Client-Side Embedding of Screen-Shooting Resilient Image Watermarking

Xiangli Xiao, Yushu Zhang, *Senior Member, IEEE*, Zhongyun Hua, *Senior Member, IEEE*, Zhihua Xia, *Member, IEEE*, and Jian Weng, *Senior Member, IEEE*

***Abstract*— The proliferation of portable camera devices, represented by smartphones, is increasing the risk of sensitive internal data being leaked by screen shooting. To trace the leak source, a lot of research has been done on screen-shooting resilient watermarking technique, which is capable of extracting the previously embedded watermark from the screen-shot image. However, all existing screen-shooting resilient watermarking schemes follow the owner-side embedding mode. In this mode, the management center will suffer heavy computational and communication burden in the case of numerous screens, which hinders the system scalability. As another embedding mode of digital watermarking, client-side embedding can solve the above scalability problem by migrating the watermark embedding operation to the same time when the screen decrypts the image. By designing a pair of image encryption and personalized decryption algorithms based on matrix operation, this paper is the first to realize the client-side embedding of screen-shooting resilient watermarking. In this implementation, challenges are overcome and the following key achievements are attained. First, our scheme embeds watermark using the algorithm of Fang et al. without modification, and thus fully inherits its robustness against screen shooting. Second, the original image is securely encrypted and the watermarked image can be directly retrieved through decryption. Third, the secrecy of the screen watermark is ensured by concealing the embedding pattern. Finally, our scheme is validated by experiments, which shows that the efficiency advantage of client-side embedding is realized while maintaining robustness.**

***Index Terms*— Digital watermarking, screen-shooting leakage, leak tracing, cryptography.**

I. Introduction

With the increasing popularity of portable camera devices such as smartphones, screen shooting is becoming one of the easiest to implement but the hardest to police ways of stealing sensitive internal information. For example, in 2021, a purported employee of Alibaba uploaded screen-shot images online, revealing sensitive details including the company's basic business model and strategic road map for community e-commerce. This leakage garnered wide attention as it not only unmasked Alibaba's internal positioning of its community e-commerce, but also exposed crucial information such as its strategic objective to digitally transform 6 million small stores. As a result, this poses a potential risk of loss for Alibaba.

In view of the above issue, how to combat the leakage of screen shooting leads to a research hotspot in the field of information security. As a common technique for leak tracing, digital watermarking is a promising solution. Following this idea, many screen-shooting resilient watermarking schemes [1], [2], [3], [4], [5], [6], [7] were designed, which are robust to various distortions during screen shooting, such as lens distortion, light source distortion, and moirè pattern distortion.

Taking an image as the internal sensitive information to be protected, Fig. 1 illustrates the workflow of screen-shooting resilient watermarking. As shown therein, a watermark bound to the screen is implicitly embedded into the original image before it is imported for display. The screen and its control device are assumed to be hardware-isolated, Bluetooth-free, and not connected to the Internet. Nevertheless, leaks may happen if someone uses a portable camera device, such as a smartphone, to take an image in front of the screen. With the support of screen-shooting resilient watermarking technique, the previously embedded watermark can be extracted from the screen-shot image. In this way, the source of the leak can be

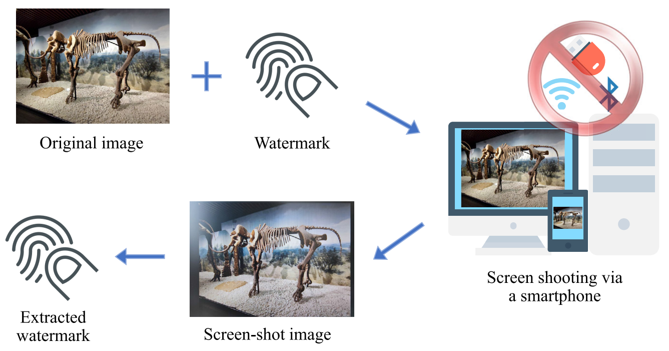


Fig. 1. The workflow of screen-shooting resilient watermarking.

Manuscript received 2 November 2023; revised 21 March 2024; accepted 2 May 2024. Date of publication 6 May 2024; date of current version 14 May 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62122032 and Grant U23B2023 and in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant KYCX23_0397. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Qinghua Li. *(Corresponding authors: Yushu Zhang; Zhihua Xia.)*

Xiangli Xiao and Yushu Zhang are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China (e-mail: xiaoxiangli@nuaa.edu.cn; yushu@nuaa.edu.cn).

Zhongyun Hua is with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Campus, Shenzhen 518055, China (e-mail: huazhongyun@hit.edu.cn).

Zhihua Xia and Jian Weng are with the College of Cyber Security, Jinan University, Guangzhou 510632, China (e-mail: xia_zhihua@163.com; cryptjweng@gmail.com).

Digital Object Identifier 10.1109/TIFS.2024.3397043

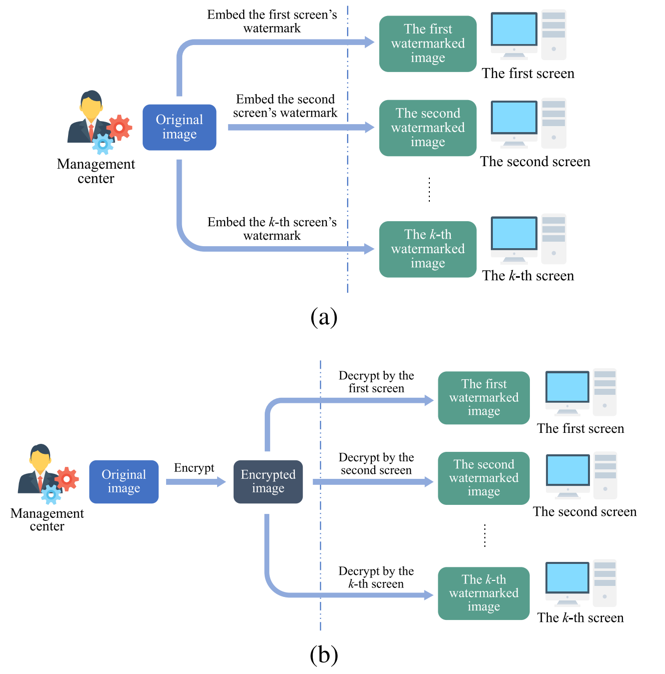

Embed the first screen's watermark
The first watermarked image
The first screen
Management center
Original image
Embed the second screen's watermark
The second watermarked image
The second screen
Embed the k-th screen's watermark
The k-th watermarked image
The k-th screen
(a)
Management center
Original image
Encrypt
Encrypted image
Decrypt by the first screen
The first watermarked image
The first screen
Decrypt by the second screen
The second watermarked image
The second screen
Decrypt by the k-th screen
The k-th watermarked image
The k-th screen
(b)

Fig. 2. Two modes of watermark embedding. (a) Owner-side embedding. (b) Client-side embedding.

traced back to the screen that the watermark is affixed. Given that screens are typically associated with specific workstations or display events during a defined time period, this approach effectively narrows the scope of traceability to individuals affiliated with the particular workstation or event. Consequently, the efficiency and precision of subsequent investigations can be enhanced.

However, all existing screen-shooting resilient watermarking schemes follow the owner-side embedding mode. In this mode, as shown in Fig. 2(a), a management center is required to generate different watermarked images and distribute them to each screen. Here, the management center is an internal department responsible for preserving all original images and watermark information, providing watermarked images for all screens, and extracting watermarks for tracing when screen-shooting leakage occurs. In the owner-side embedding mode [8], [9], [10], [11], [12], the management center could be stumped by heavy computational and communication burdens if there are a large number of screens, resulting in poor scalability.

In contrast, another mode of watermark embedding, namely client-side embedding [13], [14], [15], [16], [17], can solve the above problem. In this mode, as shown in Fig. 2(b), the management center only needs to encrypt the original image once and share the result to each screen, who implements watermark embedding while decrypting it. In fact, the decryption here is done by the control device of the screen, and this paper does not make an expressive distinction between the screen and its control device for simplicity. Since the image encryption operation is executed only once and each screen is offered with the same encrypted image, some special bandwidth-saving mechanisms can be employed for image transmission, such as multicasting [18], [19] and caching [20], [21]. As a result, the management center can gain significant saving in computing and communication costs, thereby greatly improving scalability.

In this paper, we design a screen-shooting resilient image watermarking scheme with the client-side embedding mode. The design is not trivial and there are three main challenges to be solved. First, the watermarking scheme must be robust enough against screen-shooting distortions. Second, the original image can only be accessed by the management center. To this end, two things need to be noted here. For one thing, the image encryption algorithm must be sufficiently secure. For another, the watermarked image should be obtained directly after decryption, rather than taking the original image as an intermediate product. Third, the watermark of each screen is specified by the management center and cannot be disclosed to anyone else.

In the proposed scheme, the above challenges are solved by elaborately designing a pair of matrix operation based image encryption and personalized decryption algorithms, in which the watermark is embedded into the image according to the algorithm proposed by Fang et al. [1] while decrypting. Since the watermark embedding follows the algorithm of Fang et al. without modification, the proposed scheme achieves a perfect inheritance of robustness against screen-shooting distortions. Regarding the security aspect of the proposed scheme, for one thing, the secrecy of the original image in terms of both pixel values and positions is well preserved by a series of carefully designed matrix operations, and the watermarked image can be obtained directly after decryption without any intermediate product. For another, the secrecy of the embedding pattern in the proposed scheme contributes to the unaware of the screen watermark by anyone except the management center.

To sum up, the main contributions of this paper are summarized as follows.

- To the best of our knowledge, we are the first to design a screen-shooting resilient watermarking scheme with the client-side embedding mode, which effectively alleviates the computing and communication workload of the management center compared with the conventional owner-side embedding mode.
- The challenges are well solved. For one thing, the robustness against screen-shooting distortions of the proposed watermarking scheme is consistent with that of Fang et al. [1]. For another, the secrecy of the original image and the screen watermark is well preserved.
- A formal security analysis is carried out. Experiments are conducted to demonstrate the maintained robustness and improved efficiency of the proposed scheme.

The rest of this paper is organized below. The next section reviews the related work. Section III covers the system model, threat model, design goals, as well as basic technique. After that, Section IV elaborates the scheme construction. Security analysis is given in Section V and experiments are carried out in Section VI. The last section concludes the paper.

II. Related Work

This section reviews the research progress of the two most relevant techniques, i.e., screen-shooting resilient watermarking and client-side embedded watermarking.

A. Screen-Shooting Resilient Watermarking

Screen shooting exposes sensitive internal data even if they are stored in a hardware-isolated, Bluetooth-free, and Internet-disconnected environment. The digital watermarking technique that can trace the source of screen-shooting leakage is called screen-shooting resilient watermarking. As analyzed by Fang et al. [1], the distortions resulted from the process of screen shooting mainly include lens distortion, light source distortion, and moiré pattern distortion. Because of these distortions, the conventional watermarking technique fails to resist the screen shooting process. In view of this, Fang et al. [1] put forward a small-size template algorithm, based on which the screen-shooting resilient embedding of watermarking was achieved. In the realization, the Intensity-Based Scale-Invariant Feature Transform (I-SIFT) algorithm is adopted to locate embedding regions. The watermark is then repeatedly embedded in each region by altering the relative magnitude of two intermediate-frequency Discrete Cosine Transform (DCT) coefficients.

After that, Chen et al. [2] came up with another embedding approach for screen-shooting resilient watermarking, in which the Harris-Laplace algorithm and the speeded-up robust feature algorithm are used to select embedding regions and construct orientation descriptors for the selected regions, respectively. By arranging the watermark into a circle, the scheme of Chen et al. is robust against rotation operation. In addition, in the scheme of Chen et al., the vertex information of the screen-shot image and the size information of the original image are not necessary in the process of watermark extraction. Chen et al. [3] also proposed a screen-shooting resilient watermarking scheme suitable for satellite images. In the scheme, a watermarking synchronization method is designed and the watermark is embedded in the discrete Fourier transform domain, thus obtaining decent robustness for stitching, cropping, and translation operations. In addition to the embedding principle adopted in the schemes mentioned above, namely modifying the frequency-domain coefficients, there are other means to construct screen-shooting resilient watermarking. Cheng et al. [4] proposed to modulate the moiré pattern with watermark information and then extract the watermark by analyzing the moiré pattern generated on the screen-shot image. Fang et al. [5] adopted the template watermarking method, in which the watermark template and the locating template are embedded into the blue and red channels of color images, respectively, then deep learning was used to achieve watermark extraction. After that, Fang et al. [22] proposed a noise layer for training deep learning networks to enhance the screen-shooting robustness. To combat the screen-shooting leakage of documents, Fang et al. [6] suggested embedding the watermark into the underpainting of documents. In contrast, Yang et al. [7] chose to embed the screen-shooting resilient watermark directly into the text via changing the centroid of the text.

Regarding the research on cryptography-related watermarking, Chen et al. [23] combined screen-shooting resilient watermarking with cryptography to embed two watermarks within one image, among which the plaintext-domain watermark and the ciphertext-domain watermark are applied for leak tracing and real-time information reading, respectively. Additionally, Dong et al. [24] proposed a ciphertext-domain screen-shooting resilient watermarking protocol with cloud participation that enables the tracing of both cloud server leakage behavior and user redistribution actions. However, no work has been done to realize the client-side embedding of screen-shooting resilient watermarking.

B. Client-Side Embedded Watermarking

Compared with the owner-side embedding mode, the client-side embedding mode has better system scalability. To migrate the watermark embedding operation to the media decryption process, the idea of Chameleon cipher [25] is often borrowed, which allows users (correspond to the screens in this paper) to remove most of the influence of media encryption through personalized decryption, but at the same time leave a small part of the influence specifically for marking watermark information. Based on this, a representative research practice was made by Adelsbach et al. [13]. After that, Celik et al. [14] skillfully applied the lookup table technique to media encryption and personalized decryption, and came up with a spread-spectrum watermarking scheme with the client-side embedding mode. Later, Pun et al. [26] further developed this scheme for its application to audio files, while Sun et al. [27] focused on reducing the size of keys. Following a similar technique to Celik et al., Piva et al. [15] went on to propose a client-side watermarking scheme using spread transform dither modulation. Subsequently, based on the schemes of Celik et al. and Piva et al., Bianchi and Piva [16] realized the secure distribution of decryption lookup tables in the ciphertext domain using homomorphic encryption, thus fairly protecting media copyright and the rights of users. The robustness performance of this scheme against collusion attacks was further enhanced in [17]. Building on the efforts of Xiao et al. [28] introduced the client-side watermarking into blockchain for copyright protection in multi-owner media sharing.

Unlike the technical approach of relying on lookup tables proposed by Celik et al., there are also subsequent works to implement the client-side embedding of watermarking based on vector quantization [29], [30], [31], quaternion rotation [32], and matrix operation [33]. Additionally, Lin et al. [34] conducted research on the integrity authentication of encrypted images in client-side watermarking and proposed a potential solution. Inspired by the client-side watermarking pattern, Mareen et al. [35] devised a scalable architecture for the uncompressed-domain video watermarking algorithm. In this paper, rather than based on any existing algorithm, we realize the client-side embedding of screen-shooting resilient watermarking by originally designing a pair of image encryption and personalized decryption algorithms based on matrix operation.

III. Preliminaries

This section firstly expounds the scientific issue studied in this paper from the aspects of system model, threat model, and design goals. Subsequently, the embedding algorithm of the screen-shooting resilient watermarking scheme proposed by Fang et al. [1] is briefly described.

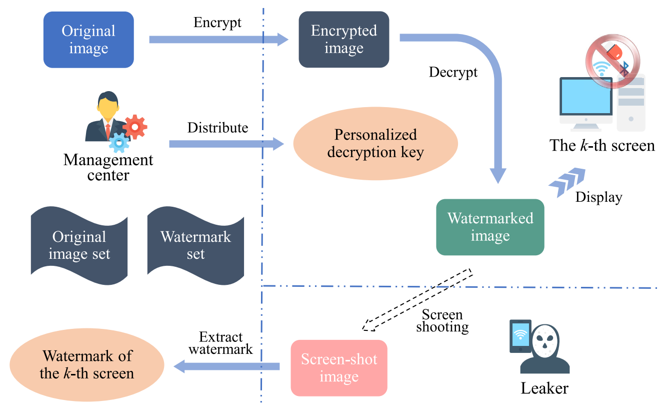


Fig. 3. System model.

A. System Model

As illustrated in Fig. 3, the system model consists of three entities: the management center, the screens, and the leaker. Their roles and mutual relations are elaborated as follows.

- **Management center.** The management center is a trusted internal department, which is responsible for providing watermarked images for all screens and extracting watermarks for tracing when screen-shooting leakage occurs. The management center preserves all the original images and watermark information. In the client-side embedding mode, to provide watermarked images, the management center encrypts each original image once and distributes a personalized decryption key for each screen.
- **Screens.** The screens are the media for displaying watermarked images. Each screen has its own control device in practice, e.g., a screen can be seen as a desktop computer at a workstation. The control device is responsible for performing required calculations such as image decryption. In this paper, we do not make an expressive distinction between the screen and its control device for simplicity. Each screen and its control device operate within a secure environment, are hardware-isolated, Bluetooth-free, and Internet-disconnected. As a result, the leaker cannot steal data from the screen and its control device via media such as USB flash disks, Bluetooth, and the Internet.
- **Leaker.** The leaker is malicious, who illegally shoots an image in front of the screen through a portable camera device, such as a smartphone, and then leaks the screen-shot image. In practice, both internal hires and external visitors may be leakers.

To clarify the applicable scenario of the proposed scheme, we present an illustrative example within a company setting here, where the need arises to share a sensitive image with many employees for business research. Each employee has an independent workstation with a computer that is hardware-isolated, Bluetooth-free, and Internet-disconnected. The management center multicasts the encrypted image to these computers, which are programmed to automatically decrypt and display the image. If the image is leaked via screenshot, the embedded watermark can be extracted to identify the originating screen, narrowing the investigation scope to relevant employees or visitors.

B. Threat Model and Design Goals

The threats considered in this paper stem from two entities: leakers and eavesdroppers. Leakers pose a threat by capturing screenshots and then releasing them, whereas eavesdroppers intercept the communication between the management center and the screen, aiming to obtain the plaintext version of the image (by cracking the encrypted image) and the watermark information of the screen. The eavesdroppers could either be internal employees authorized to view the image or unauthorized external entities. In comparison to external entities, internal employees are assumed to have stronger eavesdropping capabilities, who can intercept all communications between the management center and the screen, encompassing even those transmitted over an secure internal channel. In contrast, external entities are unable to eavesdrop on communications over secure internal channels. The eavesdroppers' access to the unencrypted and unwatermarked image leads to the untraceable revelation of sensitive internal information, while their possession of watermark information enhances their potential to disrupt traceability, ultimately elevating the overall risk.

Taking into account the above threat model and adhering to the general requirements of client-side embedding, the main design goals of this paper are summarized as follows. First, the proposed scheme must be robust enough against various distortions during screen shooting, so that the watermark can be accurately extracted from the screen-shot image to trace which screen the leakage came from. Second, the proposed scheme must be able to effectively reduce the overhead of the management center compared to the owner-side embedding mode. Third, to mitigate risks and facilitate management, all original images and watermark information should only be accessed by the management center, which requires that the image encryption is secure enough, the watermarked image is directly obtained after decryption, and the watermark embedding pattern is confidential.

C. Basic Technique

Here, we briefly describe the embedding algorithm of Fang et al. [1]. In this algorithm, if the image is colored, it needs to be converted to the YCbCr colorspace, and then the Y channel is used as the host image. Subsequently, the I-SIFT algorithm [36] is executed to acquire the keypoints (i.e., K_1, K_2, ...). As shown in Fig. 4, every embedding region is centered on a keypoint and has a size of $a \times b$. The embedding regions are initially selected using the greedy algorithm, i.e.,

$$\max \left(\sum_{i=1}^{n} \mathrm{In}\left(K_i\right) \right)$$
$$\text{s.t.} \quad \mathrm{R}\left(K_i\right) \cap \mathrm{R}\left(K_j\right) = \varnothing \qquad i, j \in [1, n], \ i \neq j, \tag{1}$$

where $\mathrm{In}\left(K_i\right)$ and $\mathrm{R}\left(K_i\right)$ output the intensity of the keypoint K_i and the region centered on the keypoint K_i, respectively. The n embedding regions initially selected are used to repeatedly embed the BCH-encoded watermark $\mathbf{W}_k$ of the k-th screen, as shown in Fig. 4. Here, every watermark bit is embedded into an 8×8 block. To achieve this embedding,

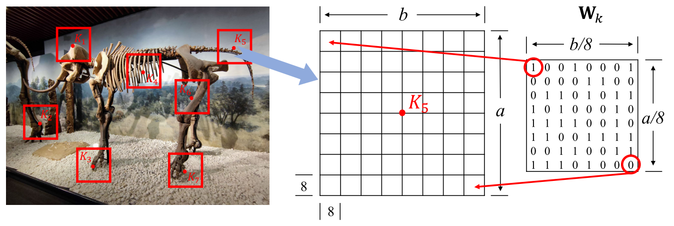


Fig. 4. Partial flow of the embedding algorithm proposed by Fang et al. [1].

DCT is first performed on the 8×8 block and then a pair of DCT coefficients C_1 and C_2 are selected to compute the following formula for watermark embedding:

$$\begin{cases} d \geq |q_2 - q_1| \dfrac{C_1 \cdot \max(q_1, q_2) + C_2 \cdot \min(q_1, q_2)}{2q_1 q_2} \\ \qquad + \dfrac{(q_1 + q_2)\, r}{2}, \\ C_1 = \max(C_1, C_2) + \dfrac{d}{2},\ C_2 = \min(C_1, C_2) - \dfrac{d}{2}, \\ \qquad \text{if } w = 0, \\ C_1 = \min(C_1, C_2) - \dfrac{d}{2},\ C_2 = \max(C_1, C_2) + \dfrac{d}{2}, \\ \qquad \text{otherwise}, \end{cases} \tag{2}$$

where r is used to set the watermark embedding strength, w is the watermark bit to be embedded, and q_1 and q_2 are the JPEG quantization steps of C_1 and C_2, respectively. After the calculation, Inverse DCT (IDCT) is performed on the 8×8 block. The pair of coefficients most recommended by Fang et al. are located at $(5, 4)$ and $(4, 5)$. After $\mathbf{W}_k$ is embedded into all the regions, the Structure Similarity Index Measure (SSIM) values between each region and its original version is computed. Finally, m watermarked regions with the largest SSIM value are selected to replace the original regions while the remaining $n - m$ regions remain unchanged.

IV. Scheme Construction

This section elaborates on the scheme construction, which is divided into four stages, i.e., image encryption, distribution of personalized decryption keys, image decryption, and watermark extraction. At the end of this section, the flow of the proposed scheme is summarized.

A. Image Encryption

The image encryption stage consists of two processes, i.e., default watermark embedding and encryption processing.

1) Default Watermark Embedding: As in Section III-C, if the original image $\mathbf{I}$ is colored, it needs to be converted to the YCbCr colorspace and the Y channel is used as the host image; otherwise, the grayscale image itself serves as the host image. Subsequently, the same region selection method is performed to initially select n embedding regions $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_n$ of size $a \times b$ from the host image, and the set composed of these n embedding regions is denoted as $\mathbb{R}$.

After completing the above region selection operation, the management center randomly generates a uniform binary vector of length $ab/64$ and arranges it into a matrix as shown in Fig. 4. The obtained matrix is denoted as $\mathbf{W}^+$ and is called the default watermark. In addition, the management center also needs to randomly generate a uniform binary matrix $\mathbf{L}$ with the same size as $\mathbf{W}^+$ for obfuscating the embedding pattern. Here, both $\mathbf{W}^+$ and $\mathbf{L}$ are the key to concealing the embedding pattern and need to be kept confidential. The same $\mathbf{W}^+$ is reused by different images but each image can use a different $\mathbf{L}$. With the above input parameters, Algorithm 1 is executed to implement the default watermark embedding operation. In Algorithm 1, the function embed$(\mathbf{T}, w^+, loc_1, loc_2, r)$ is used to embed the watermark bit w^+ into the 8×8 block $\mathbf{T}$ and the embedding mode is consistent with that described in Section III-C. Here, the inputted loc_1 and loc_2 are the coordinates of C_1 and C_2, respectively. Finally, as an option, the management center can discard some watermarked embedding regions with relatively more distortion. In this paper, we assume that there are $n - m$ regions discarded here. The discarded regions will be removed from $\mathbb{R}_e$ and the coordinates of the keypoints of the remaining m regions are recorded into a set $\mathbb{K}_p$.

In subsequent steps, $\mathbb{K}_p$ will be transmitted to the screen, which seemingly reveals the embedding location of the watermark directly. However, contrary to this initial perception, the application scenario considered in this paper demonstrates a different reality. In this scenario, the decryption operation is seamlessly executed in the background of a screen that is hardware-isolated, Bluetooth-free, and Internet-disconnected, utilizing an automated program. As a result, potential leakers will be unable to access $\mathbb{K}_p$ unless they have the capability to eavesdrop on the secure transmission channel. In addition, if $\mathbb{K}_p$ is transmitted based on a key exchange protocol, eavesdropping will become futile. In fact, in Fang et al. scheme [1], the embedding location of the watermark is not strictly confidential, since the selection of the watermark embedding region is based on the public ISIFT algorithm. Therefore, as long as the transmission process of $\mathbb{K}_p$ is secure enough, our scheme will not increase the risk of the watermark embedding region being exposed.

2) Encryption Processing: First, the management center replaces the pixel values of the m selected embedding regions in the host image with random integers in the range of $[0, 255]$. The image after the above processing is denoted as $\mathbf{I}_p$, which has c channels and a size of $M \times N$. If the original image is colored, then $c = 3$ and the three channels in turn represent the Y, Cb, and Cr channels; otherwise, $c = 1$. To encrypt $\mathbf{I}_p$, the management center needs to generate the encryption keys $\mathbf{P}_l$, $\mathbf{P}_r$, $\mathbf{u}$, and $\mathbf{v}$. Among them, $\mathbf{P}_l$ and $\mathbf{P}_r$ are two matrices with sizes of $M \times c$ and $N \times c$, respectively. $\mathbf{u}$ and $\mathbf{v}$ are two vectors with lengths of M and N, respectively. The same $\mathbf{P}_l$ and $\mathbf{P}_r$ are reused by different images but each image uses a different pair of $\mathbf{u}$ and $\mathbf{v}$. $\mathbf{P}_l$ and $\mathbf{P}_r$ are generated by

$$\begin{cases} \mathbf{P}_l(:, j) = \text{RandPerm}(M), \\ \mathbf{P}_r(:, j) = \text{RandPerm}(N), \end{cases} \tag{3}$$

where $j = 1, 2, \ldots, c$ and the function RandPerm(M) returns a vector of length M whose elements are integers and randomly selected from the interval $[1, M]$ without repeating. In other words, the function randomly scrambles the elements in the vector $(1, 2, \ldots, M)^T$. Here, each output is independent

Algorithm 1 embedding the Default Watermark

```
Input: $\mathbb{R}$, $\mathbf{W}^+$, $\mathbf{L}$, $n$, $r$, $a$, and $b$
Output: $\mathbb{R}_e$
1  $ind = 1$;
2  for $i = 1 : n$ do
3      for $x = 8 : 8 : a$ do
4          for $y = 8 : 8 : b$ do
5              if $\mathbf{L}(x/8, y/8) == 0$ then
6                  $loc_1 = (5, 4)$; $loc_2 = (4, 5)$;
7              else
8                  $loc_1 = (4, 5)$; $loc_2 = (5, 4)$;
9              end
10             $\mathbf{T} = \mathbf{R}_i(x-7:x,\ y-7:y)$;
11             $w^+ = \mathbf{W}^+(x/8, y/8)$;
12             $w^- = |w^+ - 1|$;
13             $\mathbf{temp}^+ = \text{embed}(\mathbf{T}, w^+, loc_1, loc_2, r)$;
14             $\mathbf{temp}^- = \text{embed}(\mathbf{T}, w^-, loc_1, loc_2, r)$;
15             $\mathbf{R}_e^{ind}(x-7:x,\ y-7:y) = \mathbf{temp}^+$;
16             $\mathbf{R}_e^{ind+1}(x-7:x,\ y-7:y) = \mathbf{temp}^-$;
17         end
18     end
19     Save $\mathbf{R}_e^{ind}$ and $\mathbf{R}_e^{ind+1}$ into the set $\mathbb{R}_e$;
20     $ind = ind + 2$;
21 end
22 return $\mathbb{R}_e$.
```

when the function is called multiple times. Subsequently, $\mathbf{u}$ and $\mathbf{v}$ are generated by

$$\begin{cases} \mathbf{u} = 256 \cdot \text{Rand}(M, 1), \\ \mathbf{v} = \lceil \hat{q} \cdot \text{Rand}(1, N) \rceil, \end{cases} \tag{4}$$

where $\hat{q}$ is an integer arbitrarily selected by the management center, $\lceil \cdot \rceil$ is the upward rounding operation, and the function Rand$(M, 1)$ returns a vector of size $M \times 1$ whose elements are uniformly selected from the interval $(0, 1)$. After generating $\mathbf{u}$ and $\mathbf{v}$, the management center calculates

$$\mathbf{Q} = \lfloor \mathbf{u} \cdot \mathbf{v} \rfloor \bmod 256, \tag{5}$$

where $\lfloor \cdot \rfloor$ represents rounding down. Finally, the management center encrypts $\mathbf{I}_p$ by

$$\mathbf{E}_p^j = \mathbf{H}_l^j \cdot (\mathbf{I}_p^j \oplus \mathbf{Q}) \cdot \mathbf{H}_r^j, \tag{6}$$

where $\oplus$ is the XOR operation, $\mathbf{E}_p^j$ and $\mathbf{I}_p^j$ represent the j-th channel of $\mathbf{E}_p$ and $\mathbf{I}_p$ respectively, and

$$\begin{cases} \mathbf{H}_l^j(\hat{x}, \hat{y}) = \Phi_{\mathbf{P}_l(\hat{x}, j), \hat{y}}, \\ \mathbf{H}_r^j(\overline{x}, \overline{y}) = \Phi_{\mathbf{P}_r(\overline{x}, j), \overline{y}}, \end{cases} \tag{7}$$

where $\hat{x}, \hat{y} = 1, 2, \ldots, M$, $\overline{x}, \overline{y} = 1, 2, \ldots, N$, and

$$\Phi_{\mathbf{P}_l(\hat{x}, j), \hat{y}} = \begin{cases} 0, & \text{if } \mathbf{P}_l(\hat{x}, j) \neq \hat{y}, \\ 1, & \text{if } \mathbf{P}_l(\hat{x}, j) = \hat{y}. \end{cases} \tag{8}$$

The definition of $\Phi_{\mathbf{P}_r(\overline{x}, j), \overline{y}}$ is the same as above. As $\mathbf{H}_l^j$ and $\mathbf{H}_r^j$ are sparse matrices, Eq. (6) can be computed by executing Algorithm 2 in pursuit of better efficiency. From Algorithm 2, it can be seen that the roles of $\mathbf{H}_l^j$ and $\mathbf{H}_r^j$ are to implement row and column scrambling on $\mathbf{T}$, respectively.

Algorithm 2 an Efficient Way to Compute Eq. (6)

```
Input: $\mathbf{I}_p^j$, $\mathbf{Q}$, $\mathbf{P}_l$, $\mathbf{P}_r$, $c$, $M$, and $N$
Output: $\mathbf{E}_p^j$
1  for $j = 1 : c$ do
2      $\mathbf{T} = \mathbf{I}_p^j \oplus \mathbf{Q}$;
3      for $i = 1 : M$ do
4          $\mathbf{temp} = \mathbf{T}(i, :)$;
5          $\mathbf{T}(i, :) = \mathbf{T}(\mathbf{P}_l(i, j), :)$;
6          $\mathbf{T}(\mathbf{P}_l(i, j), :) = \mathbf{temp}$;
7      end
8      for $i = 1 : N$ do
9          $\mathbf{temp} = \mathbf{T}(:, i)$;
10         $\mathbf{T}(:, i) = \mathbf{T}(:, \mathbf{P}_r(i, j))$;
11         $\mathbf{T}(:, \mathbf{P}_r(i, j)) = \mathbf{temp}$;
12     end
13     $\mathbf{E}_p^j = \mathbf{T}$;
14 end
15 return $\mathbf{E}_p^j$.
```

In addition to encrypt $\mathbf{I}_p$ according to the above approach, the management center also needs to encrypt $\mathbb{R}_e$. To this end, four encryption keys $\overline{\mathbf{P}}_l$, $\overline{\mathbf{P}}_r$, $\overline{\mathbf{U}}$, and $\overline{\mathbf{V}}$ need to be generated by the management center first. Here, $\overline{\mathbf{P}}_l$, $\overline{\mathbf{P}}_r$, $\overline{\mathbf{U}}$, and $\overline{\mathbf{V}}$ are four matrices with sizes of $a \times m$, $b \times m$, $8 \times (ab/32)$, and $(ab/32) \times 8$, respectively. The same $\overline{\mathbf{U}}$ and $\overline{\mathbf{V}}$ are reused by different images while each image uses a different pair of $\overline{\mathbf{P}}_l$ and $\overline{\mathbf{P}}_r$, which are generated by

$$\begin{cases} \overline{\mathbf{P}}_l(:, h) = \text{RandPerm}(a), \\ \overline{\mathbf{P}}_r(:, h) = \text{RandPerm}(b), \end{cases} \tag{9}$$

where $h = 1, 2, \ldots, m$. $\overline{\mathbf{U}}$ and $\overline{\mathbf{V}}$ are generated by

$$\begin{cases} \overline{\mathbf{U}}(:, z) = 256 \cdot \text{Rand}(8, 1), \\ \overline{\mathbf{V}}(z, :) = \lceil \hat{q} \cdot \text{Rand}(1, 8) \rceil, \end{cases} \tag{10}$$

where $z = 1, 2, \ldots, ab/32$. To implement the encryption, the management center first calculates

$$\overline{\mathbf{Q}}_z = \lfloor \overline{\mathbf{U}}(:, z) \cdot \overline{\mathbf{V}}(z, :) \rfloor \bmod 256. \tag{11}$$

Then the obtained result is used to encrypt the 8×8 blocks $\mathbf{B}_q^i$ and $\mathbf{B}_q^{i+1}$ in the regions $\mathbf{R}_e^i$ and $\mathbf{R}_e^{i+1}$, i.e.,

$$\begin{cases} \mathbf{F}_q^i = \mathbf{B}_q^i \oplus \overline{\mathbf{Q}}_z, \\ \mathbf{F}_q^{i+1} = \mathbf{B}_q^{i+1} \oplus \overline{\mathbf{Q}}_{z+1}, \end{cases} \tag{12}$$

where $i = 1, 3, \ldots, 2m-1$, $q = 1, 2, \ldots, ab/64$, and $z = 2q - 1$. The regions obtained by encrypting $\mathbf{R}_e^i$ with Eq. (12) are denoted as $\mathbf{D}_i$, where $i = 1, 2, \ldots, 2m$. Subsequently, $\mathbf{D}_i$ is further encrypted by

$$\mathbf{R}_c^i = \overline{\mathbf{H}}_l^h \cdot \mathbf{D}_i \cdot \overline{\mathbf{H}}_r^h, \tag{13}$$

where $h = \lceil i/2 \rceil$ and $\overline{\mathbf{H}}_l^i$ and $\overline{\mathbf{H}}_r^i$ are generated based on $\overline{\mathbf{P}}_l$ and $\overline{\mathbf{P}}_r$ following the same method as Eqs. (7) and (8). $\mathbf{R}_c^i$ are the final results of region encryption, and together they form a set $\mathbb{R}_c$. The calculation of Eq. (13) can be accelerated in a similar way to Algorithm 2.

After completing the above encryption operations, the management center releases the results $\mathbf{E}_p$ and $\mathbb{R}_c$ on the internal Local Area Network (LAN) for the screens to download. The transmission of $\mathbf{E}_p$ and $\mathbb{R}_c$ from the management center to the screens does not necessitate a strictly secure internal channel as it is capable of tolerating eavesdropping (please see the security analysis given in Section V-A). Nevertheless, in the event of an active attack on the channel, the employment of a message authentication code becomes imperative.

Algorithm 3 key Distribution for the k-Th Screen

```
Input: $\mathbf{W}_k^+$, $\overline{\mathbf{U}}$, $\overline{\mathbf{V}}$, $a$, and $b$
Output: $\mathbf{U}_k$ and $\mathbf{V}_k$
1  $ind = 1$;
2  for $i = 1 : a/8$ do
3      for $j = 1 : b/8$ do
4          $num = b/8 \times (i-1) + j$;
5          if $\mathbf{W}_k^+(i, j) == 0$ then
6              $\mathbf{U}_k(:, num) = \overline{\mathbf{U}}(:, ind)$;
7              $\mathbf{V}_k(num, :) = \overline{\mathbf{V}}(ind, :)$;
8          else
9              $\mathbf{U}_k(:, num) = \overline{\mathbf{U}}(:, ind+1)$;
10             $\mathbf{V}_k(num, :) = \overline{\mathbf{V}}(ind+1, :)$;
11         end
12     end
13     $ind = ind + 2$;
14 end
15 return $\mathbf{U}_k$ and $\mathbf{V}_k$.
```

B. Distribution of Personalized Decryption Keys

At this stage, the management center distributes a personalized decryption key for each screen. To this end, the management center first selects a non-repeating binary watermark sequence for each screen and performs BCH encoding on the selected sequences. Then the BCH-encoded sequences are supplemented to a length of $ab/64$ by trailing 0s. After that, the management center arranges the supplemented sequences into the matrix form by rows as shown in Fig. 4. Assuming that the watermark of the k-th screen is $\mathbf{w}_k$ and the processed version is $\mathbf{W}_k$, the management center calculates the XOR of $\mathbf{W}_k$ and the default watermark $\mathbf{W}^+$, i.e.,

$$\mathbf{W}_k^+ = \mathbf{W}_k \oplus \mathbf{W}^+. \tag{14}$$

After that, Algorithm 3 is executed by the management center to obtain $\mathbf{U}_k$ and $\mathbf{V}_k$. $\mathbb{K}_k = \{\mathbf{U}_k, \mathbf{V}_k, \mathbf{W}_k^+\}$ is called the personalized decryption key of the k-th screen, which must be transmitted to the k-th screen over a secure internal channel or based on a key exchange protocol.

C. Image Decryption

Suppose the image $\mathbf{I}$ is to be displayed on the k-th screen, the management center needs to share a session key $\mathbb{S}_I$ with

Algorithm 4 decryption of the Embedding Regions

```
Input: $\mathbb{D}$, $\mathbf{W}_k^+$, $m$, $a$, and $b$
Output: $\mathbb{R}_d$
1  $ind = 1$;
2  for $h = 1 : m$ do
3      for $x = 8 : 8 : a$ do
4          for $y = 8 : 8 : b$ do
5              if $\mathbf{W}_k^+(x/8, y/8) == 0$ then
6                  temp $= \mathbf{D}_{ind}(x-7 : x, y-7 : y)$;
7              else
8                  temp $= \mathbf{D}_{ind+1}(x-7 : x, y-7 : y)$;
9              end
10         end
11         $\mathbf{R}_d^h(x-7 : x, y-7 : y) =$ temp;
12     end
13     Save $\mathbf{R}_d^h$ into the set $\mathbb{R}_d$;
14     $ind = ind + 2$;
15 end
16 return $\mathbb{R}_d$.
```

the k-th screen before image decryption. $\mathbb{S}_I$ is different for each image and $\mathbb{S}_I = \{\mathbf{u}, \mathbf{v}, \overline{\mathbf{P}}_l, \overline{\mathbf{P}}_r, \mathbb{K}_p\}$. Besides that, the management center needs to share $\mathbf{P}_l$ and $\mathbf{P}_r$ with all the screens, which is only performed once as they are the same for all screens and images. Here, the sharing of $\mathbb{S}_I$, $\mathbf{P}_l$, and $\mathbf{P}_r$ must be over a secure internal channel or based on a key exchange protocol. After having $\mathbb{K}_k$, $\mathbb{S}_I$, $\mathbf{P}_l$, and $\mathbf{P}_r$, the k-th screen downloads $\mathbf{E}_p$ and $\mathbb{R}_c$ from the LAN and then performs the following image decryption processes. At the same time of image decryption, the watermark $\mathbf{W}_k$ of the k-th screen is successfully embedded into the image.

The image decryption stage mainly includes two processes, i.e., decryption of $\mathbf{E}_p$ and decryption of $\mathbb{R}_c$. To decrypt $\mathbf{E}_p$, the k-th screen first calculates $\mathbf{H}_l^j$ and $\mathbf{H}_r^j$ based on Eqs. (7) and (8) with $\mathbf{P}_l$ and $\mathbf{P}_r$ as inputs. In addition, $\mathbf{u}$ and $\mathbf{v}$ are used to calculate $\mathbf{Q}$ by Eq. (5). With $\mathbf{H}_l^j$, $\mathbf{H}_r^j$, and $\mathbf{Q}$, the k-th screen decrypt $\mathbf{E}_p$ by

$$\mathbf{I}_p^j = \left(\left(\mathbf{H}_l^j\right)^T \cdot \mathbf{E}_p^j \cdot \left(\mathbf{H}_r^j\right)^T\right) \oplus \mathbf{Q}, \tag{15}$$

which is the inverse of Eq. (6). Thus, pixels of the image $\mathbf{I}$ other than the embedding regions are decrypted without loss.

To decrypt the embedding regions $\mathbb{R}_c$, the k-th screen first calculates $\overline{\mathbf{H}}_l^h$ and $\overline{\mathbf{H}}_r^h$ based on Eqs. (7) and (8) with $\overline{\mathbf{P}}_l$ and $\overline{\mathbf{P}}_r$ as inputs and then calculates

$$\mathbf{D}_i = \left(\overline{\mathbf{H}}_l^h\right)^T \cdot \mathbf{R}_c^i \cdot \left(\overline{\mathbf{H}}_r^h\right)^T, \tag{16}$$

which is the inverse of Eq. (13). Both Eqs. (15) and (16) can be computed more efficiently in a similar way to Algorithm 2. After obtaining $\mathbb{D} = \{\mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_{2m}\}$, the k-th screen performs further decryption by Algorithm 4 and obtains $\mathbb{R}_d = \{\mathbf{R}_d^1, \mathbf{R}_d^2, \ldots, \mathbf{R}_d^m\}$. Afterwards, the k-th screen computes

$$\overline{\mathbf{Q}}_q^k = \lfloor \mathbf{U}_k(:, q) \cdot \mathbf{V}_k(q, :) \rfloor \bmod 256, \tag{17}$$

where $q = 1, 2, \ldots, ab/64$. $\overline{\mathbf{Q}}_q^k$ are used to decrypt the 8×8 blocks $\mathbf{F}_q^h$ of $\mathbf{R}_d^h$, i.e.,

$$\mathbf{B}_q^h = \mathbf{F}_q^h \oplus \overline{\mathbf{Q}}_q^k, \tag{18}$$

where $h = 1, 2, \ldots, m$ and $\mathbf{B}_q^h$ is different for each screen. In this way, the embedding regions $\mathbf{R}_k^h$ generated by decrypting $\mathbf{R}_d^h$ with Eq. (18) are obtained by the k-th screen.

Finally, the k-th screen overwrites the decrypted m watermark embedding regions $\mathbb{R}_k = \{\mathbf{R}_k^1, \mathbf{R}_k^2, \ldots, \mathbf{R}_k^m\}$ to the corresponding positions in $\mathbf{I}_p$ based on $\mathbb{K}_p$ and then converts the resulting image to the RGB colorspace, thereby obtaining the watermarked image $\mathbf{I}_k$ embedded with its own watermark.

Algorithm 5 watermark Extraction

```
Input: $\mathbf{I}_s'$, $\mathbf{loc}_x^i$, $\mathbf{loc}_y^i$, $\mathbf{L}$, $a$, and $b$
Output: $\mathbf{W}_k^i$
1  for $z = 1:9$ do
2      $ind = 1$;
3      $\mathbf{temp} = \mathbf{I}_s'(\mathbf{loc}_x^i(z) - b/2 + 1 : \mathbf{loc}_x^i(z) + b/2,$
           $\mathbf{loc}_y^i(z) - a/2 + 1 : \mathbf{loc}_y^i(z) + a/2)$;
4      for $x = 8:8:a$ do
5          for $y = 8:8:b$ do
6              if $\mathbf{L}(x/8, y/8) == 0$ then
7                  $loc_1 = (5,4)$; $loc_2 = (4,5)$;
8              else
9                  $loc_1 = (4,5)$; $loc_2 = (5,4)$;
10             end
11             $\mathbf{temp\_s} = \mathbf{temp}(x-7:x, y-7:y)$;
12             $\mathbf{temp\_s} = \text{dct2}(\mathbf{temp\_s})$; /*perform
                 2D-DCT on $\mathbf{temp\_s}$*/
13             $C_1 = \mathbf{temp\_s}(loc_1(1), loc_1(2))$;
14             $C_2 = \mathbf{temp\_s}(loc_2(1), loc_2(2))$;
15             if $C_1 > C_2$ then
16                 $\mathbf{W}_k^i(ind, z) = 0$;
17             else
18                 $\mathbf{W}_k^i(ind, z) = 1$;
19             end
20             $ind = ind + 1$;
21         end
22     end
23 end
24 return $\mathbf{W}_k^i$.
```

D. Watermark Extraction

After capturing a screenshot of the watermarked image $\mathbf{I}_k$ using a portable camera device, an image $\mathbf{I}_s$ is obtained. To extract the watermark from $\mathbf{I}_s$, the management center first performs a distortion correction operation on $\mathbf{I}_s$, which was described by Fang et al. [1], and thereby obtains $\hat{\mathbf{I}}_s$. Subsequently, the management center converts $\hat{\mathbf{I}}_s$ to the YCbCr colorspace and extracts its Y channel to be represented by $\mathbf{I}_s'$ if $\mathbf{I}_s$ is colored, otherwise $\hat{\mathbf{I}}_s$ is converted to a grayscale image and assigned to $\mathbf{I}_s'$. After that, the management center performs the I-SIFT algorithm on $\mathbf{I}_s'$ and extracts $2m$ regions of size $a \times b$ from the image based on the greedy algorithm described in Eq. (1). For the i-th of these regions, the coordinates of its center point and the eight points around the center point are recorded in two vectors $\mathbf{loc}_x^i$ and $\mathbf{loc}_y^i$ of length 9, and then nine watermarks are extracted by executing Algorithm 5.

The watermarks extracted from the $2m$ regions are recorded into a set $\mathbb{W}_k = \{\mathbf{W}_k^1, \mathbf{W}_k^2, \ldots, \mathbf{W}_k^{2m}\}$, which is inputted into Algorithm 6 to perform cross-validation. In Algorithm 6, th is an adjustable threshold. The resulting $\widetilde{\mathbf{W}}_k$ is used in the next watermark extraction step, i.e.,

$$\widetilde{\mathbf{w}}_k(q) = \begin{cases} 0, & \text{if } \sum_{i=1}^{f} \widetilde{\mathbf{W}}_k(q,i) < \dfrac{f}{2}, \\ 1, & \text{otherwise,} \end{cases} \tag{19}$$

where $q = 1, 2, \ldots, ab/64$ and f is the number of columns in $\widetilde{\mathbf{W}}_k$. Finally, the management center performs BCH decoding on $\widetilde{\mathbf{w}}_k$ to obatin $\mathbf{w}_k'$. If $\mathbf{w}_k' = \mathbf{w}_k$ is true within the allowable error range, then the leak source is considered to be the k-th screen.

Algorithm 6 Cross-Validation

```
Input: $\mathbb{W}_k$, $m$, and $th$
Output: $\widetilde{\mathbf{W}}_k$
1  $ind = 1$;
2  for $i = 1:2m$ do
3      for $z = 1:9$ do
4          $w_1 = \mathbf{W}_k^i(:, z)$;
5          for $g = i+1:2m$ do
6              for $t = 1:9$ do
7                  $w_2 = \mathbf{W}_k^g(:, t)$;
8                  $num = w_1 \oplus w_2$;
9                  if $num < th$ then
10                     $\widetilde{\mathbf{W}}_k(:, ind) = w_1$;
11                     $\widetilde{\mathbf{W}}_k(:, ind+1) = w_2$;
12                     $ind = ind + 2$;
13                 end
14             end
15         end
16     end
17 end
18 return $\widetilde{\mathbf{W}}_k$.
```

E. Summary

We summarize the contents of the above four subsections as Figs. 5 and 6, which illustrate the watermark embedding flow and watermark extraction flow of the proposed scheme, respectively. In Fig. 6, $\mathbb{loc}_x = \{\mathbf{loc}_x^1, \mathbf{loc}_x^2, \ldots, \mathbf{loc}_x^{2m}\}$ and $\mathbb{loc}_y = \{\mathbf{loc}_y^1, \mathbf{loc}_y^2, \ldots, \mathbf{loc}_y^{2m}\}$. As shown in the two figures, only image decryption is implemented by the screen and the remaining three stages are implemented by the management center. Nevertheless, when a new screen is to participate in the system, the management center only needs to implement key distribution instead of image encryption. The encryption only needs to be performed once for each image. The overhead of

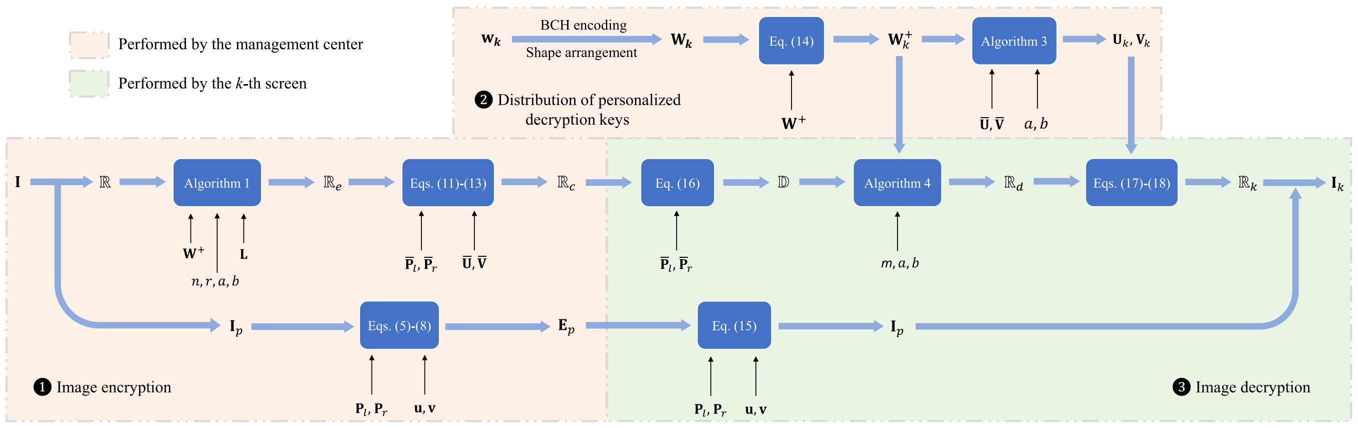


Fig. 5. Summary of the watermark embedding flow.

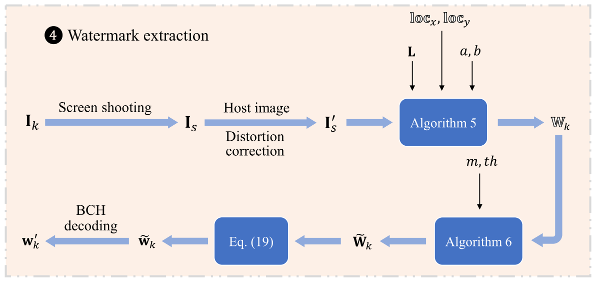


Fig. 6. Summary of the watermark extraction flow.

key distribution is much less than that of image encryption. As a result, the proposed scheme can well cope with the increase in the number of screens.

V. Security Analysis

In this section, we conduct theoretical analysis on the third point of the design goals given in Section III-A, i.e., analyze the security of image encryption as well as the concealment of watermark embedding pattern.

A. Security of Image Encryption

As shown in Fig. 5, the encryption of the original image $\mathbf{I}$ consists of two parts, i.e., the encryption of $\mathbf{I}_p$ by Eqs. (5)-(8) and the encryption of $\mathbb{R}_e$ by Eqs. (11)-(13). It is easy to find that the encryption processes of these two parts is completely consistent in principle. Therefore, the following analysis takes the encryption of $\mathbf{I}_p$ as an example without loss of generality.

The encryption of $\mathbf{I}_p$ can be broken down into two steps, namely $\mathbf{T}^j = \mathbf{I}_p^j \oplus \mathbf{Q}$ and $\mathbf{E}_p^j = \mathbf{H}_l^j \cdot \mathbf{T}^j \cdot \mathbf{H}_r^j$, where $\mathbf{Q} = \lfloor \mathbf{u} \cdot \mathbf{v} \rfloor \bmod 256$, $\mathbf{H}_l^j$ and $\mathbf{H}_r^j$ are generated by Eq. (7) based on $\mathbf{P}_l$ and $\mathbf{P}_r$, and $j = 1, 2, \ldots, c$. For the former step, we call it Step 1 and give the following theorem.

Theorem 1: Step 1 is computationally secure in value against chosen-plaintext attacks if each encryption uses a pair of re-generated $\mathbf{u}$ *and* $\mathbf{v}$.

Proof: We formalize Step 1 as $C = \mathrm{Enc}_K(M)$, where M, C, and K refer to an element of $\mathbf{I}_p^j$, $\mathbf{T}^j$, and $\mathbf{Q}$, respectively. We also employ $\mathcal{M}$ and $\mathcal{C}$ to represent the plaintext space and the ciphertext space, respectively. With these symbolic definitions, we first compute $\Pr[C = c'|M = m']$ for arbitrary $c' \in \mathcal{C}$ and $m' \in \mathcal{M}$:

$$\begin{aligned}\Pr[C = c'|M = m'] &= \Pr[\mathrm{Enc}_K(m') = c'] \\ &= \Pr[m' \oplus K = c'] \\ &= \Pr[K = m' \oplus c']. \end{aligned} \tag{20}$$

Based on Eq. (4), it can be seen that the elements of $\mathbf{u}$ and $\mathbf{v}$ are uniformly distributed within $(0, 256)$ and random integers within $[1, \hat{q}]$, respectively. Therefore, for any element of $\mathbf{Q}$, it is uniformly taken from integers within $[0, 255]$, which implies that $\Pr[K = m' \oplus c'] = 1/256$. As a result,

$$\Pr[C = c'|M = m'] = \Pr[K = m' \oplus c'] = \frac{1}{256}. \tag{21}$$

For any $c' \in \mathcal{C}$, we then compute

$$\begin{aligned}\Pr[C = c'] &= \sum_{m' \in \mathcal{M}} \Pr[C = c'|M = m'] \cdot \Pr[M = m'] \\ &= \frac{1}{256} \cdot \sum_{m' \in \mathcal{M}} \Pr[M = m'] = \frac{1}{256}. \end{aligned} \tag{22}$$

Finally, based on Bayes' Theorem, we have

$$\begin{aligned}\Pr[M = m'|C = c'] &= \frac{\Pr[C = c'|M = m'] \cdot \Pr[M = m']}{\Pr[C = c']} \\ &= \frac{\frac{1}{256} \cdot \Pr[M = m']}{\frac{1}{256}} \\ &= \Pr[M = m'], \end{aligned} \tag{23}$$

which indicates that the encryption of each element in $\mathbf{I}_p^j$ by $\mathrm{Enc}_K(M)$ is secure against ciphertext-only attacks according to the security definition in [37]. On this basis, and thanks to the assumption that each encryption uses a pair of regenerated $\mathbf{u}$ and $\mathbf{v}$, in other words, the key K is assumed not to be reused each time the encryption oracle is called, we can conclude that Step 1 is secure in value against chosen-plaintext attacks. Considering that in practice, $\mathbf{u}$ and $\mathbf{v}$ can only be generated with pseudorandom numbers rather than true random numbers, the security here is computational. □

For the latter step, i.e., $\mathbf{E}_p^j = \mathbf{H}_l^j \cdot \mathbf{T}^j \cdot \mathbf{H}_r^j$, we call it Step 2 and give the following theorem.

Theorem 2: Step 2 is computationally secure in position against chosen-plaintext attacks if each encryption uses a pair of regenerated $\mathbf{P}_l$ *and* $\mathbf{P}_r$*.*

Step 2 is actually a widely used secure permutation operation in the field of computing outsourcing [38], [39], [40], [41], and its security conclusion, i.e., Theorem 2, has been formally proved by Liao et al. [41], thus we do not go into this proof here.

To sum up, Steps 1 and 2 are aimed at protecting the values and positions of elements in $\mathbf{I}_p$, respectively. In this paper, the encryption of $\mathbf{I}_p$ is a concatenation of Steps 1 and 2. Therefore, according to the conclusions of Theorems 1 and 2, computational security in both value and position against chosen-plaintext attacks can be achieved if each encryption uses a set of regenerated $\mathbf{u}$, $\mathbf{v}$, $\mathbf{P}_l$, and $\mathbf{P}_r$. During the encryption stage of the proposed scheme, at least one of the two keys used in Steps 1 and 2 is regenerated for different $\mathbf{I}_p$ and different channels of $\mathbf{I}_p$, thereby effectively ensuring the security of the image and overfulfilling the security requirements in the case of eavesdropping.

B. Concealment of Watermark Embedding Pattern

For the concealment of watermark embedding pattern in the proposed scheme, we give the following theorem.

Theorem 3: Even with the ability to eavesdrop on all channels of communication between the management center and the screen, the eavesdropper is computationally incapable of obtaining the watermark of the screen.

Proof: Of all the materials that the management center transmits to the screen through the channels, only the personalized decryption key $\mathbb{K}_k = \{\mathbf{U}_k, \mathbf{V}_k, \mathbf{W}_k^+\}$ is related to the watermark of the screen. In the following, we discuss the cracking of $\mathbf{W}_k^+$ and the cracking of $\mathbf{U}_k$ and $\mathbf{V}_k$ separately.

For the former, as shown in Eq. (14), $\mathbf{W}_k$ is concealed by an equal-length $\mathbf{W}^+$ through an XOR operation, where $\mathbf{W}^+$ is a uniform binary sequence randomly generated and secretly held by the management center without disclosure. According to the knowledge of One-Time Pad, this concealment of $\mathbf{W}_k$ is perfect if the generation of $\mathbf{W}^+$ is truly random and each encryption uses a regenerated $\mathbf{W}^+$. However, in the proposed scheme, $\mathbf{W}^+$ is reused and in practice can only be generated with a pseudorandom number generator. In this case, the perfect security is reduced to the computational security against ciphertext-only attacks, which is equivalent to computational security in the eavesdropping scenario.

For the latter, as described in Algorithm 3, the generation of $\mathbf{U}_k$ and $\mathbf{V}_k$ relies on $\mathbf{W}_k^+$, $\overline{\mathbf{U}}$, and $\overline{\mathbf{V}}$. Notably, among these inputs, only $\mathbf{W}_k^+$ is related to the watermark of the screen, and the remaining $\overline{\mathbf{U}}$ and $\overline{\mathbf{V}}$ are unrelated to $\mathbf{W}^+$. Therefore, if $\mathbf{W}_k$ can be cracked from $\mathbf{U}_k$ and $\mathbf{V}_k$, it can be cracked from $\mathbf{W}_k^+$ in the same way with the assistance of $\overline{\mathbf{U}}$ and $\overline{\mathbf{V}}$. In other words, cracking $\mathbf{U}_k$ and $\mathbf{V}_k$ is not an easier task than cracking $\mathbf{W}_k^+$. □

Theorem 3 confirms that even an internal employee cannot obtain the watermark of the screen through eavesdropping. In the following, we proceed with further analysis to verify that the watermark cannot be extracted from the screen-shot image beyond the management center, which further ensures the security of the screen watermark.

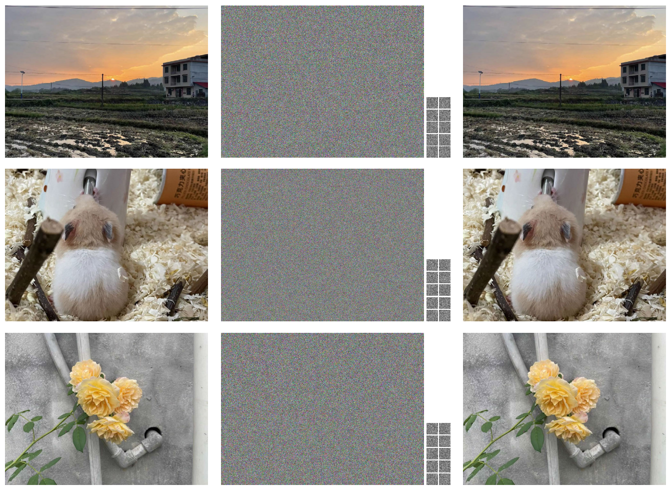


Fig. 7. Visual effects of images at key stages of the proposed scheme. The leftmost column is the original image $\mathbf{I}$. The middle column is the encrypted image, including $\mathbf{E}_p$ and $\mathbb{R}_c$. The rightmost column is the watermarked image $\mathbf{I}_k$ obtained after decryption.

As described in Algorithm 5, the extraction of watermark requires $\mathbf{L}$ as input, which cannot be obtained by the screen. As a result, the screen is prevented from extracting its own watermark from $\mathbf{I}_k$. In the case of ciphertext-only attack, the screen has no other way to acquire the elements of $\mathbf{L}$ than to guess. Since $\mathbf{L}$ is a uniform binary matrix generated using a pseudorandom number generator, the probability of guessing correctly for each element of $\mathbf{L}$ is close to $1/2$ by a negligible margin. Therefore, we conclude that the concealment of embedding pattern imposed by $\mathbf{L}$ is computationally secure against ciphertext-only attacks.

In practical applications, better security of screen watermark can be achieved by changing the $\mathbf{W}^+$ and $\mathbf{L}$ used from time to time. Here, each change of $\mathbf{W}^+$ requires redistributing the personalized key for all screens, but the change of $\mathbf{L}$ does not.

VI. EXPERIMENTS

In this section, we implement the proposed scheme using Matlab programming,[1] show the visual effect, and undertake experimental evaluations with regard to the first and second design goals, specifically to assess the robustness and efficiency performance of the proposed scheme.

In the implementation of the proposed scheme, both length a and width b of the embedding region are set to 64. Each screen is bound to a 24-bit binary watermark sequence, which is then encoded to 63 bits by BCH encoding. In this case, the BCH encoding is able to correct 7-bit errors. The experimental equipment is a desktop computer with Intel(R) Core(TM) i7-10700 CPU and 16 GB of RAM.

A. Visual Effect

As a visual illustration of the flow of the proposed scheme, Fig. 7 shows the visual effects of the original image, encrypted

[1]Our code is available at https://github.com/XIAO-Xiangli/SW.git

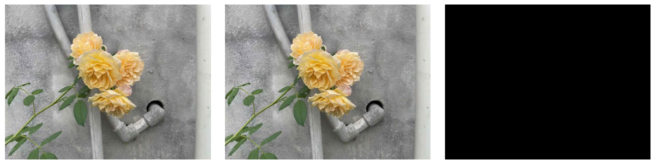


Fig. 8. Comparison of watermarked images generated by the scheme proposed in this paper (image located on the left) and that of Fang et al. [1] (image located in the middle). The image on the right is the difference between the two watermarked images, where all pixels are 0. The embedded watermark and parameter settings are the same. Some key parameters are set as follows: $M = 1365$, $N = 1024$, $r = 2$, $n = 5$, $m = 5$, and the elements of $\mathbf{L}$ are all 1.

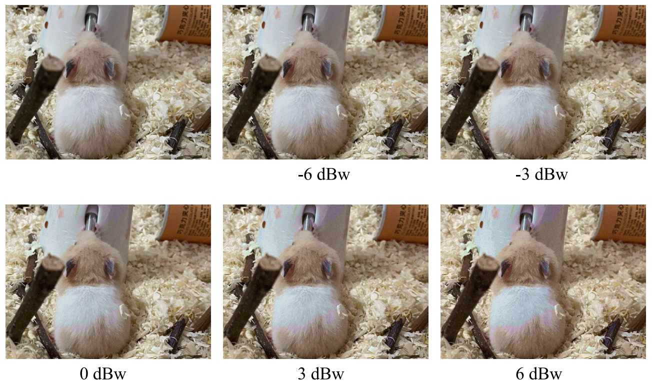


Fig. 9. Visual effect of decrypted images following transmission of encrypted images (including $\mathbf{E}_p$ and $\mathbb{R}_c$) through a lossy channel, resulting in exposure to Gaussian white noise of varying powers. Gaussian white noise is added to the encrypted image via the function "awgn()" of Matlab. The first image is the watermarked image without being affected by noise.

TABLE I
THE IMPACT OF LOSSY CHANNEL NOISE ON THE LOCALIZATION OF KEYPOINTS

Noise power	Coordinates of keypoints
No noise	(199, 751),(129, 552),(877, 912),(967, 838),(277, 34),(344, 472),(802, 983),(551, 163),(63, 675),(462, 262)
-6 dBw	(199, 751),(129, 551),(877, 912),(968, 837),(277, 34),(344, 472),(802, 983),(532, 120),(63, 675),(462, 262)
-3 dBw	(199, 751),(129, 552),(878, 912),(967, 838),(277, 34),(344, 472),(802, 983),(551, 163),(63, 675),(462, 261)
0 dBw	(199, 751),(129, 552),(877, 912),(967, 838),(272, 136),(344, 472),(802, 983),(63, 64),(462, 262),(718, 394)
3 dBw	(199, 751),(129, 552),(877, 912),(967, 837),(277, 34),(455, 987),(344, 472),(551, 162),(50, 676),(718, 394)
6 dBw	(199, 751),(43, 535),(877, 912),(967, 838),(277, 35),(109, 630),(396, 452),(801, 982),(532, 120),(462, 261)

TABLE II
THE TIME (IN SECOND) IT TAKES TO RUN EACH STAGE

	Image encryption	Key distribution	Image decryption	Watermark extraction
1024×1024	0.4477	0.0107	0.0282	0.6357
1532×1532	0.9818	0.0111	0.0621	1.1651
2048×2048	1.8487	0.0105	0.1183	2.0007
2560×2560	2.8916	0.0107	0.1728	3.0349
3072×3072	4.5213	0.0109	0.2380	4.6195
3584×3584	6.0174	0.0134	0.3281	5.9810
4096×4096	8.1462	0.0115	0.4493	8.1251

image, and watermarked image. The encrypted image consists of two parts: the encrypted image body $\mathbf{I}_p$ and the encrypted embedding region $\mathbb{R}_c$. In Fig. 7, the size of the three original images is 1365×1024 and the number n of initial embedding regions is set to 6. After filtering, the number m of remaining watermark embedding regions is set to 5. The embedding strength r is set to 2. As displayed in Fig. 7, the image encrypted by the management center does not visually reveal any information of the original image and the watermarked image obtained after decryption by the screen is almost the same visually as the original image.

B. Robustness

In this paper, watermark embedding follows the algorithm of Fang et al. [1]. For any watermark embedding region, the watermark embedding implemented by the proposed scheme is the same as that implemented by Fang et al., i.e., every pixel in the watermarked region is exactly the same. As a result, the proposed scheme can achieve the same level of robustness in watermark extraction as that achieved by Fang et al. As an illustrative example, Fig. 8 compares the watermarked image generated by the proposed scheme with that generated by Fang et al., which embed the same watermark without discarding any watermark embedding region under identical parameter settings. As can be seen from Fig. 8, the watermarked images generated by both approaches exhibit complete consistency at the bit-level.

Then, we turn to evaluate the robustness of the proposed image encryption algorithm against lossy channels. As can be seen from Fig. 9, the image encryption algorithm proposed in this paper is robust against Gaussian white noise. Even when the encrypted image is contaminated by noise, it can still be successfully decrypted, displaying an acceptable visual effect. Although the decrypted image contains some noisy pixels, it is noteworthy that, as indicated in Table I, these noisy pixels do not have a significant impact on the ISIFT algorithm in terms of keypoint location. Furthermore, the watermark extraction algorithm itself has a substantial tolerance for the location errors of keypoints. Consequently, these noisy pixels will not pose a significant hindrance to watermark extraction.

C. Computational Overhead

We first test the time consumption of the four stages in the proposed scheme using color images of various sizes, and the results are shown in Table II. It can be clearly seen from Table II that the stages of image encryption and watermark extraction demand a greater amount of time, whereas key distribution and image encryption necessitate less time. Notably, the time required for key distribution is the shortest, and its duration is independent of image size. This ensures the scalability of the system, enabling it to maintain stable performance in response to an increase in the number of screens.

In the experiments, n is set to 6, 8, 10, 12, 14, 16, and 18 for color images of sizes 1024×1024, 1536×1536, 2048×2048, 2560×2560, 3072×3072, 3584×3584, and 4096×4096, respectively. After filtering, the number m of remaining watermark embedding regions is set to $0.8n$ (the result will be rounded off). In the following, we carry out a

TABLE III
THE COMPUTATIONAL COST (IN SECOND) ON THE MANAGEMENT CENTER SIDE IN OUR SCHEME AND [1] UNDER DIFFERENT SCREEN NUMBERS AND IMAGE SIZES

		10	50	100	400	1000	3000	5000
1024×1024	Our scheme	0.5920	0.9730	1.4650	4.4160	10.3230	29.8480	51.0210
	[1]-I*	4.4768	21.6498	42.9328	171.7079	431.7597	1305.2247	2251.9745
	[1]-II*	0.9400	3.0664	5.7870	21.9389	54.5988	177.7780	306.8669
2048×2048	Our scheme	2.0690	2.4820	2.9750	5.9840	12.0090	32.1710	52.5240
	[1]-I*	17.4097	86.4322	172.0932	707.9261	1826.9068	5658.1504	–*
	[1]-II*	2.8810	5.7735	11.5243	40.9937	99.8466	343.4152	563.0499
3072×3072	Our scheme	5.0100	5.2600	5.7060	8.7220	14.8880	35.0020	55.0760
	[1]-I*	45.4195	227.6048	444.7742	1729.6861	4650.5821	–*	–*
	[1]-II*	5.5788	9.7766	19.4965	62.7155	151.1909	495.7950	826.2500
4096×4096	Our scheme	7.7750	8.0820	9.1130	12.1440	18.1500	37.9820	57.9380
	[1]-I*	72.8430	391.4299	829.6969	3030.3526	7899.3682	–*	–*
	[1]-II*	9.4651	16.9333	30.6588	99.9020	248.2063	718.2150	1196.5283

* [1]-II represents the case of accelerating calculation by additionally storing the coordinates of the watermark embedding regions, while [1]-I represents the case without such acceleration. '–' indicates that the running time is too long to facilitate testing.

TABLE IV
THE MANAGEMENT CENTER-SIDE EFFICIENCY GAINS ACHIEVED BY OUR SCHEME RELATIVE TO [1] UNDER DIFFERENT SCREEN NUMBERS AND IMAGE SIZES

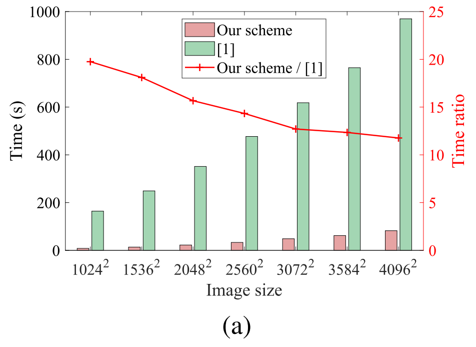
		10	50	100	400	1000	3000	5000
1024×1024	Our scheme / [1]-I	7.56	22.25	29.31	38.88	41.83	43.73	44.14
	Our scheme / [1]-II	1.59	3.15	3.95	4.97	5.29	5.96	6.01
2048×2048	Our scheme / [1]-I	8.41	34.82	57.85	118.30	152.13	175.88	–
	Our scheme / [1]-II	1.39	2.33	3.87	6.85	8.31	10.67	10.72
3072×3072	Our scheme / [1]-I	9.07	43.27	77.95	198.31	312.37	–	–
	Our scheme / [1]-II	1.11	1.86	3.42	7.19	10.16	14.14	15.00
4096×4096	Our scheme / [1]-I	9.37	48.43	91.05	249.54	435.23	–	–
	Our scheme / [1]-II	1.22	2.10	3.36	8.23	13.68	18.91	20.65

meticulous test of the computational costs on the management center side and the screen side. The test results are compared with those of the owner-side watermarking, namely the scheme of Fang et al. [1].

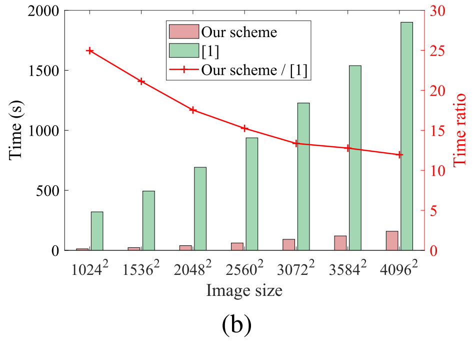
1) Management Center Side: Table III details the computational cost on the management center side of our scheme and Fang et al. scheme [1] under different screen numbers and image sizes. Regarding the scheme proposed by Fang et al., in scenarios with multiple screens, an efficient measure to expedite the computation process involves leveraging additional storage space to store the coordinates of the watermark embedding regions. By doing so, the positioning operation for the watermark embedding regions is necessitated only once. In Table III, we concurrently assess the performance of Fang et al.'s scheme in two distinct scenarios: one incorporating the aforementioned measure and another excluding it. Correspondingly, the efficiency gains achieved by our scheme are shown in Table III. As evident from Table III, our scheme exhibits a notable advantage in terms of management center-side computation efficiency regardless of whether Fang et al.'s scheme adopts the aforementioned measure to accelerate computing, and this advantage becomes increasingly prominent in general as the number of screens and the size of images increase.

In the above experiment, the number of images is assumed to be 1. Next, we assume that the management center needs to distribute 10/20 images to each of 400 screens, and test the computational overhead on the management center side at the settings of [1]-II. The test results are shown in Fig. 10. As demonstrated therein, our scheme is capable of achieving an efficiency gain of over 10 times. We then vary the experimental parameters and acquire the results presented in Fig. 11. It can be observed from the figure that the efficiency gain in this case goes towards tens of times and scales up with the increasing number of images and screens.

2) Screen Side: In the proposed scheme, the screen needs to decrypt the received image, consequently incurring a slight computational overhead. This overhead is tested and the corresponding results are depicted in Fig. 12. From the figure, it can be observed that regardless of whether the image size reaches 4096×4096 or the number of images to be decrypted reaches 60, the time cost on the screen side remains less than 35 seconds. As for the scheme of Fang et al. [1], screens are exempt from computational overhead as they directly receive the watermarked image. Although our scheme introduces some computational overhead on the screen side, it is actually quite small. As shown in Table V, even with a high-resolution image containing 4096×4096 pixels, the decryption process on the

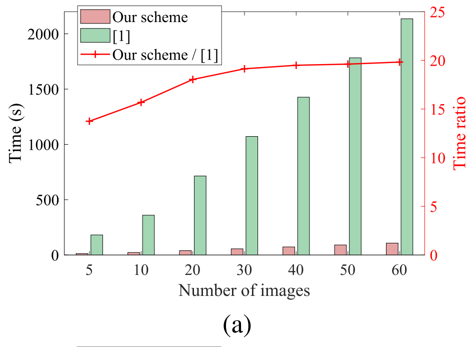


(a)

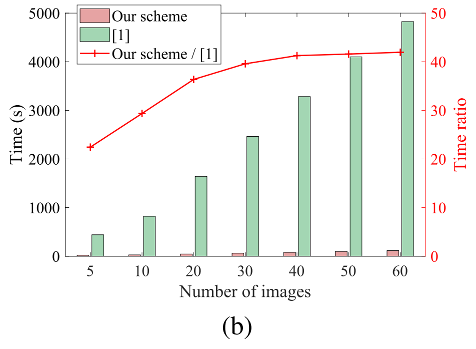


(b)

Fig. 10. The effect of changes in image size on the management center-side efficiency in our scheme and [1]. The bars and polyline correspond to the left and right Y-axes, respectively. The number of screens is set to 400. The number of images in (a) and (b) is set to 10 and 20, respectively.



(a)



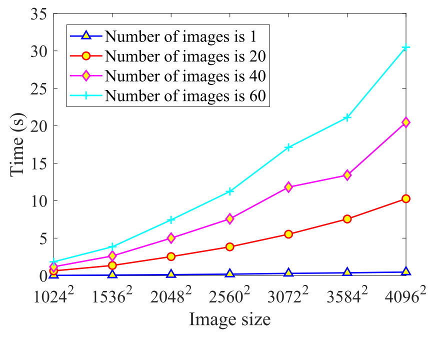
(b)

Fig. 11. The effect of changes in image number on the management center-side efficiency in our scheme and [1]. The bars and polyline correspond to the left and right Y-axes, respectively. The image size is set to 2048×2048. The number of screens in (a) and (b) is set to 400 and 1000, respectively.

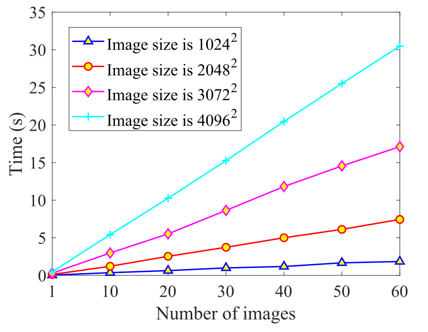
TABLE V
COMPARISON OF OUR SCHEME WITH [14] REGARDING SCREEN-SIDE (USER-SIDE) COMPUTING TIME

		1	5	10	20
1024^2	Our scheme	0.03	0.17	0.29	0.63
	[14]-I*	1.38	6.68	13.41	25.82
	[14]-II*	0.84	4.07	8.06	16.06
2048^2	Our scheme	0.12	0.64	1.18	2.53
	[14]-I*	6.04	28.49	57.54	115.52
	[14]-II*	3.71	19.43	39.35	77.96
3072^2	Our scheme	0.30	1.41	2.65	5.52
	[14]-I*	12.90	65.10	130.04	269.68
	[14]-II*	9.37	45.97	91.91	183.47
4096^2	Our scheme	0.47	2.40	4.71	10.26
	[14]-I*	24.42	118.33	241.78	493.73
	[14]-II*	17.14	80.09	163.98	329.78

* [14]-I and [14]-II refer to embedding watermarks in the DCT domain and the spatial domain, respectively, utilizing the scheme of Celik *et al.* [14].



(a)



(b)

Fig. 12. The computational cost on the screen side in our scheme under different image sizes and numbers. (a) The impact of image size on the cost. (b) The impact of the number of images on the cost.

screen side takes less than 0.5 seconds. Moreover, Table V also compares this overhead to a classic design of client-side watermarking [14]. While this design is somewhat outdated, subsequent studies have not focused on improving its user-side efficiency. Evidently, our scheme outperforms [14] significantly, primarily because our image decryption mechanism avoids large-scale matrix multiplication operations, which are necessary in [14].

D. Communication Overhead

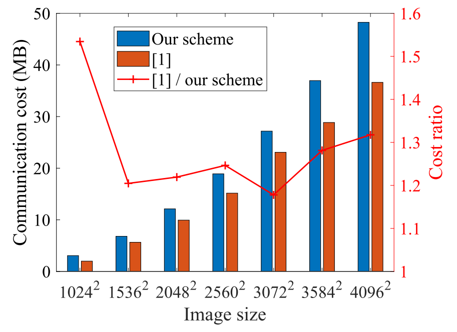
Before evaluating the communication overhead on the management center side and the screen side in detail, we first test the transmission cost of each component in our scheme using color images of various sizes, and the results are presented in Table VI. Here, we measure the communication cost by the size of the file. As can be seen from the table, the encrypted

TABLE VI
THE COMMUNICATION COST (IN KB) OF TRANSMITTING EACH COMPONENT

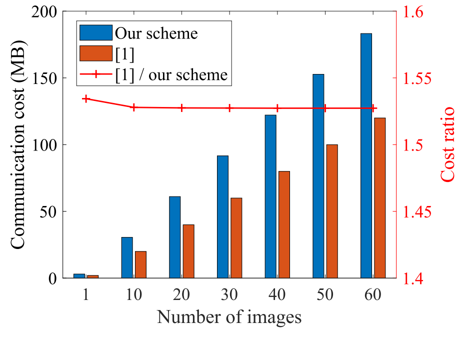
	Multicast communication									On-demand communication		
	$\mathbb{R}_c$	$\mathbf{E}_p$	$\mathbf{u}$	$\mathbf{v}$	$\overline{\mathbf{P}}_l$	$\overline{\mathbf{P}}_r$	$\mathbb{K}_p$	$\mathbf{P}_l$	$\mathbf{P}_r$	$\mathbf{U}_k$	$\mathbf{V}_k$	$\mathbf{W}_k^+$
1024×1024	40.2705	3075.9561	7.7656	1.7422	0.4346	0.4336	0.3633	4.8174	4.8154	4.0010	0.9883	0.1963
1536×1536	48.2803	6884.6416	11.5195	2.4990	0.4863	0.4854	0.3633	7.3672	7.3652			
2048×2048	64.2998	12303.2061	15.3379	3.2695	0.5713	0.5713	0.3711	9.8916	9.8955			
2560×2560	80.3193	19223.6064	19.1201	4.0361	0.6758	0.6729	0.3848	12.6680	12.6631			
3072×3072	88.3291	27681.9180	22.9053	4.7939	0.7227	0.7207	0.3877	15.4346	15.4385			
3584×3584	104.3486	37678.1328	26.6934	5.5508	0.8145	0.8125	0.3945	18.1318	18.1270			
4096×4096	112.3584	49212.2061	30.4980	6.3271	0.8672	0.8652	0.3994	20.8408	20.8359			

TABLE VII
THE COMMUNICATION COST (IN MB) ON THE MANAGEMENT CENTER SIDE IN OUR SCHEME AND [1]

Image size	Number of images	Number of screens	Our scheme		[1]
			Multicast	On-demand	On-demand
1024^2	1	10	30.6273	0.0506	19.9938
		50	153.1365	0.2532	99.9688
		100	306.2730	0.5065	199.9375
		400	1225.0919	2.0260	799.7501
		1000	3062.7298	5.0650	1999.3753
		3000	9188.1895	15.1949	5998.1260
		5000	15313.6492	25.3248	9996.8767
	10	10	305.4263	0.0506	199.9375
	20		610.7586		399.8751
	30		916.0909		599.8126
	40		1221.4232		799.7501
	50		1526.7555		999.6877
	60		1832.0877		1199.6252
1532^2	1		67.9981		56.4849
2048^2			121.1662		99.4146
2560^2			189.0053		151.6710
3072^2			271.7837		230.7790
3584^2			369.6583		288.5364
4096^2			482.4726		366.2500



(a)



(b)

Fig. 13. The communication cost on the screen side in our scheme and [1] under different image sizes and numbers. The bars and polyline correspond to the left and right Y-axes, respectively. (a) The impact of image size on the cost. (b) The impact of the number of images on the cost.

image body $\mathbf{E}_p$ exhibits a significantly larger size compared to other components. Notably, in our solution, $\mathbf{E}_p$ is transmitted via multicast, thus effectively bypassing the primary source of communication cost. As for $\mathbf{U}_k$, $\mathbf{V}_k$, and $\mathbf{W}_k^+$, which require on-demand transmission, their combined size is merely a few KB, facilitating the system scalability when accommodating a substantial number of screens.

In the experiments, the value of n is set as in the previous subsection. In the following, we proceed to thoroughly evaluate and compare the communication overhead of our scheme and the scheme of Fang et al. [1], separately considering the management center and screen perspectives.

1) Management Center Side: Table VII presents a side-by-side comparison of the screen-side communication overhead between our scheme and Fang et al. scheme [1], with varying parameter settings. As evident from the table, our scheme is able to convert the vast majority of on-demand communica-tion overhead into more bandwidth-saving multicast communication overhead, with only a minimal amount of on-demand communication overhead left. Meanwhile, it can be observed that the multicast communication overhead in our scheme is slightly higher, which is mainly due to the size expansion of encrypted images. This is not a hindrance because multicast communication is far more efficient than multiple one-to-one on-demand communications, as tested by Lin and Ni [18]. In addition, even in cases where the same image is required for different screens at different times, the availability of multicast can still be ensured with the help of techniques such as coded caching [20], [21].

2) Screen Side: Fig. 13 displays the test results regarding the communication cost on the screen side. As can be observed therein, our scheme incurs a slight increase in the communication cost on the screen side compared to Fang et al. scheme [1]. This increase is also primarily attributed to the hindrance caused by the encrypted image to the compression effect of the image compression algorithm, leading to an expansion of the size. Given that the increase in screen-side communication overhead is only about 1.5 times, we do not anticipate it to be an excessive burden on the screens.

VII. Conclusion

This paper designs the first screen-shooting resilient image watermarking scheme with client-side embedding mode. In the proposed scheme, the screen-shooting resilient watermark is embedded in a manner that follows the existing algorithm [1] without sacrificing robustness. The most important difference is that the screen watermarks are embedded at the same time as the image is decrypted, rather than being embedded one by one in the plaintext domain by the management center. As a result, the proposed scheme has significantly higher efficiency on the management center side than existing ones of owner-side embedding, thereby better ensuring the system scalability. This efficiency advantage enables the proposed scheme to be suitable for situations with a large number of screens.

While the efficiency improvements on the management center side do lead to some additional overhead on the screen side, this additional overhead is minor and will not constrain the application. The primary restriction of the proposed scheme lies in its inability to further enhance the robustness of current state-of-the-art screen-shooting resilient image watermarking methods. Consequently, exploring schemes to simultaneously implement client-side embedding and enhance robustness remains a crucial area of focus for our future research.

References

[1] H. Fang, W. Zhang, H. Zhou, H. Cui, and N. Yu, "Screen-shooting resilient watermarking," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 6, pp. 1403–1418, Jun. 2019.

[2] W. Chen, N. Ren, C. Zhu, Q. Zhou, T. Seppänen, and A. Keskinarkaus, "Screen-cam robust image watermarking with feature-based synchronization," *Appl. Sci.*, vol. 10, no. 21, p. 7494, Oct. 2020.

[3] W. Chen, C. Zhu, N. Ren, T. Seppänen, and A. Keskinarkaus, "Screen-cam robust and blind watermarking for tile satellite images," *IEEE Access*, vol. 8, pp. 125274–125294, 2020.

[4] Y. Cheng, X. Ji, L. Wang, Q. Pang, Y.-C. Chen, and W. Xu, "mID: Tracing screen photos via Moiré patterns," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 2969–2986.

[5] H. Fang et al., "Deep template-based watermarking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 4, pp. 1436–1451, Apr. 2021.

[6] H. Fang et al., "A camera shooting resilient watermarking scheme for underpainting documents," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 11, pp. 4075–4089, Nov. 2020.

[7] X. Yang, W. Zhang, H. Fang, Z. Ma, and N. Yu, "Language universal font watermarking with multiple cross-media robustness," *Signal Process.*, vol. 203, Feb. 2023, Art. no. 108791.

[8] Y.-G. Wang, G. Zhu, and Y.-Q. Shi, "Transportation spherical watermarking," *IEEE Trans. Image Process.*, vol. 27, no. 4, pp. 2063–2077, Apr. 2018.

[9] Y. Huang, B. Niu, H. Guan, and S. Zhang, "Enhancing image watermarking with adaptive embedding parameter and PSNR guarantee," *IEEE Trans. Multimedia*, vol. 21, no. 10, pp. 2447–2460, Oct. 2019.

[10] L. Y. Zhang, Y. Zheng, J. Weng, C. Wang, Z. Shan, and K. Ren, "You can access but you cannot leak: Defending against illegal content redistribution in encrypted cloud media center," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 6, pp. 1218–1231, Nov. 2020.

[11] R. Hu and S. Xiang, "Cover-lossless robust image watermarking against geometric deformations," *IEEE Trans. Image Process.*, vol. 30, pp. 318–331, 2021.

[12] F. Peng, W. Jiang, M. Long, and K. Li, "A reversible watermarking for 2D engineering graphics based on difference expansion with adaptive interval partitioning," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 3, pp. 1867–1881, May/Jun. 2023.

[13] A. Adelsbach, U. Huber, and A.-R. Sadeghi, "Fingercasting—Joint fingerprinting and decryption of broadcast messages," *Transactions on Data Hiding and Multimedia Security II*. Berlin, Germany: Springer, 2007, pp. 1–34.

[14] M. U. Celik, A. N. Lemma, S. Katzenbeisser, and M. van der Veen, "Lookup-table-based secure client-side embedding for spread-spectrum watermarks," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 3, pp. 475–487, Sep. 2008.

[15] A. Piva, T. Bianchi, and A. De Rosa, "Secure client-side ST-DM watermark embedding," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 1, pp. 13–26, Mar. 2010.

[16] T. Bianchi and A. Piva, "TTP-free asymmetric fingerprinting based on client side embedding," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 10, pp. 1557–1568, Oct. 2014.

[17] T. Bianchi, A. Piva, and D. Shullani, "Anticollusion solutions for asymmetric fingerprinting protocols based on client side embedding," *EURASIP J. Inf. Secur.*, vol. 2015, no. 1, pp. 1–17, Dec. 2015.

[18] X. Lin and L. M. Ni, "Multicast communication in multicomputer networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 10, pp. 1105–1117, Oct. 1993.

[19] W. Gao, Q. Li, B. Zhao, and G. Cao, "Social-aware multicast in disruption-tolerant networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1553–1566, Oct. 2012.

[20] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 836–845, Apr. 2016.

[21] A. Tang, S. Roy, and X. Wang, "Coded caching for wireless backhaul networks with unequal link rates," *IEEE Trans. Commun.*, vol. 66, no. 1, pp. 1–13, Jan. 2018.

[22] H. Fang, Z. Jia, Z. Ma, E.-C. Chang, and W. Zhang, "PIMoG: An effective screen-shooting noise-layer simulation for deep-learning-based watermarking network," in *Proc. 30th ACM Int. Conf. Multimedia*, Oct. 2022, pp. 2267–2275.

[23] W. Chen, N. Ren, C. Zhu, A. Keskinarkaus, T. Seppänen, and Q. Zhou, "Joint image encryption and screen-cam robust two watermarking scheme," *Sensors*, vol. 21, no. 3, p. 701, Jan. 2021.

[24] X. Dong, W. Zhang, H. Fang, and N. Yu, "Interactive screen-shooting resilient ciphertext watermarking protocol between the cloud and the user," *SCIENTIA SINICA Informationis*, vol. 52, no. 7, pp. 1272–1286, Jul. 2022.

[25] R. Anderson and C. Manifavas, "Chameleon—A new kind of stream cipher," in *Proc. Int. Workshop Fast Softw. Encryp.*, 1997, pp. 107–113.

[26] C.-M. Pun, J.-J. Jiang, and C. L. P. Chen, "Adaptive client-side LUT-based digital watermarking," in *Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Nov. 2011, pp. 795–799.

[27] J.-H. Sun, Y.-H. Lin, and J.-L. Wu, "Secure client side watermarking with limited key size," in *Proc. 21st Int. Conf.*, 2015, pp. 13–24.

[28] X. Xiao, Y. Zhang, Y. Zhu, P. Hu, and X. Cao, "FingerChain: Copyrighted multi-owner media sharing by introducing asymmetric fingerprinting into blockchain," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 2869–2885, Sep. 2023.

[29] P. Prangjarote, C.-Y. Lin, L.-W. Kang, and C.-H. Yeh, "Joint fingerprinting and decryption for VQ images through bipartite matching," in *Proc. 6th Int. Conf. Genetic Evol. Comput.*, Aug. 2012, pp. 27–30.

[30] C.-Y. Lin, P. Prangjarote, L.-W. Kang, W.-L. Huang, and T.-H. Chen, "Joint fingerprinting and decryption with noise-resistant for vector quantization images," *Signal Process.*, vol. 92, no. 9, pp. 2159–2171, Sep. 2012.

[31] C.-Y. Lin, P. Prangjarote, C.-H. Yeh, and H.-F. Ng, "Reversible joint fingerprinting and decryption based on side match vector quantization," *Signal Process.*, vol. 98, pp. 52–61, May 2014.

[32] B. Czaplewski, "Joint fingerprinting and decryption method for color images based on quaternion rotation with cipher quaternion chaining," *J. Vis. Commun. Image Represent.*, vol. 40, pp. 1–13, Oct. 2016.

[33] B. Czaplewski and R. Rykaczewski, "Matrix-based robust joint fingerprinting and decryption method for multicast distribution of multimedia," *Signal Process.*, vol. 111, pp. 150–164, Jun. 2015.

[34] C.-Y. Lin, K. Muchtar, C.-H. Yeh, and C.-S. Lu, "Secure multicasting of images via joint privacy-preserving fingerprinting, decryption, and authentication," *J. Vis. Commun. Image Represent.*, vol. 38, pp. 858–871, Jul. 2016.

[35] H. Mareen, J. D. Praeter, G. V. Wallendael, and P. Lambert, "A scalable architecture for uncompressed-domain watermarked videos," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 6, pp. 1432–1444, Jun. 2019.

[36] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, Oct. 1999, pp. 1150–1157.

[37] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Boca Raton, FL, USA: CRC Press, 2020.

[38] X. Lei, X. Liao, T. Huang, H. Li, and C. Hu, "Outsourcing large matrix inversion computation to a public cloud," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 78–87, Jan./Jun. 2013.

[39] X. Lei, X. Liao, T. Huang, and H. Li, "Cloud computing service: The caseof large matrix determinant computation," *IEEE Trans. Services Comput.*, vol. 8, no. 5, pp. 688–700, Sep. 2015.

[40] S. Salinas, C. Luo, X. Chen, W. Liao, and P. Li, "Efficient secure outsourcing of large-scale sparse linear systems of equations," *IEEE Trans. Big Data*, vol. 4, no. 1, pp. 26–39, Mar. 2018.

[41] W. Liao, C. Luo, S. Salinas, and P. Li, "Efficient secure outsourcing of large-scale convex separable programming for big data," *IEEE Trans. Big Data*, vol. 5, no. 3, pp. 368–378, Sep. 2019.

Xiangli Xiao received the B.E. degree from the College of Electronic and Information Engineering, Southwest University, Chongqing, China, in June 2020. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. His current research interests include multimedia security, digital watermarking, blockchain, and cloud computing security.

Yushu Zhang (Senior Member, IEEE) received the B.S. degree from the School of Science, North University of China, Taiyuan, China, in 2010, and the Ph.D. degree from the College of Computer Science, Chongqing University, Chongqing, China, in 2014. He held various research positions with the City University of Hong Kong, Southwest University, the University of Macau, and Deakin University. He is currently a Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include multimedia security, blockchain, and artificial intelligence. He has coauthored more than 200 refereed journal articles and conference papers in these areas. He is an Associate Editor of *Information Sciences*, *Journal of King Saud University-Computer and Information Sciences*, and *Signal Processing*.

Zhongyun Hua (Senior Member, IEEE) received the B.S. degree from Chongqing University, Chongqing, China, in 2011, and the M.S. and Ph.D. degrees from the University of Macau, Macau, China, in 2013 and 2016, respectively. He is currently an Associate Professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His works have appeared in prestigious venues, such as IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON SIGNAL PROCESSING, and ACM Multimedia. His current research interests include chaotic systems, multimedia security, and secure cloud computing. He has published about seventy articles on the subject and received more than 6700 citations. He has been recognized as a "Highly Cited Researcher 2022".

Zhihua Xia (Member, IEEE) received the B.S. degree from Hunan City University, Yiyang, China, in 2006, and the Ph.D. degree in computer science and technology from Hunan University, China, in 2011. He is currently a Professor with the College of Cyber Security, Jinan University, Guangzhou, China. His research interests include digital forensic and encrypted image processing.

Jian Weng (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from Shanghai Jiao Tong University in 2008. From 2008 to 2010, he held a post-doctoral position with the School of Information Systems, Singapore Management University. He is currently a Professor and the Vice President of Jinan University. He has published more than 100 papers in cryptography and security conferences and journals, such as *Cryptography*, Eurocrypt, Asiacrypt, ACM CCS, and USENIX Security. His research interests include cryptography, data security, and blockchain. He received the Innovation Award from Chinese Association for Cryptologic Research in 2015 and the National Science Fund for Distinguished Young Scholars in 2018. He served as a PC member or the PC co-chair for more than 50 international conferences.