# RDIAS: Robust and Decentralized Image Authentication System

ALI GHORBANPOUR and MOHAMMAD AMIN ARAB, School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada
MOHAMED HEFEEDA, School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canadaand Qatar Computing Research Institute, Hamad Bin Khalifa University, Doha, Qatar

Recent AI tools can subtly manipulate images, eroding users' trust in the authenticity of images they see on their displays. Current image authentication methods either detect artifacts that may result from manipulations or attach hashes of images as metadata for users to verify. The efficacy of the first approach is rapidly deteriorating with the continuous improvements in AI tools, leading to missing many serious manipulations. Hashes become invalid once images are subjected to any processing, such as re-sizing and transcoding. This makes the second approach impractical as most platforms, e.g., Facebook and X, perform several *legitimate* operations on images. Further, most platforms remove the metadata attached to images. We propose RDIAS, a robust and practical image authentication system. RDIAS securely embeds representative fingerprints into images without damaging their visual quality. We design these fingerprints to robustly detect malicious manipulations, e.g., adding/removing objects, while tolerating legitimate operations, e.g., image resizing and transcoding. Rigorous evaluation of RDIAS with diverse image datasets and realistic manipulations conducted by human subjects utilizing AI tools shows its high accuracy and efficiency. For example, RDIAS detects DeepFake manipulations that change facial features/expressions with an accuracy of 99%. The results also show that RDIAS preserves image quality and verifies authenticity in real time.

CCS Concepts: • **Applied computing** → **Computer forensics**;

Additional Key Words and Phrases: Multimedia Forensics, Image Authentication, Image Watermarking, Image Fingerprinting, Error Correction Code

**ACM Reference format:**
Ali Ghorbanpour, Mohammad Amin Arab, and Mohamed Hefeeda. 2025. RDIAS: Robust and Decentralized Image Authentication System. *ACM Trans. Multimedia Comput. Commun. Appl.* 21, 10, Article 303 (October 2025), 28 pages.
https://doi.org/10.1145/3760260

Authors' Contact Information: Ali Ghorbanpour, School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada; e-mail: ali_ghorbanpour@sfu.ca; Mohammad Amin Arab, School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada; e-mail: maarab@sfu.ca; Mohamed Hefeeda (corresponding author), School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada and Qatar Computing Research Institute, Hamad Bin Khalifa University, Doha, Qatar; e-mail: mhefeeda@sfu.ca.


ACM 1551-6865/2025/10-ART303
https://doi.org/10.1145/3760260

## 1 Introduction

Recent advances in AI have improved many aspects of everyday life, from assisting in the early diagnosis of dangerous diseases to summarizing documents and generating content. However, such advances have also introduced major concerns for society. One of these concerns is the erosion of trust in the authenticity of visual content, especially images. Specifically, AI tools, e.g., Adobe Photoshop [2], Adobe Firefly [1], Canva [15], and OpenAI DALL·E [38], have made it relatively easy to manipulate images by, for example, adding, removing, or modifying objects to alter the semantic meaning of images. AI tools can also blend various images and backgrounds as well as generate new images from scratch.

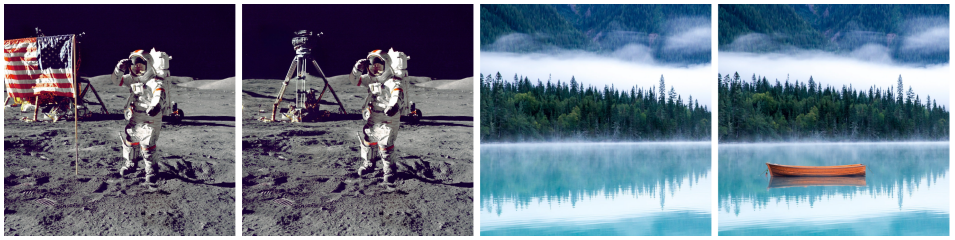
The goal of this article is to *partially* address the content authenticity concern. Specifically, we consider the problem of authenticating images published on *arbitrary* image-hosting platforms such as social media and news Web sites. Addressing this complex research problem faces multiple challenges. First, image-hosting platforms typically transcode images, remove/modify associated metadata, and resize images to fit their storage systems and user interfaces. While these operations do not change the semantic meaning of images, they may completely change the bits representing these images. Thus, approaches that rely on computing digital signatures or hashes of images and attaching them as metadata, e.g., [14, 19] would fail in practice. To demonstrate this, in Section 2.2, we evaluate the recent framework for image authentication developed by the **Coalition for Content Provenance and Authenticity (C2PA)** [14] and show that it does not support common social media sites such as Facebook and X, which is a major limitation.

The second challenge is the wide availability of AI tools and computing power that allow attackers to manipulate images in sophisticated and hard-notice ways. We demonstrate examples of such manipulations in Figure 1, which we easily created using the Adobe Firefly tool [1]. In addition, and perhaps more seriously, recent deep-learning models may enable attackers to remove/change/transfer watermarks embedded in them. Specifically, several methods, e.g., [4, 8, 13, 21, 58] have been proposed in the literature to embed watermarks in images using deep-learning models. These watermarks may carry authentication information, such as the identity of the image creator. However, attackers can use AI models to, for example, extract the watermark from an authentic image and embed it in a totally different one, tricking users into trusting manipulated images. Embedding simple image hashes as watermarks may not solve this problem, as attackers can replace them with hashes of the manipulated image. Attackers may also transcode the image with a different encoder and/or quality factor as well as apply simple image filters, complicating the authentication process as these operations invalidate the embedded watermarks.

The third challenge is that image quality is critically important for users. Thus, the authentication system cannot degrade the quality of images for the sake of ensuring their authenticity. This makes it harder to *imperceptibly* embed authentication information into images. It also limits the amount of information that can be embedded.

Ensuring *robustness* is the fourth challenge for designing practical image authentication systems. Robustness is the ability to reliably authenticate images in the presence of *legitimate* operations performed by hosting sites, such as image resizing and transcoding. It also means detecting manipulations even if attackers attempt to hide them by applying operations such as image filtering and noise addition. Robustness could potentially be achieved by embedding more information in images to consider various operations. This, however, would negatively impact the visual quality. Effectively managing the *tradeoff* between robustness and quality (or equivalently, imperceptibility of the embedded information) further complicates this challenge for image authentication systems.

The fifth and final challenge is the massive volume of images that need to be authenticated. This requires the image authentication system to be computationally efficient, i.e., *scalable*.



(a) Original (b) Flag Removed (c) Original (d) Boat Added

Fig. 1. Examples of AI-powered image manipulations performed using Adobe Firefly.

In this article, we propose RDIAS, an end-to-end system for robust image authentication. We design RDIAS to efficiently and practically address the above-mentioned challenges. At a high level, RDIAS *securely* and *robustly* embeds *representative* authentication information into images without damaging their visual quality. This authentication information, which we call fingerprint, is created based on the image's semantic contents using a perceptual hashing method. We design fingerprints that detect malicious manipulations, e.g., adding/removing objects, while tolerating legitimate operations, e.g., image resizing and transcoding. We secure the fingerprint embedding by a private key held by the image creator, preventing attackers from changing the fingerprint or transferring it to other images.

Verifying the authenticity of an image is done by first extracting the fingerprint and then decrypting it using the image creator's public key. However, fingerprint embedding and extraction are done using deep-learning models, which produce non-deterministic results. This means some bits of the extracted fingerprint may differ from the original ones, compromising the system's robustness. In addition, the various legitimate and malicious operations performed on the image may also affect the fingerprint, further diminishing robustness. To achieve robust authentication, we model the image as a *noisy* communication channel and the embedded fingerprint as the message to be sent on that channel. We then utilize an **Error-Correcting Code (ECC)** to *reliably* embed and extract fingerprints from images.

We note that this is a *systems* paper, which designs a novel and practical image authentication system composed of several components, including perceptual hashing, watermarking, and ECC. Numerous methods exist in the literature for each component, with different goals and assumptions that do not necessarily consider the characteristics of the image authentication problem. Our contributions include identifying the most suitable method for each component that meets the requirements of image authentication and analyzing the tradeoffs among the various components. In particular, the contributions of this article are as follows:

- We analyze the state-of-the-art image authentication framework, C2PA [14], and demonstrate its limitations when deployed on social media platforms in Section 2.2.
- We propose a robust image authentication system, RDIAS, that meets the requirements of common image-hosting sites, including image resizing and transcoding. This is presented in Section 3.
- We analyze the suitability of current perceptual hashing methods for creating representative fingerprints for image authentication systems in Section 4.
- We compare various methods for embedding information in images (aka watermarking) and analyze their performance tradeoffs in Section 5.
- We propose utilizing ECCs to improve the robustness of image authentication systems, as described in Section 6.

—We implement all components of RDIAS and evaluate its performance with diverse image datasets, complex manipulations, and transformations performed by social media platforms, e.g., Facebook. We recruit ten human subjects to realistically manipulate 1,300 diverse images using AI tools. Our results show the accuracy and robustness of RDIAS under a wide range of practical scenarios. For example, RDIAS achieves an accuracy of up to 98.11% for detecting AI-powered manipulations that add/remove/change objects in arbitrary images. For DeepFake face manipulations, RDIAS achieves an accuracy of up to 99.0%, with 100% Recall (*all* manipulated faces were identified) and 98.0% Precision (a very low false alarm rate of 2%). The results also show that RDIAS is computationally efficient: it immunizes an image in under 850 ms and verifies authenticity in under 50 ms on average, indicating that it can run in real time. Our evaluation is presented in Section 7.

## 2 Related Work and Case Study

This section provides a brief background on image authentication and summarizes the related works in the literature. It also analyzes the recent C2PA authentication framework as a case study.

### 2.1 Background and Related Work

*Image Authentication.* The goal of image authentication methods is to verify the authenticity of images by ensuring that they have not been manipulated [48]. These methods are essential in systems and applications where image integrity is critical, such as digital forensics [5], medical imaging [7], and photojournalism [41]. They are also useful in fighting misinformation and detecting forged images [33].

We divide image authentication methods into two main categories: passive and active. The former detects artifacts introduced by manipulations, while the latter computes and attaches authentication information to images. We describe each category in the following.

*Passive Authentication Methods.* These methods generally rely on detecting inconsistencies and artifacts left by manipulations to identify and sometimes localize image alterations. These methods can further be divided into two sub-categories: those that depend on the type of forgery and those that focus solely on artifacts. For example, some approaches target specific manipulation types, such as copy-move forgeries [31, 53] and inpainting [52], leveraging the unique characteristics of each to improve the detection accuracy. Others rely purely on the artifacts or inconsistencies introduced by manipulations, such as forensic pattern analysis or error-level analysis, often combining these techniques with deep neural networks for more effective classification and localization [3, 11].

The effectiveness of passive methods is diminishing due to recent advances in AI tools, which can realistically edit images without leaving significant visual artifacts or inconsistencies. Nonetheless, they remain useful, especially when an image has not been secured before distribution.

*Active Authentication Methods.* Active image authentication methods aim to *proactively* protect images from future manipulations. They compute authentication information from the original images. They can be divided into two sub-subcategories: digital signatures and data hiding. Digital signatures involve generating a unique signature from the original image, which changes if the image is manipulated or edited. By comparing the signatures of the original and the queried image, we can determine whether the image has been altered and confirm its authenticity [17, 25, 50]. On the other hand, data hiding involves embedding information, such as a unique identifier or a representation of the image, directly into the pixel values in a way that is imperceptible to the human eye. This embedded information can later be extracted to verify the authenticity of the image [26, 55, 56].

The key shortcomings of current active image authentication methods are their limited robustness and accuracy. For example, digital signatures attached as metadata are usually removed by hosting

sites. Even if not removed, signatures may become invalid because of normal operations that hosting sites typically perform on images, such as resizing and transcoding. Similarly, current active methods that embed information into images may be vulnerable in the presence of AI-powered attackers [6]. Such attackers may remove the authentication information from image pixels, change the image, and recompute and embed new authentication information without leaving noticeable visual artifacts [28, 42].

The proposed system in this article belongs to the active image authentication category, as it embeds authentication information into images. However, it is designed to be robust, secure, and accurate even in the presence of powerful attackers.

*Image Manipulations versus Transformations.* Images can be subjected to numerous operations during distribution. Some operations are malicious, i.e., performed by attackers to mislead users, which we refer to as *manipulations*. Other operations can be performed by hosting sites and are considered part of the normal processing; we call these operations *transformations*.

We consider AI-powered attackers, which use AI tools to alter images and blend the modifications with the rest of the images, leaving minimal inconsistencies and visual artifacts, as shown by the examples in Figure 1. This is unlike many previous image authentication works, which consider simpler manipulations such as cropping out an area in the image and copying an object from one location to another. Such manipulations are less challenging to detect.

We divide manipulations into three *broad* categories: (i) object addition, (ii) object removal, and (iii) cropping. The first adds one or more objects to an image, while the second deletes object(s). The third category removes areas from the image. All use AI to hide the modifications and smooth out the resulting image. These three categories and their combinations cover all practical malicious manipulations, including the ones reported in prior works.

For transformations, we consider realistic scenarios where hosting sites, e.g., Facebook and X, typically perform multiple operations on images before storing and serving them. These include two main operations: (i) transcoding and (ii) resizing. Transcoding means recompressing the image with a different encoder and/or using different compression parameters. For example, as shown by the case study below, Facebook transcodes all uploaded images to JPEG with a quality factor between 70 and 90, depending on the image size. Image resizing is also a common operation performed by most hosting sites, where the size is adjusted to match the characteristics of the storage, processing, and user interface systems of the hosting site.

In addition to the legitimate operations by hosting sites, we expand the considered transformations to include two other categories: (i) filtering and (ii) noise addition. The first allows different filters to be applied to images, such as averaging neighboring pixels. The second category subjects the image pixels to different types of noise, such as Gaussian noise. Including these additional categories makes our authentication system more robust and practical. This is because sophisticated attackers may apply various filters or add slight noise to images to hide their manipulations.

### 2.2 Case Study: Analysis of C2PA

Due to the increased importance of the image authentication problem, recent works in industry and academia have started designing complete end-to-end frameworks, e.g., C2PA [14] and AMP [19]. C2PA is currently considered the most comprehensive, state-of-the-art framework. It is designed by a coalition of major companies, including Adobe, BBC, and Microsoft.

C2PA defines APIs for publishers to compute hashes of their images and attach them as metadata. When a user receives an image, they can compute the hash of the image and compare it against the one in the metadata. C2PA also allows storing a copy of the hash on a secure cloud server in case the metadata is removed from the image. In this case, the user downloads the original hash from the cloud server.

We analyze the suitability of C2PA for images distributed over *social media platforms*. We implemented an image authentication application using the C2PA Python APIs [14]. We tested our implementation with three social media platforms: Facebook, X, and Instagram. We computed the hashes of 20 different images and attached them as metadata to the image files according to the C2PA guidelines. We ensured the metadata's correctness in all cases using the C2PA verification tool and the C2PA online verification Web site. We then uploaded the image files to the three platforms. Shortly after, we downloaded the same images to verify their authenticity locally. We did not manipulate the images before or after uploading/downloading them. In addition, we kept local copies of the metadata as if they were downloaded from a secure cloud server.

To our surprise, the authenticity verification of *all* 20 images downloaded from each of the three social media platforms *failed*, even when we used the local copies of the metadata. We inspected the downloaded files and found that all platforms removed the metadata. Thus, we had to rely on the local copies of the metadata. However, each platform significantly changed the bits of the images, invalidating the hashes and forcing the verification process to fail. For example, Facebook resized any image with a dimension greater than 2,048 pixels down to 2,048 pixels. X performs a similar adjustment for large images but resizes them to 4,096 pixels. Instagram takes a slightly different approach: it reduces images with dimensions between 1,080 and 2,160 pixels down to 1,080 pixels and those exceeding 2,160 pixels down to 2,160 pixels on the larger side. Further, each platform transcoded the images into different qualities and utilized various codecs. Thus, the resulting images have vastly different sizes from the original ones. Specifically, the total size of the original 20 images was 99.5 MB, whereas the total sizes of the downloaded ones from Facebook, X, and Instagram were 17.7, 12.3, and 4.9 MB, respectively.

*In summary*, social media platforms remove the metadata containing the authentication information, and they resize and encode images with different ratios and quality parameters, making the C2PA framework fail when used with such platforms. C2PA is effective when images are hosted on the publishers' sites, e.g., BBC and CNN, where the metadata is kept and the operations performed on images are limited and known ahead of time. Publishers' sites, however, represent a tiny fraction of locations distributing images.

## 3 Proposed Solution

This section first specifies the design objectives of RDIAS and summarizes our approaches to achieve them. It then presents the overall architecture of RDIAS and its operation. Finally, it discusses various practical considerations and limitations of RDIAS.

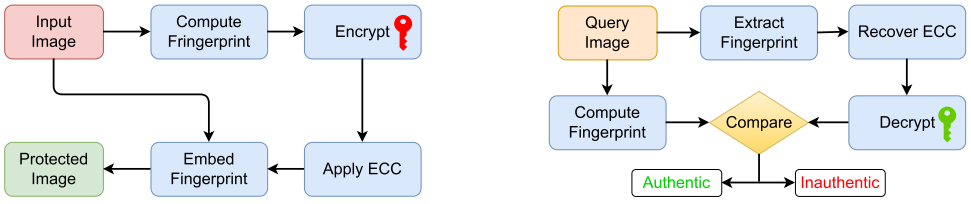
### 3.1 Design Goals and Approaches

RDIAS is an *active* image authentication system. It proactively protects images before distributing them. We call this step immunization. Then, as images circulate on different sites, they could be manipulated by malicious attackers or transformed through regular operations performed by hosting sites. RDIAS strives to verify the authenticity of these images obtained from arbitrary sites.

Specifically, we design RDIAS to achieve the following goals:

— *Robust against Legitimate Image Transformations*. As discussed in Section 2, hosting sites, e.g., Facebook and X, perform various operations on images that do not change their semantic meaning, such as resizing and transcoding. Image authentication systems should tolerate these operations. RDIAS achieves this by computing fingerprints that represent the visual content of images using perceptual hashing.

— *Preserve Image Quality*. Authentication systems should not significantly change the visual quality of images, i.e., the minor changes performed by these systems to protect images should




(a) Immunization (b) Verification

Fig. 2. Overview of the proposed image authentication system, RDIAS.

be imperceptible. To achieve this, RDIAS first determines the smallest size of fingerprints that can reliably represent the visual contents of images. It then embeds these fingerprints in images using deep-learning watermarking models that have minimal impact on quality.

— *Secure against Powerful AI Attackers*. To be relevant in the AI age, authentication systems must be able to detect sophisticated manipulations performed by modern AI tools. Further, authentication systems cannot rely on *security by obscurity*, i.e., hiding the details of the used models and algorithms from attackers. To achieve this objective, RDIAS assumes attackers have full access to all models and algorithms. It designs robust and representative fingerprints. It then utilizes the well-established public key infrastructure security framework, where the fingerprint is encrypted by the private key of the image creator/publisher and decrypted by the corresponding public key. Further, since some bits of the (encrypted) fingerprints may be changed because of the various image transformations and manipulation, rendering these fingerprints unusable, RDIAS employs an error-correcting method to mitigate these bit changes.

— *Decentralized and Scalable*. Authentication systems must scale to the massive volumes of images they aim to protect. RDIAS achieves scalability through decentralization. Specifically, since it embeds the authentication information inside the image itself, users can *independently* verify the authentication of the image *without* relying on any central entity. This is in contrast to some of the current image authentication systems, e.g., C2PA [14] that may store copies of the authentication information on *trusted* locations (e.g., cloud servers), which not only increases their vulnerability but also imposes significant storage and communication overheads, ultimately limiting their scalability. We note that users of RDIAS do need to obtain the image publisher's public key. However, there is only one key per publisher/creator (not per image), and keys can easily be distributed and/or locally cached, as regularly done with, for example, financial systems. Further, public keys do not leak any information about images, and they are typically digitally signed to prevent forgery.

### 3.2 Architecture and Operation of RDIAS

The proposed system has two parts. The first is used to *immunize* an image before distributing it, done once by the image creator/publisher. The second is used to *verify* the authenticity of an image, which is done by each receiver/viewer of the image. Figure 2 illustrates the architecture of each part.

For the immunization part, illustrated in Figure 2(a), RDIAS first computes a small fingerprint to abstractly represent the visual contents of the input image using a carefully selected perceptual hashing method, as detailed in Section 4. It then encrypts this fingerprint using the image creator's private key. Any of the well-known public key cryptography systems, e.g., RSA can be used in this step. To mitigate potential bit changes in the encrypted fingerprint during distribution and

increase robustness, RDIAS adds redundant bits to the fingerprint using an error-correcting method customized for image authentication systems, as described in Section 6. Finally, the expanded and encrypted fingerprint is embedded in the image using a watermarking method that retains the image quality while achieving robustness to various image transformations, as described in Section 5.

As illustrated in Figure 2(b), the verification part of RDIAS mostly has the reverse operations of the immunization part. Specifically, the fingerprint is first extracted using the watermarking method. Then, the error-correcting method recovers from the bit errors that may have occurred on the fingerprint. Then, the image creator's public key is used to decrypt the fingerprint. Finally, the decrypted fingerprint is then compared against the one computed from the query image to decide on the authenticity of the image.

We have implemented the verification part of RDIAS as a *web browser plugin*, which we describe in Appendix A. The plugin automatically verifies images in real time, and it presents a small authentic or manipulated mark at the top left corner.

### 3.3 Practical Considerations and Limitations

RDIAS utilizes a public-private key infrastructure. Such infrastructures are already available and used by many organizations.

RDIAS embeds information in the image's pixels. Thus, it may be less effective in verifying the authenticity of low-resolution images (below $256 \times 256$ pixels). This is because it may not have sufficient space to embed representative fingerprints in them. Further, excessive downscaling of images (to lower than $256 \times 256$) may damage the embedded fingerprints. Such downscaling also removes critical image details, which makes the verification process harder and RDIAS may not produce accurate results in this case. We note, however, that most images that are target for manipulations have high resolutions so that attackers can reasonably change their details.

By robustly embedding the authentication information directly into the pixels, RDIAS offers resiliency to common operations performed by hosting sites, such as removing the metadata associated with images. Although challenging, it is still possible for attackers to remove the embedded authentication information from the images using recent deep-learning methods such as [6, 32, 42]. These methods, however, often leave visible artifacts and significantly degrade the image quality, making any potential manipulations easily detectable by users. Further, attackers cannot re-generate new fingerprints after manipulating the images since they do not have access to the private keys of the images' owners.

Finally, RDIAS embeds authentication information into images before they are distributed. Images released without this immunization step cannot be verified by RDIAS, and a complementary passive image authentication would be needed in this case.

## 4 Image Fingerprinting

The goal of image fingerprinting is to represent an image by a small binary digest. Fingerprints created by common hash functions like SHA-256 and MD5 do not tolerate any image changes. Thus, they are unsuitable for *robust* image authentication systems, which must handle image transformations such as resizing and transcoding. In this section, we first summarize the current robust fingerprinting methods in the literature. Then, we analyze and compare their performance to identify the most suitable method for image authentication.

### 4.1 Current Fingerprinting Methods

Robust fingerprinting methods strive to capture the visual and perceptual details of images, ideally providing fingerprints that can tolerate benign operations that do not change the semantic meaning

of images, e.g., transcoding. At the same time, they detect malicious modifications, e.g., adding/removing objects. Many fingerprinting methods have been proposed in the literature [16], targeting different application(s) with various requirements. However, at a high level, most involve a few main steps. They first perform some preprocessing on the input image, e.g., transform it to the frequency domain. Then, they extract features representing this image, such as color distribution, edge maps, relative patterns between objects, or deep features obtained by neural networks. They finally compute the fingerprint by applying a locality-sensitive hashing function that maps similar features close to each other.

Image fingerprinting methods come in a wide variety [16]. We select *six* representative samples covering the entire spectrum of possible fingerprinting methods in the literature, ranging from simple pixel averaging to complex neural network methods. We provide brief descriptions of these methods in the following:

— *Average Hash (aHash)* [43]: It first calculates the average pixel intensity. Then, each pixel is compared to this average: if the pixel's intensity is higher, it is represented as a 1 and as 0 otherwise. This process creates a binary map that captures the image's overall appearance.
— *Difference Hash (dHash)* [49]: It also calculates the average intensity but computes the difference between adjacent pixels. If the pixel on the right has a higher intensity than the one on the left, it is represented as 1 and as 0 otherwise. This generates a map that reflects the image's gradients and edges.
— *Wavelet Hash (wHash)* [29]: It utilizes the Discrete Wavelet Transform to break down the image into different frequency components. It then hashes the low-frequency components, which represent the basic shapes and patterns of the image.
— *Perceptual Hash (pHash)* [24]: It applies the **Discrete Cosine Transform (DCT)** to transform the image to the frequency domain, and it then hashes the most significant frequency components that represent the essential features of the image.
— *Facebook's PDQ* [20]: It is developed to identify and prevent the spread of harmful images, e.g., violent and inappropriate images on Facebook. Like pHash, it uses DCT, but it incorporates various optimizations to increase the likelihood of detecting near-duplicate images.
— *Apple's NeuralHash* [12]: It is designed to detect Child Sexual Abuse Material. It uses a neural network to extract features from images.

### 4.2 Analysis of Fingerprinting Methods

*Performance Metrics.* We first define the following three performance metrics to analyze the various dimensions of fingerprinting methods and their suitability for image authentication. We normalize all three metrics and make their ranges between 0.0 and 1.0, where 0.0 indicates the worst performance.

— *Sensitivity* $\eta$: It is the ability to differentiate between original images and their manipulated versions, and it is computed as the normalized Hamming distance between their fingerprints:

$$\eta(F, F^M) = \frac{1}{|F|} \sum_{i=1}^{|F|} (F_i \oplus F_i^M), \tag{1}$$

where $F$ and $F^M$ are the fingerprints of the original and manipulated images, respectively, and $\oplus$ is the bitwise XOR operation. Manipulations refer to the malicious modification of images by, for example, adding/removing objects or cropping areas to change the semantic meaning of images. Higher $\eta$ values indicate the fingerprinting method has greater sensitivity to manipulations, i.e., it offers higher chances of detecting them.

— *Robustness* $\rho$: It measures the ability to *tolerate* legitimate transformations performed on images and is given by:

$$\rho(F, F^T) = 1 - \frac{1}{|F|} \sum_{i=1}^{|F|} (F_i \oplus F_i^T), \tag{2}$$

where $F^T$ is the fingerprint after applying legitimate transformations on the image by hosting sites, such as compression and resizing. These transformations may also include image filtering and noise addition, which attackers can perform to make it harder for the system to detect their manipulations. Higher $\rho$ values imply that the fingerprint does not significantly change because of transformations, providing higher system robustness.

— *Discrimination* $\delta$: It measures how well fingerprints *uniquely* represent images. It indicates the chances of two different images having the same fingerprint, i.e., collision. It is given by:

$$\delta(\tau) = 1 - P_c(\tau), \tag{3}$$

where $P_c(\tau)$ is the collision probability of fingerprints, and $\tau$ is a threshold for considering two images as visually identical. If $\tau = 2$, for instance, two images would be considered visually identical if their fingerprints have a Hamming distance $\leq 2$, i.e., the fingerprints differ only in two or fewer bits.

The Hamming distance between two fingerprints $F^1$ and $F^2$ is computed as the summation over $X_i = F_i^1 \oplus F_i^2$. Considering each $X_i$ as a random variable, then, by the Central Limit Theorem, the Hamming distances between fingerprints can be approximated by a Normal distribution. A collision occurs if the distance $\geq \tau$. Thus, the collision probability can be computed as:

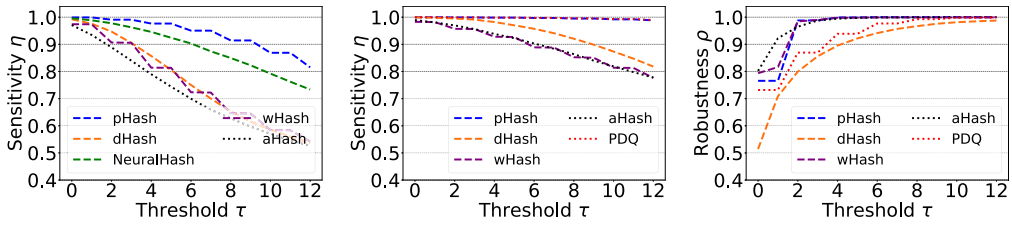
$$P_c(\tau) = \frac{1}{2} \cdot \text{erfc}\left(\frac{\mu - \tau}{\sqrt{2}\delta}\right), \tag{4}$$

where $\mu$ and $\sigma$ are the mean and standard deviation of the Normal distribution of the Hamming distances between fingerprints and $\text{erfc}(\cdot)$ is the complementary error function, which computes the probability of the normally distributed Hamming distances exceeding the parameter of the function.

*Implementation and Experimental Setup.* We implemented all considered fingerprinting methods. We evaluate and compare their performance on 2,000 randomly selected images from the diverse datasets described in Section 7. We compute the fingerprint of each image using each of the six fingerprinting methods. PDQ only creates fingerprints of size 256 bits, whereas NeuralHash only creates 96-bit fingerprints. The other four methods (aHash, dHash, wHash, and pHash) can be configured to create fingerprints of different sizes, but the sizes must be square numbers. Thus, for fair comparisons, we consider two fingerprint sizes for them: 256 and 100 (the closest square number to 96).

After computing the fingerprint, we then randomly subject the image to the various manipulations and transformations described in Section 2, where each manipulation and transformation can have multiple configuration parameters. We compute the averages of the three performance metrics, $\eta$, $\rho$, and $\delta$ across the entire dataset.

*Summary of the Results.* The sensitivity of all considered fingerprinting methods is plotted in Figure 3(a) and (b) for two fingerprints of sizes 100 and 256 bits, respectively. Multiple observations can be made on this figure. First, the results show that pHash consistently provides the highest sensitivity $\eta$ in all scenarios. This means pHash will likely detect more malicious image manipulations than other methods, which is the most critical aspect for authentication systems. Second,




(a) 100-bit Fingerprint (Sensitivity)

(b) 256-bit Fingerprint (Sensitivity)

(c) 256-bit Fingerprint (Robustness)

Fig. 3. Sensitivity and robustness of different fingerprinting methods for various thresholds $\tau$.

Table 1. Collision Probability ($= 1 - \delta$) for Different Fingerprinting Methods across Representative Fingerprint Sizes and Threshold Values

| **Method** | **Size** | $\tau = 0$ | $\tau = 4$ | **Size** | $\tau = 0$ | $\tau = 4$ |
|---|---|---|---|---|---|---|
| aHash | 100 | $1.02 \times 10^{-5}$ | $4.67 \times 10^{-5}$ | 256 | $9.68 \times 10^{-7}$ | $2.15 \times 10^{-5}$ |
| dHash | 100 | $1.01 \times 10^{-14}$ | $9.75 \times 10^{-13}$ | 256 | $1.07 \times 10^{-30}$ | $6.98 \times 10^{-29}$ |
| wHash | 100 | $5.54 \times 10^{-7}$ | $3.67 \times 10^{-6}$ | 256 | $1.22 \times 10^{-6}$ | $2.59 \times 10^{-6}$ |
| pHash | 100 | $\mathbf{6.38 \times 10^{-21}}$ | $\mathbf{2.24 \times 10^{-18}}$ | 256 | $1.09 \times 10^{-51}$ | $\mathbf{3.53 \times 10^{-51}}$ |
| NeuralHash | 96 | $2.69 \times 10^{-16}$ | $6.16 \times 10^{-14}$ | 96 | – | – |
| PDQ | 256 | – | – | 256 | $\mathbf{4.22 \times 10^{-55}}$ | $1.24 \times 10^{-50}$ |

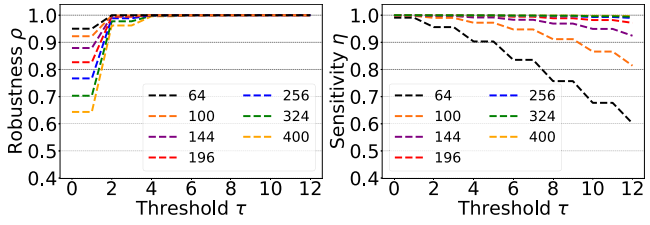
Smaller values indicate better discrimination by the fingerprint method. Bold numbers indicate the best-performing results in each column.

the sensitivity for fingerprint sizes of 100 bits quickly deteriorates with increasing the threshold value $\tau$, compromising the accuracy of detecting manipulations. However, 256-bit fingerprints produce a sensitivity close to 1.0 for multiple thresholds for pHash and PDQ.

We plot the robustness $\rho$ of all considered fingerprinting methods in Figure 3(c) for fingerprints of size 256 bits. The results show that pHash consistently produces the best robustness for threshold $\tau \geq 2$. This means that pHash tolerates more benign image transformations, e.g., resizing and transcoding, than other methods. We compute the collision probability, which is equal to $1 - \delta$, for all fingerprinting methods and summarize the results in Table 1. The table confirms that pHash produces the best performance in most cases. For example, for $\tau = 4$ and a fingerprint size of 256 bits, the discrimination value is close to 1.0, i.e., the probability of fingerprint collision is almost zero. Thus, we select pHash for image authentication.

Next, we determine the most suitable fingerprint size for pHash, which supports different sizes, unlike PDQ and NeuralHash. We consider all possible (square) fingerprint sizes between 64 and 400 bits. We repeat the above experiments and measure the robustness $\rho$ and sensitivity $\eta$ achieved for each fingerprint size as the threshold $\tau$ varies between 0 and 12. The results, in Figure 4, show that a fingerprint size of 256 bits yields the best overall performance. In addition, the results show that the threshold $\tau$ offers a tradeoff between $\eta$ and $\rho$: increasing $\tau$ reduces $\eta$ but improves $\rho$. By jointly inspecting the results for both $\eta$ and $\rho$ versus $\tau$, we found that $\tau$ values between 3 and 8 represent the acceptable operating range for the fingerprinting method. In our end-to-end evaluation of the entire system in Section 7, we analyze the accuracy across this range.

*In summary*, our analysis shows that *pHash with 256-bit fingerprints is the most suitable* method for image authentication systems.




(a) Robustness (b) Sensitivity

Fig. 4. Performance of pHash with different fingerprint sizes.

## 5 Embedding Information in Images

After computing a representative fingerprint of an image, we need to embed this fingerprint in the pixels of the image. This embedding should not damage the quality of the image, i.e., it should be imperceptible. Embedding information into images is sometimes referred to as watermarking in the literature. In recent years, many deep watermarking methods have been proposed. Generally, there are two primary approaches for designing watermarking methods. The first targets watermarking outputs of generative AI models. This helps in distinguishing these outputs from other works, providing clear content ownership and facilitating digital rights management. Methods in this category, e.g., [22, 51, 54] typically integrate watermarking directly into the image generation pipeline (often using Latent Diffusion Models), ensuring robust and high-quality watermark embedding. The second approach, known as *post hoc* watermarking, is designed as a post-processing step for arbitrary images, e.g. [4, 21, 58]. Our research belongs to the second approach because we aim to protect all types of images, whether AI-generated or captured by cameras.

In this section, we analyze the state-of-the-art methods for embedding information in images and identify the most suitable one for image authentication.

### 5.1 Current Embedding Methods

We summarize the main methods for embedding information in images in the following:

— *ReDMark* [4]: ReDMark improves upon HiDDeN [58], an early deep neural network for embedding data in images. HiDDeN employs adversarial perturbations to manage diverse image transformations. It uses an end-to-end deep-learning approach, where the encoder and decoder are trained jointly. However, it does not effectively handle important transformations, such as compression. ReDMark addresses this issue by introducing a differentiable module to approximate JPEG compression. ReDMark also has deeper and more complex encoder and decoder architectures based on ResNet [27].

— *FIN* [21]: FIN employs invertible neural networks [9] instead of the conventional encoder/decoder architecture. FIN uses the same neural network for the encoder and decoder, unlike previous works that designed separate networks and trained them jointly. This allows FIN to ensure the encoder does not embed redundant data that the decoder does not need to extract, which enables FIN to embed more information and improve visual quality.

— *FlexMark* [8]: FlexMark offers an adaptable design that can effectively control the tradeoff between the size of the embedded information and the system's robustness. FlexMark is trained once, and the size of the embedded information can be dynamically adjusted during inference without the need to retrain the model, which is a desirable feature for practical systems. It also includes a comprehensive transformation emulation module, which improves its robustness to a wide range of transformations.

— *TrustMark* [13]: TrustMark focuses on handling images of different resolutions. This is an important advantage because hosting sites tend to change image resolutions to fit their systems. In addition, TrustMark includes a combined GAN-based [30] module that can handle various transformations, including compression, color distortions, and additive noises.

### 5.2 Analysis of Embedding Methods

*Performance Metrics.* We define the following metrics to rigorously assess the performance of embedding methods.

— *Capacity*: is the amount of information that can be embedded in an image, measured in **Bits Per Pixel (bpp)**.
— *Robustness*: assesses resiliency to various transformations. It is defined as the **Bit Accuracy Rate (BAR)**, which is the fraction of bits of the embedded information that is correctly extracted.
— *Imperceptibility*: refers to how unnoticeable the embedded information is within the image. It is assessed by measuring the quality of the image after embedding the information relative to the original image. We consider three quality metrics: **Peak Signal-to-Noise Ratio (PSNR)**, **Structural Similarity Index (SSIM)**, and **Learned Perceptual Image Patch Similarity (LPIPS)** [57]. LPIPS is a relatively recent image quality metric that uses deep neural networks. It compares the activations of image patches within a pre-trained neural network. These activations capture complex features that are more aligned with human perception compared to traditional pixel-wise metrics like PSNR. It is trained on a dataset where human judgments of image similarity are available. This training helps LPIPS to better reflect how humans perceive differences between images. Lower LPIPS values indicate higher similarity between images and, therefore, better imperceptibility.

*Implementation and Experimental Setup.* We implemented all four considered information embedding methods, starting from their public repositories. We trained and tested each of them on randomly selected 2,000 images from the datasets described in Section 7.1. Every image was subjected to 11 transformations, each with three different parameter configurations. Thus, in total, each information embedding method was evaluated on 66,000 images. For fair comparisons, we fixed the capacity for all methods at $256/(512 \times 512)$ bpp.

*Summary of the Results.* We summarize the average robustness and imperceptibility of the four embedding methods in Tables 2 and 3, respectively. The details of the results for individual transformations are given in Appendix D. The results highlight a clear overall advantage for TrustMark compared to the other methods. Specifically, TrustMark achieves significantly higher average bit accuracy across various categories of transformation, demonstrating superior robustness. In contrast, the alternative methods either fail to maintain accuracy or exhibit substantially lower performance under the same conditions.

Regarding imperceptibility, TrustMark consistently outperforms the other methods, as reflected by its higher PSNR and SSIM and lower LPIPS values. This indicates that the resulting images after embedding the information using TrustMark are nearly indistinguishable from the original images.

*In summary*, our analysis shows that *TrustMark is the most suitable for information embedding* in image authentication systems.

## 6 ECCs

ECCs add redundancy to the original data to allow errors to be detected and corrected without retransmitting the data. In RDIAS, we model an image as a communication channel that carries the embedded fingerprint as a message. We then model the impact of the changes that occur on

Table 2. Robustness of Different Information Embedding Methods, Quantified by the BAR

| | ReDMark | FlexMark | FIN | TrustMark |
|---|---|---|---|---|
| Transcode | 84.46 | 84.72 | 67.06 | **98.56** |
| Resize | 81.42 | 50.00 | 85.22 | **99.84** |
| Noise | **100.0** | 89.94 | 99.98 | 99.84 |
| Filter | 87.38 | 90.06 | 88.45 | **98.08** |
| Average | 88.315 | 78.68 | 85.18 | **99.08** |

Bold numbers indicate the best-performing results in each row.

Table 3. Imperceptibility of Different Information Embedding Methods, Quantified by Three Quality Metrics

| | ReDMark | FlexMark | FIN | TrustMark |
|---|---|---|---|---|
| PSNR ↑ | 43.02 | 35.15 | 35.90 | **44.04** |
| SSIM ↑ | 0.988 | 0.939 | 0.872 | **0.997** |
| LPIPS ↓ | 0.002 | 0.012 | 0.171 | **0.001** |

Bold numbers indicate the best-performing results in each row.

images, e.g., transcoding and resizing, as bit errors in the fingerprint. Then, we use an ECC method to correct these bit errors. In this section, we analyze various ECC methods and identify the most suitable one for the proposed image authentication system.

### 6.1 Current ECC Methods

Many ECC methods have been proposed for different communication and storage systems. Each method is typically optimized for target communication channels, message lengths, and error patterns. We summarize the main categories of ECC methods in the literature in the following:

— *Polar Codes* [10]: Polar codes are based on the principle of channel polarization. The idea is to transform communication channels into a mix of reliable and unreliable subchannels using recursive transformations. During encoding, information bits are assigned to reliable subchannels while the others are frozen. Polar codes achieve optimal performance in theory but require long messages, large block sizes, and complex decoding.
— *Low-Density Parity-Check (LDPC) Codes* [35] *and Turbo Codes* [44]: LDPC and Turbo codes are iterative ECCs rooted in graph theory and probabilistic reasoning. LDPC codes utilize sparse bipartite graphs to define parity-check matrices, enabling efficient message-passing algorithms for decoding. Turbo codes leverage parallel concatenation of convolutional codes and iterative decoding to approach Shannon's channel capacity. They both approach optimal performance for long messages (thousands of bits) in noisy environments but lose efficiency for short messages.
— *Reed-Solomon (RS) Codes* [40]: RS codes are block-based error-correction codes grounded in finite field (Galois field) arithmetic. They are optimal for correcting burst errors, as their structure divides data into symbols and operates over blocks of data. RS codes work by representing data as polynomials and encoding them with redundant parity symbols. Errors are detected and corrected through interpolation and root-finding techniques, making RS codes widely used in storage media, CDs, and QR codes. RS codes are suitable for burst errors but less efficient for random errors.

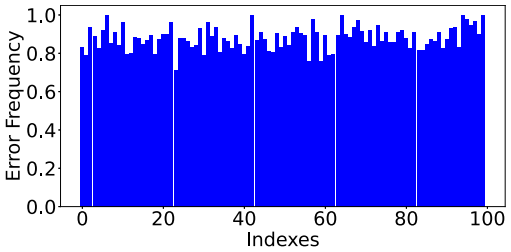



Fig. 5. Normalized distribution of bit errors.

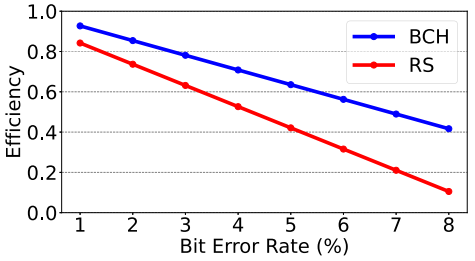



Fig. 6. Efficiency comparison of BCH and RS.

— *Bose–Chaudhuri–Hocquenghem (BCH) Codes* [45]: BCH codes are cyclic block codes designed to correct multiple random errors. They leverage the properties of Galois fields to construct parity-check equations that detect and correct errors efficiently. The codes are defined by their generator polynomials, which are algebraically designed for specific error-correction capabilities. BCH codes are particularly effective in applications like memory storage and digital communications, where precise control over error correction is required. They offer robust error correction for short messages and are computationally efficient.

### 6.2 Analysis of ECC Methods

ECC methods have been used mainly in communication and storage systems, where the error patterns have been heavily studied. Image authentication systems, however, have quite different characteristics, and it is unclear which ECC method(s) would suit these systems. Thus, we first model and analyze the error pattern resulting from embedding and extracting fingerprints in images. Then, we analyze the performance of candidate ECC methods and identify the most suitable one for RDIAS.

*Modeling Bit Errors in Image Authentication.* We define three important parameters to analyze the bit errors that result from embedding and extracting fingerprints in images, which are: (i) error distribution, (ii) bit flip pattern, and (iii) burstiness index.

*Error distribution* shows how bit errors are distributed within the received message (embedded fingerprint in our case). It indicates whether errors are concentrated in specific areas or are randomly distributed throughout. This is important because if, for example, errors are predominantly concentrated in one area, a localized ECC method can be used for this area, which can be different from the one(s) used for other areas.

To analyze error distribution, we embed fingerprints in images. Then, we randomly subject these images to various transformations. Then, we extract the fingerprints and compare them to the original ones. We repeat this experiment for 2,000 images and four transformations including transcoding, resizing, filtering, and noise addition. We compute the normalized error frequency for each bit location in the fingerprint, and we plot the results in Figure 5. The results indicate that fingerprint embedding/extraction and image transformations collectively yield a fairly uniform distribution of errors in the fingerprint. Thus, a single ECC method is sufficient for our case.

*Bit flip pattern* indicates whether there is a bias toward flipping more bits to either 1 or 0, i.e., whether we need a symmetric or asymmetric ECC method. We compute the percentages of bits that were flipped from 1 to 0 and vice versa during the experiments mentioned above. The results, summarized in Table 4, show that there is no significant bias in the bit flip pattern. Thus, a symmetric ECC method is desirable for our image authentication system.

Table 4. Bit Flip Pattern and Burstiness Index of Bit Errors in Fingerprints

| Transformation | 0–1 | 1–0 | Burstiness Index |
|---|---|---|---|
| Transcode | 50.47 | 49.50 | 0.979 |
| Resize | 51.87 | 48.13 | 0.993 |
| Noise | 51.29 | 48.71 | 0.992 |
| Filter | 50.72 | 49.28 | 0.963 |
| Average | 51.09 | 48.91 | 0.982 |

*Burstiness index* indicates whether errors tend to occur in clusters (bursts of size $\geq 2$ bits) or individually. It is calculated as:

$$\text{Burstiness Index} = \frac{\text{Number of Error Bursts}}{\text{Total Number of Error Bits}}.$$

A burstiness index closer to 1 suggests that errors are more dispersed and random, while a value closer to 0 indicates that errors are more likely to cluster in bursts. We compute and summarize the burstiness index in Table 4, which shows that errors in image authentication systems are not bursty.

*Identifying the Most Suitable ECC Method.* Polar codes, LDPC, and Turbo codes provide optimal efficiency. However, they require large message sizes (thousands of bytes). Fingerprints in image authentication systems are in the order of a few hundred bits (as discussed in Section 4). In addition, these ECC methods have high computational complexity due to their iterative and probabilistic designs. Thus, they are not suitable for image authentication systems.

On the other hand, BCH and RS methods can work with short messages as needed by image authentication systems. Both have simple encoding and decoding operations, which help with the real-time nature of image authentication. We conduct experiments to compare BCH and RS methods to choose the most suitable one. Specifically, we compute the efficiency achieved by each method in the context of image authentication. Efficiency is defined as the original message size divided by the message after adding redundancy by the ECC method. Figure 6 summarizes the results, which show that BCH consistently outperforms RS over the whole range of bit error rates.

*In summary*, our analysis shows that *BCH is the most suitable ECC method* for image authentication systems.

## 7 End-to-End Evaluation of RDIAS

RDIAS is composed of multiple components. We have analyzed each component separately in Sections 4–6. These analyses guided the selection of the most suitable method for each component and the recommended parameters for each method. Specifically, RDIAS employs: (i) fingerprinting using pHash with 256-bit fingerprints, (ii) information embedding using TrustMark, and (iii) error correction using BCH. Once chosen, these methods and parameters are *fixed* in RDIAS.

In this section, we evaluate the end-to-end performance of RDIAS on large and diverse datasets. We consider various malicious manipulations performed using AI tools, where we recruit participants to act as attackers and realistically change images. We also consider the usual image transformations imposed by hosting sites. In addition, we demonstrate the accuracy and robustness of RDIAS in complex scenarios where images are sequentially uploaded to multiple social platforms. For example, protected images are first uploaded to Facebook, then downloaded and

uploaded to Telegram, and finally posted on WhatsApp. Each platform performs different image transformations. Finally, we analyze all overheads imposed by RDIAS and show they are negligible.

Due to space limitations, some details and results are presented in Appendices A–D. We have also implemented the verification component of RDIAS as a web browser plugin, which seamlessly verifies image authenticity and displays the results in the top left corner of the image in real time. This plugin is described in Appendix A.

### 7.1 Datasets and Performance Metrics

*Datasets.* We use the following four diverse image datasets:

- *DIVerse 2K (DIV2K)* [18]: It consists of 1,000 high-resolution (2K) images of various real-world scenes, such as urban environments, natural landscapes, plants, and animals.
- *Challenge on Learned Image Compression (CLIC)* [36]: It has high-quality photographs sourced primarily from specialized platforms like Unsplash.
- *Flickr-Faces-HQ (FFHQ)* [47]: It contains 70,000 high-resolution (1,024 × 1,024 pixels) images of human faces of various ages, ethnicities, and backgrounds. Faces may also have different accessories like eyeglasses, sunglasses, and hats.
- *ArtBench-10* [34]: It is a curated dataset of paintings with 60,000 images representing 10 distinct artistic styles. It encompasses many art forms, including paintings, murals, and sculptures from the 14th to the 21st century.

*Performance Metrics.* The goal of RDIAS is to distinguish authentic from manipulated images, which is a binary classification problem. Thus, we compute all important standard metrics to comprehensively analyze the performance of RDIAS, including Precision, Recall, Accuracy, F1 Score, the **Receiver Operating Characteristic (ROC)** curve, and the Precision–Recall curve. All these metrics are computed based on the correctness of the binary outcome (Positive or Negative) of the system. Specifically, True Positive means a manipulated image is correctly identified as manipulated, whereas False Positive means an authentic image is incorrectly declared as manipulated (i.e., false alarm). Similarly, True Negative means correctly declaring an authentic image as not manipulated, and False Negative means a manipulated image was identified as authentic.

### 7.2 Results for Detecting AI Manipulations

*Experiments.* We randomly selected 2,600 images from three datasets: DIV2K [18], CLIC [36], and ArtBench [34]. Using the immunization module of RDIAS, we processed all these images to make them immune to potential manipulations by embedding authentication information in them.

Then, we recruited ten volunteers and instructed them to "manipulate the images to change their semantics to deceive users." Each participant was assigned 130 images and asked to apply at least one manipulation to each image using Adobe Firefly [1], ChatGPT DALL.E3 [38], Meta LLaMA.4 [37], and Google Gemini FLASH2.5 [39] which all are AI-powered image editing and generation tool. Participants employed various techniques, including cropping image borders, expanding and filling borders with generative AI, adding and removing objects using the magic brush, and writing prompts to generate new objects and blend them with the images. They also removed any artifacts left by the manipulations to make the images appear realistic and undetectable. An illustration of this process can be found in Appendix B.

Each of the 10 participants freely edited 130 images, resulting in a total of 1,300 manipulated images. We then upload these 1,300 manipulated images and the other 1,300 unmodified ones to three widely used platforms: Facebook (1,000 images), WhatsApp (1,000 images), and Telegram (600 images). The 2,600 images were in .png format. The three platforms transcoded all images to JPEG, but with different quality parameters according to the settings of each platform. Each

Fig. 7. Sample AI manipulations detected by RDIAS. Top: original. Bottom: manipulated.

Table 5. Summary of the Performance of RDIAS on Detecting AI-Powered Manipulations of General Images

| **Threshold** | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|
| Accuracy | 94.07 | 96.88 | 97.15 | 97.84 | 98.11 | 97.77 |
| Recall | 99.69 | 99.46 | 99.46 | 99.08 | 99.00 | 98.07 |
| Precision | 89.61 | 94.58 | 95.07 | 96.69 | 97.27 | 97.47 |
| F1-Score | 0.943 | 0.969 | 0.972 | 0.978 | 0.981 | 0.977 |

The presented results are averages across the three platforms: Facebook, WhatsApp, and Telegram.

platform also scaled down large images to lower resolutions according to its settings. Further, all platforms removed the metadata from the uploaded images.

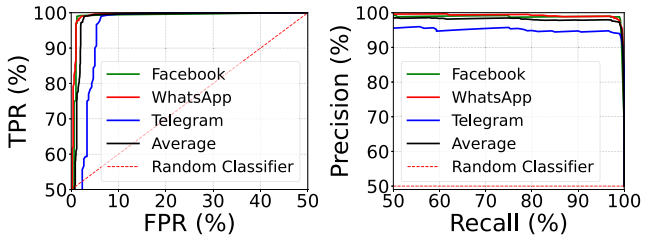
*Sample Visual Results.* We first present some *qualitative* results to demonstrate our approach in Figure 7, where we show multiple original images (top row) and their manipulated versions created by participants (bottom row). Original images have green bounding boxes, and manipulated images have red boxes. The manipulations included seamlessly adding/removing objects and subtly changing some image areas. RDIAS was able to correctly identify all original and manipulated images in these cases, even in the presence of these sophisticated AI manipulations and the various transformations and scaling operations performed on the images.

*Overall Objective Performance.* We report the Accuracy, Precision, Recall, and F1-Score for different threshold values $\tau$ in Table 5. The best performance is achieved for $\tau = 7$, where the overall Accuracy is 98.11%. The Recall, which measures how successful the system is in detecting manipulations, is even higher at 99.00%. This means that RDIAS missed only 1.0% of these sophisticated AI manipulations. The Precision is also high at 97.27% (indicating a very low false alarm rate), which is important for authentication systems. Further, the F1-Score is close to 1.0, indicating a good balance between Precision and Recall.

In addition, Table 5 shows that RDIAS performs well across multiple threshold values, specifically $\tau$ between 4 and 8; the performance gradually drops outside this range. This shows the practicality and robustness of the system, as it does not need to be fine-tuned on a specific threshold value.

*Detailed Analysis.* We plot the ROC and the Precision–Recall curves in Figure 8 to demonstrate the performance of RDIAS across all possible thresholds. We plot an ROC curve for each social media platform and also include the overall average ROC curve. ROC curves show the tradeoff between the **True-Positive Rate (TPR)** and **False-Positive Rate (FPR)** achieved by RDIAS.

The results in Figure 8 can be used to configure RDIAS for different scenarios in practice. For example, some applications may tolerate a higher FPR for the sake of minimizing the risk of missing




(a) ROC Curve (b) PR Curve

Fig. 8. Detailed performance analysis of RDIAS for each platform.

any manipulation. As the figure shows, RDIAS is capable of achieving an average TPR of 99.69%, but in this case, the FPR is 11.54%. This is obtained by setting $\tau = 3$. On the other hand, a balanced performance can be achieved by setting $\tau = 7$, as indicated in Table 5. In this case, the TPR is 99.00%, and the FPR is 2.78%. In addition, Figure 8(b) demonstrates the stability of RDIAS, because both the Precision and Recall stay fairly high and consistent for most $\tau$ values.

*Robustness of RDIAS in Challenging Scenarios.* We next stress-test RDIAS by considering challenging but practical scenarios, where images can spread from one site to another. Specifically, users may immunize their images and post them on a platform, say Facebook. Attackers may download these images, manipulate them, and repost them on Facebook. Facebook typically transcodes images using a specific set of parameters and scales them to suit its storage and processing pipelines. These operations affect the embedded watermarks, making it harder for the protection system to detect manipulations. Nonetheless, we have shown above that RDIAS can accurately detect such manipulations.

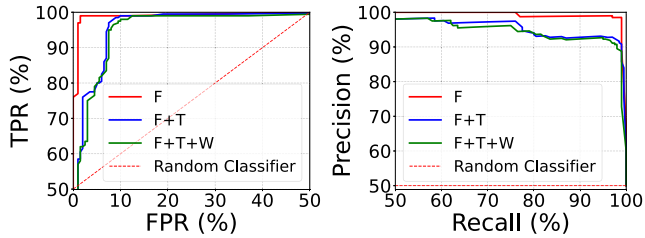
Attackers, however, may post the manipulated images on another platform, e.g., Telegram. Telegram, in turn, performs an additional set of transformations on the images, further complicating the job of the image protection system. We note that Telegram resized and compressed images with lower quality factors than Facebook and WhatsApp. It resized high-resolution images so that the longer side was 1,280 pixels (compared to 2,048 for Facebook and 1,600 for WhatsApp), keeping the aspect ratio unchanged. It also applied JPEG compression with an average quality factor of 75 (compared to 92 for Facebook and 80 for WhatsApp). This caused more errors for RDIAS, leading to a slightly lower accuracy in the case of Telegram. To analyze the performance of RDIAS in these challenging scenarios, we consider sequentially uploading and downloading images to three quite different platforms: Facebook, Telegram, and WhatsApp. Specifically, we randomly selected 400 images from our combined dataset, where half had been manipulated by participants and the other half were authentic. We first upload all images to Facebook. Then, download them after being processed by Facebook. Then, upload these images to Telegram and download them again. Finally, we upload the images to WhatsApp and download them. We then use RDIAS to authenticate the images after all such transformations and manipulations.

Table 6 summarizes the *final* results, i.e., after the images have sequentially been posted to three different social media platforms. The results show that RDIAS still achieves an accuracy above 93% even in these challenging scenarios. All other performance metrics are also fairly high. We present a detailed analysis in Figure 9, which shows the ROC and Precision–Recall curves after each stage of uploading and downloading images to the different platforms. The results in Figure 9 show the minor drop in performance due to the successive and extensive image processing operations performed by various platforms. The results confirm the robustness and accuracy of RDIAS in realistic scenarios. The details of the results for Various Transformations and Manipulations are given in Appendix C.

Table 6. Performance of RDIAS in Challenging Scenarios, Where Protected Images Are First Uploaded to Facebook (F), Then Downloaded from Facebook and Uploaded to Telegram (T), and Finally Downloaded from Telegram and Posted on WhatsApp (W)

| **Threshold** | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|
| Accuracy | 81.50 | 88.75 | 90.25 | 91.75 | 93.25 | 93.25 |
| Recall | 99.00 | 99.00 | 99.00 | 99.00 | 99.00 | 98.00 |
| Precision | 73.33 | 82.16 | 84.26 | 86.46 | 88.79 | 89.50 |
| F1-Score | 0.8426 | 0.8980 | 0.9103 | 0.9231 | 0.9362 | 0.9356 |

Each social platform performs different image transformations. Yet, RDIAS still achieves a fairly high accuracy.




(a) ROC Curve (b) PR Curve

Fig. 9. Detailed performance analysis of RDIAS in challenging scenarios. While various social media platforms transcode and store images differently, RDIAS achieves robust and consistent performance.

Fig. 10. Sample DeepFake manipulations detected by RDIAS. Top: original. Bottom: manipulated.

### 7.3 Results for Detecting DeepFake Face Manipulations

*Experiments.* We consider two DeepFake manipulations on faces: Expression Change and Face Swap. Expression Change involves altering a person's facial expression, such as making them appear happy, sad, drowsy, or blinking. We performed these manipulations using a well-known AI model [23] available on HuggingFace. We show sample images in Figure 10.

For Face Swap, we utilized the tool in [46], which transfers a face from a source image to a target image. This process allows placing someone's face onto another person, making them appear in contexts they were not originally present in (see Figure 11).

For our experiments, we randomly selected 100 images from the FFHQ dataset. We applied Face Swap manipulations to 50 images and Expression Change manipulations to the remaining 50.

Fig. 11. Sample Face Swap manipulations detected by RDIAS. Top: original. Middle: target. Bottom: output.

Table 7. Summary of the Overall Performance of RDIAS on Detecting DeepFake Face Manipulations

| **Threshold** | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|
| Accuracy | 97.0 | 98.5 | 98.5 | 98.5 | 99.0 | 99.0 |
| Recall | 100 | 100 | 100 | 100 | 100 | 100 |
| Precision | 94.3 | 97.0 | 97.0 | 97.0 | 98.0 | 98.0 |
| F1-Score | 0.97 | 0.98 | 0.98 | 0.98 | 0.99 | 0.99 |

In addition, we applied image transformations similar to those performed by Facebook. Specifically, we transcoded all images using JPEG with a quality parameter between 70 and 90, and we randomly scaled images by ratios between 0.5 and 0.9.

*Summary of the Results.* First, for subjective demonstration, all sample cases shown in Figures 10 and 11 were correctly identified by RDIAS.

Next, we summarize the objective results of all DeepFake face manipulations in Table 7. The results confirm the high accuracy of RDIAS in detecting such serious and increasingly common attacks. A Recall of 100% is reported for all threshold values, indicating that RDIAS was able to identify all manipulated faces. A Precision of 98.0% is reported for the best threshold ($\tau = 7$), indicating a very low false alarm rate of 2%. The overall Accuracy and F1-Score achieved by RDIAS are 99.0% and 0.99, respectively. Collectively, all performance metrics indicate the robustness, stability, and accuracy of the performance of RDIAS.

We note that the results in Table 7 are slightly better than the ones reported in Table 5 for the general AI manipulations of scenes. This is because manipulations changing facial expressions or swapping faces impact a substantial portion of the image, making such alterations less challenging for RDIAS to identify compared to single object addition or removal in general images.

## 8 Conclusion

The rise of advanced AI tools has made image manipulation more accessible and sophisticated, posing serious challenges to the authenticity of digital content. In this article, we introduced RDIAS, a robust and decentralized system for image authentication that addresses these challenges by embedding semantic fingerprints directly into images. RDIAS integrates perceptual hashing,

deep-learning-based information embedding, secure encryption, and ECCs to provide a scalable and practical solution capable of withstanding benign transformations and detecting malicious manipulations. We analyzed various methods for each system component and identified the most suitable option for image authentication systems. We also analyzed the tradeoffs among components to maximize the end-to-end performance of RDIAS.

We conducted extensive evaluations with diverse datasets and scenarios, including AI-powered attacks like DeepFake manipulations. Our experiments considered common transformations performed by various social platforms and image-hosting sites, such as transcoding, resizing, and removing metadata. They also considered image transformations that attackers may use to hide their manipulations, such as adding noise and/or applying filters. Our results demonstrated the RDIAS's effectiveness, achieving up to 99% accuracy in distinguishing authentic from manipulated images in these challenging scenarios. In addition, RDIAS preserves image quality, operates efficiently in real time detailed in Appendix E, and eliminates the need for centralized verification, making it suitable for widespread adoption across various platforms.

## References


[1] Adobe. 2024. Adobe Firefly. Retrieved from https://www.adobe.com/sensei/generative-ai/firefly.html

[2] Adobe. 2024. Adobe Photoshop. Retrieved from https://www.adobe.com/products/photoshop.html

[3] Shruti Agarwal and Hany Farid. 2020. Photo forensics from rounding artifacts. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, 103–114.

[4] Mahdi Ahmadi, Alireza Norouzi, S. M. Reza Soroushmehr, Nader Karimi, Kayvan Najarian, Shadrokh Samavi, and Ali Emami. 2020. ReDMark: Framework for residual diffusion watermarking on deep networks. *Expert Systems with Applications* 145 (Oct. 2020), 113157–113172.

[5] Irene Amerini, Aris Anagnostopoulos, Luca Maiano, and Lorenzo Ricciardi Celsi. 2021. Deep learning for multimedia forensics. *Foundations and Trends® in Computer Graphics and Vision* 12 (2021), 309–457.

[6] Bang An, Mucong Ding, Tahseen Rabbani, Aakriti Agrawal, Yuancheng Xu, Chenghao Deng, Sicheng Zhu, Abdirisak Mohamed, Yuxin Wen, Tom Goldstein, et al. 2024. Benchmarking the robustness of image watermarks. In *Proceedings of the International Conference on Machine Learning (ICML '41)*.

[7] Ashima Anand and Amit Kumar Singh. 2021. Watermarking techniques for medical data authentication: A survey. *Multimedia Tools and Applications* 80 (2021), 30165–30197.

[8] Mohammad Amin Arab, Ali Ghorbanpour, and Mohamed Hefeeda. 2024. FlexMark: Adaptive watermarking method for images. In *Proceedings of the ACM Multimedia Systems (MMSys '24)*, 56–66.

[9] Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W. Pellegrini, Ralf S. Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. 2019. Analyzing inverse problems with invertible neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR '19)*.

[10] Erdal Arikan. 2009. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory* 55, 7, (Jul. 2009), 3051–3073.

[11] Quentin Bammey, Rafael Grompone von Gioi, and Jean Michel Morel. 2020. An adaptive neural network for unsupervised mosaic consistency analysis in image forensics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR '20)*, 14182–14192.

[12] Abhishek Bhowmick, Dan Boneh, Steve Myers, Kunal Talwar, and Karl Tarbe. 2021. *The Apple PSI System*. Technical Report.

[13] Tu Bui, Shruti Agarwal, and John Collomosse. 2023. TrustMark: Universal watermarking for arbitrary resolution images. arXiv:2311.18297. Retrieved from https://arxiv.org/abs/2311.18297

[14] C2PA. 2024. Coalition for Content Provenance and Authenticity. Retrieved from https://c2pa.org/

[15] Canva. 2024. Graphic Design Software. Retrieved from https://www.canva.com/

[16] Ling Du, Anthony T. S. Ho, and Runmin Cong. 2020. Perceptual hashing for image authentication: A survey. *Signal Processing: Image Communication* 81 (2020), 115713.

[17] Nigel Earnshaw, Jonathan Dupras, and Bruce Maccormack. 2023. Fighting misinformation with authenticated C2PA provenance metadata the growing threat to the news ecosystem. In *Proceedings of the NAB Broadcast Engineering and Information Technology Conference (NAB '23)*.

[18] Eirikur Agustsson and Radu Timofte. 2017. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17) Workshops*.

[19] Paul England, Henrique S. Malvar, Eric Horvitz, Jack W. Stokes, Cédric Fournet, Rebecca Burke-Aguero, Amaury Chamayou, Sylvan Clebsch, Manuel Costa, John Deutscher, et al. 2021. AMP: Authentication of media via provenance. In *Proceedings of the ACM Multimedia Systems (MMSys '21)*, 109–121.
[20] Facebook. 2019. *The TMK + PDQF Video-Hashing Algorithm and the PDQ Image-Hashing Algorithm*. Technical Report.
[21] Han Fang, Yupeng Qiu, Kejiang Chen, Jiyi Zhang, Weiming Zhang, and Ee Chien Chang. 2023. Flow-based robust watermarking with invertible noise layer for black-box distortions. In *Proceedings of the Conference on Artificial Intelligence (AAAI '23)*.
[22] Pierre Fernandez, Guillaume Couairon, Herve Jegou, Matthijs Douze, and Teddy Furon. 2023. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE International Conference on Computer Vision*, 22409–22420.
[23] Sylvain Filoni. 2024. Expression Editor - A Hugging Face Space. Retrieved from https://huggingface.co/spaces/fffiloni/expression-editor
[24] Azhar Hadmi, William Puech, Brahim Ait, Es Said, and Abdellah Ait Ouahman. 2012. Perceptual image hashing. *Watermarking* 2 (May 2012), 17–42.
[25] Maamar Hamadouche, Khalil Zebbiche, Mohamed Guerroumi, Hanane Tebbi, and Youcef Zafoune. 2021. A comparative study of perceptual hashing algorithms: Application on fingerprint images. In *Proceedings of the International Conference on Computer Science's Complex Systems and Their Applications (ICCSA '21)*.
[26] Basna Mohammed Salih Hasan, Siddeeq Y. Ameen, and Omer Mohammed Salih Hasan. 2021. Image authentication based on watermarking approach: Review. *Asian Journal of Research in Computer Science* 9 (Jun. 2021), 34–51.
[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition (CVPR '16)*.
[28] Yuepeng Hu, Zhengyuan Jiang, Moyang Guo, and Neil Gong. 2024. A transfer attack to image watermarks. arXiv:2403.15365. Retrieved from https://arxiv.org/abs/2403.15365
[29] Wei Chung Huang, Fabio Di Troia, and Mark Stamp. 2018. Robust hashing for image-based malware classification. In *Proceedings of the International Workshop on Behavioral Analysis for System Security (BASS '18)*.
[30] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. 2018. Multimodal unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition (CVPR '18)*.
[31] Ashraful Islam, Chengjiang Long, Arslan Basharat, and Anthony Hoogs. 2020. DOA-GAN: Dual-order attentive generative adversarial network for image copy-move forgery detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR '20)*.
[32] Zhengyuan Jiang, Jinghuai Zhang, and Neil Zhenqiang Gong. 2023. Evading watermark based detection of AI-generated content. In *Proceedings of the ACM Conference on Computer and Communications Security (SIGSAC '23)*.
[33] Pica Johansson, Florence Enock, Scott Hale, Bertie Vidgen, Cassidy Bereskin, Helen Margetts, and Jonathan Bright. 2022. *How Can We Combat Online Misinformation? A Systematic Overview of Current Interventions and Their Efficacy*. Technical Report.
[34] Peiyuan Liao, Xiuyu Li, Xihui Liu, and Kurt Keutzer. 2022. The ArtBench dataset: Benchmarking generative models with artworks. arXiv:2206.11404. Retrieved from https://arxiv.org/pdf/2206.11404
[35] Gianluigi Liva, Shumei Song, Lan Lan, Yifei Zhang, Shu Lin, and William E. Ryan. 2006. Design of LDPC codes: A survey and new results. *Communications Software and Systems* 2, 3 (Sep. 2006), 191–211.
[36] Luca Versari, Ross Cutler, and Nick Johnston. 2023. Learned image compression and perceptual metric challenge. In *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Workshop (CVPR '22)*.
[37] Meta AI. 2025. Llama 4. Retrieved from https://ai.meta.com/blog/llama-4-multimodal-intelligence/ Multimodal LLM with mixture-of-experts architecture.
[38] OpenAI. 2024. DALL·E 3. Retrieved from https://www.openai.com/research/dall-e
[39] Google DeepMind or Google Cloud. 2025. Gemini 2.5 Flash. Retrieved from https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash Multimodal "Flash" model with hybrid reasoning.
[40] I. S. Reed and G. Solomon. 1960. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics* 8, 2 (Jun. 1960), 300–304.
[41] Leonard Rosenthol, Andy Parsons, Eric Scouten, Jatin Aythora, Bruce MacCormack, Paul England, Marc Levallee, Jonathan Dotan, Sherif Hanna, Hany Farid, et al. 2020. *The Content Authenticity Initiative*. Technical Report.
[42] Mehrdad Saberi, Vinu Sankar Sadasivan, Keivan Rezaei, Aounon Kumar, Atoosa Chegini, Wenxiao Wang, and Soheil Feizi. 2023. Robustness of AI-image detectors: Fundamental limits and practical attacks. In *Proceedings of the International Conference on Learning Representations (ICLR '23)*.
[43] Sam Farisa Chaerul Haviana and Dedy Kurniadi. 2016. Average hashing for perceptual image similarity in mobile phone application. *Journal of Telematics and Informatics* 4 (Mar 2016), 12–18.

[44] Shuai Shao, Peter Hailes, Tsang Yi Wang, Jwo Yuh Wu, Robert G. Maunder, Bashir M. Al-Hashimi, and Lajos Hanzo. 2019. Survey of turbo, LDPC, and polar decoder ASIC implementations. *IEEE Communications Surveys & Tutorials* 21 (Jan. 2019), 2309–2333.
[45] N. J. A. Sloane. 1972. A survey of constructive coding theory, and a table of binary codes of highest known rate. *Discrete Mathematics* 3 (Nov. 1972), 265–294.
[46] AI Face Swap. 2024. An Online Face Editing Platform. Retrieved from https://aifaceswap.io/
[47] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks timo aila NVIDIA. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR '19)*.
[48] Luisa Verdoliva, Sencar Husrev Taha, and Memon Nasir (Eds.). 2022. *Multimedia Forensics*. Springer.
[49] De Zhi Wang and Jun Yan Liang. 2019. Research and design of theme image crawler based on difference hash algorithm. In *Proceedings of the Conference Series: Materials Science and Engineering (AEMCME '19)*, Vol. 563.
[50] Xiaofeng Wang, Kemu Pang, Xiaorui Zhou, Yang Zhou, Lu Li, and Jianru Xue. 2015. A visual model-based perceptual image hash for content authentication. *IEEE Transactions on Information Forensics and Security* 10, 7 (2015), 1336–1349.
[51] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. 2023. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust.arXiv:2305.20030. Retrieved from https://arxiv.org/abs/2305.20030
[52] Haiwei Wu and Jiantao Zhou. 2022. IID-Net: Image inpainting detection network via neural architecture search and attention. *IEEE Transactions on Circuits and Systems for Video Technology* 32, (Mar. 2022), 1172–1185.
[53] Yue Wu, Wael Abd-Almageed, and Prem Natarajan. 2018. BusterNet: Detecting copy-move image forgery with source/target localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR '18)*, Vol. 11210, 170–186. DOI: https://doi.org/10.1007/978-3-030-01231-1_11
[54] Cheng Xiong, Chuan Qin, Guorui Feng, and Xinpeng Zhang. 2023. Flexible and secure watermarking for latent diffusion model. In *Proceedings of the 31st ACM International Conference on Multimedia*, 1668–1676.
[55] Qichao Ying, Zhenxing Qian, Hang Zhou, Haisheng Xu, Xinpeng Zhang, and Siyi Li. 2021. From image to image: Immunized image generation. In *Proceedings of the ACM Conference on Multimedia (MM '21)*, 3565–3573.
[56] Qichao Ying, Hang Zhou, Zhenxing Qian, Sheng Li, and Xinpeng Zhang. 2023. Learning to immunize images for tamper localization and self-recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 11 (2023), 13814–13830.
[57] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition (CVPR '18)*, 586–595.
[58] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. 2018. HiDDeN: Hiding data with deep networks. In *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition (CVPR '18)*.

## Appendices

### A RDIAS Plugin for Web Browsers

To demonstrate the effectiveness of RDIAS and evaluate its real-world performance, we developed a browser plugin that verifies images in real-time as users browse the web. By installing this plugin, users can automatically see verification results regarding the authenticity of images on Web sites such as social media platforms or news sites. These images may have undergone common transformations applied by online platforms or have been altered using AI tools to modify their content. The RDIAS plugin automatically verifies images and presents the results to users in a clear and visually minimal format positioned at the top left corner of the Web page. In Figure A1, we showcase examples of our plugin in action on the Chrome browser for images shared on Facebook.

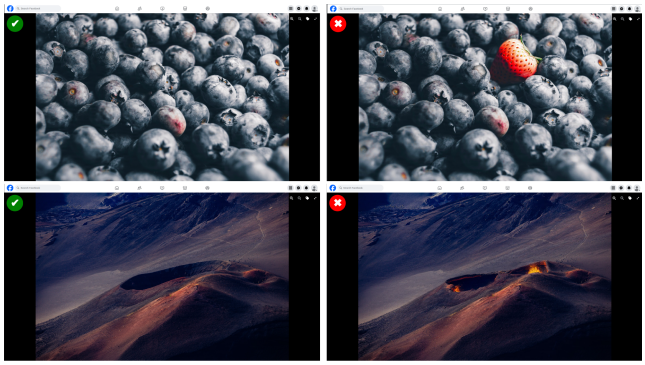


Fig. A1. Web browser plugin for image authentication using the RDIAS verification module tested on Facebook. A green checkmark or a red cross is displayed at the top left corner of the Web page, indicating the verification result based on the plugin's analysis in real time.

## B Image Manipulation Using Adobe Firefly

Figure B1 shows the procedure of applying manipulations to an image. Starting from an original image, one object has been added, and another object has been removed from the original image.

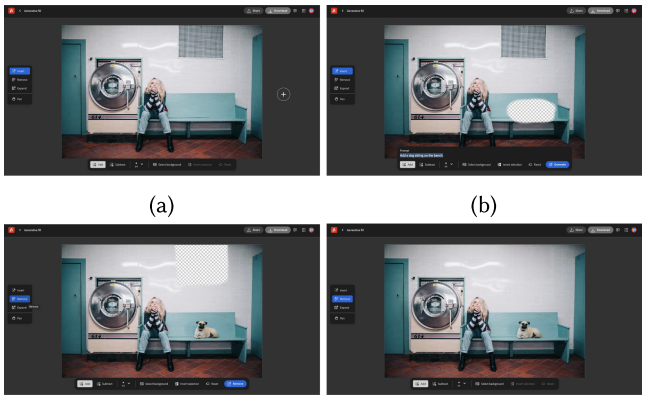



Fig. B1. An example of using Adobe Firefly to manipulate an image from left to right: (a) An original image uploaded to Adobe Firefly. (b) An object added to the image using magic brush and prompt. (c) An object removed from the image using magic brush. (d) Manipulated image.

## C Detailed Analysis of Various Transformations and Manipulations

We analyze how different transformations and manipulations affect our system's performance. To evaluate the system at scale, we developed a pipeline that includes the RDIAS immunization and verification modules. The pipeline randomly applies transformations and manipulations to images to stress-test RDIAS.

Initially, we immunized 2,000 images from the aggregated dataset. For each immunized image, we independently applied one transformation or manipulation from Table C1, generating seven new images per original image. This experiment resulted in individually transformed or manipulated

images. We verified each of them independently, and the results are presented in Tables C2 and C3. In total, we evaluated 14,000 images, all of which were either transformed or manipulated.

Table C1. Operations and Parameter Ranges for Automated Transformation/Manipulation

| Category | Transformation | Parameter | Range |
|---|---|---|---|
| Benign Transformations | Transcode | Quality | 70–90 |
| | Resize | Ratio | 0.5–0.9 |
| | Noise | Std | 0.1–0.5 |
| | Filter | Radius | 0.1–0.5 |
| Malicious Manipulations | Cropping | Ratio | 0.5–0.75 |
| | Object addition | Scale | 0.04–0.16 |
| | Object removal | Scale | 0.04–0.16 |

Table C2. Summary of the Performance on Randomly Chosen Individual Benign Transformations to Stress-Test RDIAS

| Threshold | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|
| Transcoding | 88.55 | 94.20 | 94.85 | 95.80 | 96.25 | 96.50 |
| Resizing | 92.85 | 98.30 | 98.35 | 98.80 | 99.00 | 99.05 |
| Noise addition | 93.35 | 98.00 | 98.25 | 98.90 | 98.95 | 99.00 |
| Filtering | 93.80 | 98.15 | 98.40 | 99.00 | 99.05 | 99.05 |

Table C3. Summary of the Performance on Randomly Chosen Individual Malicious Manipulations to Stress-Test RDIAS

| Threshold | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|
| Cropping | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| Object addition | 99.85 | 99.85 | 99.85 | 99.65 | 99.65 | 99.45 |
| Object removal | 99.90 | 99.60 | 99.55 | 99.20 | 99.20 | 98.10 |

Table C4. Summary of the Performance on Combinations of Randomly Chosen Transformations and Manipulations to Stress-Test RDIAS

| Threshold | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|
| Transformation | 89.65 | 94.85 | 95.65 | 96.80 | 96.95 | 97.40 |
| Manipulation | 99.85 | 99.80 | 99.80 | 99.50 | 99.50 | 99.00 |

In another experiment, we evaluated the effect of combined operations. For transformations, we randomly applied two transformations to each image. For manipulations, we randomly applied one manipulation and two transformations to each image, mimicking the environment of image-hosting Web sites where transformations are routinely applied to uploaded images. In this experiment, we evaluated 4,000 images, each of which underwent a combination of transformations and/or manipulations. The results are presented in Table C4.

## D Detailed Analysis of Embedding Methods

To assess the embedding methods and their suitability for our system, we evaluated their robustness against various image transformations, as summarized in Table D1. The evaluation covered a range of transformation levels to analyze the effects of each transformation comprehensively. TrustMark [13] demonstrated consistent performance across a wide range of transformations, unlike other methods, which exhibited inconsistent results. While some methods achieved excellent performance (e.g., greater than 99%) for specific transformations, they generally lacked robustness against key transformations such as transcoding and resizing—critical factors for our system. Consequently, TrustMark [13] emerged as the most suitable choice. This evaluation was conducted on 2,000 images for each transformation-parameter pair, resulting in 66,000 image variations tested per method.

Table D1. Robustness of Different Information Embedding Methods, Quantified by the BAR over Different Image Transformations (T) and Parameters (P)

| Category | T | P | ReDMark | FlexMark | FIN | TrustMark |
|---|---|---|---|---|---|---|
| Transcoding | JPEG | 50 | 60.93 | 78.03 | 66.23 | 97.97 |
| | | 70 | 71.57 | 81.14 | 67.43 | 99.14 |
| | | 90 | 98.10 | 88.11 | 68.08 | 99.73 |
| | WebP | 50 | 83.72 | 85.29 | 65.20 | 96.75 |
| | | 70 | 92.45 | 87.43 | 66.76 | 98.12 |
| | | 90 | 99.99 | 88.32 | 68.66 | 99.60 |
| Resize | Resize | 0.5 | 54.55 | 50.0 | 85.03 | 99.85 |
| | | 0.75 | 89.82 | 50.0 | 85.29 | 99.84 |
| | | 1.5 | 99.90 | 50.0 | 85.35 | 99.84 |
| Noise | Gaussian | 0.02 | 100 | 92.06 | 99.98 | 99.84 |
| | | 0.04 | 100 | 90.08 | 99.98 | 99.84 |
| | | 0.08 | 100 | 87.68 | 99.98 | 99.84 |
| Filter | G. Blur | 1 | 100 | 95.47 | 99.99 | 99.84 |
| | | 3 | 50.32 | 84.89 | 99.53 | 99.64 |
| | | 5 | 49.59 | 72.04 | 23.30 | 95.42 |
| | Median Blur | 1 | 100 | 95.48 | 99.99 | 99.84 |
| | | 3 | 73.09 | 91.56 | 99.98 | 99.80 |
| | | 5 | 67.28 | 88.92 | 99.83 | 99.74 |
| | Average Filter | 1 | 100 | 95.48 | 100 | 99.83 |
| | | 3 | 36.13 | 93.08 | 100 | 99.81 |
| | | 5 | 64.79 | 92.74 | 99.98 | 99.32 |
| | Contrast | 0.5 | 100 | 96.54 | 83.06 | 99.25 |
| | | 1.5 | 99.93 | 95.78 | 85.16 | 95.85 |
| | | 2 | 99.58 | 93.05 | 83.40 | 90.46 |
| | Brightness | 0.5 | 100 | 90.23 | 81.95 | 99.40 |
| | | 1.5 | 98.69 | 90.46 | 76.89 | 94.49 |
| | | 2 | 95.56 | 89.23 | 70.03 | 89.06 |
| | Saturation | 0.5 | 100 | 84.12 | 99.99 | 99.73 |
| | | 1.5 | 100 | 83.72 | 99.94 | 99.66 |
| | | 2 | 100 | 80.21 | 99.82 | 99.12 |
| | Sharpness | 0.5 | 100 | 93.69 | 84.06 | 99.83 |
| | | 1.5 | 100 | 93.10 | 84.92 | 99.83 |
| | | 2 | 100 | 91.53 | 85.72 | 99.81 |

## E Computational Analysis and Overheads

We assess all overheads associated with RDIAS.

*Quality Overhead.* Embedding fingerprints in images may impact their visual quality. We measure the visual quality after embedding the fingerprints using three metrics PSNR, SSIM, and LPIPS. We compute the quality of images with fingerprints relative to the original images. We report the average results across 2,600 images, which are: PSNR of 43.91, SSIM of 0.995, and LPIPS of 0.001. These results show that the reduction in quality is negligible and imperceptible to the human eye.

*Storage Overhead.* Embedding fingerprints in images may slightly increase their size. This is because the embedding process changes the image pixels, which may reduce the compression efficiency. Across the 2,600 images in our experiments, the average increase in file size is about 4.14% (from 4.63 MB to 4.83 MB), which is small.

*Computational Analysis.* We measure the processing time of all components of RDIAS. All experiments were conducted on a system with an Intel Core i7-12700 (20 cores), 32 GB of RAM, and an NVIDIA 3080 GPU with 10 GB of VRAM.

Table E1. Computational Analysis of RDIAS

| Step | FNG | ENC | ECC | EMB | E2E |
|---|---|---|---|---|---|
| Immunization | 76 ms | 1 ms | 1 ms | 765 ms | 843 ms |
| Verification | 27 ms | 1 ms | 1 ms | 17 ms | 46 ms |

The results are summarized in Table E1. The immunization process takes less than a second (on average 843 ms). Thus, the immunization process can easily be done without imposing any noticeable delay on the users before distributing their images. Recall that the immunization is done once. On the other hand, the verification step is done every time an image is downloaded from a Web site. The total verification time in RDIAS is 46 ms, on average. Thus, the verification step can easily be performed on images in real time, ensuring seamless authentication of images.

Received 30 November 2024; revised 4 June 2025; accepted 25 July 2025