	44.31	11.77	37.66	6.97	N/A	N/A
Jetplane	44.78	12.69	38.14	7.51	34.67	3.95
Average	44.44 (4)	12.77	37.74	7.56	34.8	3.95

Table 10. Comparison between the method proposed and [17,18,35].

Cover Image	Proposed Method		[18]		[17]		[35]	
	PSNR	bpp	PSNR	bpp	PSNR	bpp	PSNR	bpp
Blonde	44.35	15.98	37.48	9.46	35.23	5.65	37.31	3.04
Pepper	44.29	15.90	37.48	9.41	36.19	5.65	37.27	3.05
Jetplane	44.78	12.69	38.14	7.51	36.84	4.77	33.85	3.91
Boat	44.30	16.25	37.46	9.62	37.00	5.70	33.57	3.91
Average	44.43	15.20	37.64	9.00	36.31	5.43	35.50	3.47

Table 11. Comparison between proposed method and [17,18,34–37].

Method	PSNR	Embedding Rate (bpp)
[17]	36.31	5.43
[18]	37.83	7.98
[34]	34.8	3.95
[35]	35.50	3.47
[36]	49.24	4
[37]	46.51	6
Proposed method	44.35	13.48

5. Discussion

In our study, we proposed a novel LSB matching algorithm based on information pre-processing. In the experiments, we proved that our scheme with high capacity and low distortion. To the best of our knowledge, it is the first information pre-processing for a novel LSB matching algorithm. Furthermore, we want to discuss two issues:

(1) Application: In our opinion, our study is most suitable for a digital office because with the digital development of life and office, the secure transfer of documents and mail through the internet became necessary.
(2) Future work: Because the authors did not evaluate our study against the most common attacks, it is just a data-hiding scheme for pre-processed secret data. In the future, we plan to strengthen the study of robustness.

6. Conclusions

We present in this paper a novel, efficient LSB matching algorithm. Experiments showed that it had the lowest distortion, outperforming other related schemes. Before embedding secret data, the information was pre-processed by halftone, quadtree, decimal coding, and substitution treatment, and the size was reduced by at least a factor of eight. In the data hiding step, the cover image was divided into 1 × 1 sub-blocks. The compressed information was inserted into pixels by a new revisited LSB matching scheme based on the golden ratio. The receiver can extract the information without any knowledge. Therefore, our method has general applicability and provides the best trade-off between capacity and PSNR.

In our study, we saved the additional information and sent it to the receiver secretly. In future work, we plan to improve the speed of pre-processing and reconstruct the cover image completely. Therefore, it is suggested that a more efficient scheme for text documents should be developed.

Author Contributions: Conceptualization, Y.H. and X.L.; methodology, X.L. and J.M.; validation, X.L. and Y.H.; writing—review and editing, X.L., Y.H., and J.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work were supported by the Foundation of Science and Technology on Information Assurance Laboratory (No. KJ-15-108) and Hainan Provincial Reform in Education Project of China (No. Hnjg2020-31).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The corresponding author can provide the data sets utilized in this work upon reasonable request.

Acknowledgments: The authors thank the review of Yang for the deep and constructive comments on the first version of the paper.

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

References

1. Liu, X.M.; Choo, K.K.R.; Deng, R.H.; Lu, R.X.; Weng, J. Efficient and Privacy-Preserving Outsourced Calculation of Rational Numbers. *IEEE Trans. Dependable Secur. Comput.* **2018**, *15*, 27–39. [CrossRef]
2. Xiong, J.B.; Ma, R.; Chen, L.; Tian, Y.L.; Li, Q.; Liu, X.M.; Yao, Z.Q. A personalized privacy protection framework for mobile crowdsensing in IIoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 4231–4241. [CrossRef]
3. Chen, Z.; Tian, Y.; Peng, C. An incentive-compatible rational secret sharing scheme using blockchain and smart contract. *Sci. China Inf. Sci.* **2021**, *64*, 202301. [CrossRef]
4. Liu, X.M.; Deng, R.H.; Choo, K.K.R.; Yang, Y. Privacy-Preserving Outsourced Support Vector Machine Design for Secure Drug Discovery. *IEEE Trans. Cloud Comput.* **2020**, *8*, 610–622. [CrossRef]
5. Fridrich, J.; Goljan, M.; Du, R. Invertible authentication. In Proceedings of the Security and Watermarking of Multimedia Contents III, San Jose, CA, USA, 20 January 2001. [CrossRef]
6. Fridrich, J.; Goljan, M.; Du, R. Lossless data embedding for all image formats. In Proceedings of the Security and Watermarking of Multimedia Contents IV, San Jose, CA, USA, 29 April 2002. [CrossRef]
7. Celik, M.U.; Sharma, G.; Tekalp, A.M.; Saber, E. Lossless generalized-LSB data embedding. *IEEE Trans. Image Process.* **2005**, *14*, 253–266. [CrossRef] [PubMed]
8. Jun, T. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 890–896. [CrossRef]
9. Alattar, A.M. Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Trans. Image Process.* **2004**, *13*, 1147–1156. [CrossRef]
10. Thodi, D.M.; Rodriguez, J.J. Expansion embedding techniques for reversible watermarking. *IEEE Trans. Image Process.* **2007**, *16*, 721–730. [CrossRef]
11. Li, X.L.; Li, J.; Li, B.; Yang, B. High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion. *Signal Process.* **2013**, *93*, 198–205. [CrossRef]
12. Ou, B.; Li, X.L.; Zhao, Y.; Ni, R.R. Reversible data hiding using invariant pixel-value-ordering and prediction-error expansion. *Signal Process. Image Commun.* **2014**, *29*, 760–772. [CrossRef]
13. Qu, X.C.; Kim, H.J. Pixel-based pixel value ordering predictor for high-fidelity reversible data hiding. *Signal Process.* **2015**, *111*, 249–260. [CrossRef]
14. Ni, Z.C.; Shi, Y.Q.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362. [CrossRef]
15. Lin, C.C.; Hsueh, N.L. A lossless data hiding scheme based on three-pixel block differences. *Pattern Recognit.* **2008**, *41*, 1415–1425. [CrossRef]

16. Tsai, P.; Hu, Y.C.; Yeh, H.L. Reversible image hiding scheme using predictive coding and histogram shifting. *Signal Process.* **2009**, *89*, 1129–1143. [CrossRef]
17. Soleymani, S.H.; Taherinia, A.H. High capacity image steganography on sparse message of scanned document image (SMSDI). *Multimed. Tools Appl.* **2017**, *76*, 20847–20867. [CrossRef]
18. Soleymani, S.H.; Taherinia, A.H. High capacity image data hiding of scanned text documents using improved quadtree. *arXiv* **2018**, arXiv:1803.11286.
19. Basheer, N.M.; Aaref, A.M.; Ayyed, D.J. Proposed method of text hiding in image edges. *Int. J. Comput. Appl.* **2015**, *126*, 33–37. [CrossRef]
20. Hussein, H.L.; Abbass, A.A.; Naji, S.A.; Al-Augby, S.; Lafta, J.H. Hiding text in gray image using mapping technique. *J. Phys. Conf. Ser.* **2018**, *1003*, 012032. [CrossRef]
21. Cox, I.J.; Kilian, J.; Leighton, T.; Shamoon, T. A secure, robust watermark for multimedia. In *Information Hiding*; Springer: Berlin/Heidelberg, Germany, 1996.
22. Chan, C.K.; Cheng, L.M. Hiding data in images by simple LSB substitution. *Pattern Recognit.* **2004**, *37*, 469–474. [CrossRef]
23. Thien, C.C.; Lin, J.C. A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function. *Pattern Recognit.* **2003**, *36*, 2875–2881. [CrossRef]
24. Wang, S.J. Steganography of capacity required using modulo operator for embedding secret image. *Appl. Math Comput.* **2005**, *164*, 99–116. [CrossRef]
25. Mielikainen, J. LSB matching revisited. *IEEE Signal Process. Lett.* **2006**, *13*, 285–287. [CrossRef]
26. Wu, N.I.; Hwang, M.S. A novel LSB data hiding scheme with the lowest distortion. *Imaging Sci. J.* **2017**, *65*, 371–378. [CrossRef]
27. Chen, L.S.; Shen, S.Y. *Modern Cryptography*, 2nd ed.; Science Press: Beijing, China, 2008.
28. Liu, L.Y.; Chen, W.; Zheng, W.T.; Geng, W.D. Structure-aware error-diffusion approach using entropy-constrained threshold modulation. *Vis. Comput.* **2014**, *30*, 1145–1156. [CrossRef]
29. Li, X. Edge-directed error diffusion halftoning. *IEEE Signal Process. Lett.* **2006**, *13*, 688–690. [CrossRef]
30. Singh, Y.K. Generalized error diffusion method for halftoning. In Proceedings of the IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Tamil Nadu, India, 5–7 March 2015. [CrossRef]
31. Zhou, Z.; Arce, G.R.; Crescenzo, G.D. Halftone visual cryptography. *IEEE Trans. Image Process.* **2006**, *15*, 2441–2453. [CrossRef]
32. Alasseur, C.; Constantinides, A.G.; Husson, L. Colour quantisation through dithering techniques. In Proceedings of the International Conference on Image Processing (Cat. No.03CH37429), Barcelona, Spain, 14–17 September 2003. [CrossRef]
33. Li, X.Y.; Zhou, X.B.; Zhou, Q.L.; Han, S.J.; Liu, Z. High-capacity reversible data hiding in encrypted images by information preprocessing. *Complexity* **2020**, *2020*, 6989452. [CrossRef]
34. Jana, B. High payload reversible data hiding scheme using weighted matrix. *Optik* **2016**, *127*, 3347–3358. [CrossRef]
35. Bai, J.L.; Chang, C.C.; Nguyen, T.S.; Zhu, C.; Liu, Y.J. A high payload steganographic algorithm based on edge detection. *Displays* **2017**, *46*, 42–51. [CrossRef]
36. Lu, T.C.; Tseng, C.Y.; Wu, J.H. Dual imaging-based reversible hiding technique using LSB matching. *Signal Process.* **2015**, *108*, 77–89. [CrossRef]
37. Aditya, K.S.; Gandharba, S. Reversible Image Steganography Using Dual-Layer LSB Matching. *Sens. Imaging* **2020**, *21*, 1. [CrossRef]