future internet

MDPI

Article

A High-Capacity Reversible Data Hiding Scheme for Encrypted Hyperspectral Images Using Multi-Layer MSB Block Labeling and ERLE Compression

Yijie Lin [1], Chia-Chen Lin [2,*], Zhe-Min Yeh [1], Ching-Chun Chang [3] and Chin-Chen Chang [1,*]

[1] Department of Information Engineering and Computer Science, Feng Chia University, Taichung 407, Taiwan; p1263670@o365.fcu.edu.tw (Y.L.); m1271774@o365.fcu.edu.tw (Z.-M.Y.)
[2] Department of Computer Science and Information Engineering, National of Chin-Yi University of Technology, Taichung 411, Taiwan
[3] Information and Communication Security Research Center, Feng Chia University, Taichung 407, Taiwan; ccc@fcu.edu.tw
* Correspondence: ally.cclin@ncut.edu.tw (C.-C.L.); ccc@o365.fcu.edu.tw (C.-C.C.)

Abstract

In the context of secure and efficient data transmission over the future Internet, particularly for remote sensing and geospatial applications, reversible data hiding (RDH) in encrypted hyperspectral images (HSIs) has emerged as a critical technology. This paper proposes a novel RDH scheme specifically designed for encrypted HSIs, offering enhanced embedding capacity without compromising data security or reversibility. The approach introduces a multi-layer block labeling mechanism that leverages the similarity of most significant bits (MSBs) to accurately locate embeddable regions. To minimize auxiliary information overhead, we incorporate an Extended Run-Length Encoding (ERLE) algorithm for effective label map compression. The proposed method achieves embedding rates of up to 3.79 bits per pixel per band (bpppb), while ensuring high-fidelity reconstruction, as validated by strong PSNR metrics. Comprehensive security evaluations using NPCR, UACI, and entropy confirm the robustness of the encryption. Extensive experiments across six standard hyperspectral datasets demonstrate the superiority of our method over existing RDH techniques in terms of capacity, embedding rate, and reconstruction quality. These results underline the method's potential for secure data embedding in next-generation Internet-based geospatial and remote sensing systems.

Keywords: reversible data hiding; encrypted hyperspectral images; MSB prediction; Extended Run-Length Encoding; high-capacity embedding; image recovery

Academic Editor: Gianluigi Ferrari

Received: 24 July 2025
Revised: 14 August 2025
Accepted: 19 August 2025
Published: 21 August 2025

Citation: Lin, Y.; Lin, C.-C.; Yeh, Z.-M.; Chang, C.-C.; Chang, C.-C. A High-Capacity Reversible Data Hiding Scheme for Encrypted Hyperspectral Images Using Multi-Layer MSB Block Labeling and ERLE Compression. *Future Internet* **2025**, *17*, 378. https://doi.org/10.3390/fi17080378

1. Introduction

In the digital era, safeguarding information has emerged as a critical challenge and a key area of focus in both industry applications and academic research. One notable technique gaining significant interest is data embedding, a subset of information security techniques that involves concealing confidential data into various types of media. This approach serves multiple purposes, such as verifying ownership, authenticating images, and enabling covert communication [1].

Conventional data hiding techniques typically introduce irreversible changes to the original content, preventing full recovery of the unaltered media after the embedded information is extracted [2]. This limitation poses a significant challenge in sensitive domains

such as defense, aerospace, remote sensing, and geospatial analysis, where the preservation of original data fidelity is critical, and any form of distortion is generally unacceptable. To address this, various reversible data hiding (RDH) methods in the spatial domain have been extensively investigated, including difference expansion (DE) [3], histogram shifting (HS) [4], and prediction-error expansion (PEE) [5]. In 2003, Tian introduced the DE method, which enables data embedding by expanding the intensity difference between neighboring pixels and inserting secret data into the least significant bit (LSB) of one of the pixels. Over time, this technique has been enhanced through several follow-up studies [6–8]. Subsequently, in 2006, Ni et al. [4] proposed the HS method, which embeds data into the peak regions of the grayscale histogram to minimize distortion. However, as its embedding capacity is highly dependent on the statistical features of the host image, multiple refined versions have been developed to overcome this constraint [9–12]. Later, Thodi and Rodríguez introduced the PEE approach, which combines prediction-error expansion with histogram shifting to improve embedding performance [5]. Their experiments demonstrated that PEE achieves higher capacity and lower distortion by effectively exploiting local pixel correlations. Since then, a wide range of PEE-based methods have been proposed [13–17], including a notable variant called pixel-value ordering (PVO) [15], which organizes pixel values within local blocks to facilitate more efficient and less intrusive data embedding. In addition, several deep learning-based schemes have been proposed over the past three years [18,19].

Beyond conventional natural images or light-field images [20], RDH has also been actively researched in contexts such as encrypted image domain [21–25], where it plays a key role in secure cloud computing and privacy-aware systems [26]. Furthermore, RDH techniques have been investigated for various specialized image types, such as halftone images [27], high dynamic range (HDR) images [28], and hyperspectral images (HSIs) [29–32]. However, in comparison to these other image types, significantly fewer RDH methods have been developed specifically for HSIs.

Hyperspectral imagery (HSI) is designed to acquire detailed spectral information across a wide range of contiguous bands, allowing for the precise identification and analysis of materials based on their distinct spectral signatures [33]. The comprehensive nature of this data makes HSI highly valuable in diverse domains, including environmental monitoring, precision agriculture, mineral mapping, and defense applications [34]. For example, in agriculture, HSI supports early detection of crop distress or disease by capturing subtle variations in plant reflectance patterns [35]. Due to the sensitive and potentially strategic nature of the information extracted from hyperspectral data—such as the identification of resource-rich regions or surveillance targets—it is vital to maintain both its confidentiality and integrity. Unauthorized access or manipulation of such data can pose risks to national security, breach intellectual property, or lead to flawed scientific interpretations [36]. Therefore, safeguarding the authenticity of HSI is critical for ensuring reliable, data-driven decisions. To meet these requirements, reversible data hiding (RDH), also referred to as reversible watermarking, has emerged as a robust solution. RDH enables the accurate extraction of embedded information while fully restoring the original imagery without any loss, ensuring both security and data fidelity.

Hyperspectral imagery (HSI) collects detailed spectral data across a continuous range of narrow bands, enabling precise identification of materials through their distinct spectral signatures [33]. This capability supports a broad range of applications, including environmental monitoring, agriculture, mineral exploration, geology, atmospheric research, and defense [36–38]. Due to the sensitive and potentially strategic nature of the information extracted from HSIs—such as locating resource-rich regions or surveillance targets—ensuring the confidentiality and integrity of the data is critical. Unauthorized access or manipulation

may result in national security risks, intellectual property breaches, or flawed scientific outcomes [36]. To ensure reliable and secure use of hyperspectral data, reversible data hiding (RDH), or reversible watermarking, has been developed as a viable solution. RDH enables the embedding of additional information while allowing the exact restoration of the original image, thus preserving both data fidelity and security.

In 2019, Carpentieri et al. designed the first RDH method for HSIs [29]. In their work, they introduced a one-pass framework that utilizes predictive techniques to simultaneously perform lossless data embedding and compression on the modified image stream. With just a single processing pass, their proposed method generates a stego image that is both compressed and embedded with hidden data. At the receiver's side, the original HSI can be precisely reconstructed through decompression and reversible recovery. Three years later, Fang et al. introduced a pixel type classification based RDH for HSIs [30]. With their proposed method, they utilized inter-band pixel intensity correlations to classify each pixel into one of five distinct types, based on the interval into which its value falls—determined by the four nearest neighboring pixels. For each of these five categories, a corresponding adaptive prediction strategy is employed to optimize prediction accuracy. By employing five tailored prediction strategies in the spatial domain, their proposed method achieves an embedding rate of up to 0.04 bpppb while preserving image quality with a peak signal-to-noise ratio (PSNR) close to 98 dB. In 2024, Zhang et al. [31] expanded the scope of RDH for HSIs by introducing its application to the compression domain. They proposed an innovative embedding framework that leverages predictive modeling to conceal secret information directly within bitmap structures, while preserving the standard format of Absolute Moment Block Truncation Coding (AMBTC) and enabling exact reconstruction of the original image. To reduce embedding distortion, a specialized embedding pattern was developed, in which secret data are embedded through predictive modification of bitmaps. Based on this designed embedding pattern, an adaptive embedding sequence was subsequently designed by exploiting the statistical properties of AMBTC codes to further improve performance. According to their findings, the proposed method achieves a data hiding capacity of approximately 4.19×10^6 bits, corresponding to an embedding rate of about 0.4624 bpppb, while maintaining a PSNR of around 60 dB.

Although many RDH schemes for HSIs have focused on either the spatial domain or the compression domain, for example, the method proposed by Zhang et al. [31] achieves an embedding rate of up to 0.4624 bpppb, but the hiding capacity still needs to be prompted to meet the broader demands of real-world HSI usage. Take the case of wildfire assessment: although HSIs can accurately detect and measure burn severity, their usefulness depends heavily on additional information such as the date of the fire, the type of vegetation present beforehand, and the weather conditions during image capture. Without this context, the spectral data alone may lead to misinterpretation. Similarly, important processing parameters—like the classification models used or threshold values applied—are not embedded in the HSI file and may be lost or overlooked, resulting in errors in data analysis at the receiving end. To solve these issues, it is necessary to expand the embedding capacity of HSIs so that auxiliary metadata can be included directly within the image before transmission. This enables the receiving end to perform more accurate, reliable, and meaningful analysis by having access to both the image data and the necessary contextual cues. However, increasing embedding capacity alone is not enough. Without proper protection mechanisms, the HSI or the embedded metadata could be altered by unauthorized users, undermining both data integrity and analysis outcomes.

Currently, studies on RDH schemes tailored specifically for encrypted HSIs remains limited. However, developing such techniques is essential to guarantee the secure and interpretable transmission of HSI data in practical applications. To meet this need, this

paper proposes a high-capacity RDH method designed for encrypted HSIs. The superiority of the proposed method is demonstrated by comparing with the hyperspectral image RDH methods [31,32]. The contributions of our proposed method are as follows:

- High embedding rate: The scheme utilizes a multi-layer block labeling mechanism based on the similarity of most significant bits (MSBs), which allows it to efficiently locate smooth regions for embedding. This leads to a high embedding capacity—up to 3.76 bits per pixel per band (bpppb)—and outperforms previous methods like those by Zhang et al. [31] and Yeh et al. [32] in both embedding capacity and rate.
- Lossless image recovery with high visual quality: Using the reference pixel MSBs and reserved space strategies, the original hyperspectral image can be fully recovered with minimal visual distortion, as confirmed by the high PSNR and visual comparison results.
- Strong encryption security: The block-based encryption approach achieves excellent encryption performance, with NPCR values above 99% and UACI values around 48%, indicating high resistance to differential attacks and effective information concealment.

The rest of this paper is organized as follows. The well-known Extended Run-Length Encoding [39] is introduced in Section 2. Section 3 describes the proposed method. The experimental results and the comparisons to other methods are given in Section 4. Finally, Section 5 summarizes this paper.

2. Related Work

Run-Length Encoding (RLE) [40] is a technique for compressing consecutive symbols. Its variant called Extended Run-Length Encoding (ERLE) will be used in our proposed scheme to encrypt the layer label, which will be described in detail in Section 3.2. To offer sufficient background knowledge, ERLE is introduced in this section. ERLE was proposed by Chen et al. in 2019 [39] for application in RDHEI. Specifically, both fixed-length and variable-length codewords are introduced. If the length of the run of consecutive symbols is less than four, the fixed-length codeword is used; otherwise, the variable-length codeword is applied.

The binary format of the ERLE compression codes consists of a sequence of variable-length bit patterns. Two types of codewords are defined: fixed-length codewords for short runs and variable-length codewords for longer runs. In a fixed-length codeword, an indicator bit specifies the code type, followed directly by a fixed number of bits representing the run symbols. In a variable-length codeword, the indicator consists of a prefix of consecutive 1s ending with a 0, which determines the number of subsequent bits encoding the run-length difference. The run symbol bit is appended at the end. During decoding, the bitstream is read sequentially: the indicator bits are interpreted first to determine the codeword type and length; then, the following bits are used to reconstruct the original run of symbols.

For example, suppose we have the sequence 11010000000000. First, we read a run of the symbol 1 and find that the length is two, which is less than four, so the fixed-length codeword is used. The indicator bit is 0, and the next four bits, 1101, are directly read, so this part is encoded as 01101. Next, we continue to read a run of the symbol 0 and find that the length is 10, which is greater than 4, so the variable-length codeword is used. First, the prefix length is $\lfloor \log_2 10 \rfloor = 3$. It starts with two 1s and ends with a 0, so the indicator 110 is obtained. Then the length information is calculated as $10 - 2^3 = 2$, expressed as 010 in binary. Finally, the run symbol bit 0 is appended, resulting in the code 1100100. Combining these two parts, the final code sequence is 011011100100. In other words, the ERLE data structure comprises an indicator followed by either a fixed-length or a variable-length codeword. This structure is applied iteratively until the entire raw data

bitstream is processed. Overall, the format of ERLE compression codes can be regarded as bitstream-oriented.

In the decoding process, when you receive the ERLE compressed code 011011100100, you first read the indicator bit 0 and directly read the next four bits, which are 1101. Then, continue to read the indicator 110, indicating that the prefix length is three. Next, read the following three bits, 010, which represent the value two. The calculated length is $2^3 + 2 = 8$. Finally, read the run symbol bit 0. Therefore, it is decoded to 0000000000. Hence, the final decoded sequence is 11010000000000.

3. Proposed Scheme

This section elaborates on the proposed reversible data hiding scheme in encrypted hyperspectral images. The overall process is illustrated in Figure 1. The content owner first encrypts the original hyperspectral image using the encryption key K_e to obtain the encrypted image I_e. The image is then divided into blocks, and the label map LM is generated as auxiliary information. This label map is embedded into the encrypted image to produce the encrypted image I'_e with LM, which is then sent to the data hider. The data hider uses the data hiding key K_d to embed the secret data into the reserved space, resulting in the marked encrypted image I_m. Upon receiving I_m, the receiver can recover the original image if they possess the encryption key K_e or extract the secret data if they have the data hiding key K_d.

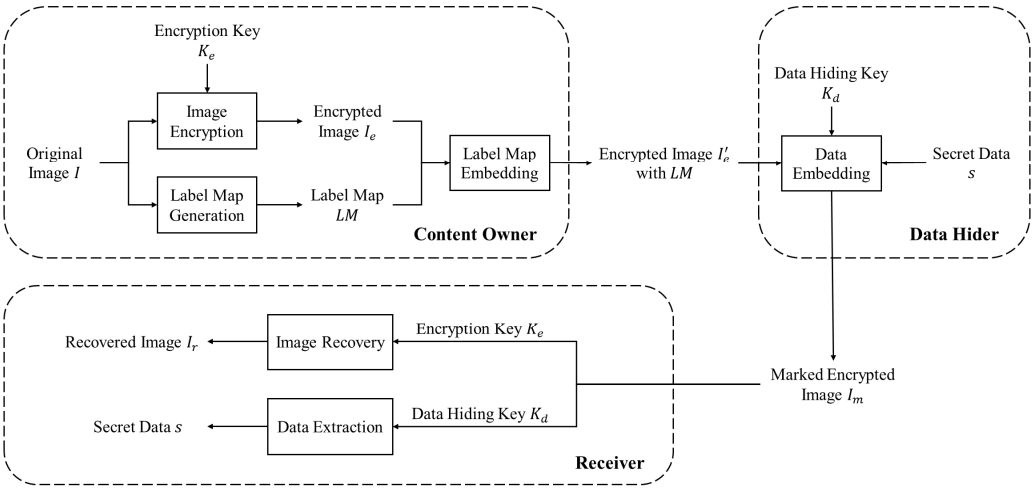

Figure 1. Flow of the proposed scheme.

3.1. Image Encryption

First, the encryption key K_e is used to encrypt each pixel of the original image I. A pseudo-random matrix R of the same size as the image is generated using K_e. Then, the encrypted image I_e is obtained by performing the following encryption operation as shown in Equation (1), where i and j represent the pixel positions, and $\oplus$ denotes the bitwise exclusive OR operation.

$$I_e(i,j) = I(i,j) \oplus R(i,j) \tag{1}$$

3.2. Label Map Generation

Since the bit depth of hyperspectral images is usually 16 bits, but the highest brightness level is rarely used in practical applications, the most significant bits (MSBs) of many pixels are usually zero. In addition, due to the strong correlation between adjacent pixels, these characteristics create favorable conditions for reserved space. To take advantage of this, each band of the image is divided into equal-sized, non-overlapping blocks. In each block,

the first pixel is selected as the reference pixel. Then, the MSBs of the other pixels are compared with those of the reference. If all MSBs match the reference pixel, the block is labeled as 1, indicating it is suitable for data embedding; otherwise, it is labeled as 0. For blocks labeled as 1, the reference pixel is retained, and the MSBs of the other pixels provide reserved space that can be used for embedding.

It is worth mentioning that the proposed scheme supports multi-layer operations and allows customization of the number of MSBs used for comparison. Specifically, when we assume three layers, suppose that the first layer uses 8×8 blocks, the second layer uses 4×4 blocks, and the third layer uses 2×2 blocks. Also, assume that five MSBs are used for comparison. If, in a first-layer 8×8 block, the five MSBs of the first pixel match those of all the other pixels, the block is labeled as 1, and no further processing is required. Otherwise, it is labeled as 0 and proceeds to the second layer for evaluation. The second layer divides each 8×8 block into four 4×4 blocks, and each 4×4 block undergoes the same MSBs matching process. Similarly, if all MSBs within a 4×4 block match, its second-layer label is set to 1, and no further processing is needed. Otherwise, it is labeled as 0 and moves to the third layer for evaluation. In the third layer, each 4×4 block is further divided into four 2×2 blocks, and the same matching process is applied to obtain the third-layer labels.

After traversing and processing the entire image, the label map LM is generated. This label map consists of all first-layer labels, all second-layer labels, and all third-layer labels across the image, rather than storing the first, second, and third-layer labels for each individual block. The advantage of this structure is that, due to the correlation between neighboring blocks, the embeddable and non-embeddable regions tend to be spatially continuous. As a result, we adopt the ERLE algorithm to compress the first-layer labels.

Figure 2 shows an example. Figure 2a is an 8×8 original image, which is converted into a 16-bit binary representation, as shown in Figure 2b–d. The area inside the orange box indicates the region that the current layer needs to process. The first pixel is used as the reference pixel, and its five MSBs are marked in blue. The five MSBs of other pixels are marked in red if they differ from the reference; otherwise, they remain black. As seen in Figure 2b, there are red pixels in the 8×8 block, indicating that the block cannot be embedded, so the first-layer label is 0. The process then moves to the second layer, as shown in Figure 2c, where the 8×8 block is divided into four 4×4 blocks. Similarly, the first pixel of each 4×4 block serves as the reference pixel, and its five MSBs are marked in blue. If other pixels differ from the reference, they are marked in red. It can be seen that only the 4×4 block in the upper right corner contains red pixels. Therefore, the labels of the four second-layer blocks are 1, 0, 1, and 1, respectively. Only the 4×4 block labeled as 0 in the upper right corner proceeds to the third layer, as shown in Figure 2d. This 4×4 block is further divided into four 2×2 blocks. Again, the five MSBs of the first pixel are marked in blue to indicate the reference pixel, and other pixels are marked in red if they are different. Only the 2×2 block in the upper left corner has no red pixels, so the labels of the third-layer blocks are 1, 0, 0, and 0. As a result, the label map is 010111000, and its tree diagram is shown in Figure 3. It should be noted that this is only an illustrative example. In practical applications, the image would be much larger than the 8×8 example, and the label map would also be significantly longer. Therefore, the first-layer labels can be compressed using the ERLE algorithm.

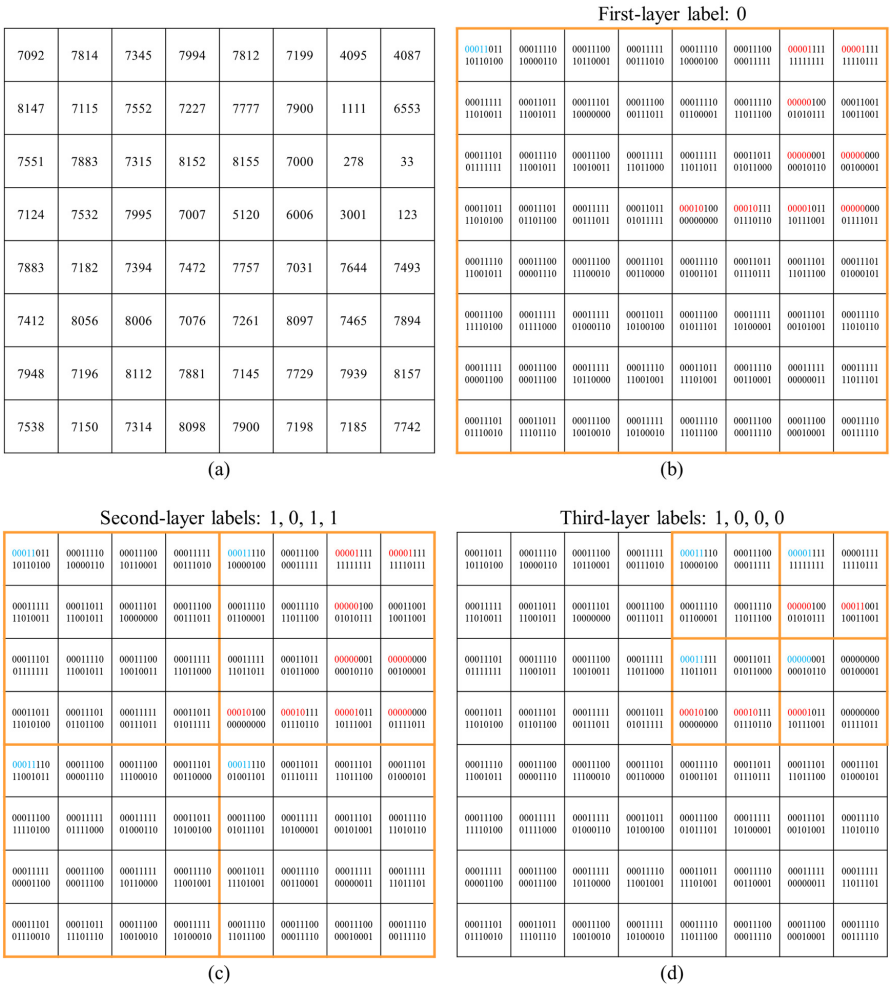

Figure 2. Example of label map generation: (**a**) original image; (**b**) first layer; (**c**) second layer; (**d**) third layer.

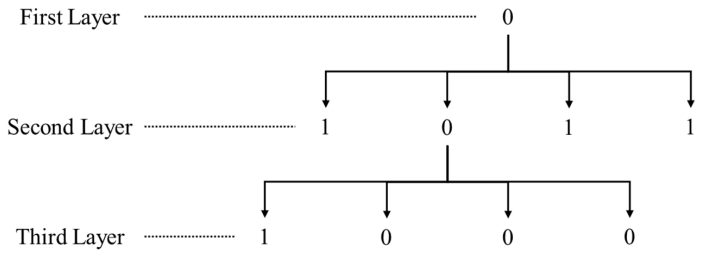

Figure 3. Tree diagram of the label map.

3.3. Label Map Embedding

After the processes in Sections 3.1 and 3.2, we obtain the encrypted image I_e and the label map LM. In this subsection, we embed LM into I_e. The auxiliary information we actually want to embed includes the length of LM and LM itself. The length information does not need to be stored explicitly, as the required number of bits can be determined based on the image size and the theoretical maximum length of LM. We embed the auxiliary information directly into the least significant bit (LSB) of each pixel of each band of I_e in sequence. Then, in order to enable recovery, the replaced LSBs are embedded into the blocks labeled as 1. Their length does not need to be recorded additionally, as it is the same as the length of the label map, which has already been recorded. After this process, we obtain the encrypted image I'_e with LM. The remaining embeddable space is used for data embedding in the next subsection.

3.4. Data Embedding

In this subsection, the data hider receives the encrypted image I'_e with the label map. Before embedding data, it first extracts the LSB from each pixel of each band of I'_e, reads the length information of the label map, and then retrieves the entire label map. Based on the label map, the embeddable positions can be identified, after which the region used to store the replaced LSBs is skipped. The remaining embeddable space is reserved for data embedding. It should be noted that, for the security of the secret data, it is encrypted using the data hiding key K_d before embedding. After embedding, the marked encrypted image I_m is obtained.

The proposed Algorithm 1, which performs data embedding followed by XOR encryption, is presented below:

Algorithm 1: Embedding with XOR Encryption

Input:
- Original hyperspectral image HSI (uint16, size H × W × B)
- Block size blk (e.g., 16)
- Number of MSB bits n_msb used for testing/embedding
- Maximum quadtree depth max_depth
- Secret bitstream S (binary sequence)
- Encryption key K (same size as HSI)

Output:
- Encrypted and embedded hyperspectral image HSI_emb
- Label maps L for all bands and levels
- Final embedding capacity
- Embedded secret bits S_used

Stage A—Label Map Generation (using the original HSI):
1. For each spectral band b:
    1.1 Partition the image into non-overlapping blk × blk blocks.
    1.2 Perform quadtree recursion up to max_depth:
        a. For each block at the current level, compare the n_msb of all pixels with the top-left pixel's n_msb.
        b. If all match, label the block as '1'; otherwise, label as '0'.
        c. If label = '0' and the block size > 1 and level < max_depth, split into 4 sub-blocks for the next level.
        d. If label = '1', the block will still be embedded in Stage C.
    1.3 Store the binary labels for each level in L_b.

Stage B—XOR Encryption:
1. Generate the encrypted image: Enc_HSI ← bitxor(HSI, K)

Stage C—Label-Guided Embedding on Enc_HSI:
1. Initialize pointer p ← 1.
2. For each spectral band b:
    2.1 For each block at each level following L_b:
        - If label = '1':
            a. Skip the reference pixel (top-left).
            b. Replace the n_msb of each remaining pixel with the next n_msb bits from S, padding with zeros if S is exhausted.
            c. Increment p accordingly.
        - If label = '0': do not embed bits; follow splitting rules from Stage A.
    2.2 For edge (incomplete) blocks in the right, bottom, and bottom-right regions, repeat the same embedding rule if label = '1'.
5. Output HSI_emb, L, final embedding capacity, and S_used = S[1:p].

Figure 4 presents an example of label map embedding and data embedding. In the original image I shown in Figure 4a, blue indicates the reference bits, while green denotes the embeddable space. In the encrypted image I_e shown in Figure 4b, the green region remains identical to that in Figure 4a and serves as the embeddable space. In the encrypted image I'_e with the label map shown in Figure 4c, the label map is determined to be 010111000 based on the original image in Figure 4a. It should be noted that this is just a simplified example. In practical applications, the label map would be compressed using the ERLE algorithm. However, since there is only one label in the first layer in this example, compression is not applied here. This label map is sequentially embedded into the least significant bits, represented in orange. The original least significant bits at those positions are relocated to the embeddable space as data, shown in purple. After receiving I'_e, the data hider embeds the secret data into the embeddable space indicated in green in Figure 4c. As illustrated in Figure 4d, the embedded secret data is represented in pink, resulting in the marked encrypted image I_m.

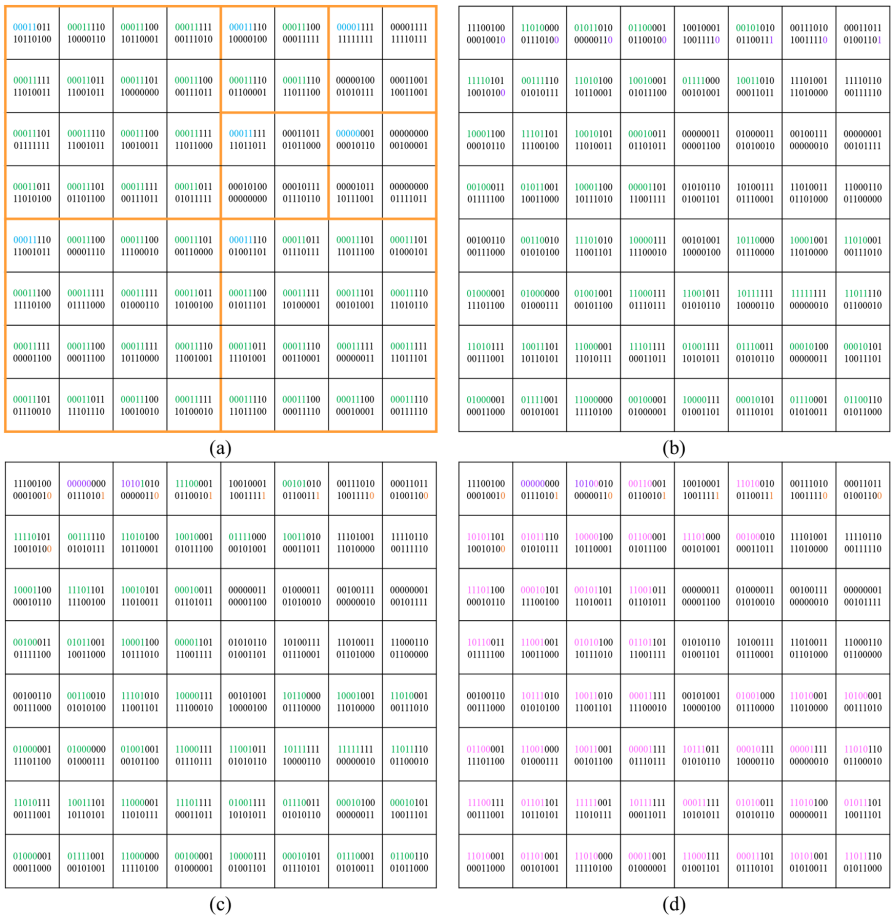

Figure 4. Example of label map embedding and data embedding: (**a**) original image; (**b**) encrypted image; (**c**) encrypted image with label map; (**d**) marked encrypted image.

3.5. Image Recovery and Data Extraction

When the receiver obtains the marked encrypted image I_m, it first extracts the auxiliary information from the LSB of each pixel in each band of the encrypted image to obtain the label map. This process does not require any key. If the receiver possesses the image encryption key K_e, it first extracts the replaced LSBs from the embedding positions indicated by the label map. These extracted LSBs are then restored to their original positions to complete the LSB recovery. Next, Equation (2) is applied to I_m, where I_r denotes the recovered image, R is the pseudo-random matrix generated by K_e, and i and j represent the pixel positions. However,

the recovery is not yet complete. The extracted replaced LSBs are first restored to their original positions to complete the LSB recovery. Then, in the blocks labeled as 1, the other pixels still have their MSBs occupied by embedded data. To fully restore these pixels, the MSBs of the corresponding reference pixels in I_r are directly used to replace the embedded MSBs, thereby completing the recovery of the original image.

$$I_r(i,j) = I_m(i,j) \oplus R(i,j) \tag{2}$$

If the receiver possesses the data hiding key K_d, the replaced LSBs indicated by the label map are skipped, and the remaining data in the embeddable regions correspond to the secret data. By decrypting this data using K_d, the original secret information can be successfully extracted. In summary, this means that only when the receiver possesses both K_e and K_d can the original image be recovered and the secret data extracted. The proposed Algorithm 2, which performs data extraction followed by XOR decryption, is presented below:

Algorithm 2: Extraction with XOR Decryption

Input:
- Encrypted and embedded hyperspectral image HSI_emb (uint16, size H × W × B)
- Block size blk
- Number of MSB bits n_msb
- Maximum depth max_depth
- Expected number of secret bits N_exp (∞ for all)
- Encryption key K

Output:
- Extracted secret bitstream S_out
- Number of bits read N_read
- Restored original HSI HSI_restored

Stage A—Decode/Recover Label Maps from HSI_emb:
1. Read the serialized label-map payload from its predefined storage in HSI_emb (e.g., header/auxiliary area compressed by ERLE).
2. Decompress and reconstruct per-band, per-level label strings L_b for quadtree traversal.
3. (If a headerless variant is used) Re-infer labels by applying the n_msb-homogeneity test and the same quadtree policy used at embedding time.

Stage B—Label-Guided Extraction on Enc_HSI:
1. Initialize write pointer p ← 1 and an empty buffer S_out.
2. For each spectral band b:
    2.1 Traverse blocks level-by-level following the reconstructed labels L_b.
        - If label = '1':
            a. Use the top-left pixel as reference; skip it.
            b. Read n_msb from each remaining pixel in scan order; append to S_out (truncate to N_exp if needed).
            c. Restore each pixel by replacing its n_msb with the reference n_msb.
        - If label = '0': do not extract; if the block is splittable and level < max_depth, descend to its 4 children.
    2.2 For right, bottom and bottom-right edge regions (level-1 only), apply the same rule when the edge label = '1'.

Stage C—XOR Decryption:
1. Apply decryption to obtain the original image: HSI_restored ← bitxor(HSI_restored, K).
Return S_out, N_read (|S_out|), HSI_restored.

Figure 5 shows a specific example of image recovery and data extraction. When the receiver obtains the marked encrypted image, as shown in Figure 5a, the label of the first layer can be determined as only one bit, according to the image size of 8×8. As indicated by the orange bit in the figure, the receiver first reads the LSB of the first pixel and obtains 0, which means that embedding is not possible in the first layer. Therefore, the process moves to the second layer, where the next four LSBs of subsequent pixels are read, resulting in a second-layer label of 1, 0, 1, 1. Since a 0 appears in this label, the process proceeds to the third layer, reading another four LSBs and obtaining the label 1, 0, 0, 0. At this point, the complete label map is determined, and the location of each layer is indicated with orange boxes. Based on the label instructions, the areas containing embedded data can be identified. The replaced LSBs are marked in purple, and the embedded secret data is marked in pink. If the receiver possesses the image encryption key K_e, as illustrated in Figure 5b, the replaced LSBs can be restored to their original positions. Then, by applying Formula 2, the partially recovered image shown in Figure 5c is obtained. In the block marked as 1, the MSBs of the remaining pixels are still occupied by embedded data, highlighted in red. To fully recover these pixels, as shown in Figure 4d, the MSB of the reference pixel marked in blue is directly used to replace the embeddable space marked in green, completing the recovery of the original image. If the receiver also possesses the data hiding key K_d, the pink bits shown in Figure 5b can be extracted and decrypted to successfully extract the secret data.

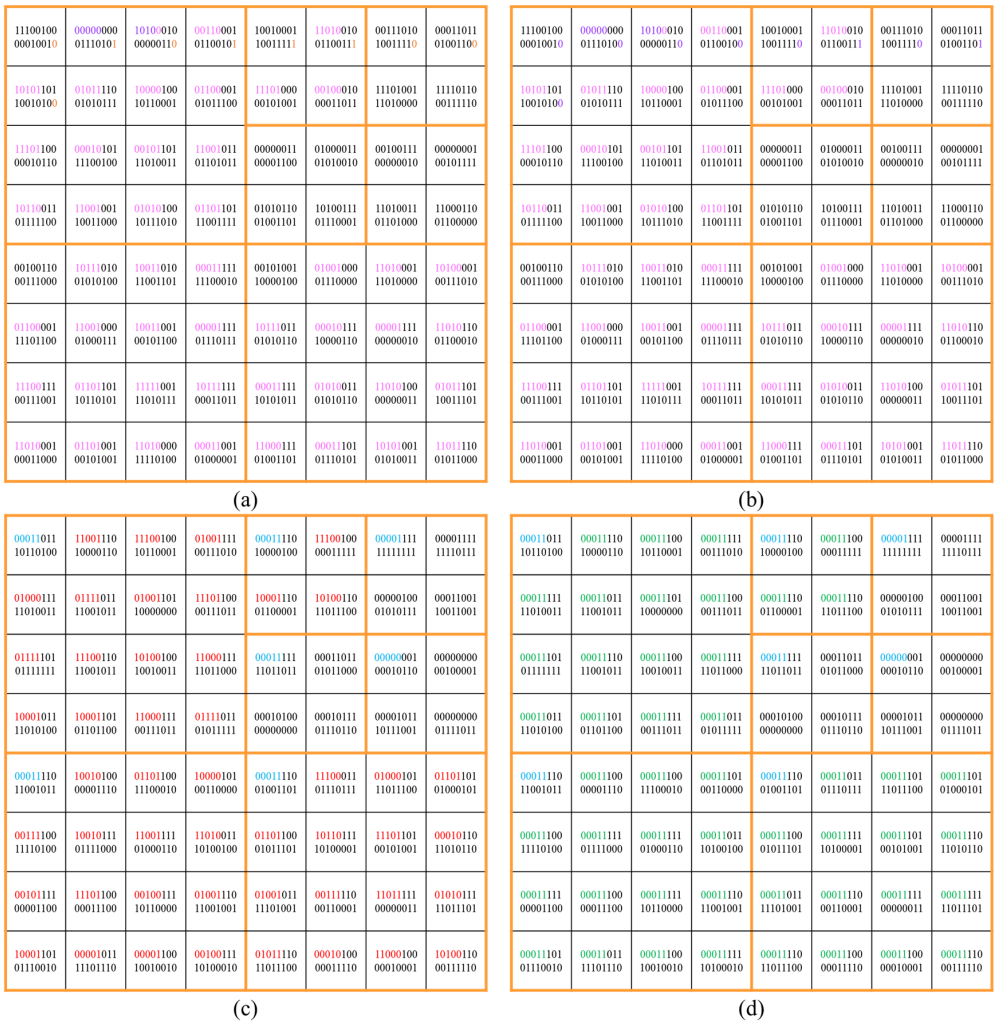

Figure 5. Example of image recovery and data extraction: (**a**) marked encrypted image; (**b**) marked encrypted image with replaced LSB; (**c**) partially recovered image; (**d**) fully recovered image.

4. Experimental Results

This section evaluates the effectiveness of the proposed scheme. To demonstrate the performance of our scheme alongside that of Zhang et al. [31], six hyperspectral test images: "Salinas," "Pavia University," "Indian pines," "Botswana," "Cuprite," and "KSC," shown in Figure 6, are served as the test images. The dimensions of these images are as follows: the first image (Salinas) is sized as 512 × 217 × 204 pixels, the second image (Pavia University) is sized as 1096 × 715 × 102 pixels, the third image (Indian pines) is sized as 145 × 145 × 200 pixels, the fourth image (Botswana) is sized as 1476 × 256 × 145 pixels, the fifth image (Cuprite) is sized as 512 × 614 × 224 pixels, and the sixth image (KSC) is sized as 512 × 614 × 176 pixels. All computations were conducted on a PC equipped with a 3.6 GHz Intel(R) Core (TM) i7-4790 CPU and 8 GB of RAM, running Windows 10 Professional. The algorithms were implemented in MATLAB R2023a.

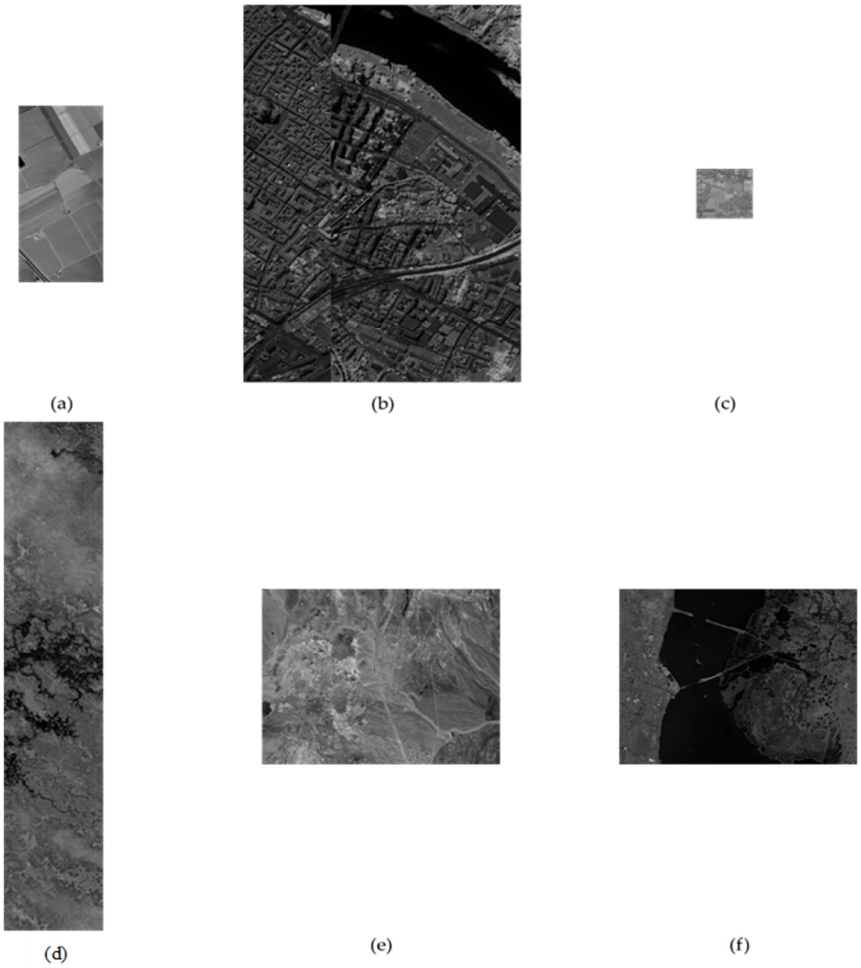

Figure 6. One selected band of six hyperspectral test images. (**a**) Salinas, sized as 512 × 217 × 204 pixels; (**b**) Pavia University, sized as 1096 × 715 × 102 pixels; (**c**) Indian pines, sized as 145 × 145 × 200 pixels; (**d**) Botswana, sized as 1476 × 256 × 145 pixels; (**e**) Cuprite, sized as 512 × 614 × 224 pixels; and (**f**) KSC, sized as 512 × 614 × 176 pixels.

To evaluate the effectiveness of the encryption process, the number of pixels change rate (NPCR), unified average changing intensity (UACI), and peak signal-to-noise ratio (PSNR) analyses were conducted on the images before and after encryption, as summarized in Table 1. NPCR is calculated using the formula in Equation (3).

$$\mathrm{NPCR} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} D(i,j) \times 100\%, \tag{3}$$

where $D(i,j) = 1$ if the pixel values at position (i,j) are different in the original and encrypted images; otherwise $D(i,j) = 0$. A higher NPCR indicates that the encryption method effectively alters the pixel values, enhancing the security of the encrypted image. The ideal NPCR value is greater than 99%. UACI is calculated as in Equation (4).

$$\text{UACI} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} \frac{|C(i,j) - P(i,j)|}{65535} \times 100\%, \tag{4}$$

where $P(i,j)$ and $C(i,j)$ denote the pixel values of the original and encrypted images, respectively. A higher UACI value implies greater intensity differences, contributing to enhanced resistance against attacks. The ideal UACI value is around 33%. From Table 1, it is noted that when the original test image is encrypted, the NPCR of each test image is greater than 99%, and the corresponding UACI is around 48%. Additionally, the PSNR of encrypted image remains only about 5 dB. Therefore, we can conclude that the block-based encryption method employed in our approach provides a certain level of security, the original pixel information is not preserved visually.

Table 1. Analysis of NPCR, UACI, and PSNR before and after image encryption.

Images	NPCR (%)			UACI (%)			PSNR (dB)		
	MAX	MIN	AVG	MAX	MIN	AVG	MAX	MIN	AVG
Salinas	99.9973	99.9973	99.9973	50.0701	45.0952	48.3238	5.46	4.76	5.00
Pavia University	99.9986	99.9986	99.9986	49.0135	47.4750	48.3281	5.11	4.76	4.92
Indian Pines	99.9952	99.9952	99.9952	48.3062	41.2542	45.9932	6.08	4.76	5.32
Botswana	99.9981	99.9981	99.9981	49.8532	43.4363	47.4423	5.73	4.76	5.09
Cuprite	99.9997	99.9984	99.9984	50.0826	42.2797	46.4075	5.91	4.76	5.27
KSC	99.9987	99.9987	99.9987	50.0020	49.7562	49.8843	5.39	4.77	4.90

Figure 7 illustrates a visual comparison between the original hyperspectral image and the reconstructed image after applying our proposed scheme, revealing a highly effective reconstruction with minimal visual differences detectable to the human eye. The proposed scheme successfully preserves the essential spectral characteristics of the original image, ensuring that the embedded information remains imperceptible while maintaining excellent image quality. This robustness is further confirmed by Figure 8, where the histograms of (a) the original image and (d) the restored image are nearly identical, indicating the full reversibility of the scheme. Meanwhile, the histograms of (b) the encrypted image and (c) the encrypted and embedded image demonstrate the effectiveness of the encryption process. These results emphasize the effectiveness of our technique in securing hidden data without compromising image fidelity. Figures 7 and 8 together demonstrate how our scheme achieves a delicate balance between data hiding and image quality, offering strong invisibility guarantees; the embedded information remains imperceptible even under close scrutiny, underscoring the reliability and practicality of the proposed scheme.

Unlike Table 1, Figure 9 presents the line charts of NPCR and UACI for the encrypted images. Notably, in the Cuprite test image, four completely black bands appear in bands 108 to 111, and twelve completely black bands appear in bands 155 to 166. This explains why the UACI values in the line chart of Cuprite test image exhibit one short and one long significant fluctuation.

Notably, in Tables 2–4, as the block size increases to 32 × 32, the area of each block grows larger. This results in a decreased likelihood of maintaining pixel similarity within the same block. In other words, a 32 × 32 block requires further division to preserve pixel consistency within smaller sections. As a result, for the 32 × 32 block size, the likelihood of

having three or four layers increases. This explains why in Tables 2–4, the reported number of layers for the 32 × 32 block size is three and four, while for the 8 × 8 and 16 × 16 block sizes, the number of layers reported is only two and three.

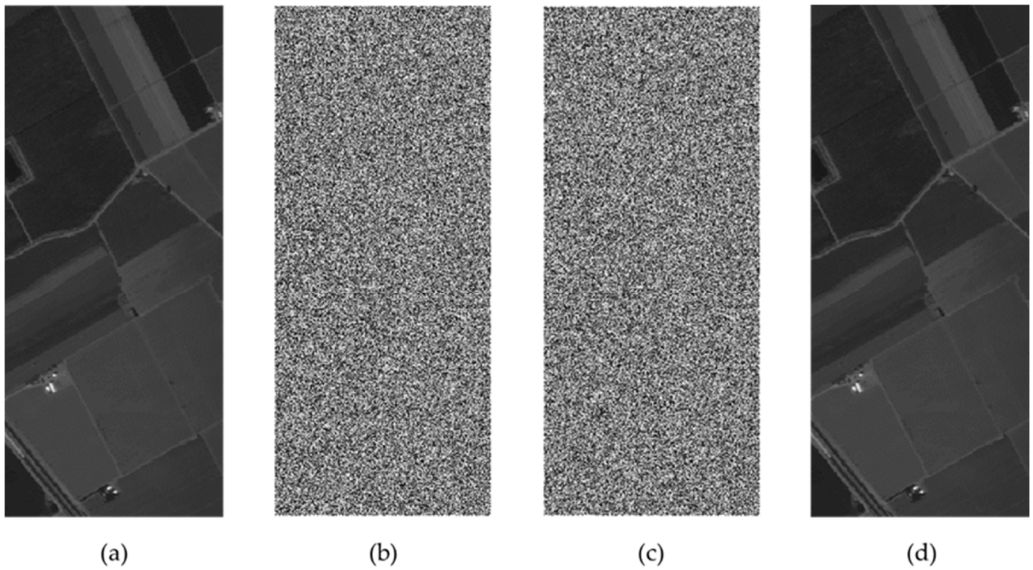

Figure 7. Results of our method applied to the Salinas image: (**a**) original image: Salinas; (**b**) encrypted image; (**c**) encrypted and embedded image; (**d**) restored image.

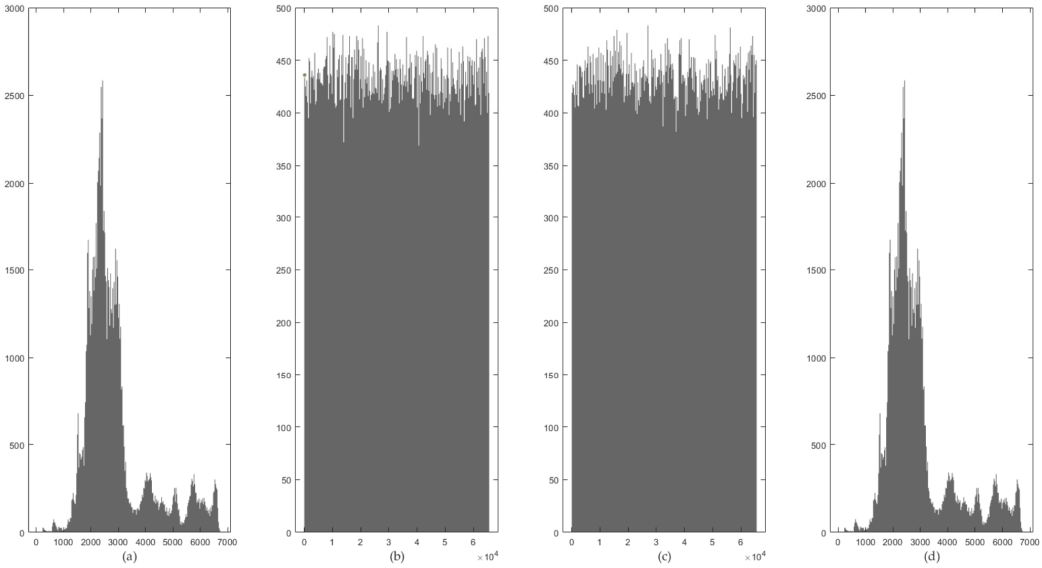

Figure 8. Histograms of our method applied to the Salinas image: (**a**) original image: Salinas; (**b**) encrypted image; (**c**) encrypted and embedded image; (**d**) restored image.

In Tables 2–4, although Cuprite offers a higher hiding capacity than KSC, its embedding rate consistently falls short compared to KSC. This discrepancy is due to the bandwidth: Cuprite has a bandwidth of 224, while KSC has a bandwidth of 176. As a result, the embedding rate performance of the two does not correlate with their respective hiding capacities.

In contrast to Tables 2–4, which report the embedding capacities of the six test images, Table 5 presents the horizontal and vertical correlation coefficients along with the entropy values of the images after being processed by the proposed encryption method. These metrics are provided to illustrate the effectiveness of the proposed approach in disrupting spatial correlations and enhancing encryption security. Table 5 demonstrates the encryption performance of the proposed method by analyzing the horizontal and vertical correlations,

as well as the entropy values of six test images. The results show that after encryption, both horizontal and vertical correlation values are close to zero, with some even approaching negative values. This indicates that the strong spatial correlations originally present in the images have been effectively disrupted. Furthermore, the entropy values of the encrypted images are consistently high (ranging from approximately 14.05 to 15.94), approaching the theoretical maximum for 16-bit data. These results confirm that the proposed encryption method successfully produces encrypted images that exhibit high randomness and low predictability, thereby ensuring strong encryption security.

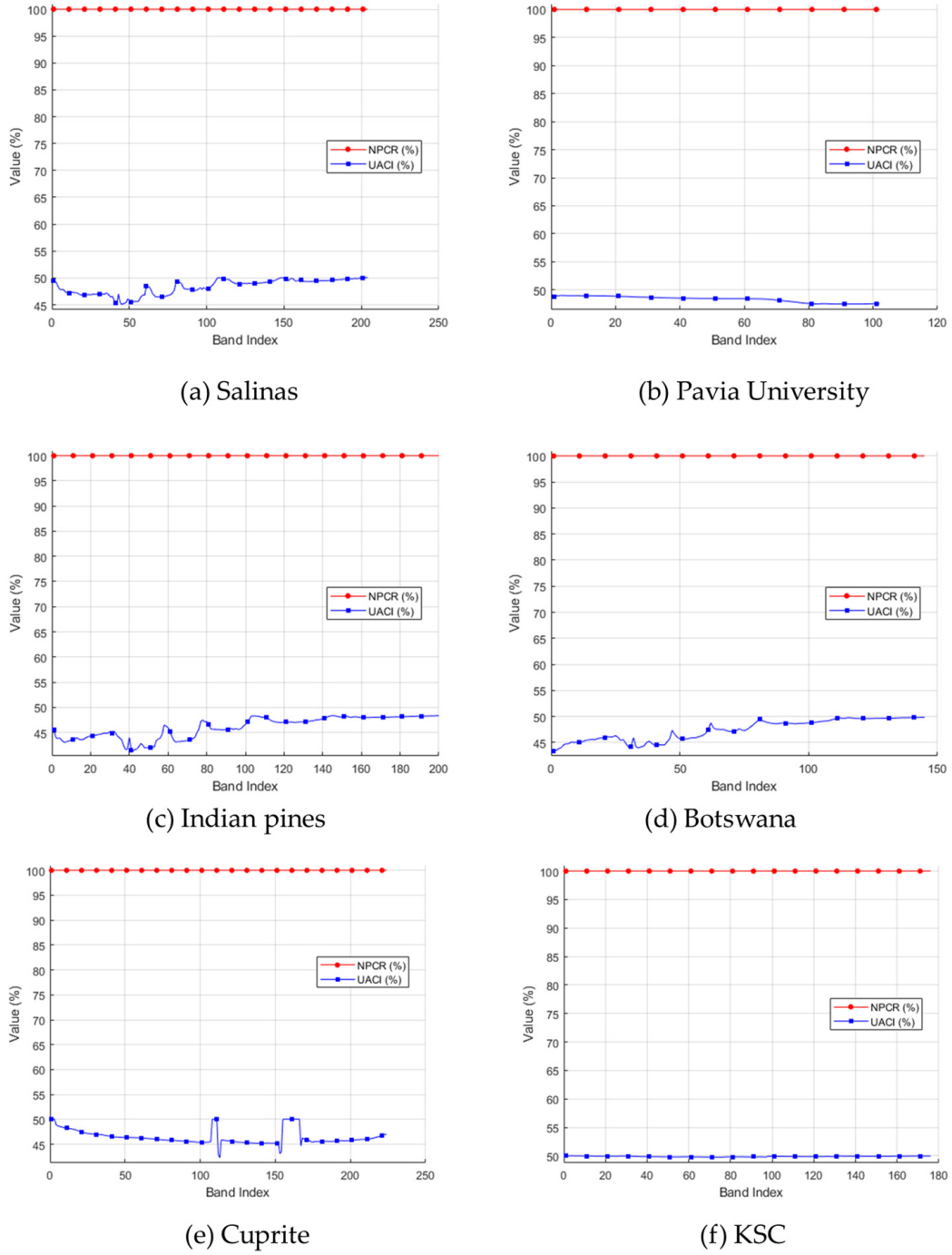

Figure 9. NPCR and UACI comparison across different images.

Table 2. Comparison of embedding capacity based on the 4 MSBs.

	Images	First-Layer Block Size = 8 × 8		First-Layer Block Size = 16 × 16		First-Layer Block Size = 32 × 32	
		Layers = 2	Layers = 3	Layers = 2	Layers = 3	Layers = 3	Layers = 4
Embedding Capacity	Salinas	8.81×10^7	8.85×10^7	8.83×10^7	8.91×10^7	8.84×10^7	8.91×10^7
	Pavia University	3.05×10^8	3.08×10^8	2.98×10^8	3.08×10^8	2.98×10^8	3.08×10^8
	Indian Pines	1.47×10^7	1.53×10^7	1.38×10^7	1.48×10^7	1.34×10^7	1.42×10^7
	Botswana	1.92×10^8	2.00×10^8	1.83×10^8	1.94×10^8	1.83×10^8	1.95×10^8
	Cuprite	2.66×10^8	2.69×10^8	2.63×10^8	2.69×10^8	2.64×10^8	2.70×10^8
	KSC	2.15×10^8	2.16×10^8	2.14×10^8	2.17×10^8	2.15×10^8	2.18×10^8
Embedding Rate	Salinas	3.89	3.90	3.89	3.93	3.90	3.93
	Pavia University	3.82	3.86	3.73	3.85	3.73	3.86
	Indian Pines	3.49	3.64	3.28	3.53	3.18	3.38
	Botswana	3.51	3.64	3.34	3.55	3.35	3.55
	Cuprite	3.78	3.82	3.74	3.83	3.75	3.84
	KSC	3.88	3.90	3.87	3.92	3.88	3.93

Table 3. Comparison of embedding capacity based on the 5 MSBs.

	Images	First-Layer Block Size = 8 × 8		First-Layer Block Size = 16 × 16		First-Layer Block Size = 32 × 32	
		Layers = 2	Layers = 3	Layers = 2	Layers = 3	Layers = 3	Layers = 4
Embedding Capacity	Salinas	1.03×10^8	1.06×10^8	9.90×10^7	1.04×10^8	9.77×10^7	1.02×10^8
	Pavia University	2.84×10^8	3.23×10^8	2.21×10^8	2.84×10^8	2.20×10^8	2.83×10^8
	Indian Pines	1.68×10^7	1.82×10^7	1.47×10^7	1.69×10^7	1.39×10^7	1.57×10^7
	Botswana	2.13×10^8	2.32×10^8	1.89×10^8	2.15×10^8	1.89×10^8	2.15×10^8
	Cuprite	3.01×10^8	3.14×10^8	2.81×10^8	3.04×10^8	2.81×10^8	3.04×10^8
	KSC	2.68×10^8	2.70×10^8	2.68×10^8	2.71×10^8	2.69×10^8	2.72×10^8
Embedding Rate	Salinas	4.56	4.69	4.37	4.59	4.31	4.51
	Pavia University	3.55	4.05	2.77	3.55	2.76	3.54
	Indian Pines	4.00	4.32	3.50	4.03	3.31	3.73
	Botswana	3.89	4.23	3.44	3.92	3.45	3.93
	Cuprite	4.27	4.46	3.99	4.31	3.99	4.32
	KSC	4.85	4.87	4.84	4.91	4.85	4.92

Table 4. Comparison of embedding capacity based on the 6 MSBs.

	Images	First-Layer Block Size = 8 × 8		First-Layer Block Size = 16 × 16		First-Layer Block Size = 32 × 32	
		Layers = 2	Layers = 3	Layers = 2	Layers = 3	Layers = 3	Layers = 4
Embedding Capacity	Salinas	1.15×10^8	1.22×10^8	1.06×10^8	1.16×10^8	1.03×10^8	1.12×10^8
	Pavia University	1.91×10^8	2.81×10^8	1.11×10^8	1.89×10^8	1.10×10^8	1.88×10^8
	Indian Pines	1.47×10^7	1.78×10^7	1.15×10^7	1.48×10^7	1.07×10^7	1.33×10^7
	Botswana	1.89×10^8	2.32×10^8	1.49×10^8	1.90×10^8	1.49×10^8	1.91×10^8
	Cuprite	2.55×10^8	3.09×10^8	1.78×10^8	2.55×10^8	1.78×10^8	2.55×10^8
	KSC	3.22×10^8	3.24×10^8	3.21×10^8	3.26×10^8	3.22×10^8	3.27×10^8
Embedding Rate	Salinas	5.09	5.37	4.68	5.11	4.55	4.95
	Pavia University	2.39	3.51	1.38	2.36	1.37	2.35
	Indian Pines	3.50	4.24	2.74	3.52	2.53	3.15
	Botswana	3.45	4.24	2.72	3.47	2.72	3.48
	Cuprite	3.62	4.39	2.53	3.63	2.52	3.62
	KSC	5.82	5.85	5.81	5.89	5.82	5.90

In the proposed scheme the label map is first compressed using ERLE to minimize the size of the auxiliary data. Accordingly, Table 6 presents the impact of applying or not applying ERLE compression on the auxiliary data size under different MSB configurations, with the block size set to 8 × 8 and the number of layers set to three. Table 6 shows that when the MSB is set to six, the improvement ranges from 19.46% to 85.38%. In comparison, with the MSB set to four, the average improvement reaches up to 60% when using

ERLE. This demonstrates that ERLE effectively reduces the size of the auxiliary data in the proposed scheme.

Table 5. Encryption performance analysis.

Images	Horizontal Correlation			Vertical Correlation			Entropy		
	MAX	MIN	AVG	MAX	MIN	AVG	MAX	MIN	AVG
Salinas	0.0048	0.0034	0.0041	0.0004	−0.0005	0.0002	15.5154	15.5014	15.5096
Pavia University	−0.0013	−0.0016	−0.0014	−0.0011	−0.0015	−0.0013	15.9397	15.9380	15.9388
Indian Pines	0.0010	−0.0039	−0.0018	0.0062	0.0017	0.0042	14.0707	14.0461	14.0585
Botswana	0.0019	0.0009	0.0016	−0.0000	−0.0010	−0.0004	15.8719	15.8677	15.8701
Cuprite	−0.0007	−0.0021	−0.0017	0.0003	−0.0008	−0.0003	15.8443	15.8403	15.8422
KSC	−0.0028	−0.0043	−0.0037	−0.0011	−0.0027	−0.0018	15.8445	15.8398	15.8422

Table 6. Embedding capacity comparison with and without ERLE based on 4, 5, and 6 MSBs.

Images	4 MSBs			5 MSBs			6 MSBs		
	With ERLE	*w/o* ERLE	Improvement Rate	With ERLE	*w/o* ERLE	Improvement Rate	With ERLE	*w/o* ERLE	Improvement Rate
Salinas	39,451	365,568	89.21%	130,484	365,568	64.31%	192,930	365,568	47.22%
Pavia University	502,152	1,257,660	60.07%	1,006,492	1,257,660	19.97%	741,380	1,257,660	41.05%
Indian Pines	25,436	72,000	64.67%	39,153	72,000	45.62%	41,823	72,000	41.91%
Botswana	231,949	858,400	72.98%	366,505	858,400	57.30%	373,875	858,400	56.45%
Cuprite	241,464	1,103,872	78.13%	538,980	1,103,872	51.17%	889,107	1,103,872	19.46%
KSC	125,739	867,328	85.50%	125,739	867,328	85.50%	126,813	867,328	85.38%

Table 7 summarizes the complexity of the proposed scheme, showing that the time and space complexities for data embedding, data extraction, and image recovery are all O(n). Table 8 reports the execution times for data embedding and extraction under the configuration of an 8×8 block size, four MSBs, and a three-layer setting across various images. Additionally, Table 9 presents the execution times for encryption and decryption, demonstrating that both operations are efficient. These results collectively indicate that the proposed method achieves good results in terms of computational complexity and practical execution efficiency.

Table 7. Complexity of the proposed scheme.

Process	Data Embedding	Data Extraction	Image Recovery
Time	$O(n)$	$O(n)$	$O(n)$
Space	$O(n)$	$O(n)$	$O(n)$

Table 8. Execution time (seconds) based on the 4 MSBs.

Images	Data Embedding	Data Extraction	Image Recovery
Salinas	1.9990	0.0321	0.6697
Pavia University	9.8202	0.1037	4.7538
Indian Pines	1.1081	0.0484	0.2373
Botswana	14.4178	0.0537	7.9359
Cuprite	13.3442	0.0569	8.6266
KSC	13.4145	0.0610	9.0590

Table 10 compares the embedding capacity and embedding rate of the proposed scheme with two existing methods, Yeh et al. [32] and Zhang et al. [31], under the four-MSB configuration. The results show that the proposed scheme consistently achieves the highest embedding capacities across most test images, with an average of

1.82×10^8 bits—slightly higher than Yeh et al. [32] (1.80×10^8 bits) and significantly outperforming Zhang et al. [31], which reports a much lower average of only 8.62×10^6 bits. Similarly, in terms of embedding rate, the proposed scheme maintains a slightly higher average rate of 3.79 bpppb compared to 3.76 bpppb for Yeh et al. [32], while Zhang et al. [31] lags far behind at 0.36 bpppb. These results highlight the advantage of the proposed method in achieving both high embedding capacity and efficiency, making it more suitable for practical high-capacity data hiding applications.

Table 9. Execution time (seconds) of encryption and decryption.

Images	Encryption	Decryption
Salinas	0.0803	0.0823
Pavia University	0.3495	0.4040
Indian Pines	0.0205	0.0195
Botswana	0.2551	0.2593
Cuprite	0.3002	0.3025
KSC	0.2711	0.2427

Table 10. Comparison of embedding capacity (bits) and embedding rate (bpppb) for each method.

Measurements	Methods	Salinas	Pavia University	Indian Pines	Botswana	Cuprite	KSC	Average
Embedding Capacity	Yeh et al. [32] (4 MSB)	8.76×10^7	3.04×10^8	1.51×10^7	2.02×10^8	2.60×10^8	2.16×10^8	1.80×10^8
	Proposed (4 MSB)	8.85×10^7	3.08×10^8	1.53×10^7	2.00×10^8	2.69×10^8	2.16×10^8	1.82×10^8
	Zhang et al. [31]	4.19×10^6	1.89×10^7	2.78×10^6	-	-	-	8.62×10^6
Embedding Rate	Yeh et al. [32] (4 MSB)	3.86	3.80	3.59	3.69	3.69	3.90	3.76
	Proposed (4 MSB)	3.90	3.86	3.64	3.64	3.82	3.90	3.79
	Zhang et al. [31]	0.18	0.24	0.66	-	-	-	0.36

Note: "-" indicates no data has been reported in Zhang et al. [31].

Figure 10 shows the embedding capacity results on the Salinas image under different MSB bit-depths and varying block sizes. The results indicate that embedding capacity generally increases with higher MSB bit-depths and larger block sizes, demonstrating the trade-off between payload and block granularity. Additionally, the three-layer configuration consistently outperforms the two-layer setup. These findings provide valuable insight into parameter selection for optimizing data embedding performance.

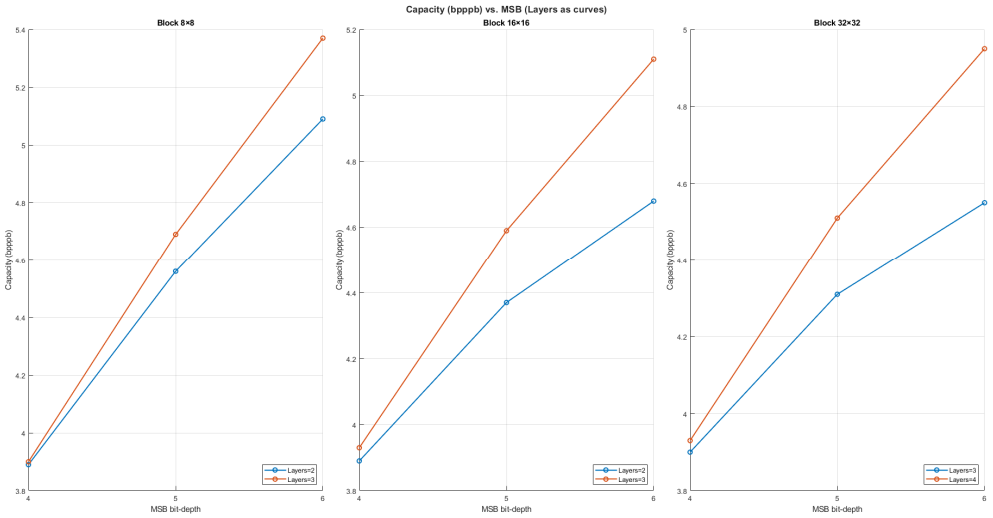

Figure 10. Embedding capacity on the Salinas image at MSB bit-depths of 4, 5, and 6 and block sizes of 8×8, 16×16, and 32×32.

Table 11 presents the results of key sensitivity testing on six hyperspectral images using five randomly generated keys (R1 to R5). The consistently high NPCR values near 100% and UACI values around 45% to 50% indicate that the encryption performance is robust and not affected by the choice of key. These results confirm that any randomly selected key can achieve strong encryption effectiveness.

Table 11. Results of key sensitivity testing.

Images	NPCR (%)					UACI (%)				
	R1	**R2**	**R3**	**R4**	**R5**	**R1**	**R2**	**R3**	**R4**	**R5**
Salinas	99.9991	99.9991	99.9973	100.0000	99.9982	48.2532	48.2385	48.1744	48.2673	48.2962
Pavia University	99.9985	99.9987	99.9982	99.9985	99.9983	48.3774	48.3648	48.3040	48.3415	48.3696
Indian Pines	99.9952	100.0000	99.9952	100.0000	100.0000	46.0360	46.1793	46.1840	46.3834	45.9621
Botswana	99.9987	99.9981	99.9984	99.9984	99.9981	47.4504	47.3650	47.4480	47.4984	47.4199
Cuprite	99.9984	99.9975	99.9994	99.9981	99.9994	46.3555	46.3894	46.3308	46.4088	46.2571
KSC	99.9990	99.9978	99.9981	99.9984	99.9984	49.7794	49.9240	49.9250	49.8523	49.7747

5. Conclusions

This paper presents a high-capacity RDH scheme tailored for encrypted HSIs, incorporating a multi-layer MSB block labeling mechanism and ERLE for efficient label map compression. Since it relies on multi-MSB matching, when large-scale matching is difficult, the multi-layer strategy divides the image into smaller blocks to enable more effective matching and to improve embedding capacity. The proposed method achieves competitive embedding capacity—up to 3.79 bpppb—and enables lossless recovery of both the original image and the embedded data. While image quality is not the primary focus due to the encrypted domain setting, the scheme ensures full reversibility and satisfactory encryption performance, as demonstrated by PSNR, NPCR, UACI, entropy, and correlation metrics. Compared to existing RDHEI methods, the proposed approach offers advantages in embedding flexibility and auxiliary data compression. These characteristics make it a promising candidate for secure and high-capacity data embedding in encrypted hyperspectral imagery.

Several directions are worth exploring in future work. One potential extension is to develop adaptive MSB selection strategies, potentially combined with deep learning-based [41] or statistical approaches, to dynamically adjust to local image characteristics and further improve embedding efficiency. Another avenue is to extend the proposed scheme to hyperspectral video or real-time data streams to enhance its practical applicability. Finally, incorporating the multi-image encryption concept [42] into the multi-channel structure of HSIs, along with lightweight encryption or compression modules, could reduce computational complexity, th ereby making the method more suitable for deployment in edge computing or satellite-based systems.

Author Contributions: Conceptualization, Y.L. and C.-C.L.; methodology, Y.L. and C.-C.L.; software, Z.-M.Y.; validation, Z.-M.Y.; writing—original draft preparation, Y.L., C.-C.L., and Z.-M.Y.; writing—review and editing, Y.L., C.-C.L., Z.-M.Y., C.-C.C. (Ching-Chun Chang), and 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: Data are contained within the article.

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

References

1. Comesaña, P.; Pérez-Freire, L.; Pérez-González, F. Fundamentals of Data Hiding Security and Their Application to Spread-Spectrum Analysis. In *Information Hiding 7th International Workshop, IH 2005*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 146–160.
2. Chang, C.C.; Nguyen, T.S.; Lin, C.C. A reversible compression code hiding using SOC and SMVQ indices. *Inf. Sci.* **2015**, *300*, 85–99. [CrossRef]
3. Tian, J. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 890–896. [CrossRef]
4. Ni, Z.; Shi, Y.Q.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
5. Thodi, D.M.; Rodríguez, J.J. Expansion embedding techniques for reversible watermarking. *IEEE Trans. Image Process.* **2007**, *16*, 721–730. [CrossRef]
6. Alattar, A.M. Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Trans. Image Process.* **2004**, *13*, 1147–1156. [CrossRef] [PubMed]
7. Peng, F.; Li, X.; Yang, B. Adaptive reversible data hiding scheme based on integer transform. *Signal Process.* **2012**, *92*, 54–62. [CrossRef]
8. Coltuc, D. Low distortion transform for reversible watermarking. *IEEE Trans. Image Process.* **2012**, *21*, 412–417. [CrossRef] [PubMed]
9. Pan, Z.; Hu, S.; Ma, X.; Wang, L. Reversible data hiding based on local histogram shifting with multilayer embedding. *J. Vis. Commun. Image Represent.* **2015**, *31*, 64–74. [CrossRef]
10. Zhang, T.; Hou, T.; Weng, S.; Zou, F.; Zhang, H.; Chang, C.C. Adaptive reversible data hiding with contrast enhancement based on multi-histogram modification. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 5041–5054. [CrossRef]
11. Weng, S.; Tan, W.; Ou, B.; Pan, J.S. Reversible data hiding method for multi-histogram point selection based on improved crisscross optimization algorithm. *Inf. Sci.* **2021**, *549*, 13–33. [CrossRef]
12. Fan, G.; Pan, Z.; Zhou, Q.; Gao, X.; Zhang, X. Multiple histogram based adaptive pairwise prediction-error modification for efficient reversible image watermarking. *Inf. Sci.* **2021**, *581*, 515–535. [CrossRef]
13. Ou, B.; Li, X.; Zhao, Y.; Ni, R.; Shi, Y.-Q. Pairwise prediction-error expansion for efficient reversible data hiding. *IEEE Trans. Image Process.* **2013**, *22*, 5010–5021. [CrossRef]
14. Li, X.; 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]
15. Li, X.; Zhang, W.; Gui, X.; Yang, B. Efficient reversible data hiding based on multiple histograms modification. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 2016–2027. [CrossRef]
16. Jia, Y.; Yin, Z.; Zhang, X.; Luo, Y. Reversible data hiding based on reducing invalid shifting of pixels in histogram shifting. *Signal Process.* **2019**, *163*, 238–246. [CrossRef]
17. Roy, A.; Chakraborty, R.S. Toward optimal prediction error expansion-based reversible image watermarking. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *30*, 2377–2390. [CrossRef]
18. Li, Q.; Ma, B.; Wang, X.; Wang, C.; Gao, S. Image steganography in color conversion. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *71*, 106–110. [CrossRef]
19. Li, Q.; Wang, X.; Ma, B.; Wang, X.; Wang, C.; Gao, S.; Shi, Y. Concealed attack for robust watermarking based on generative model and perceptual loss. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *32*, 5695–5706. [CrossRef]
20. Wang, C.; Zhang, Q.; Wang, X.; Zhou, L.; Li, Q.; Xia, Z.; Ma, B.; Shi, Y.Q. Light-Field Image Multiple Reversible Robust Watermarking Against Geometric Attacks. *IEEE Trans. Dependable Secur. Comput.* **2025**, 1–15. [CrossRef]
21. Zhang, X. Reversible data hiding in encrypted image. *IEEE Signal Process. Lett.* **2011**, *18*, 255–258. [CrossRef]
22. Zhang, M.; Tong, X.J.; Liu, J.; Wang, Z.; Liu, J.; Liu, B.; Ma, J. Image compression and encryption scheme based on compressive sensing and Fourier transform. *IEEE Access* **2020**, *8*, 40838–40849. [CrossRef]
23. Singh, R.; Vaish, A. MSB/LSB prediction based reversible data hiding in encrypted images: A survey. In *Machine Intelligence and Smart Systems: Proceedings of MISS 2020*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 11–24.
24. Ji, C.; Gao, G.; Shi, Y.Q. Reversible data hiding in encrypted images with adaptive Huffman code based on dynamic prediction axes. *IEEE Trans. Multimed.* **2023**, *26*, 5962–5975. [CrossRef]
25. Yao, Y.; Wang, K.; Chang, Q.; Weng, S. Reversible data hiding in encrypted images using global compression of zero-valued high bit-planes and block rearrangement. *IEEE Trans. Multimed.* **2023**, *26*, 3701–3714. [CrossRef]
26. Wang, J.; Ou, B. Video reversible data hiding: A systematic review. *J. Vis. Commun. Image Represent.* **2024**, *98*, 104029. [CrossRef]
27. Yin, X.; Liu, W.; Zhang, J.; Liu, W. Reversible data hiding in halftone images based on minimizing the visual distortion of pixels flipping. *Signal Process.* **2020**, *170*, 107605. [CrossRef]
28. Gao, X.; Pan, Z.; Gao, E.; Fan, G. Reversible data hiding for high dynamic range images using two-dimensional prediction-error histogram of the second time prediction. *Signal Process.* **2020**, *173*, 107579. [CrossRef]

29. Carpentieri, B.; Castiglione, A.; De Santis, A.; Palmieri, F.; Pizzolante, R. One-pass lossless data hiding and compression of remote sensing data. *Future Gener. Comput. Syst.* **2019**, *90*, 222–239. [CrossRef]
30. Fan, G.; Pan, Z.; Zhou, Q.; Dong, J.; Zhang, X. Pixel type classification based reversible data hiding for hyperspectral images. *Knowl.-Based Syst.* **2022**, *254*, 109606. [CrossRef]
31. Zhang, X.; Pan, Z.; Zhou, Q.; Fan, G.; Dong, J. A reversible data hiding method based on bitmap prediction for AMBTC compressed hyperspectral images. *J. Inf. Secur. Appl.* **2024**, *81*, 103697. [CrossRef]
32. Yeh, Z.M.; Lin, C.C.; Lin, Y.; Chang, C.C.; Chang, C.C. High-Capacity reversible data hiding in encrypted hyperspectral images using MSB prediction and arithmetic coding. *J. Inf. Hiding Multimed. Signal Process.* **2025**, *16*, 772–790.
33. Chang, C.-I. *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*; Springer: Berlin/Heidelberg, Germany, 2003.
34. Goetz, A.F.H.; Vane, G.; Solomon, J.E.; Rock, B.N. Imaging spectrometry for Earth remote sensing. *Science* **1985**, *228*, 1147–1153. [CrossRef]
35. Mulla, D.J. Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps. *Biosyst. Eng.* **2013**, *114*, 358–371. [CrossRef]
36. Rasti, B.; Hong, D.; Ghamisi, P.; Alizadeh, S.; Plaza, A. Deep learning for hyperspectral image classification: An overview. *IEEE Geosci. Remote Sens. Mag.* **2020**, *8*, 60–88. [CrossRef]
37. Tao, R.; Zhao, X.; Li, W.; Li, H.-C.; Du, Q. Hyperspectral anomaly detection by fractional fourier entropy. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2019**, *12*, 4920–4929. [CrossRef]
38. Huang, S.; Zhang, H.; Pižurica, A. Subspace clustering for hyperspectral images via dictionary learning with adaptive regularization. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5524017. [CrossRef]
39. Chen, K.; Chang, C.C. High-capacity reversible data hiding in encrypted images based on extended run-length coding and block-based MSB plane rearrangement. *J. Vis. Commun. Image Represent.* **2019**, *58*, 334–344. [CrossRef]
40. Golomb, S. Run-length encodings (corresp.). *IEEE Trans. Inf. Theory* **1966**, *12*, 399–401. [CrossRef]
41. Liu, Y.; Wang, C.; Lu, M.; Yang, J.; Gui, J.; Zhang, S. From simple to complex scenes: Learning robust feature representations for accurate human parsing. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 5449–5462. [CrossRef]
42. Gao, S.; Ding, S.; Ho-Ching Iu, H.; Erkan, U.; Toktas, A.; Simsek, C.; Wu, R.; Xu, X.; Cao, Y.; Mou, J. A three-dimensional memristor-based hyperchaotic map for pseudorandom number generation and multi-image encryption. *Chaos Interdiscip. J. Nonlinear Sci.* **2025**, *35*, 073105. [CrossRef]