JPEG Compatible Secret Sharing Scheme with Enhanced Performance

Xiaotian Wu, *Member, IEEE*, Dingbin Chen, Yuyang Xiong, Ching-Nung Yang, *Senior Member, IEEE*, WeiQi Yan, *Senior Member, IEEE*, Zhihua Xia, *Member, IEEE*

***Abstract*—Secret leakage and DC overflow are two critical challenges in secret image sharing (SIS) for JPEG images, with the latter also leading to format incompatibility. To tackle these two issues, a systematic investigation on SIS for JPEG images is presented. The primary contribution of this paper is the design of two fundamental components, DC secret sharing and AC secret sharing, for encoding the DC and AC coefficients, respectively, of a JPEG secret image. In DC secret sharing, DCA sharing and post-processing are combined together to deal with DC components. Notably, a partial shadow-adjustable secret sharing (PSASS) mechanism and a sample-recalculate strategy are devised to prevent DC overflow. In AC secret sharing, the AC cross-segment permutation and global scrambling are adopted to disorder AC coefficients, enabling the dramatic alternation of AC block features. The ACA sharing is then employed to encode the permuted AC coefficients. Extensive experiments and comparisons confirm the effectiveness and superiority of the proposed scheme. Compared to recent methods, our approach successfully eliminates secret leakage and avoids DC overflow, thereby ensuring full compatibility with the JPEG format.**

***Index Terms*—Secret sharing, secret image sharing, JPEG image, security, overflow.**

I. Introduction

Nowadays, secret sharing (SS) [1]–[7] has been extensively adopted in various security applications. Among them, multimedia security techniques such like reversible data hiding [1], [5], [8]–[11], image encryption [12]–[15], and secret image sharing (SIS) [16]–[21], has received much attention from the research community. The well-known (k, n) SIS safeguards a secret image by dividing the secret into n random-looking shadows such that any collection having k or more shadows can decode the secret. Whereas any $(k-1)$ or fewer shadows are not feasible to learn any information about the secret.

Xiaotian Wu, Dingbin Chen, and Yuyang Xiong are with the College of Cyber Security, College of Information Science and Technology, and Guangdong Key Laboratory of Data Security and Privacy Preserving, Jinan University, Guangzhou, China. E-mail: wxt.sysu@gmail.com

Ching-Nung Yang is with the Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan. E-mail: cnyang@gms.ndhu.edu.tw

WeiQi Yan is with the Department of Computer and Information Sciences, Auckland University of Technology, Auckland, New Zealand. E-mail: wyan@aut.ac.nz

Zhihua Xia is with the College of Cyber Security, Engineering Research Center of Trustworthy AI, Ministry of Education, Jinan University, Guangzhou, China. E-mail: xia_zhihua@163.com

This work was partially supported by National Key Research and Development Program of China (Grant No. 2022YFB3103100), National Natural Science Foundation of China (Grant No. U23B2023), Guangdong Key Laboratory of Data Security and Privacy Preserving (Grant No. 2023B1212060036), and National Science and Technology Council (Grant No. 112-2221-E-259-007-MY2). (Corresponding authors: Zhihua Xia and Xiaotian Wu)

Visual secret sharing (VSS) and polynomial-based SIS represent two primary schemes for protecting sensitive images. VSS employs base matrices to encode a black-and-white secret image into multiple shadows. The secret can then be reconstructed visually by printing the shadows on transparencies and stacking them together, thereby allowing decoding without the aid of computational devices. Building upon the foundational principles of VSS [22], various extensions of VSS were developed. Important attributes such as pixel expansion, reconstructed image quality, secret image format, and sharing policy were extensively studied. To reduce pixel expansion, probabilistic VSS , random grid-based VSS, and VSS formulated via integer linear programming [19] were presented. Meanwhile, XOR-based VSS schemes [19], [23] were introduced to achieve higher recovered image quality. To deal with grayscale images, halftoning techniques can first convert them into approximate binary versions, enabling the application of VSS to generate shadows. Moreover, advanced halftoning methods, such as parallel error diffusion [24] and analysis-by-synthesis [25], can be integrated with VSS to produce visually pleasing halftone shadows. To support flexible secret sharing policies, VSS schemes were also designed for general access structures [26], evolving access structures [27], and essential access structures [28].

On the other hand, the pioneer work of SIS was presented in [29], where Shamir's SS [30] was adopted for encrypting image pixels. Typically, the k coefficients of a $(k-1)$ degree polynomial serve to conceal k secret pixels. By utilizing Lagrange interpolation, the secret can be reconstructed from any k shadows. As the shadows are noise-like, the attention of eavesdropper would be readily aroused while random-looking data are transmitted. Consequently, steganography is available in SIS [31], where the shared data is embedded into a cover image to produce meaningful stego images, preventing an eavesdropper from learning that a secret exists. With the purpose of non-destructively extracting the hidden shared data from the stego images and facilitating lossless secret reconstruction, reversible and lossless SIS [21] was established. SIS with essential participants, where the participants are categorized as essential and non-essential, was proposed in [32] to offer flexible access structures. Only when sufficient number of essential participants are involved, the secret can be decoded. The SIS approaches in [33] [17] investigate the mechanism for determining whether a shadow is fake or not. Moreover, SIS with a variety of functionalities, such as cheating-prevention [34], and outsourcing computation [35], was also introduced.

However, the aforementioned SIS schemes are only applicable for uncompressed images. They cannot deal with the secret that is encoded by popular image compression technique such like JPEG. As the use of JPEG images continues to grow, ensuring their security has become increasingly necessary. JPEG encryption serves as an effective means to this end. Existing approaches have primarily focused on DC and AC components separately. For DC components, stream cipher is often applied to encrypt DC coefficients produced by specialized DCT transforms [36], which effectively introduces a high degree of disorder. The method in [37] processes DC codes (DCCs) through permutation and encrypts the appended bits of DCCs using a stream cipher, though it carries a risk of DC overflow. Another technique [38] encrypts DCCs by scrambling consecutive DCCs with the same sign and performing iterative grouping and swapping. However, leaving DCC values unchanged may present certain security risks. In [13], DC coefficients are encrypted by predicting them from the pixel correlation of decoded AC coefficients. The DC prediction errors are then grouped, and a random number is added to each group to improve security. A different strategy maps DC coefficients to variable-length integer codes within the same category [14], where coefficients at risk of overflow are labeled and rotated. For AC components, several encryption approaches have been presented, including stream cipher encryption of the appended bits of AC codes (ACCs) [39], run/size, value (RSV) permutation with identical run length [38], [40], ACC sign encryption [41], and intra-block ACC permutation [13].

A series of SIS schemes designed for JPEG images have also been reported. In [42], a joint SIS and JPEG compression method is presented, which employs Shamir's SS during the Huffman coding stage to maintain the visual security of the secret image directly in the compressed domain. A meaningful SIS scheme is proposed in [43] that operates on the DCT coefficients of JPEG images. It constructs meaningful shadow images via a random element utilization model, thereby improving the visual quality of both the shadow images and the recovered secret image. Furthermore, a robust SIS based on a stable block condition was developed in [44] to resist JPEG recompression. This method ensures the invariance of DCT coefficients during recompression, significantly enhancing the robustness of the scheme. Puteaux et al. [18] introduced an SIS technique applicable to JPEG images, focusing on sharing the AC components. In their method, the amplitude values of AC coefficients are concatenated to form a secret, which is then encrypted using Shamir's SS. However, this approach does not guarantee visual security, as the generated shadow images may still disclose private information despite encryption of the DC coefficients. Moreover, the issue of DC overflow is not considered, which could cause a large number of shared DC coefficients to exceed the valid range and lead to format incompatibility.

In this paper, we focus on the area of SIS for JPEG images, and present a secret JPEG image sharing (SJIS) scheme to address the security vulnerability and DC overflow issue in [18]. Main contribution of this paper is summarized below.

- A secret JPEG image sharing scheme is presented. To encrypt a JPEG secret image, two essential components, DC secret sharing and AC secret sharing, are investigated. The DC secret sharing deals with DC data while the AC secret sharing is responsible for encoding AC information. The shared DC and AC data are eventually synthesized to form the JPEG shadows.
- The DC secret sharing mechanism leverages a two-stage process: DCA sharing and subsequent post-processing. The DCA sharing utilizes the proposed partial-shadow adjustable secret sharing (PSASS) to split DC coefficients into shadows, while a sample-recalculation strategy refines the DC items to preemptively minimize overflow occurrences. A post-processing module incorporating DC shifting and reversible data hiding is then applied, guaranteeing that any overflow is entirely eliminated.
- The AC secret sharing framework incorporates adaptive key generation, AC permutation, and ACA sharing. The AC permutation employs a two-level strategy: first, AC cross-segment permutation randomizes the coefficients within designated blocks; second, a global permutation scrambles all blocks, excluding DC coefficients. The permuted coefficients are subsequently encoded using ACA sharing. By applying AC secret sharing, the structural features of AC blocks are substantially modified, resulting in a dramatic improvement in security.

The rest of this paper is organized as follows. Section II describes the JPEG bitstream and traditional SIS for uncompressed images. Motivation and overview are illustrated in Section III. Details of the proposed scheme, including DC secret sharing and AC secret sharing, are given in Sections IV and V. Experiments, discussions, and comparisons are demonstrated in Section VI. Section VII concludes this work.

II. Preliminaries

Detailed information of JPEG bitstream is described in this section, as well as traditional SIS for uncompressed images.

A. JPEG Bitstream

One popular format for storing and sending digital images is JPEG compression [45]. A JPEG image is established by performing the DCT, quantization, zig-zag scanning, and entropy coding. The entropy encoded data is constituted by N minimum coded units (MCUs, denoted as MCU_i, $1 \leq i \leq N$), with N representing the total number of non-overlapping JPEG image blocks. An MCU is achieved by encoding a DCT block after zig-zag scanning. A DCT block actually contains 1 DC coefficient and 63 AC coefficients. Fig. 1 demonstrates the structure of entropy encoded data of a JPEG image.

Each MCU_i has one DC code (DCC, specified by DCC_i) for the DC coefficient, several AC codes (ACCs, represented by ACC_i^j where $1 \leq j \leq t_i$ and t_i is the quantity of non-zero AC coefficients of the DCT block) for the AC coefficients and an end-of-block (EOB) code. Usually, $0 \leq t_i \leq 63$ for an 8×8 DCT block and ACC_i^j denotes the j-th ACC of the i-th DCT block. Additionally, the EOB code is absent when the last AC coefficient in the zig-zag sequence is non-zero.

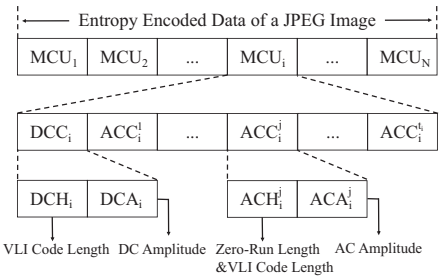

Fig. 1. The structure of entropy encoded data of a JPEG image.

One Huffman code (DCH, denoted by DCH_i) and the appended bits (DCA, given as DCA_i) are included in each DCC_i. DCA is obtained by performing the variable-length integer (VLI) coding to encode the difference between the quantized DC coefficient of the current block and that of the previous block. Let d_i $(1 \leq i \leq N)$ be the amplitude encoded by DCA_i. According to differential pulse code modulation (DPCM), the quantized DC coefficient of the i-th DCT block is determined as $D_i = \sum_{j=1}^{i} d_j$. DCH is derived by applying Huffman coding to encode the VLI code length of d_i.

Every ACC_i^j has one Huffman code (ACH, denoted as ACH_i^j), and the appended bits (ACA, represented by ACA_i^j). The ACA_i^j is obtained by applying VLI coding to encode the amplitude of the current non-zero AC coefficient. The zero-run length (i.e., the number of zero AC coefficients preceding the current non-zero AC coefficient) and the VLI code length of the current non-zero AC coefficient are simultaneously encoded as ACH_i^j by using Huffman coding.

B. Traditional Secret Image Sharing

Shamir's (k, n) SS [30] utilizes a $(k-1)$-degree polynomial $f(x) = (a_0 + \sum_{j=1}^{k-1} a_j x^j) \pmod p$ to encrypt a secret s, where the constant coefficient a_0 of $f(x)$ is replaced by secret s. The $(k-1)$ coefficients $a_1, \cdots, a_{k-1}$ are randomly chosen. $f(x)$ yields n shadows $f(x_1), \cdots, f(x_n)$ when n distinct values of $x_1, \cdots, x_n$ are fed to it. Lagrange interpolation can be used to reconstruct the $(k-1)$-degree polynomial $f(x)$ from any k shadows. Secret s is thus recovered as $s = f(0)$. In order to encode an uncompressed secret image, Shamir's SS was extended in [29]. First, the secret image is permuted at random. Then, any k secret pixels are embedded into the k coefficients $a_0, \cdots, a_{k-1}$ of the $(k-1)$-degree polynomial $f(x)$ to create n shared pixels. By collecting the corresponding shared pixels together, the shadow images are produced.

III. Motivation and Overview

In this section, limitations of previous techniques are summarized. Then, the motivation of this paper is explained, as well as the overview of the proposed approach.

A. Analysis and Motivation

The paramount concerns on recent SIS scheme for JPEG images [18] are summarized as follows.

- **Secret leakage.** In [18], various AC block features of the JPEG shadows, including NCC (nonzero coefficients count), PLZ (position of last nonzero coefficients), EAC (energy of non-zero AC coefficients) and RLS (run length sequence), closely resemble those of the original image. This similarity may lead to information disclosure, and sensitive content might be directly perceived on shadows. The experiment given in Fig. 2 further confirms the foregoing security flaw. One can observe the private information from the generated shadows.
- **Overflow of shared DC coefficients.** A DC coefficient must remain within a predefined numerical range. Since the DC coefficients in [18] are encoded directly, the generated shared values may exceed the permissible maximum or minimum values, resulting in DC overflow. More seriously, such overflow can ultimately cause format incompatibility.

Additionally, some existing approaches [17] [34] are only applicable to uncompressed images. JPEG compressed images, which are frequently utilized in various practical applications, are not taken into consideration. As a result of this, we are motivated to design a secure and format compatible secret JPEG image sharing technique. The proposed approach can effectively prevent secret leakage, and resolve the DC overflow issue, ensuring full compatibility with the JPEG format.

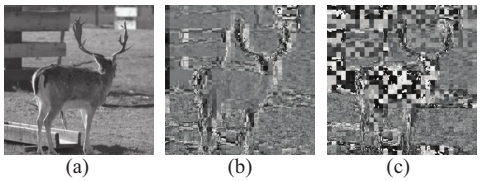

Fig. 2. The shadows generated by Puteaux *et al.*'s technique [18]. (a) The JPEG secret image, (b) a JPEG shadow by sharing only AC components, (c) a JPEG shadow by sharing both DC and AC components.

B. Overview of The Proposed Scheme

The proposed technique splits a JPEG secret image into n JPEG-format shadows in such a way that any collection having k or more shadows can disclose the secret. Whereas any less than k shadows are unable to reveal any clue about the secret. For a JPEG secret image, the compressed information can be broadly categorized into two types: DC coefficients represented by DCCs and AC coefficients denoted as ACCs. Accordingly, the proposed approach consists of two primary building blocks: DC secret sharing and AC secret sharing. The DC secret sharing is responsible for sharing the DCC bitstreams, while the AC secret sharing is in charge of encrypting the ACC data. After all DCCs and ACCs are processed, n DCC-whole-shadows and n ACC-whole-shadows are generated. These are then combined correspondingly to form n JPEG shadows. Fig. 3 depicts our scheme.

DCA sharing and post-processing are the two primary parts of DC secret sharing. The DCA sharing mainly deals with the DCAs (i.e., the encoded amplitudes of DC coefficients) from DCCs. In each round, a certain number of DCAs are concatenated together to obtain a DCA-secret, which is later split into n DCA-shadows. More concretely, a partial shadow-adjustable secret sharing (PSASS) technique is introduced in

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2026.3714465

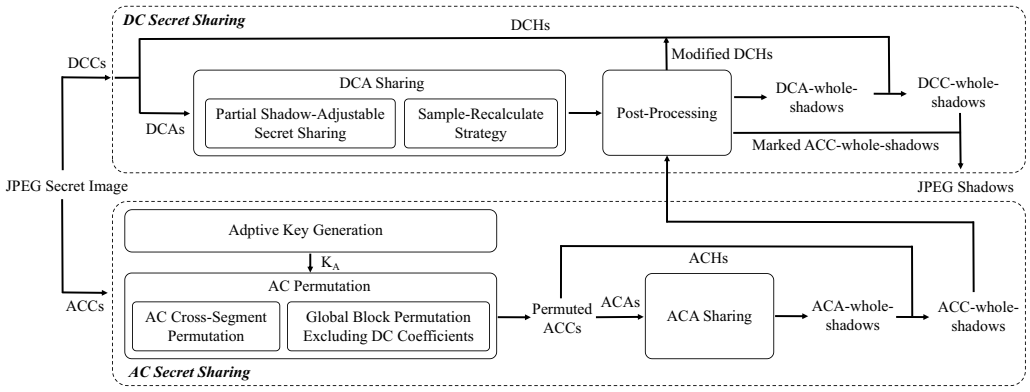

Fig. 3. Diagram of the proposed scheme.

order to alter the values of DCA-shadows. In this scenario, the number of overflow shared DC coefficients can be reduced. If the generated DCA-shadows are still unsuitable (i.e., causing overflow), a sample-recalculate strategy is employed to re-produce appropriate shared data. Furthermore, a post-processing is developed to completely prevent DC overflow. When all DCAs are encrypted, n DCA-whole-shadows are derived. By subsequently combining with the DCHs, n DCC-whole-shadows are created.

The AC secret sharing involves three main building blocks: adaptive key generation, AC permutation, and ACA sharing. The adaptive key generation is responsible for providing a secret key for the AC permutation. To boost the security, the AC permutation significantly changes the primary characteristics of AC blocks by randomly rearranging the AC coefficients. Additionally, all ACAs (i.e., the encoded amplitudes of AC coefficients) are encrypted in order to produce n ACA-whole-shadows by using the ACA sharing technique. At last, the ACA-whole-shadows are assembled with the original ACHs to yield n ACC-whole-shadows.

The ACC-whole-shadows are subsequently utilized in the post-processing stage of DC secret sharing, yielding marked shared data. By merging the DC shared data with the marked AC shared data, n JPEG-format shadows are produced. The original JPEG secret can be reconstructed when any k of n JPEG shadows are collected. For clarity, the key notations used in this paper are summarized in Table I

TABLE I
NOTATION USED IN THIS PAPER

Notation	Description
D_i	a quantized DC coefficient
$D_i^{(j)}$	the j-th shared value of D_i
$\tilde{D}_i^{(j)}$	a shifted value for $D_i^{(j)}$ to avoid overflow
d_i	the DC difference encoded by a DCA_i
$d_i^{(j)}$	the j-th shared value of d_i
$\tilde{d}_i^{(j)}$	a shifted difference calculated from $\tilde{D}_i^{(j)}$
$DCA_i^{(j)}$	the j-th shared value of DCA_i
D_{min}, D_{max}	the minimum and maximum DC values
$\overline{DCA}_{a,t}$	a DCA-secret by concatenating t consecutive DCAs
$\overline{DCA}_{a,t}^{(j)}$	the j-th shared value of $\overline{DCA}_{a,t}$
$p_i^{(j)}$	the distance from $D_i^{(j)}$ to the nearest DC boundary
$\mathcal{R}_\mathcal{S}, \mathcal{R}_\mathcal{N}, \mathcal{R}_\mathcal{O}$	the secure range, near-overflow range, and overflow range for DC coefficients
η_i	a variable to indicate whether the shared DC coefficients at location i might overflow or not
$\mathcal{F}_\mathcal{N}(\cdot), \mathcal{F}_\mathcal{P}(\cdot)$ $\mathcal{F}_\mathcal{E}(\cdot), \mathcal{F}_\mathcal{R}(\cdot)$	four procedures to calculate the NCC, PLZ, EAC, and RLS features of an AC block
I, II, III	three types of AC blocks
$\overline{ACA}$	an ACA-secret fetched from an ACA sequence

IV. DC Secret Sharing

This section begins by identifying the key challenge in DC secret sharing, followed by a description of the proposed solution, which consists of DCA sharing and post-processing. Generally, a DCA-secret is first constructed by concatenating consecutive DCAs from DCCs. Based on the DCA-secret, the proposed PSASS technique is adopted to produce n DCA-shadows. In case that the generated shadows are inappropriate (i.e., causing DC overflow), the sample-recalculate strategy is employed to create suitable ones once more. The usage of both PSASS and sample-recalculate strategy effectively diminishes the number of overflow shared DC coefficients. However, the overflow issue cannot be prevented completely. In order to fully turn the overflow DC coefficients into a valid range, a post-processing is described. When all secret DCAs are encrypted, n DCA-whole-shadows are achieved. By combining the DCA-whole-shadows with the original DCHs, n DCC-whole-shadows are constructed.

A. Key Issue of DC Secret Sharing

When dealing with DC data, the problem of DC overflow should be taken into account. Herein, we explain the DC overflow phenomenon. And then, the limitation of directly applying secret sharing to tackle DC information is discussed.

1) ***DC Overflow****:* A quantized DC coefficient D_i, $1 \leq i \leq N$, must be within a valid range, as denoted by

$$D_i \in [D_{min}, D_{max}] \tag{1}$$

where D_{min} and D_{max} are the minimum and maximum DC values calculated by

$$\begin{cases} D_{min} = round(-1024/Q_1), \\ D_{max} = round(1016/Q_1). \end{cases} \tag{2}$$

In the above equations, Q_1 denotes the first entry in the quantization table, which is related to the quality factor of a JPEG image. Moreover, suppose d_i is the value encoded by a DCA_i where $1 \leq i \leq N$, the corresponding DC coefficient D_i is determined as $D_i = \sum_{j=1}^{i} d_j$.

Building upon the preceding analysis, a key challenge in designing DC secret sharing is the potential overflow of D_i beyond the permissible range $[D_{min}, D_{max}]$ when the values of d_js are sufficiently large or small. This overflow would cause severe visual distortion in the decoded image [38], undermining the assessment of visual security. Moreover, such overflow may render the resulting bitstream incompatible with standard image applications (i.e., format incompatibility), leading to format errors and processing failure.

*2) **Limitation of Applying SS to Share DCA**:* We discuss the case of directly applying a (k, n) SS [30] to tackle a DCA value DCA_i. The following two steps are usually considered. (i) Construct a $(k-1)$-degree polynomial

$$f(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1} \tag{3}$$

under Galois Field $GF(2^M)$ where $a_0 = DCA_i$. Additionally, $a_1, \cdots, a_{k-1}$ are $(k-1)$ random numbers. (ii) By using different values of $x_1, \cdots, x_n$, n shadows are calculated by

$$DCA_i^{(1)} = f(x_1), \cdots, DCA_i^{(n)} = f(x_n). \tag{4}$$

Since the shared value $DCA_i^{(j)}$ $(1 \leq j \leq n)$ differs from the original DCA_i, the corresponding $d_i^{(j)}$ decoded from $DCA_i^{(j)}$ might deviate substantially from the original d_i converted from DCA_i. This discrepancy can lead to an overflow problem when these shared values are utilized. Example 1 provides an instance of DC overflow.

Example 1. *Consider an overflow situation of* $(2, 3)$ *scheme when sharing the DCAs* $DCA_1 = 41$ *and* $DCA_2 = 120$*. The corresponding DC differences decoded from* 41 *and* 120 *are* $d_1 = 41$ *and* $d_2 = -135$*, respectively. Let the DC valid range be* $[-204, 203]$*. Suppose the two polynomials for sharing* DCA_1 *and* DCA_2 *are*

$$\begin{cases} f_1(x) = 41 + 10x \pmod{GF(2^6)}, \\ f_2(x) = 120 + 217x \pmod{GF(2^8)}, \end{cases} \tag{5}$$

where 10 *and* 217 *are randomly chosen. Let the* 3 *IDs be* $x_1 = 1$*,* $x_2 = 2$*, and* $x_3 = 3$*. The DCA-shadows of the two secrets are computed by*

$$\begin{cases} DCA_1^{(1)} = f_1(x_1) = 35, DCA_1^{(2)} = f_1(x_2) = 61, \\ DCA_1^{(3)} = f_1(x_3) = 55, DCA_2^{(1)} = f_2(x_1) = 161, \\ DCA_2^{(2)} = f_2(x_2) = 215, DCA_2^{(3)} = f_2(x_3) = 14. \end{cases} \tag{6}$$

Further, the shared DC differences from the shared DCAs are

$$\begin{cases} d_1^{(1)} = 35, d_1^{(2)} = 61, d_1^{(3)} = 55, \\ d_2^{(1)} = 161, d_2^{(2)} = 215, d_2^{(3)} = -241. \end{cases} \tag{7}$$

The corresponding shared DC coefficients are

$$\begin{cases} D_1^{(1)} = d_1^{(1)} = 35, D_2^{(1)} = d_1^{(1)} + d_2^{(1)} = 196, \\ D_1^{(2)} = d_1^{(2)} = 61, D_2^{(2)} = d_1^{(2)} + d_2^{(2)} = 276, \\ D_1^{(3)} = d_1^{(3)} = 55, D_2^{(3)} = d_1^{(3)} + d_2^{(3)} = -186. \end{cases} \tag{8}$$

We notice that the shared DC coefficient $D_2^{(2)}$ *is not in the valid range (i.e.,* $276 \notin [-204, 203]$*).*

B. DCA Sharing

The DCA sharing is designed to encrypt the DCAs while leaving the DCHs unaltered. Essentially, a DCA-secret is created by extracting the DCAs from the DCCs of a JPEG secret image and concatenating parts of them. Based on the DCA-secret, a PSASS method is employed to create n DCA-shadows. It should be noted that $(k-1)$ out of n DCA-shadows in PSASS are allowed to be modified. The sample-recalculate strategy serves for DCA-shadow re-generation in the scenario that the constructed DCA-shadows are unacceptable. The DC overflow issue can be considerably alleviated with the assistance of PSASS and the sample-recalculate strategy. Herein, we outline the fundamental processes of DCA sharing after presenting the PSASS. The sample-recalculate strategy is given at the end of this sub-section.

*1) **Partial Shadow-Adjustable Secret Sharing**:* Recall that, typical (k, n) SS randomly selects $(k-1)$ coefficients $a_1, \cdots, a_{k-1}$ of a $(k-1)$-degree polynomial to produce n shadows. Because the shadows are randomly distributed, their values cannot be predicted in advance. Consequently, some shadows may fall outside the acceptable range (i.e., be too large or too small), leading to overflow. To mitigate this issue, the PSASS approach is devised. Its core idea is to proactively adjust a subset of the shadows during generation, thereby reducing the likelihood of DC overflow. The following three steps comprise our PSASS.

(i) Build a $(k-1)$-degree polynomial with the constant coefficient being the secret s, as given by

$$f(x) = s + a_1 x + \cdots + a_{k-1} x^{k-1} \pmod{GF(2^M)}. \tag{9}$$

In contrast to traditional SS, the coefficients $a_1, \cdots, a_{k-1}$ are not randomly chosen. Instead, they remain unknown and are determined from the $(k-1)$ shadows given in the next step.

(ii) Select $(k-1)$ values as shadows to calculate the $(k-1)$ coefficients $a_1, \cdots, a_{k-1}$. Without loss of generality, let the chosen $(k-1)$ values be $f(x_{j_1}), \cdots, f(x_{j_{k-1}})$, where the subscripts $j_1, \cdots, j_{k-1}$ represent the corresponding $(k-1)$ participants. Given $x_{j_1}, \cdots, x_{j_{k-1}}$ and $f(x_{j_1}), \cdots, f(x_{j_{k-1}})$, a system of $(k-1)$ equations is formed as

$$\begin{cases} s + a_1 x_{j_1} + \cdots + a_{k-1} x_{j_1}^{k-1} = f(x_{j_1}), \\ s + a_1 x_{j_2} + \cdots + a_{k-1} x_{j_2}^{k-1} = f(x_{j_2}), \\ \quad \vdots \\ s + a_1 x_{j_{k-1}} + \cdots + a_{k-1} x_{j_{k-1}}^{k-1} = f(x_{j_{k-1}}). \end{cases} \tag{10}$$

The $(k-1)$ coefficients $a_1, \cdots, a_{k-1}$ can be computed from the foregoing system of equations.

(iii) As the coefficients $a_1, \cdots, a_{k-1}$ are calculated, a $(k-1)$-degree polynomial $f(x)$ is determined. By employing $(n-k+1)$ values of $x_{i_1}, \cdots, x_{i_{n-k+1}}$, the remaining shadows $f(x_{i_1}), \cdots, f(x_{i_{n-k+1}})$ are created.

To sum up, one can determine the values of $(k-1)$ shadows in advance and then use them to generate the remaining $(n-k+1)$ ones by PSASS.

*2) **Main Process**:* With the aid of PSASS, the DCA sharing technique is presented, as given below.

(i) **DCA-secret generation.** For a JPEG secret image having N DCCs, there exist N DCAs $DCA_1, \cdots, DCA_N$. A length parameter M_D for producing a DCA-secret is first determined, where $M_D \geq \lceil \log_2(n+1) \rceil$ and n is the number of participants. For each time, t consecutive DCAs $DCA_a, \cdots, DCA_{a+t-1}$ are chosen and concatenated together to create a DCA-secret $\overline{DCA}_{a,t}$ by

$$\overline{DCA}_{a,t} = DCA_a || \cdots || DCA_{a+t-1} \tag{11}$$

where $||$ is the bit concatenation. Remember that, the value of t depends on the DCAs and it might vary for each DCA-secret creation. Essentially, those t chosen DCAs $DCA_a, \cdots, DCA_{a+t-1}$ should satisfy

$$L(\overline{DCA}_{a,t-1}) < M_D \leq L(\overline{DCA}_{a,t}) \tag{12}$$

where $\overline{DCA}_{a,t-1} = DCA_a || \cdots || DCA_{a+t-2}$ and $L(\cdot)$ denotes the bit length of the input. It implies that the bit length of t DCAs must be no less than M_D while the bit length of $(t-1)$ DCAs should be smaller than M_D. Additionally, Galois Field $GF(2^{M_t})$ is subsequently employed to generate shadows where $M_t = L(\overline{DCA}_{a,t})$.

(ii) **Boundary distance calculation and priority sorting.** Let $D_{a-1}^{(j)}$ be the j-th shared DC coefficient in location $(a-1)$, as computed by $D_{a-1}^{(j)} = \sum_{b=1}^{a-1} d_b^{(j)}$. The distance from $D_{a-1}^{(j)}$ to the nearest DC boundary (i.e., D_{max} or D_{min}) is calculated by

$$p_{a-1}^{(j)} = \begin{cases} D_{max} - D_{a-1}^{(j)}, & \text{if } D_{a-1}^{(j)} \geq 0, \\ D_{a-1}^{(j)} - D_{min}, & \text{otherwise.} \end{cases} \tag{13}$$

One can use $p_{a-1}^{(j)}$ to estimate whether the j-th shared DC coefficient at location a is easy to overflow or not. Smaller $p_{a-1}^{(j)}$ indicates the current DC coefficient $D_{a-1}^{(j)}$ is closer to the nearest boundary. It is hence more prone to overflow. As a result of this, we sort the values of $p_{a-1}^{(1)}, \cdots, p_{a-1}^{(n)}$ and select the $(k-1)$ smallest ones with the indexes $j_1, \cdots, j_{k-1}$. The shadows with these $(k-1)$ indexes are selected for adjustment in the following step.

(iii) **Determine the values of $(k-1)$ chosen shadows for PSASS.** Let $d_a, \cdots, d_{a+t-1}$ be the t values encoded by t consecutive DCAs $DCA_a, \cdots, DCA_{a+t-1}$ which comprise the DCA-secret $\overline{DCA}_{a,t}$. For any participant $j_h \in \{j_1, \cdots, j_{k-1}\}$ among the $(k-1)$ selected ones, we first determine the shared values $d_a^{(j_h)}, \cdots, d_{a+t-1}^{(j_h)}$ via the steps given in the next paragraph. Then, the corresponding DCAs $DCA_a^{(j_h)}, \cdots, DCA_{a+t-1}^{(j_h)}$ are achieved from $d_a^{(j_h)}, \cdots, d_{a+t-1}^{(j_h)}$ by using the VLI coding. Finally, these t DCAs are concatenated to derive a DCA-shadow $\overline{DCA}_{a,t}^{(j_h)}$, as represented by

$$\overline{DCA}_{a,t}^{(j_h)} = DCA_a^{(j_h)} || \cdots || DCA_{a+t-1}^{(j_h)}. \tag{14}$$

Calculation of $d_a^{(j_h)}, \cdots, d_{a+t-1}^{(j_h)}$. Since the DCHs remain unchanged, the range of each shared value $d_u^{(j_h)}$ $(a \leq u \leq a+t-1)$ must be the same as that of the original d_u. That is, if $d_u \in [-d_{max}^{(u)}, -d_{min}^{(u)}] \cup [d_{min}^{(u)}, d_{max}^{(u)}]$, then $d_u^{(j_h)} \in [-d_{max}^{(u)}, -d_{min}^{(u)}] \cup [d_{min}^{(u)}, d_{max}^{(u)}]$ where $[-d_{max}^{(u)}, -d_{min}^{(u)}] \cup [d_{min}^{(u)}, d_{max}^{(u)}]$ is the range specified by the original DCH. According to DPCM encoding, the t consecutive DC coefficients of the j_h-th DCA-shadow are computed by

$$\begin{cases} D_a^{(j_h)} = D_{a-1}^{(j_h)} + d_a^{(j_h)}, \\ D_{a+1}^{(j_h)} = D_a^{(j_h)} + d_{a+1}^{(j_h)}, \\ \quad \vdots \\ D_{a+t-1}^{(j_h)} = D_{a+t-2}^{(j_h)} + d_{a+t-1}^{(j_h)}. \end{cases} \tag{15}$$

We should choose a $d_u^{(j_h)}$ from the acceptable interval $[-d_{max}^{(u)}, -d_{min}^{(u)}] \cup [d_{min}^{(u)}, d_{max}^{(u)}]$ to ensure that the corresponding shared DC coefficient $D_u^{(j_h)}$ can be far away from the nearest boundary (i.e., D_{max} or D_{min}). Generally, the following criteria are employed.

(1) When $D_{u-1}^{(j_h)} \in [-d_{max}^{(u)}, -d_{min}^{(u)}] \cup [d_{min}^{(u)}, d_{max}^{(u)}]$, we choose $d_u^{(j_h)} = -D_{u-1}^{(j_h)}$ so that $D_u^{(j_h)} = D_{u-1}^{(j_h)} + d_u^{(j_h)} = D_{u-1}^{(j_h)} - D_{u-1}^{(j_h)} = 0$. In this case, $D_u^{(j_h)}$ is far away from both boundaries (i.e., D_{max} and D_{min}).

(2) When $D_{u-1}^{(j_h)} \in (d_{max}^{(u)}, +\infty)$, we ensure $d_u^{(j_h)} = -d_{max}^{(u)}$ to have $D_u^{(j_h)} = D_{u-1}^{(j_h)} + d_u^{(j_h)} = D_{u-1}^{(j_h)} - d_{max}^{(u)}$. In this situation, $D_{u-1}^{(j_h)}$ is reduced to be as far as possible from the boundary D_{max}. Similarly, when $D_{u-1}^{(j_h)} \in (-\infty, -d_{max}^{(i)})$, we have $d_u^{(j_h)} = d_{max}^{(u)}$ so that $D_u^{(j_h)} = D_{u-1}^{(j_h)} + d_u^{(j_h)} = D_{u-1}^{(j_h)} + d_{max}^{(u)}$. $D_{u-1}^{(j_h)}$ is increased to be far away from D_{min}.

(3) When $D_{u-1}^{(j_h)} \in [0, d_{min}^{(u)})$, we make $d_u^{(j_h)} = -d_{min}^{(u)}$ to have $D_u^{(j_h)} = D_{u-1}^{(j_h)} + d_u^{(j_h)} = D_{u-1}^{(j_h)} - d_{min}^{(u)}$. In this situation, $D_{u-1}^{(j_h)}$ is reduced to be far away from D_{max}. By the same way, when $D_{u-1}^{(j_h)} \in (-d_{min}^{(u)}, 0)$, we ensure $d_u^{(j_h)} = d_{min}^{(u)}$ to have $D_u^{(j_h)} = D_{u-1}^{(j_h)} + d_u^{(j_h)} = D_{u-1}^{(j_h)} + d_{min}^{(u)}$. $D_{u-1}^{(j_h)}$ is increased to be far away from D_{min}.

Randomness. To provide randomness, we introduce a perturbation term δ (randomly chosen from $[0, \sigma]$) to $d_u^{(j_h)}$, as represented by $d_u^{(j_h)} \pm \delta$ (i.e., $d_u^{(j_h)} + \delta$ or $d_u^{(j_h)} - \delta$). Note that, the final value of $d_u^{(j_h)}$ with the perturbation term δ should be within the same interval of the initial $d_u^{(j_h)}$, as represented by

$$\begin{cases} d_u^{(j_h)} \pm \delta \in [-d_{max}^{(u)}, -d_{min}^{(u)}], \text{ if } d_u^{(j_h)} \in [-d_{max}^{(u)}, -d_{min}^{(u)}], \\ d_u^{(j_h)} \pm \delta \in [d_{min}^{(u)}, d_{max}^{(u)}], \text{ if } d_u^{(j_h)} \in [d_{min}^{(u)}, d_{max}^{(u)}]. \end{cases} \tag{16}$$

If the range of the final $d_u^{(j_h)}$ is altered by δ, we should randomly re-sample another δ from $[0, \sigma]$ until we have a valid $d_u^{(j_h)}$. In summary, the rules for determining the value of $d_u^{(j_h)}$ are formulated as

$$d_u^{(j_h)} = \begin{cases} -D_{u-1}^{(j_h)} - \delta, & \text{if } D_{u-1}^{(j_h)} \in [-d_{max}^{(u)}, -d_{min}^{(u)}], \\ -D_{u-1}^{(j_h)} + \delta, & \text{if } D_{u-1}^{(j_h)} \in [d_{min}^{(u)}, d_{max}^{(u)}], \\ -d_{max}^{(u)} + \delta, & \text{if } D_{u-1}^{(j_h)} \in (d_{max}^{(u)}, +\infty), \\ d_{max}^{(u)} - \delta, & \text{if } D_{u-1}^{(j_h)} \in (-\infty, -d_{max}^{(u)}), \\ -d_{min}^{(u)} - \delta, & \text{if } D_{u-1}^{(j_h)} \in [0, d_{min}^{(u)}), \\ d_{min}^{(u)} + \delta, & \text{if } D_{u-1}^{(j_h)} \in (-d_{min}^{(u)}, 0). \end{cases} \tag{17}$$

(iv) **Generate the remaining $(n-k+1)$ shadows by PSASS.** Based on the $(k-1)$ DCA-shadows determined in previous step, PSASS is utilized to generate the remaining $(n-k+1)$ DCA-shadows. A variable η_i is further employed

to indicate whether the $(n-k+1)$ generated ones are suitable or not. When $\eta_i \leq k-1$, the $(n-k+1)$ DCA-shadows are accepted. Otherwise, they are marked as inappropriate, and the sample-recalculate strategy (given in the subsequent subsection) is utilized to re-produce all shadows. When all DCAs are processed, n DCA-whole-shadow are provided. Example 2 demonstrates the use of DCA sharing for producing valid shared DC coefficients.

Example 2. *Consider the DCA sharing of* $(2,3)$ *scheme for dealing with* $DCA_1 = 41$ *and* $DCA_2 = 120$ *(DC differences are* $d_1 = 41$ *and* $d_2 = -135$*).*

We use the same polynomial $f_1(x)$ *and IDs given in Example 1 for sharing* DCA_1*, where* 3 *generated DCA-shadows are* $DCA_1^{(1)} = 35$*,* $DCA_1^{(2)} = 61$*, and* $DCA_1^{(3)} = 55$*. The corresponding DC differences are* $d_1^{(1)} = 35$*,* $d_1^{(2)} = 61$*, and* $d_1^{(3)} = 55$*. The boundary distances of the* 1*-st shared DC coefficients belonging to* 3 *participants are computed by*

$$p_1^{(1)} = 168, p_1^{(2)} = 142, p_1^{(3)} = 148. \tag{18}$$

Since $p_1^{(2)} = 142$ *is the smallest item, the* 2*-nd DCA-shadow of* DCA_2 *is required to be adjusted. In this case, we can modify* $DCA_2^{(2)}$ *to* 122 *(i.e.,* $d_2^{(2)} = -133$*) so that the shared DC coefficient for the* 2*-nd participant is* $D_2^{(2)} = d_1^{(2)} + d_2^{(2)} = -72$*, which is within the valid range* $[-204, 203]$*.*

On the other hand, for $f_2(x) = DCA_2 + a_1 x \pmod{GF(2^8)}$*, one can calculate the coefficient* $a_1 = 1$ *of* $f_2(x)$ *by using Lagrange interpolation with* $DCA_2^{(2)} = DCA_2 + a_1 x_1 \pmod{GF(2^8)}$*. With the determined* $f_2(x)$*, we compute the remaining* 2 *DCA-shadows by* $DCA_2^{(1)} = f_2(x_1) = 121$ *and* $DCA_2^{(3)} = f_2(x_3) = 123$ *(i.e.,* $d_2^{(1)} = -134$ *and* $d_2^{(3)} = -132$*). Consequently, the two shared DC coefficients are achieved by* $D_2^{(1)} = d_1^{(1)} + d_2^{(1)} = -99$ *and* $D_2^{(3)} = d_1^{(3)} + d_2^{(3)} = -77$*, which are acceptable.*

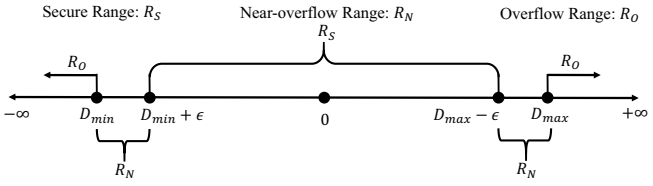

Fig. 4. Three ranges used for calculating η_i.

Calculation of η_i. A parameter ϵ is used to partition the DC valid range $[D_{min}, D_{max}]$ into two intervals: the secure range $\mathcal{R}_S$ and the near-overflow range $\mathcal{R}_N$:

$$\begin{cases} \mathcal{R}_S = (D_{min}+\epsilon, D_{max}-\epsilon), \\ \mathcal{R}_N = [D_{min}, D_{min}+\epsilon] \cup [D_{max}-\epsilon, D_{max}]. \end{cases} \tag{19}$$

Additionally, the overflow range $\mathcal{R}_O$ is defined as

$$\mathcal{R}_O = (-\infty, D_{min}) \cup (D_{max}, +\infty). \tag{20}$$

Fig. 4 summarizes these three types of intervals. Hence, η_i is computed by

$$\eta_i = \eta_i^{(1)} + \eta_i^{(2)} + \cdots + \eta_i^{(n)} \tag{21}$$

where $\eta_i^{(j)}$ corresponds to the j-th $(1 \leq j \leq n)$ shared DC coefficient at the i-th location and can be determined by

$$\eta_i^{(j)} = \begin{cases} 0, \text{ if } D_i^{(j)} \in \mathcal{R}_S, \\ 1, \text{ if } D_i^{(j)} \in \mathcal{R}_N, \\ k, \text{ if } D_i^{(j)} \in \mathcal{R}_O. \end{cases} \tag{22}$$

When $\eta_i = 0$, all shared DC coefficients at location i are inside the secure range $\mathcal{R}_S$ and the shared DC coefficients at location $(i+1)$ are not easy to overflow. When $0 < \eta_i \leq k-1$, we have less than k coefficients at location i within the near-overflow range $\mathcal{R}_N$, the shared values at location $(i+1)$ are possible to overflow. However, we can adjust the corresponding DCA-shadows at location $(i+1)$ to avoid overflow. For these two cases, the DCA-shadows are accepted. When $\eta_i \geq k$, we might have more than $(k-1)$ DC coefficients within the near-overflow range $\mathcal{R}_N$ or at least 1 DC coefficient within the overflow range $\mathcal{R}_O$. In this situation, we should re-generate all DCA-shadows.

3) ***Sample-Recalculate Strategy****:* Step (iii) of DCA sharing ensures that the shared DC coefficients decoded from $(k-1)$ chosen DCA-shadows are inside the acceptable range. The values of remaining $(n-k+1)$ DCA-shadows produced by PSASS, however, are beyond the control. DC overflow would ultimately result from these inappropriate data. In this scenario, the sample-recalculate strategy is employed to re-modify the $(k-1)$ selected DCA-shadows and later re-produce the remaining $(n-k+1)$ ones.

The two stages listed below are used to implement the sample-recalculate technique. (i) Sample different perturbation terms to derive $(k-1)$ different DCA-shadows. (ii) Utilize the pre-determined DCA-shadows to re-generate the remaining ones by PSASS. The two-step procedure is repeated until the re-generated data are appropriate.

Generally, we have $(k-1)$ DCA-shadows to be determined for PSASS. For each shadow, it consists of t components $d_a^{(j_h)}$, $\cdots$, $d_{a+t-1}^{(j_h)}$. Each component is correlated with a perturbation term. We can use a binary configuration matrix $M = [m_{i,j}]$ with $1 \leq i \leq k-1$, $1 \leq j \leq t$ to indicate which perturbation terms are re-sampled. When $m_{i,j} = 1$ (resp. 0), the j-th perturbation term of the i-th DCA-shadow is re-sampled (resp. not changed). In total, there exist $2^{(k-1)t}$ different configuration matrices. It implies we might have at least $2^{(k-1)t}$ ways to re-generate a group of $(k-1)$ pre-determined DCA-shadows if the remaining ones derived from PSASS are unsuitable. As a result, it is feasible for the sample-recalculate strategy to re-generate appropriate DCA-shadows.

C. Post-Processing: Fully Preventing DC Overflow

It is still possible to have overflow DC coefficients even though PSASS and sample-recalculate technique can effectively reduce the amount of overflow data. To fully solve this problem, a post-processing is employed. DC shifting developed in [14] and reversible data hiding for encrypted JPEG bitstreams [40] are performed to ensure that all resulting shadows remain within the allowable range.

First of all, for any DC difference $d_i^{(j)}$, $1 \leq i \leq N$, $1 \leq j \leq n$, decoded from the shared DCA, calculate the corresponding

DC coefficient $D_i^{(j)}$. If $D_i^{(j)}$ is overflow, it is shifted to a new value $\tilde{D}_i^{(j)}$ by

$$\tilde{D}_i^{(j)} = \begin{cases} D_i^{(j)} + (D_{min} - D_{max} - 1), \text{ if } D_i^{(j)} > D_{max}, \\ D_i^{(j)} + (D_{max} - D_{min} + 1), \text{ if } D_i^{(j)} < D_{min}. \end{cases} \tag{23}$$

Otherwise, $\tilde{D}_i^{(j)} = D_i^{(j)}$. Consequently, a new shared DC difference $\tilde{d}_i^{(j)}$ is determined as

$$\tilde{d}_i^{(j)} = \tilde{D}_i^{(j)} - D_{i-1}^{(j)} \tag{24}$$

where $D_{i-1}^{(j)}$ is the DC coefficient at location $(i-1)$. Based on the value of $\tilde{d}_i^{(j)}$, derive the corresponding DCH for $\tilde{d}_i^{(j)}$. To ensure that the original shared DC difference and DCH can be correctly recovered, a binary location map $LM^{(j)} = \{l_i^{(j)}\}_{i=1}^{N}$ is employed to record the overflow position, as computed by

$$l_i^{(j)} = \begin{cases} 1, \text{ if } D_i^{(j)} > D_{max} \text{ or } D_i^{(j)} < D_{min}, \\ 0, \text{ otherwise.} \end{cases} \tag{25}$$

In the next step, $LM^{(j)}$, $1 \leq j \leq n$, is compressed and then reversibly concealed into the j-th ACC-whole-shadow via the reversible data hiding technique [40]. Consequently, n marked ACC-whole-shadows are produced. Note that, the ACC-whole-shadows are generated by the proposed AC secret sharing depicted in the next section. They should be prepared prior to the post-processing.

When recovering the shared DC data, the location map $LM^{(j)}$ is extracted from the marked AC data. As reversible data hiding method is adopted, the shared AC data would be restored to the original form after data extraction. With $LM^{(j)}$, the original shared DC difference $d_i^{(j)}$ is determined as

$$d_i^{(j)} = \begin{cases} -1 \times \text{sign}(\tilde{D}_i^{(j)}) \times (\bar{D} - |\tilde{D}_i^{(j)}|), \text{ if } l_i^{(j)} = 1, \\ \tilde{D}_i^{(j)}, \text{ otherwise,} \end{cases} \tag{26}$$

where $\bar{D} = D_{max} - D_{min} + 1$.

V. AC Secret Sharing

In this section, the main problem for designing AC secret sharing is analyzed. The details of AC secret sharing, including adaptive key generation, AC permutation, and ACA sharing, are then presented. The adaptive key generation offers a secret key for the AC permutation. Further, the AC permutation, consisting of AC cross-segment permutation and global block permutation, is employed to scramble the AC coefficients within certain blocks and then globally rearrange the positions of all blocks excluding DC coefficients. Secure scrambled ACCs are enabled. The ACAs (i.e., amplitudes) of the permuted ACCs are then encrypted by the ACA sharing to produce n ACA-whole-shadows. By combining with the original ACHs, n ACC-whole-shadows are established.

A. Key Issue of AC Secret Sharing

Previous approach [18] merely changes the values of secret ACAs in each AC block. The generated shadows would probably leak the secret. Let A_i be the i-th AC block having 63 coefficients $\{a_{i,1}, \cdots, a_{i,63}\}$. According to the investigations in [46] [14], the following four types of AC block features should be carefully considered.

- **NCC**: the non-zero coefficient count of an AC block, as computed by

$$\mathcal{F}_{\mathcal{N}}(A_i) = \sum_{j=1}^{63} \mathcal{NZ}(a_{i,j}) \tag{27}$$

where

$$\mathcal{NZ}(a_{i,j}) = \begin{cases} 1, \text{ if } a_{i,j} \neq 0, \\ 0, \text{ otherwise.} \end{cases} \tag{28}$$

- **PLZ**: the position of the last non-zero coefficient of an AC block, as given by

$$\mathcal{F}_{\mathcal{P}}(A_i) = \begin{cases} j, \text{ if } a_{i,j} \neq 0, \ a_{i,j+1} = \cdots = a_{i,63} = 0, \\ 0, \text{ otherwise.} \end{cases} \tag{29}$$

- **EAC**: the energy of non-zero coefficients of an AC block, as calculated by

$$\mathcal{F}_{\mathcal{E}}(A_i) = \sum_{j=1}^{63} |a_{i,j}|. \tag{30}$$

- **RLS**: the run length sequence of an AC block, as provided by

$$\mathcal{F}_{\mathcal{R}}(A_i) = \mathcal{N}(R_i) || Sort(R_i) \tag{31}$$

where $R_i = \{r_{i,1}, \cdots, r_{i,t}\}$ is the collection of run-lengths of A_i, $\mathcal{N}(R_i)$ computes the number of zero run-lengths of R_i by

$$\mathcal{N}(R_i) = \sum_{j=1}^{t} \overline{\mathcal{NZ}}(r_{i,j}), \tag{32}$$

with

$$\overline{\mathcal{NZ}}(r_{i,j}) = \begin{cases} 1, \text{ if } r_{i,j} = 0, \\ 0, \text{ otherwise.} \end{cases} \tag{33}$$

and $Sort(R_i)$ sorts the non-zero run-lengths of R_i by

$$Sort(R_i) = \{r_{i,j_1}, \cdots, r_{i,j_h}\} \tag{34}$$

with $r_{i,j_1} \leq \cdots \leq r_{i,j_h}$ and $r_{i,j_1} \neq 0, \cdots, r_{i,j_h} \neq 0$.

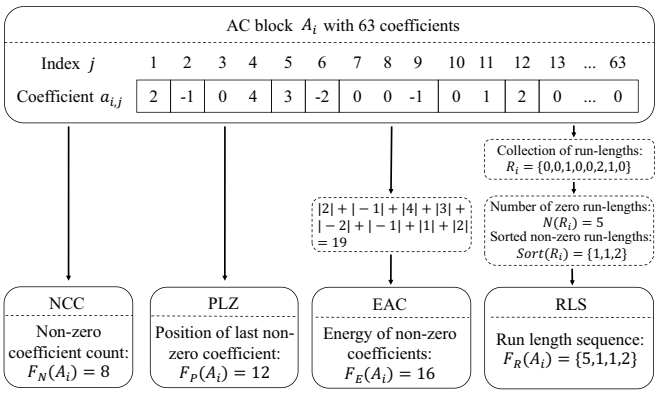

Fig. 5. Four types of AC block features.

Example 3 is presented to comprehend the concept of AC block feature. For security consideration, the AC block

features of each JPEG shadow should dramatically differ from those of the secret. To accomplish this objective, the AC secret sharing is introduced.

Example 3. *Consider the four AC block features in Fig. 5. The quantity of non-zero coefficients is* 8*. We have the NCC feature* $\mathcal{F}_{\mathcal{N}}(A_i) = 8$*. The location of last non-zero item is* 12*. Hence, the PLZ feature is* $\mathcal{F}_{\mathcal{P}}(A_i) = 12$*. The EAC feature can be evaluated by* $\mathcal{F}_{\mathcal{E}}(A_i) = |2| + |-1| + |4| + |3| + |-2| + |-1| + |1| + |2| = 16$*. The collection of run-lengths of this block is* $R_i = \{0, 0, 1, 0, 0, 2, 1, 0\}$*. Consequently, the number of zero run-lengths of* R_i *is* $\mathcal{N}(R_i) = 5$ *and the collection of sorted non-zero run-lengths is* $Sort(R_i) = \{1, 1, 2\}$*. By concatenating* $\mathcal{N}(R_i)$ *with* $Sort(R_i)$*, the RLS feature is determined as* $\mathcal{F}_{\mathcal{R}}(A_i) = \mathcal{N}(R_i)||Sort(R_i) = \{5, 1, 1, 2\}$.

B. Adaptive Key Generation

The secret key for AC permutation is generated by

$$K_A = Hash(Sort(D_1, \cdots, D_N)) \tag{35}$$

where $D_1, \cdots, D_N$ are the N DC coefficients of a JPEG secret image, $Sort(\cdot)$ is a procedure that sorts all input values in an ascending order and concatenates the sorted results together, and $Hash(\cdot)$ represents a Hash function.

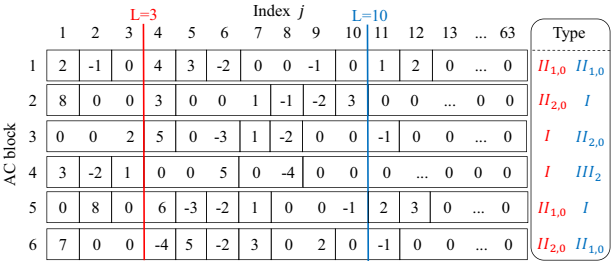

Fig. 6. Different types of AC blocks by using $L = 3$ and 10.

C. AC Permutation

The AC permutation, comprising AC cross-segment permutation and global block permutation excluding DC coefficients, is adopted to randomly scramble the ACCs. The AC cross-segment permutation is responsible for disordering the AC coefficients within specified blocks, while the global block permutation reorders the positions of all blocks, ignoring DC coefficients.

1) ***AC Cross-Segment Permutation****:* The AC cross-segment permutation is accomplished by (i) block separation and classification, and (ii) segment permutation and concatenation, as described below.

(i) Block separation and classification. By utilizing a length parameter L, an AC block is broken into the following 2 segments.

- **Head segment**: having the first L coefficients of this AC block.
- **Tail segment**: containing the remaining coefficients located from position $(L + 1)$ to 63.

Based on the characteristic of head and tail segments, every AC block is classified into the following 3 categories.

- **Type** I**.** The last coefficient of head segment is non-zero.
- **Type** II**.** The last h_1 ($h_1 > 0$) coefficients of head segment are zero and the one preceded these zero items is non-zero. The first h_2 ($h_2 \geq 0$) coefficients of tail segment are zero and the one followed these zero items is non-zero.
- **Type** III**.** The last h_3 ($h_3 > 0$) coefficients of head segment are zero and the one preceded these zero items is non-zero. All coefficients in tail segment are zero.

Note that, type II can be further separated into various sub-types with different values of h_1 and h_2. The corresponding sub-type is represented as II_{h_1,h_2}. The same argument also holds for type III by using different h_3, where the sub-type is denoted as III_{h_3}. Example 4 provides instances of 6 AC blocks for the classification where $L = 3$ and 10.

Example 4. *Consider the AC block classification by using* $L = 3$ *and* 10*, as illustrated in Fig. 6. When* $L = 3$*, the* 3*-rd and* 4*-th AC blocks are classified to type* I*, and the* 4 *remaining blocks are denoted as type* II*. As the last coefficient of the head segment is zero and the first coefficient of the tail segment is non-zero, the* 1*-st block is denoted as type* $II_{1,0}$*. By using the same approach, the* 2*-nd,* 5*-th, and* 6*-th blocks are classified to types* $II_{2,0}$*,* $II_{1,0}$*, and* $II_{2,0}$*, respectively. Moreover, if* $L = 10$*, these* 6 *AC blocks are divided into types* $II_{1,0}$*,* I*,* $II_{2,0}$*,* III_1*,* I*, and* $II_{1,0}$.

More importantly, when $h_1 = h_3$, we merge a sub-type II_{h_1,h_2} with the fewest elements and a sub-type III_{h_1} together to obtain a hybrid sub-type $II_{h_1,h_2}\&III_{h_1}$. Such an merging operation aims to increase the quantity of elements such that better permutation performance is provided.

(ii) Segment permutation and concatenation. All head segments in the same type (i.e., I), or the same sub-type (i.e., II_{h_1,h_2} or III_{h_3}), or the same hybrid sub-type (i.e., $II_{h_1,h_2}\&III_{h_1}$) are randomly permuted according to a secret key K_A, as well as the tail segments. Each permuted head segment is then concatenated with a corresponding scrambled tail segment to produce a new AC block.

Iterative execution. One can execute the AC cross-segment permutation depending on the value of L. However, conducting the permutation for only one time might not alter the block features significantly. As a result of this, iterative execution is considered to enhance the performance. In this scenario, the values of L from 1 to $\lfloor L_{Max}/2 \rfloor$ are employed to carry out the permutation iteratively, where L_{Max} is the maximum value of PLZ feature among all AC blocks. Example 5 is demonstrated to capture the concept of AC cross-segment permutation.

Example 5. *Consider a* 2*-time execution of the AC cross-segment permutation with* $L = 3$ *and* 10*, as depicted in Fig. 7. For the first time with* $L = 3$*, we have* 3 *different types* I*,* $II_{1,0}$ *and* $II_{2,0}$*. Each type has* 2 *AC blocks. By permuting both the head and tail segments, the corresponding segments are concatenated together to form* 6 *new AC blocks. For the second time, we employ* $L = 10$*. Consequently,* 4 *block types* I*,* $II_{1,0}$*,* $II_{2,0}$*, and* III_2 *are provided. Remember that,* $II_{2,0}$

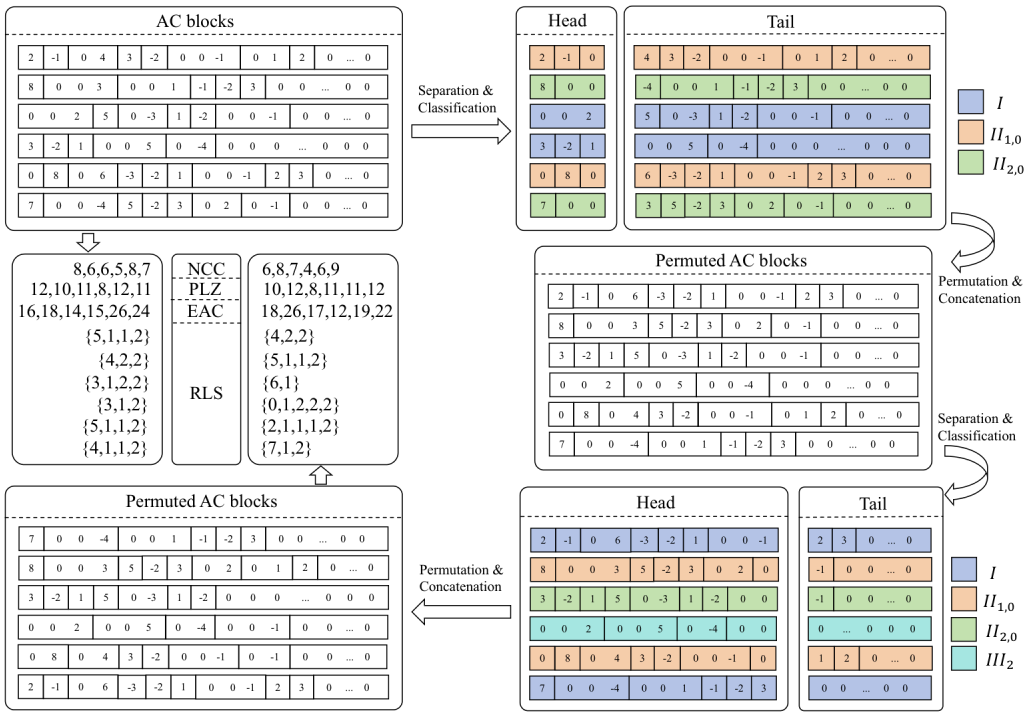

Fig. 7. A 2-time execution of the proposed AC cross-segment permutation with $L = 3$ and 10.

is merged with III_2 to produce a hybrid category $II_{2,0}\&III_2$. According to the results in Fig. 7, one can observe that the AC block features (i.e., NCC, PLZ, EAC, and RLS) are changed substantially.

*2) **Global Block Permutation Excluding DC Coefficients**:* After the iterative cross-segment permutation, all AC blocks are then globally scrambled based on the secret key K_A. It is worth noting that the DC coefficients are remained the same. Only the AC blocks are disordered.

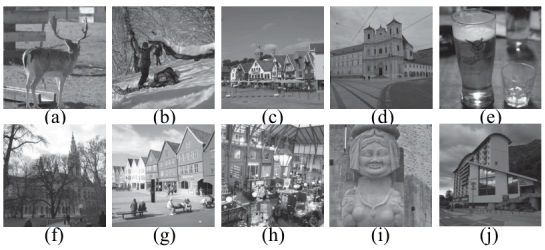

Fig. 8. 10 JPEG test images converted from BOSSbase. Image IDs: (a) 8, (b) 101, (c) 261, (d) 437, (e) 646, (f) 999, (g) 1130, (h) 2461, (i) 3197, (j) 5253.

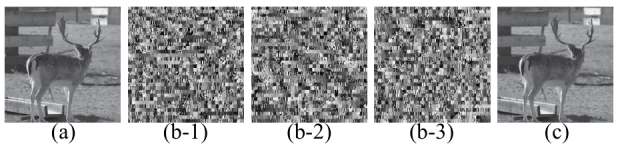

Fig. 9. A $(3, 3)$ experiment by the proposed scheme. (a) a JPEG secret image, (b) 3 JPEG shadows, (c) the decrypted JPEG image.

D. ACA Sharing

The ACA sharing is employed to split the ACAs into n ACA-whole-shadows via the (k, n) SS. First of all, extract all ACAs from the permuted ACCs and concatenate them together to obtain an ACA sequence. For each time, M_A ($\geq \lceil \log_2(n+1) \rceil$) bits, fetching from the ACA sequence, are considered as the ACA-secret, i.e., $\overline{ACA}$. Then, construct a $(k-1)$-degree polynomial and substitute the constant term of this polynomial by the ACA-secret, as represented by

$$f(x) = \overline{ACA} + a_1 x + \cdots + a_{k-1} x^{k-1} \pmod{GF(2^{M_A})} \tag{36}$$

where $a_1, \cdots, a_{k-1}$ are random numbers. By using different values of $x_1, \cdots, x_n$, n ACA-shadows $f(x_1), \cdots, f(x_n)$ are created. By concatenating all ACA-shadows together, n shared ACA sequences are achieved. One can extract the corresponding number of bits (i.e., shared ACA value) from each of the n shared sequences by referring to every ACH value. Finally, n ACC-whole-shadows are established by combining the shared ACA values with the original ACHs.

VI. Experimental Result and Discussion

A comprehensive set of experiments is conducted to demonstrate the efficacy and superiority of the proposed scheme. The test set comprises 10 images randomly chosen from the BOSSbase image database and converted to JPEG format, as shown in Fig. 8. In terms of implementation, all experiments are performed using Python 3.9 in the following environment: an Intel(R) Core(TM) i5-10300H processor, 16 GB RAM, and Windows 11. The performance of the proposed technique is evaluated from multiple perspectives: perceptual result, visual

security, PSNR and SSIM of shadow, brightness histogram of shadow, AC block feature, and DC overflow.

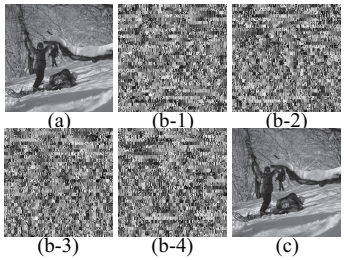

Fig. 10. A $(3, 4)$ experiment by the proposed scheme. (a) a JPEG secret image, (b) 4 JPEG shadows, (c) the decrypted JPEG image.

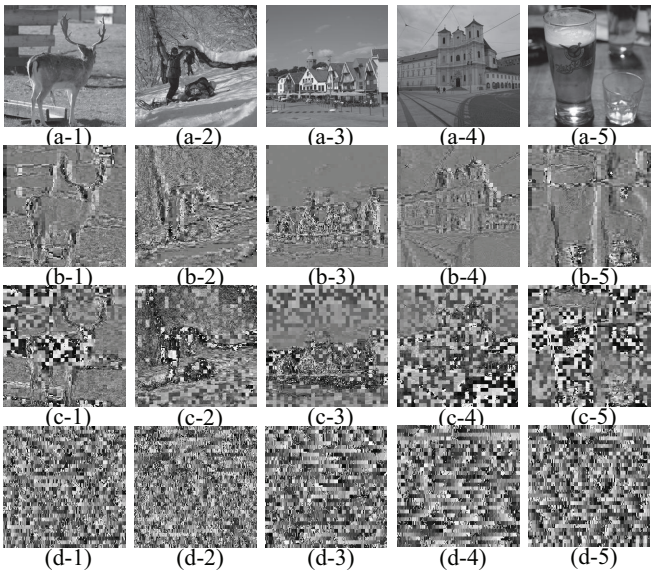

Fig. 11. Visual comparison of JPEG shadows. (a) JPEG secret images, JPEG shadows of Puteaux *et al.*'s method [18] by sharing (b) only AC, (c) both DC and AC, (d) JPEG shadows of the proposed scheme.

TABLE II
COMPARISON OF PSNR AND SSIM OF SHADOWS

Image ID	SSIM			PSNR		
	Ref. [18]		Our	Ref. [18]		Our
	AC	DC+AC		AC	DC+AC	
8	0.220	0.134	0.014	12.864	10.928	8.874
101	0.155	0.126	0.006	12.352	11.906	8.654
261	0.498	0.235	0.019	13.688	12.901	9.206
437	0.335	0.132	0.025	12.627	10.028	8.698
646	0.252	0.116	0.018	11.046	8.699	7.918
999	0.194	0.107	0.007	8.814	7.234	6.771
1130	0.269	0.182	0.009	11.235	9.933	8.249
2461	0.191	0.188	0.006	10.959	10.680	8.157
3197	0.213	0.124	0.014	15.739	13.865	9.613
5253	0.269	0.133	0.017	10.769	8.859	7.821
Average	0.260	0.148	0.014	12.009	10.503	8.396

A. Perceptual Result

A $(3, 3)$ experiment by the proposed scheme is given in Fig. 9 where the JPEG secret image and 3 generated JPEG shadows are provided in Figs. 9(a) and (b), respectively. Fig. 9(c) shows the reconstructed result by 3 shadows. Another experiment of $(3, 4)$ scheme is depicted in Fig. 10. Herein, the JPEG secret image and 4 JPEG shadows are illustrated in Figs. 10(a) and (b), respectively. By using any 3 out of 4 shadows, the secret can be recovered, as given in Fig. 10(c). The foregoing experiments demonstrate that the proposed method is effective for dealing with JPEG images. It also achieves a dramatic distortion degree of shadow images, offering visual security. Additionally, these generated shadows typically have a slight block effect, which is worth mentioning. The primary reason for this is that the proposed method actually considers the DCT block as the basic unit to implement sharing strategy. The resulting shadow is therefore unlikely to be the one produced by pixel-level sharing techniques [17], [29], [34].

B. Comparison of Visual Security

A visual comparison of shadows between our method and Puteaux *et al.*'s approach [18] is highlighted in Fig. 11 where 5 JPEG test images are employed. The shadows generated by Puteaux *et al.*'s technique [18] are shown in Figs. 11(b) and (c). Specifically, the shadows obtained by encrypting only the AC components are illustrated in Fig. 11(b) while the results generated by encoding both DC and AC coefficients are given in Fig. 11(c). The shadows by our approach are provided in Fig. 11(d). One can observe from the results that the shadows of Puteaux *et al.*'s scheme [18] are not secure. The secret information would be displayed on shadows especially when only the AC coefficients are encrypted. Nevertheless, visual security on shadows is guaranteed by our technique.

C. Quantitative Evaluation and Discussion

1) ***PSNR and SSIM of Shadows****:* Two quantitative measurements, PSNR and SSIM, are utilized to evaluate the security performance. Both PSNR and SSIM of a shadow are expected to be as small as possible so that the generated shadow leaves no trace about the secret. Comparison of PSNR and SSIM of shadows between Puteaux *et al.*'s approach [18] and our technique is emphasized in Table II where 10 JPEG test images are employed. Herein, the $(3, 4)$ threshold is implemented. According to Table II, the PSNR of a shadow by using our scheme is lower than those by using Puteaux *et al.*'s method [18]. In particular, the SSIM of a shadow from the proposed method is substantially smaller than those from Puteaux *et al.*'s scheme [18]. As the SSIM by our scheme is approximately 0, a shadow cannot expose the secret structural information. Our technique ensures a greater degree of distortion of a shadow.

2) ***Brightness Histogram of Shadows****:* Brightness histogram reflects the pixel distribution of a JPEG image. If the brightness histogram of a shadow is as flat as possible, it is difficult to correctly restore the secret content by analyzing the histogram statistics. Fig. 12 demonstrates the brightness histograms of 5 JPEG secret images and the corresponding shadows by using Puteaux *et al.*'s technique [18] and the proposed scheme. According to Fig. 12, our approach produces a far more equal and flat pixel distribution of shadows than

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2026.3714465

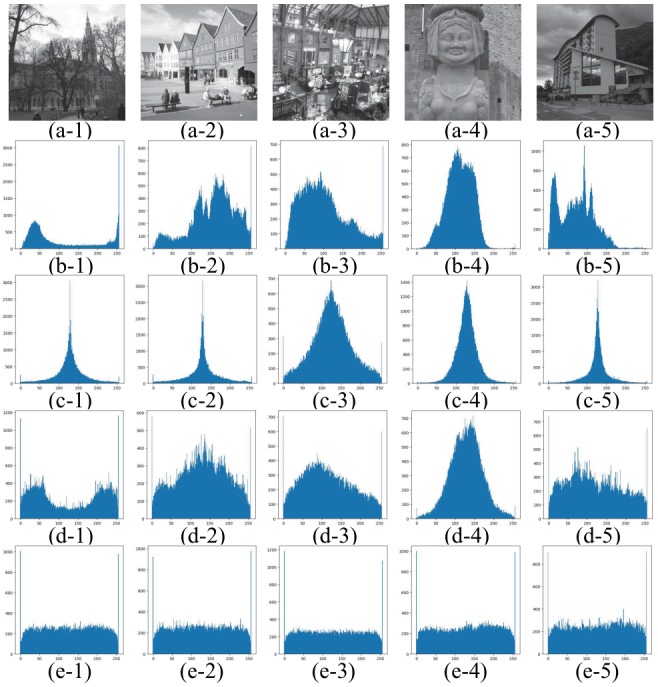

Fig. 12. Comparison of brightness histogram. (a) 5 JPEG secret images, brightness histograms of (b) 5 JPEG secret images, (c) 5 JPEG shadows by Puteaux *et al.*'s technique [18] for encoding AC coefficients, (d) 5 JPEG shadows by Puteaux *et al.*'s technique [18] for encoding DC and AC coefficients, (e) 5 JPEG shadows by the proposed method.

Puteaux et al.'s methodology [18]. The proposed technique can effectively increase the randomness of shared JPEG data.

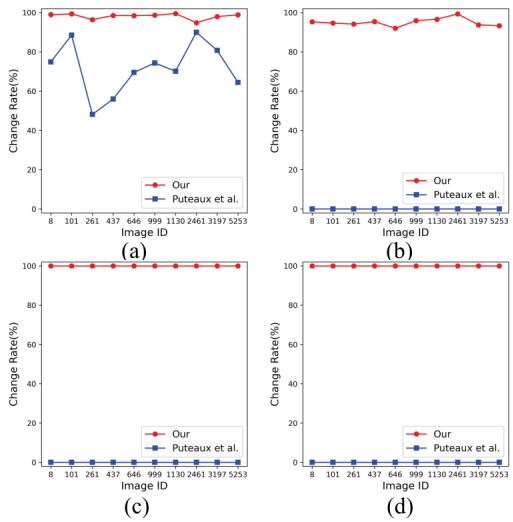

Fig. 13. Comparison of change rate of AC block feature belonging to a shadow. (a) EAC, (b) NCC, (c) PLZ, (d) RLS.

*3) **Change Rate of AC Block Feature**:* The AC block features capture the statistical properties of the secret image. Therefore, to ensure the security of the generated JPEG shadows, these features should be modified as extensively as possible. Herein, the change rate of AC block feature is employed, as calculated by

$$\nabla_{\mathcal{X}} = \frac{\sum_{i=1}^{N} \delta(\mathcal{F}_{\mathcal{X}}(\overline{A}_i), \mathcal{F}_{\mathcal{X}}(A_i))}{N} \times 100\% \tag{37}$$

where $\overline{A}_i$ and A_i are the AC blocks corresponding to a JPEG secret image and a JPEG shadow, respectively, N is the total number of blocks, $\mathcal{F}_{\mathcal{X}}$ denotes the AC block feature (i.e., EAC, NCC, PLZ, or RLS), and procedure $\delta(\mathcal{F}_{\mathcal{X}}(\overline{A}_i), \mathcal{F}_{\mathcal{X}}(A_i))$ is

$$\delta(\mathcal{F}_{\mathcal{X}}(\overline{A}_i), \mathcal{F}_{\mathcal{X}}(A_i)) = \begin{cases} 1, \text{if } \mathcal{F}_{\mathcal{X}}(\overline{A}_i) \neq \mathcal{F}_{\mathcal{X}}(A_i), \\ 0, \text{otherwise.} \end{cases} \tag{38}$$

A high change rate indicates the AC block feature of a shadow has undergone significant modification. Comparison of change rate of AC block feature is illustrated in Fig. 13, where 10 JPEG secret images are used. By referring to the results, the proposed scheme offers near-perfect change rates (i.e., close to 100%). The AC block features of the generated shadows are entirely different from those of the secret image. In contrast, the results by using Puteaux *et al.*'s technique [18] are unsatisfactory. Particularly for the NCC, PLZ, and RLS features, the corresponding change rates are 0. It implies that the AC block features of a shadow are identical to those of the secret image. As a result of this, the generated shadows might unveil the secret. In summary, our scheme provides superior performance over previous scheme [18] in terms of change rate of AC block feature.

TABLE III
COMPARISON OF DC OVERFLOW

Image ID	Total blocks	Num of overflow blocks		Overflow rate	
		Ref. [18]	Our	Ref. [18]	Our
8	1024	1002	0	0.9785	0
101	1024	1022	0	0.9980	0
261	1024	1011	0	0.9873	0
437	1024	983	0	0.9600	0
646	1024	1017	0	0.9932	0
999	1024	989	0	0.9658	0
1130	1024	999	0	0.9756	0
2461	1024	987	0	0.9639	0
3197	1024	1020	0	0.9961	0
5253	1024	973	0	0.9502	0
Average	1024	1000.3	0	0.9769	0

*4) **DC Overflow**:* Table III depicts the comparison of number of overflow DC blocks, as well as the overflow rate. According to Table III, Puteaux *et al.*'s technique [18] does not consider the DC overflow issue and produces a large number of overflow coefficients. The corresponding overflow rates are above 95%. Nearly all shared DC coefficients fall outside the valid range. On the other hand, the proposed method successfully eliminates overflow, as both the number of overflow DC coefficients and the overflow rate are reduced to 0. Our scheme is a format compatible approach which generates acceptable shared DC data.

*5) **Feature Comparison**:* Comparison of functionality among related approaches is illustrated in Table IV. The proposed method achieves the following merits.

- **Improved security.** Secret leakage is a vulnerability in [18]. The confidential information would be revealed on shadows. Evaluations based on PSNR, SSIM, brightness

TABLE IV
FEATURE COMPARISON AMONG RELATED APPROACHES

Scheme	Feature					
	Key Method	Type of Secret	Secret Leakage	DC Overflow	Decoding Operation	Lossless Recovery
Ref. [19]	LP, BM	Uncompressed, B	No	-	XOR-ing	No
Ref. [27]	SA, BM	Uncompressed, B	No	-	Stacking, XOR-ing	No
Ref. [17]	LI, BM	Uncompressed, G/C	No	-	Modular Arithmetic	Yes
Ref. [34]	LI, SM	Uncompressed, G/C	No	-	Modular Arithmetic	Yes
Ref. [18]	LI	JPEG, G/C	Yes	Yes	Modular Arithmetic	Yes
Our	LI, PSASS, SR	JPEG, G/C	No	No	Modular Arithmetic	Yes

LP: Linear Programming, BM: Base Matrix, SA: Simulated Annealing, LI: Lagrange Interpolation, PSASS: Partial Shadow-Adjustable Secret Sharing SM: Singular Matrix, SR: Sample-Recalculate Stragety, B: Binary Image, G/C: Gray-level/Color Image

histogram, and AC block feature change rate confirm that the proposed scheme produces shadows that are significantly more random and chaotic, providing no discernible information about the original secret.

- **Format compatible with non-overflow DC coefficients.** A large number of DC coefficients would overflow in Puteaux *et al.*'s scheme [18], which eventually leads to format incompatibility. However, the DC overflow problem is solved by our approach, ensuring that all shared DC coefficients fall inside the acceptable interval.
- **JPEG-aware design.** JPEG images are not taken into account by the majority of current techniques [17], [34]. The application scenario is further limited by the inability of these approaches to handle JPEG images. However, the scheme proposed in this paper can tackle JPEG images.

VII. CONCLUSION

A secret JPEG image sharing technique was investigated in this paper. In general, DC secret sharing and AC secret sharing were designed as the two main components. In order to encrypt the DC components of a JPEG secret image, we employed DCA sharing in conjunction with post-processing. To deal with the AC coefficients, AC secret sharing, which includes adaptive key generation, AC permutation, and ACA sharing, was implemented. Experimental results and comparative analyses were conducted to show the effectiveness of the proposed method, as well as its benefits such as JPEG-aware design, improved security, and format compatibility.

Future research may extend in the following directions. One is to overcome the current limitation to a fixed (k, n) threshold through an evolving access structure, which would allow unlimited participants and flexible sharing policies. Another direction is to integrate generative coverless steganography with secret sharing. By generating human-readable shadows, suspicion from malicious parties during transmission and storage can be avoided, while also facilitating data management.

REFERENCES

[1] C. Yu, X. Zhang, G. Li, P. Liu, X. Zhang, and Z. Tang, "Reversible data hiding in encrypted images with secret sharing and multivariate linear equation," *IEEE Transactions on Dependable and Secure Computing*, vol. 22, no. 6, pp. 6767–6780, 2025.

[2] S. Xu, G. Yang, X. Li, X. Han, X. Yan, and X. Huang, "Enhancing secure cloud data sharing: Dynamic user groups and outsourced decryption," *IEEE Transactions on Dependable and Secure Computing*, vol. 22, no. 6, pp. 7971–7984, 2025.

[3] J. Yin, Y. Xiao, J. Feng, M. Yang, Q. Pei, and X. Yi, "Didtrust: Privacy-preserving trust management for decentralized identity," *IEEE Transactions on Dependable and Secure Computing*, vol. 22, no. 3, pp. 3105–3120, 2025.

[4] M. Yang, H. Wang, and D. He, "Pm-abe: Puncturable bilateral fine-grained access control from lattices for secret sharing," *IEEE Transactions on Dependable and Secure Computing*, vol. 22, no. 2, pp. 1210–1223, 2024.

[5] Z. Hua, Y. Wang, S. Yi, Y. Zheng, X. Liu, Y. Chen, and X. Zhang, "Matrix-based secret sharing for reversible data hiding in encrypted images," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 5, pp. 3669–3686, 2023.

[6] J. Wu, N. Liu, and W. Kang, "The capacity region of distributed multi-user secret sharing under perfect secrecy," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 9954–9969, 2024.

[7] J. Chen, Y. Shen, and C. W. Sung, "A new shift-add secret sharing scheme for partial data protection with parallel zigzag decoding," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 10 221–10 232, 2024.

[8] C. Zhang, B. Ou, F. Peng, Y. Zhao, and K. Li, "A survey on reversible data hiding for uncompressed images," *ACM Computing Surveys*, vol. 56, no. 7, pp. 1–33, 2024.

[9] S. Weng, Y. Zhou, T. Zhang, M. Xiao, and Y. Zhao, "Reversible data hiding for jpeg images with adaptive multiple two-dimensional histogram and mapping generation," *IEEE Transactions on Multimedia*, vol. 25, pp. 8738–8752, 2023.

[10] L. Xiong, X. Han, C.-N. Yang, and Z. Xia, "Rdh-des: reversible data hiding over distributed encrypted-image servers based on secret sharing," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 19, no. 1, pp. 1–19, 2023.

[11] F. Li, Q. Wang, H. Cheng, X. Zhang, and C. Qin, "Jpeg reversible data hiding via block sorting optimization and dynamic iterative histogram modification," *IEEE Transactions on Multimedia*, vol. 27, pp. 3729–3743, 2025.

[12] Y. Zhang, X. Ye, X. Xiao, T. Xiang, H. Li, and X. Cao, "A reversible framework for efficient and secure visual privacy protection," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3334–3349, 2023.

[13] C. Qin, J. Hu, F. Li, Z. Qian, and X. Zhang, "Jpeg image encryption with adaptive dc coefficient prediction and rs pair permutation," *IEEE Transactions on Multimedia*, vol. 25, pp. 2528–2542, 2023.

[14] Y. Yuan, H. He, Y. Yang, H. Amirpour, C. Timmerer, and F. Chen, "Jpeg image encryption with dc rotation and undivided rsv-based ac group permutation," *IEEE Transactions on Multimedia*, vol. 27, pp. 1–15, 2025.

[15] Y. Ma, X. Chai, G. Long, Z. Gan, and Y. Zhang, "Tpe for jpeg images with dynamic m-ary decomposition and adaptive threshold constraints," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 35, no. 9, pp. 8864–8879, 2025.

[16] R. Wang, L. Li, G. Yang, X. Yan, and W. Yan, "Secret cracking and security enhancement for the image application of crt-based secret sharing," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 9819–9834, 2024.

[17] X. Yan, L. Li, L. Sun, J. Chen, and S. Wang, "Fake and dishonest participant immune secret image sharing," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 19, no. 4, pp. 1–26, 2023.

[18] P. Puteaux, F. Yriarte, and W. Puech, "A secret jpeg image sharing method over $gf(2^m)$ galois fields," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 6, pp. 3030–3042, 2023.

[19] X. Jia, T. Yu, X. Luo, D. Wang, and H. Zhou, "Maximizing contrast in xor-based visual cryptography schemes," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 12, pp. 12 849–12 861, 2024.

[20] X. Wu, B. Song, J. Fang, W. Yan, and Q.-Y. Peng, "Crp2-vcs: Contrast-oriented region-based progressive probabilistic visual cryptography schemes," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 35, no. 6, pp. 5501–5517, 2025.

[21] L. Xiong, X. Zhong, C.-N. Yang, and X. Han, "Transform domain-based invertible and lossless secret image sharing with authentication," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2912–2925, 2021.

[22] M. Naor and A. Shamir, "Visual cryptography," *Lecture Notes in Computer Science*, vol. 950, no. 1, pp. 1–12, 1995.

[23] P. Li, J. Ma, and Q. Ma, "(t, k, n) xor-based visual cryptography scheme with essential shadows," *Journal of Visual Communication and Image Representation*, vol. 72, p. 102911, 2020.

[24] Z. Wang, G. Arce, and G. Di Crescenzo, "Halftone visual cryptography via error diffusion," *Information Forensics and Security, IEEE Transactions on*, vol. 4, no. 3, pp. 383–396, 2009.

[25] B. Yan, Y. Xiang, and G. Hua, "Improving the visual quality of size-invariant visual cryptography for grayscale images: an analysis-by-synthesis (abs) approach," *IEEE Transactions on Image Processing*, vol. 28, no. 2, pp. 896–911, 2018.

[26] S. Shyu, "Visual cryptograms of random grids for general access structures," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, no. 3, pp. 414–424, 2013.

[27] X. Wu and X. Feng, "Size invariant visual cryptography schemes with evolving threshold access structures," *IEEE Transactions on Multimedia*, vol. 26, pp. 1488–1503, 2024.

[28] Z. Liu, G. Zhu, Y.-G. Wang, J. Yang, and S. Kwong, "A novel (t, s, k, n)-threshold visual secret sharing scheme based on access structure partition," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 16, no. 4, pp. 1–21, 2020.

[29] C.-C. Thien and J.-C. Lin, "Secret image sharing," *Computers & Graphics*, vol. 26, no. 5, pp. 765–770, 2002.

[30] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[31] C. Yang, T. Chen, K. Yu, and C. Wang, "Improvements of image sharing with steganography and authentication," *Journal of Systems and Software*, vol. 80, no. 7, pp. 1070–1076, 2007.

[32] P. Li, Z. Liu, and C.-N. Yang, "A construction method of (t, k, n)-essential secret image sharing scheme," *Signal Processing: Image Communication*, vol. 65, pp. 210–220, 2018.

[33] X. Yan, Y. Lu, C.-n. Yang, X. Zhang, and S. Wang, "A common method of share authentication in image secret sharing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 7, pp. 2896–2908, 2020.

[34] Z. Liu, G. Zhu, Y. Zhang, H. Zhang, and S. Kwong, "An efficient cheating-detectable secret image sharing scheme with smaller share sizes," *Journal of Information Security and Applications*, vol. 81, p. 103709, 2024.

[35] Z. Zhou, Y. Wan, Q. Cui, K. Yu, S. Mumtaz, C.-N. Yang, and M. Guizani, "Blockchain-based secure and efficient secret image sharing with outsourcing computation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 23, no. 1, pp. 423–435, 2024.

[36] P. Li, Z. Sun, Z. Situ, M. He, and T. Song, "Joint jpeg compression and encryption scheme based on order-8-16 block transform," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 7687–7696, 2023.

[37] V. Itier, P. Puteaux, and W. Puech, "Recompression of jpeg crypto-compressed images without a key," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 3, pp. 646–660, 2020.

[38] J. He, S. Huang, S. Tang, and J. Huang, "Jpeg image encryption with improved format compatibility and file size preservation," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2645–2658, 2018.

[39] Z. Qian, H. Zhou, X. Zhang, and W. Zhang, "Separable reversible data hiding in encrypted jpeg bitstreams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1055–1067, 2018.

[40] J. He, J. Chen, W. Luo, S. Tang, and J. Huang, "A novel high-capacity reversible data hiding scheme for encrypted jpeg bitstreams," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 12, pp. 3501–3515, 2019.

[41] K. Shimizu, Q. Wang, and T. Suzuki, "Ac prediction error propagation-based encryption for texture protection of jpeg compressed images," in *2021 Picture Coding Symposium (PCS)*. IEEE, 2021, pp. 1–5.

[42] F. Yriarte, P. Puteaux, and W. Puech, "A joint secret image sharing and jpeg compression scheme," in *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2022, pp. 581–585.

[43] Y. Jiang, X. Yan, J. Chen, J. Cheng, and J. Zhang, "Meaningful secret image sharing for jpeg images with arbitrary quality factors," *Math. Biosci. Eng*, vol. 19, no. 11, pp. 11 544–11 562, 2022.

[44] Y. Jiang, K. Chen, W. Yan, X. Yan, G. Yang, and K. Zeng, "Robust secret image sharing resistant to jpeg recompression based on stable block condition," *IEEE Transactions on Multimedia*, vol. 26, pp. 10 446–10 461, 2024.

[45] G. K. Wallace, "The jpeg still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[46] K. Minemura, Z. Moayed, K. Wong, X. Qi, and K. Tanaka, "Jpeg image scrambling without expansion in bitstream size," in *2012 19th IEEE International Conference on Image Processing*. IEEE, 2012, pp. 261–264.

Xiaotian Wu is currently an Associate Professor with the College of Cyber Security, Jinan University, Guangzhou, China. His research interests include visual cryptography, secret image sharing and multimedia security.

Dingbin Chen is currently pursuing the master's degree with the Department of Computer Science, Jinan University, China. His research interests are reversible data hiding and AI security.

Yuyang Xiong is currently pursuing the master's degree with the Department of Computer Science, Jinan University, China. Her research interests are secret image sharing and multimedia security.

WeiQi Yan is with AUT Department of Computer and Information Systems. He is an Associate Editor of ACM Transactions on Multimedia Computing, Communications and Applications, an Associate Editor of Frontiers in Neuroscience, a Section Editor of Springer Journal Discover Artificial Intelligence, an Associate Editor of Springer Nature Computer Science. Dr. Yan currently holds the position of Chair of ACM Multimedia Chapter of New Zealand, he is a Fellow of Engineering New Zealand.

Ching-Nung Yang is currently a Full Professor with the Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan. His research interests include coding theory, information security, and cryptography. He is a Fellow of IET.

Zhihua Xia is currently a Professor with the College of Cyber Security, Jinan University, Guangzhou, China. His research interests include digital forensic and encrypted image processing.