AFRL-IF-RS-TR-2002-222
Final Technical Report
August 2002

SECURE OBLIVIOUS HIDING, AUTHENTICATION, TAMPER PROOFING, AND VERIFICATION TECHNIQUES

Binghamton University

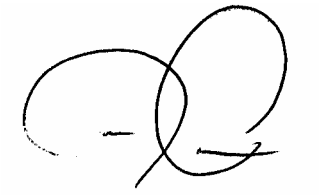
AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-222 has been reviewed and is approved for publication

APPROVED: 

RICHARD J. SIMARD
Project Engineer



FOR THE DIRECTOR:

JOSEPH CAMERA, Chief
Information & Intelligence Exploitation Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Aug 02	Final Oct 99 – Oct 00

4. TITLE AND SUBTITLE

SECURE OBLIVIOUS HIDING, AUTHENTICATION, TAMPER PROOFING, AND VERIFICATION TECHNIQUES

5. FUNDING NUMBERS

C - F30602-00-1-0502
PE - 62702F
PR - 459E
TA - SO
WU - PH

6. AUTHOR(S)

Jessica Fridrich

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Binghamton University
Center for Intelligent Systems, Dept of Systems & Industrial Engineering
Thomas J. Watson School of Engineering
PO Box 6000
Binghamton, NY 13902-6000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFEC
32 Brooks Rd
Rome, NY 13441-4114

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2002-222

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Richard J. Simard, IFEC, 315-330-4104, rsimard@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

In this report, we describe an algorithm for robust visual hash functions with applications to digital image watermarking for authentication and integrity verification of video data and still images. The robust image digest can also be used as a search index for efficient database searches. The hash function depends on a parameter K (a secret key) in a sensitive manner and on the image in a robust, continuous manner. The hash function always returns the same N bits from any image of arbitrary size. The bits obtained from two different mages or for two different keys K will generally be different (uncorrelated). However, for the same key K, two images that can be matched after applying gray scale operations, such as lossy compression, recoloring, filtering, noise adding, gamma correction, and simple geometrical operations including rotation and scaling, the extracted N-tuple will be almost the same. We also explain how the extracted N-tuple can be further utilized for synthesizing a Gaussian sequence that gradually changes with increasing number of errors in the extracted bits. Thus, the robust hash function can be used for generating pseudo-random watermark sequences that depend sensitively on a secret key yet continuously on the image. This robustness enables us to construct watermarks that depend on the original unwatermarked image in a non-trivial manner while making it possible to recover the watermark without having to access any information about the original image (oblivious watermarking). Such watermarks play an important role for authenticating videos or still images taken with a digital camera.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Digital Watermarking, Robust Visual Hash, Robust Image Digest, Image and Video Authentication	78
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Abstract

In today's world, digital images and video are gradually replacing their classical analog counterparts. This is quite understandable because digital format is easy to edit, modify, and exploit. Digital images and videos can be readily shared via computer networks and conveniently processed for queries in databases. Also, digital storage does not age or degrade with usage. On the other hand, thanks to powerful editing programs, it is very easy even for an amateur to maliciously modify digital media and create "perfect" forgeries. It is usually much more complicated to tamper with analog tapes and images. Tools that help us establish the authenticity and integrity of digital media are thus very essential and can prove vital whenever questions are raised about the origin of an image and its integrity.

In many commercial and military applications, it is highly important to know the origin of the image including the information about its author (sensor), time and place, the original parameters of the image, etc. This ancillary information can be stored either in a separate file, in the image header, or embedded in the image itself utilizing the modern concepts of steganography and data hiding. The first approach has many weaknesses because the image and the associated file can be separated and the file replaced with a different one. The second solution (embedding the associated information in the header) has similar weaknesses – the added header can be lost during lossless format conversion, the amount of information that needs to be sent is increased, and the header can be replaced. All of those weak points are successfully removed with the third approach, in which the ancillary data is embedded in the image rather than appended to it. Embedding the data in the image does not increase the image size. Also, it is possible to achieve lossless (an in some cases even lossy) format conversion without disturbing the added information. The information can be adaptively added to different objects in the image, thus making it possible to recover important message parts from a cropped image. Also, if a Message Authentication Code is added to the message to be embedded, it is possible to verify the image integrity and localize the tampered areas and in some cases identify the amount of distortion or even recover an approximation to the old content.

Fragile authentication watermarks for image integrity verification and oblivious robust watermarks for image signature embedding have been proposed in the past. In this report, we present some new tools to achieve secure oblivious robust watermarking using the concept of a visual hash and some new techniques for image authentication using fragile watermarks. Visual hash function is a very important security tool whenever possibly many images need to be authenticated (watermarked) using one secret key. In situations when one secret key must be reused for many images, such in a digital camera or when an image database is authenticated, it is highly insecure to embed an image-independent watermark. The watermark must be a non-trivial and non-invertible function of the secret key and the image content. The dependency on

the image content is the difficult part that has been successfully addressed in this report using the concept of a visual hash function. We describe how the visual hash can be modified to achieve robustness with respect to rotation, scale, and small geometric distortion as introduced with the package StirMark.

As another new paradigm for image authentication using watermarking, we present a new authentication technique based on invertible cryptographically secure global fragile watermark capable of detecting every modification to an image with extremely high probability. The security is guaranteed by a trusted cryptographic element – the MD5 hash function. This authentication is very unique because the distortion it introduces into the image can be inverted (undone) and the exact original image data can be obtained. This property is very desirable for sensitive imagery, such as medical imagery or high-importance strategic military imagery inspected by image analysts under high magnification (pixel-by-pixel level). The technique has been implemented in SecureStego.

Another capability added to the SecureStego program is a mechanism of secure distribution of imagery via adding invertible amount of distortion. Two techniques – one based on adding spatial noise and one based on bit-plane encryption have been studied and implemented in SecureStego.

TABLE OF CONTENTS

LIST OF FIGURES

Foreword

According to the deliverables as stated in the proposal, our research effort was focused towards:

- development of robust bit extraction techniques from images with emphasis on security and robustness to geometric transformations, such as rotation and change of scale, possibly general non-linear deformations as introduced by StirMark benchmark watermarking software;
- development of secure authentication techniques for tamper detection and image integrity verification;
- Implementation of the new techniques and tools in a prototype software product in Matlab and in C++ (SecureStego Windows application).

Digital watermarks have recently been proposed for authentication of both video data and still images and for integrity verification of visual multimedia. In such applications, the watermark has to depend on a secret key and on the original image. It is important that the dependence on the key be sensitive, while the dependence on the image be continuous (robust). Both requirements can be satisfied using special image digest functions that return the same bit-string for a whole class of images derived from an original image using common processing operations. It is further required that two completely different images produce completely different bit-strings. In this report, we discuss methods how such robust hash functions can be built. The focus is on robustness to common image processing operations and on geometric operations (such as rotation, scaling, and small non-linear deformations). We describe an algorithm and evaluate its performance by estimating the information content of the visual hash. We also show how the hash bits can be used to synthesize a Gaussian sequence that can be used as a source of randomness for virtually any watermarking scheme. As another application, the robust image digest can be used as a search index for an efficient image database search.

Cryptographically secure fragile watermarks [12–15] have been proposed as a means to verify image integrity without encrypting the image. Fragile watermarks typically depend on the hash of the image or an image block and are thus capable of detecting every change that occurred to the image. It can be shown that without the secret key, the probability of a modification that will not be detected can be related to the cryptographic element present in the scheme, such as a hash or a check-sum. Another advantage of digital authentication watermarks is their potential ability to localize the changes and in some cases even provide information about the distortion [16] or even reconstruct tampered/damaged areas in the image [17–19]. Those and similar techniques belong to the category of semi-fragile watermarks that allow authentication "with a degree". The goal of this soft authentication is to distinguish between malicious operations, such as feature adding/removal from non-malicious changes that do not modify the essential features in the image (slight filtering, high quality lossy compression, etc.). In this report, we limit ourselves to fragile authentication watermarking built upon secure cryptographic elements. Such schemes are designed to detect every possible change that occurred to the image with very high probability.

One possible drawback of authentication based on watermarking is the fact that the authenticated image is inevitably distorted by some small amount of noise due to the authentication itself. In all previously proposed authentication watermarking schemes, this distortion cannot be

completely removed even when the image is deemed authentic. Although the distortion is often quite small, it may be unacceptable for medical imagery (for legal reasons) or images with a high strategic importance in certain military applications. In this report, we analyze the conditions under which it is possible to "undo" the changes introduced by authentication, if the image is verified as authentic. We present some new concepts and techniques that make invertible authentication of typical images possible. Finally, we describe two watermarking techniques that embed Message Authentication Code (MAC) in images in an invertible way so that anyone who possesses the authentication key can revert to the exact copy of the original image before authentication occurred.

The last research achievement described in this report is a mechanism for secure distribution of images based on hierarchical fidelity control using invertible distortion. In order to explain the distribution mechanism; let us assume the following model situation. A satellite sensor takes high-resolution images and sends them to a Central Distribution Unit (CDU). The task of the CDU is to forward the imagery to recipients from several different security classes. Recipients from the highest security class, such as certain military subjects or government organizations, will be allowed to see the original high-resolution image. Another class of recipients, such as military subcontractors, NATO allies, or certain private companies, will be allowed to see only lower-fidelity images with lower degree of detail obtained as degraded versions of the high-fidelity images. It may be desirable to have another set of images with even more degradation available to recipients in foreign countries. Depending on the application, we may desire that one or more degraded versions exist for recipients from the lowest security class (the public access). Recipients from different security classes should see different level of detail in the images.

One way of arranging the distribution is as follows. The CDU can degrade the original image to several different fidelities, and send corresponding images to their recipients via several channels. Each channel must be either secure or the images must be encrypted with a key shared between the CDU and each security class of recipients. This method requires either multiple secure channels or encryption. It is very important that high-resolution imagery is not sent by mistake to the wrong recipient. In this report, we describe an alternative new method for secure distribution of imagery that does not require secure channels or encryption. It is based on invertible degrading of images. The CDU computes one degraded image that corresponds to the detail level available to public. The degraded image is generated using multiple secret keys whose number corresponds to the number of security levels. This degraded image is sent via public network to all recipients. Each recipient has one secret key that can be used to invert a portion of the degradation (improve the fidelity of the image). Having the key from one security level does not help in any way to obtain access to the higher detail level. Because in the new method only one image is distributed via public channels that do not need to be secure or encrypted, the new approach provides a considerable simplification of the distribution process while minimizing the error of sending a high-fidelity image to the wrong recipient.

In Section 1, we report the research results pertaining to the design of visual hash functions robust to geometrical transformations. In Section 2, we present the new invertible authentication techniques, and in the last Section 3, the new approach to secure image distribution via fidelity control is presented.

1. Visual Hash Functions for Oblivious Watermarking

Hash functions are frequently called message digest functions. Their purpose is to extract a fixed-length bit-string from a message (computer file or image) of any length. Obviously, a message digest function is a many-to-one mapping. In cryptography, hash functions are typically used for digital signatures to authenticate the message being sent so that the recipient can verify that the message is authentic and that it came from the right person. The requirements for a cryptographic hash function are [22]:

- Given a message m and a hash function H, it should be easy and fast to compute the hash $h=H(m)$
- Given h, it is hard to compute m such that $h=H(m)$ (i.e., the hash function should be one-way)
- Given m, it is hard to find another message m' such that $H(m')=H(m)$ (property of being collision free)

From the above properties it is clear that hash functions are "infinitely" sensitive in the sense that a small perturbation of the message m will give us a completely different bit-string h. In applications involving digital watermarking and authentication of digital images, the requirements on what should be a digest of an image are somewhat different. Changing the value of one pixel does not make the image different or non-trustable. Distortion introduced by lossy compression or typical image processing does not change the visual content of the image. What would be useful to have is a mechanism that would return approximately the same bit-string for all similar looking images, yet, at the same time, two completely different images would produce two uncorrelated hash strings. This is what we call in this report a visual hash function (visual hash). One can say that we want approximately the same hash bit-strings for two images whenever the human eye can say that these two images "are the same". Obviously, this is a challenging problem that can never be solved to our complete satisfaction. This is because the fuzzy concept of two images being visually the same is inherently ill defined and difficult, if not impossible, to grasp analytically. For example, changing one pixel in the pupils of a person's eye is for all purposes a negligible change. But once we change the color of every pixel in the pupil from, say, blue to brown, an important personal characteristic has been changed. Thus, we would conclude that the two images are no longer the same. However, the pupils can occupy a very small part of the image and our visual hash, not knowing the importance of eyes, may return the same hash bit-string. Being aware of these and other limitations, nevertheless, in this report, we attempt to meaningfully define the concept of a robust visual hash. Before we start with the definition and ideas how to construct such a function, we give a brief introduction into oblivious digital watermarking and explain how robust hash will play an important role in specific watermarking applications, such as authentication and fingerprinting.

1.1 Digital watermarking

Digital watermark is a perceptually invisible pattern embedded in a digital image. The watermark can carry information about the owner of the image or the recipient (watermarking for copyright protection, fingerprinting, or traitor tracing), the image itself (watermarking for tamper detection and authentication), or some additional information accompanying the image (image caption embedding). Watermarking schemes can be divided into two groups depending on whether or not the original image is required for watermark extraction. In non-oblivious watermarking, the original image is needed for watermark extraction. Although this makes non-oblivious techniques more robust to attacks, the necessity of having the original image is clearly a disadvantage that severely limits the applicability of non-oblivious techniques. In oblivious techniques, the watermark can be extracted from the watermarked / attacked image without access to the original image. In some watermarking techniques, one must have access at least to a hash of the image (or a hash of the whole video) in order to recreate the watermark sequence at the receiving end in order to be able to correlate the watermark with the watermark extracted from the image itself [1]. Such techniques are not truly oblivious because the hash needs to be exchanged prior to watermark detection.

Secure oblivious watermarking of videos for fingerprinting or authentication requires watermarks that depend on each frame. Indeed, one watermark pattern inserted into each frame would lead to a very vulnerable watermarking scheme with a serious security gap. It has been shown that by processing the images (frames), it is possible to statistically recover a good approximation to the watermark pattern [2]. However, the requirement of the technique to be oblivious means that either the watermark depends on the frame index or it is determined by the frame itself. Obviously, the latter case leads to much more versatile schemes. A reliable method for generating a good approximation of the watermark from the image itself (even after watermarking and attacks) will clearly lead to more useful and elegant oblivious watermarking schemes. For another paper that points out the importance of frame-dependent watermarking, see the paper by Yeung and Holimann [11].

Tewfik et al. [1] describe a watermarking technique in which a user-defined noise-like signature is modulated with a perceptual mask calculated from small blocks using perceptual masking. The same signature is used for all video-frames. The watermark pattern in this application is frame dependent and does not depend on the frame index. However, the frame dependency is not too strong because the perceptual mask can be calculated from each frame, which makes the technique equivalent to watermarking with a fixed watermark pattern.

Image watermarking for tamper detection leads to a similar situation as watermarking videos. Each digital image with a digital camera or digital video-camera would be watermarked on the fly so that later we can prove image integrity or indicate blocks in the image that have been tampered with. For a comprehensive review of watermarking techniques for tamper detection and common security problems, see [3]. Again, in this particular application, using one pattern that does not depend on the image would be

insecure because analyzing a relatively small number of images may reveal the watermark pattern [2].

What is needed in both applications discussed above is a watermark W that depends sensitively on a secret key K and continuously on the image I:

1. $W(K, I)$ is uncorrelated with $W(K, I')$ whenever images I and I' are dissimilar;
2. $W(K, I)$ is strongly correlated with $W(K, I')$ whenever I and I' are similar (I' is the image I after an attack comprising of a rotation, scale, and grayscale modifications);
3. $W(K, I)$ is uncorrelated with $W(K', I)$ for $K \neq K'$.

Linnartz and Cox [4,5] proposed similar requirements for watermarking digital versatile disks (DVD). The requirements 1–3 could be satisfied provided we have a robust image digest function H (visual hash function) that returns the same N bits (or almost the same N bits) for all images I that underwent a small distortion due to filtering, lossy compression, and other typical image processing operations. Visual hash is a function $H_K(I) \in \{0,1\}^N$ that depends on a secret key K. Below, we summarize the requirements:

1. Robustness to small distortions: I and J similar images, $H_K(I) \approx H_K(J)$.
2. Gradual loss of bits with increasing distortion: The Hamming distance $d(H_K(I), H_K(J))$ increases with distortion and approaches $N/2$.
3. Sufficient richness: $d(H_K(I), H_{K'}(I)) \approx N/2$ for I and J completely different images.
4. Sensitivity with respect to the key: $d(H_K(I), H_{K'}(I)) \approx N/2$ for $K \neq K'$.
5. Security: Not knowing the key, one cannot purposely modify the hash.

In a series of papers, we have described a mechanism for visual hash that has almost all of the required properties [6,10]. In particular, the visual hash was robust to all kinds of common image processing operations but was not robust to geometric transformations, such as scaling and rotation, and was weakly robust to small nonlinear warping. In addition to that, in our recent tests, we have observed that when the visual hash was applied in a hybrid watermarking scheme previously proposed by the Principal Investigator, in some rare cases, individual watermarked blocks could suddenly lose the watermark even though the actual loss of extracted bits was relatively low (46 correct hash bits out of 50). Careful analysis of this phenomenon lead us towards a new design of the watermark sequence from the hash bits and we also improved and simplified the visual hash calculation. The new visual hash runs faster, has higher robustness, and the process of synthesizing a watermarking (Gaussian) sequence from the hash bits is also faster and much more reliable.

In Section 1.2, we review ideas proposed by various researchers in the past (some ideas were posed in a different context). The old approach to visual hashing is briefly explained in Section 1.3 and the new design improvements are detailed in Section 1.4. In Section 1.5, we show how to synthesize a Gaussian sequence from the extracted hash bits so that the Gaussian sequence loses its correlation with the original sequence gradually. Finally,

in Section 1.6 we introduce special rotationally symmetrical random smooth patterns using which we made the visual hash robust to rotation and change of scale. We also address the issue of the information capacity of the visual hash and optimize the design that is robust to the change of scale and rotation.

1.2 Image invariants and robust visual hash

From the definition given in the previous section, robust image hash is a bit-string that somehow captures the essentials of the digital image or its block. Our requirement is that we need a key-dependent function that returns the same bits or numbers from similar looking images. So, the question is: "What is preserved under typical image processing operations?" Image edges typically contain the essence of an image. We could also use some relative relationship between pairs of image features [11], such as DCT coefficients. Also, it is well known that the principal directions and principal values calculated from image blocks are resistant to all kinds of grayscale image processing [23]. However, the principal directions are publicly known and the hash built from them would not have any security element in it. One could introduce a key-dependent linear or non-linear combination of the values determined from singular value decomposition of the image block, but this would provide only marginal security since the main robust values are not protected by a key, and therefore, can be intentionally manipulated. Another possibility would be to use invariant moments [20] or their key-dependent combinations for robust extraction of bits. Again, the problem with this approach is that the invariant moments are publicly known and can be purposely modified. Thus, the watermarking technique that utilizes bits derived from those moments would be inherently less secure. In [21], the authors proposed the usual hash of an edge map of a scaled-down image as a robust way of getting key-dependent hash bits for images. The logic is that edges are salient features of images and should be preserved for most image transformations. However, the usage of the cryptographic hash function will create a cliff-off effect that may not be desirable for robust watermarking. As long as the edge map does not change (after thresholding), the hash behaves in a robust manner with respect to small noise adding. However, once the edge map is modified, even in one pixel only, the hash returns a completely different bit-string. It would be nice to have a robust hash that deteriorates gradually rather than in an abrupt way, so that the watermark built from the hash is still highly correlated with the watermark used in watermark embedding.

Another approach that works quite well for small distortion especially distortion introduced by JPEG compression was introduced in [19]. The authors emphasize the fact that the mutual relationship of DCT coefficients in 8×8 blocks will be preserved no matter what quantization matrix is used for coding the image. Thus, one can extract one bit of information from predetermined pairs of DCT coefficients based on the fact if the first or the second pair member is larger than the other. The extracted bits are finally processed using a one-way function to obtain the final hash. There are several disadvantages of this method for use as a robust hash. First of all, while this method works very well for JPEG compression, its performance will be less satisfactory for a different type of distortion, such as contrast enhancement or small geometric distortions. Second, as long as the mutual relationship of the coefficient pairs is not changed, the

authentication technique based on this hash will not detect the change. And finally, one can purposely modify certain DCT coefficients to change the hash completely while making undetectable modifications to the image. This is because the DCT coefficients that enter the one-way function are publicly known.

1.3 Visual hash (old approach)

In this section, we start with a mechanism previously proposed by the Principal Investigator [6,10]. We present new, essential improvements to the scheme and describe a modification that is robust to rotation and change of scale.

We first describe the old technique as it appeared in [6,10] and then explain the new improvements. The method is based on the observation that if a low-frequency DCT coefficient of an image is small in absolute value, it cannot be made large without causing visible changes to the image. Similarly, if the absolute value of a low-frequency coefficient is large, we cannot change it to a small value without influencing the image significantly. To make the procedure dependent on a key, the DCT modes are replaced with low frequency, DC-free, (i.e., having zero mean) random smooth patterns generated from a secret key (with DCT coefficients equivalent to projections onto the patterns).

Using a secret key K (a number uniquely associated with an author, movie distributor, or a digital camera) we generate N random matrices with entries uniformly distributed in the interval [0, 1]. Then, a low-pass filter is repeatedly applied to each random matrix to obtain N random smooth patterns $P^{(i)}$, $1 \le i \le N$. An example of four random patterns and their smoothened versions are shown in Figure 1. All patterns are then made DC-free by subtracting the mean from each pattern. Considering the block and the pattern as vectors, the image I is projected on each pattern $P^{(i)}$, $1 \le i \le N$, and its absolute value is compared with the threshold Th to obtain N bits b_i:

$$\text{if } |B \cdot P^{(i)}| < Th \; b_i = 0$$
$$\text{if } |B \cdot P^{(i)}| \ge Th \; b_i = 1.$$

Since the patterns $P^{(i)}$ have a zero mean, the projections do not depend on the mean gray value of the block and only depend on the variations within the block itself. The distribution of the projections is image dependent and should be adjusted accordingly so that approximately half of the bits b_i are zeros and half are ones. This will guarantee the highest information content of the extracted N-tuple. This adaptive choice of the threshold becomes important for those image operations that significantly change the distribution of projections, such as contrast adjustment or gamma correction.

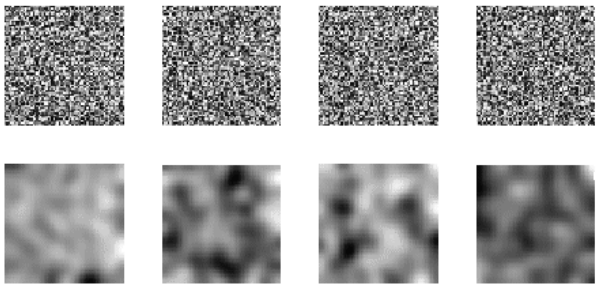


Figure 1 Examples of four random patterns and their smoothened version.

The robustness of this visual hash to grayscale modifications is very satisfactory. The results are shown in Table 1 (the numbers in brackets). Operations like embossing produce images from which the bits cannot be reliably extracted because the image has been flattened. Geometrical modifications, such as rotation, shift, and change of scale, also lead to a failure to extract the correct bits. Detailed evaluation of experiments can be found in a previous paper written by the Principal Investigator [6,10].

During our research effort, we have modified the design of the visual hash so that no adaptive threshold is needed for generating the hash bits. This threshold-free approach works faster and gives better robustness results than the old approach.

1.4 Visual hash (new approach)

The calculation of the threshold for extracting the hash bits is a relatively time consuming process because a large number of test patterns need to be projected on the block in order to determine the right threshold for each image. If the image is expected to undergo a large modification, such as removing or adding large features, the adaptive choice of the threshold may actually mislead the visual hash. In such a case, we would have to determine the threshold in each block separately. While this solves the problem, the computational time also increases significantly. The result of this consideration is that we have a strong incentive to remove the threshold from the visual hash if possible. Indeed, if we compare the values of the projections and extract bits based on those relative comparisons, we will not need a threshold for hash bit extraction. This will not only save computational time but also produce a more robust bit extraction.

Comparing the values of two projections $B \cdot P^{(i)}$ and $B \cdot P^{(j)}$ is basically the same as comparing their difference with zero because $B \cdot P^{(i)} > B \cdot P^{(j)}$ if and only if $B \cdot (P^{(i)} - P^{(j)}) > 0$. So, instead of generating the patterns and taking their differences and comparing the values, we can generate 50 patterns and compare the projections directly with zeros. Consequently, the only difference between this approach and the old approach is that we do not take the absolute value and compare it to an image-dependent threshold, but, instead, we compare the projection (without taking the absolute value) with zero. Thus, we get the following formula for the extracted hash bits:

$$\text{if } B \cdot P^{(i)} < 0 \; b_i = 0$$
$$\text{if } B \cdot P^{(i)} \geq 0 b_i = 1.$$

This formula is simpler and easier to evaluate than the old formula for bit extraction. What is even more important, however, is that the robustness of the extracted bits is also significantly better.

In all cases, we obtained an improvement. The most drastic improvement occurred for large distortions, such as StirMark, low quality JPEG compression, heavy noise adding, large positive adjustment of contrast/brightness, gamma correction, posterizing to a small number of colors, eroding, and crude mosaic filtering. The biggest improvement was for posterizing to two colors (fax dithering) in which case, we obtained an improvement from 38.51 bits per block to 43.28 bits. The new method also clearly outperformed the old approach for multiple applications of StirMark (increase from 35.47 to 40.21). Overall, we have observed a significant improvement in speed (a factor of 5) over the old technique. The visual hash has been implemented in Matlab and in SecureStego under the "Tools" menu. It has also been incorporated into the hybrid authentication scheme also implemented in SecureStego.

Distortion type	**Correct visual bits using the old (new) method**
STIRMARK 1×, 2×, 3×	44.11 (46.74), 39.18 (43.34), 35.47 (40.17)
JPEG 85%, 50%, 15%, 5% quality factor	49.89 (49.96), 49.71 (49.87), 49.04 (49.49), 47.42 (48.55)
NOISE ADDING 10%, 20%, 30% uniform noise	49.58 (49.71), 49.19 (49.44), 48.66 (49.13)
CONTRAST ADJUSTMENT –50%, –25%, 25%, 50%	49.88 (49.91), 49.91 (49.96), 49.93 (49.97), 47.76 (48.98)
BRIGHTNESS ADJUSTMENT –30%, –15%, 15%, 30%	46.71 (48.58), 49.84 (49.96), 49.87 (49.97), 46.54 (48.54)
GAMMA CORRECTION 0.7, 1.5	47.96 (49.06), 48.26 (49.28)
BLURRING 1×, 4×	49.38 (49.68), 47.97 (49.09)
SHARPENING 1×, 2×	49.85 (49.91), 49.60 (49.77)
POSTERIZING to 16, 8, 4, 2 colors	49.18 (49.58), 48.09 (48.87), 45.24 (47.39), 38.51 (43.39)
EDGE ENHANCEMENT	48.51 (49.03)
ERODE	44.36 (46.88)
HISTOGRAM equalization, stretching	47.25 (48.65), 49.89 (49.94)
MEDIAN	49.23 (49.71)
MOSAIC FILTER 2×2, 6×6	49.73 (49.91), 47.85 (48.98)

Table 1 Comparison of the performance between the old 'threshold-based" visual hash function and the new visual hash based on relative comparisons of projections.

1.5 Generating a watermark using the visual hash

Vast majority of watermarking schemes generates the watermark from a pseudo-random sequence. Below, we explain how to synthesize a pseudo-random Gaussian sequence from N visual hash bits so that the pseudo-random sequence gradually changes with increased number of errors in the hash, yet sensitively depends on the secret key. In addition to that, we require that when approximately half of the hash is incorrect, the generated Gaussian sequence should not be correlated with the sequence produced from all 50 correct bits. In our previous work [6,10], we have synthesized the Gaussian sequence by summing up uniformly distributed pseudo-random sequences obtained from a pseudo-random number generator (PRNG) seeded with a concatenation of the secret key, the block number (if the watermarking is done by blocks), and randomly chosen q-tuples of the extracted bits ($q \approx 5$). While this approach worked well in most cases, it could happen that if, say 46 out of 50 bits were correct, we lost the watermark completely because in almost all random q-tuples we had at least one bad bit. To remedy this problem, we have designed a completely different and simpler mechanism for generating the Gaussian sequence. For each block (image), we generate a Gaussian sequence ξ from a concatenation of the secret key and the block number. Then, we use the hash bits to modulate the sequence. We replace the 0's in the hash bits by -1's and expand the sequence by repeatedly concatenating randomly permuted hash bits till we get the desired number of 1's or -1's to modulate the whole sequence ξ. Since the same bits are reused in every N-tuple, if r is the percentage of the correctly recovered bits, the whole Gaussain sequence ξ is modulated correctly in r% of the cases and the correlation is lost gradually. If, however, 50% of extracted bits are wrong (either wrong key or non-watermarked image), we lose the correlation completely. This is exactly what we need to satisfy the requirements set forth in Section 1.1.

We have tested the performance of the new visual hash and the bit synthesis on real images and concluded that a significant boost in performance has been obtained. The code runs faster and is more robust and reliable.

In the next section, we propose two general strategies how to obtain visual hash functions that are robust to a wider range of grayscale operations and some simple geometric transformations including rotation and scaling.

1.6 Robustness to geometric transformations

The visual hash function described in the previous section is very robust with respect to grayscale modifications in which the geometrical location of pixels is not changed. However, a small amount of rotation or a slight change of scale can cause a rapid loss of correctly recovered bits. Also, operations, such as embossing (Laplacian filtering) prevent successful extraction of bits. Ideally, the visual hash should be robust to any combination of a rotation R_φ by an angle φ, scaling S_α by a factor α, and typical grayscale operations G. Noise adding, filtering, lossy JPEG compression, gamma correction, and histogram equalization are examples of typical grayscale operations. So, if the robust hash function H depends on a parameter K (secret key), we require that

$$H_K(R_\varphi \circ S_\alpha \circ G(I)) \approx const. \in \{0,1\}^N, \text{ for all } \varphi, \alpha, \text{ and } G. \quad (1)$$

In this section, we outline a solution to the problem of how to construct visual hash functions that will return the same bit-strings from images that have undergone a combination of rotation, scale, and a wider range of grayscale operations including an edge extractor.

To achieve robustness to scaling, we always rescale the image to a predetermined size (e.g., 256×256 pixels). Robustness to rotation will be achieved by using random smooth patterns with rotational symmetry (see Figure 2). The projections are thresholded with zero as before. Because we want the visual hash to be robust to small cropping, we center the circular patterns on a point derived from the image (its edges) instead of using a fixed point, such as the center of the image. The point we use is the center of gravity COG calculated from the edge map of the image. To increase the robustness and accuracy of the COG calculation, we preprocess the image I first by normalizing its contrast and brightness using a histogram equalization operation *Norm_Hist*(I) and use a robust edge extractor $L(\,)$ based on wavelet maxima. This algorithm gives consistent results even after operations of low-pass character.

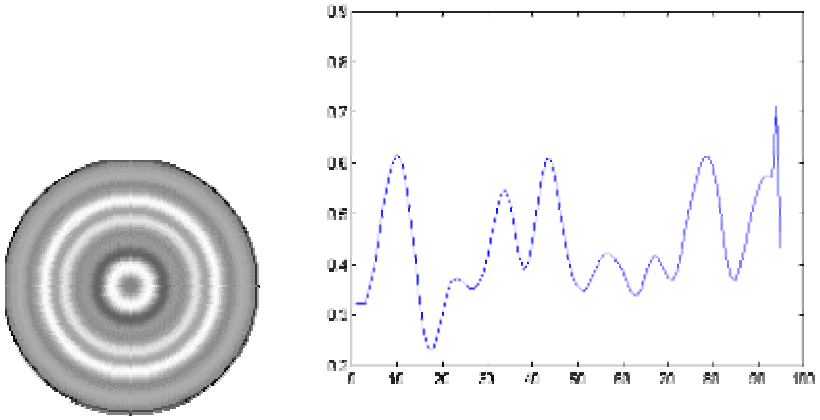


Figure 2 Example of a random circularly symmetrical pattern with its profile.

The center of gravity of the edge map seems to be very robust to common image processing operations as well as small cropping as long as the cropped out part of the image does not contain edges. Cropping an image by a large margin will cause the visual hash malfunction. It appears that this problem has no easy solution because a cropped image is actually a different image, and, therefore, it *should* produce a different hash.

The circular patterns exhibit quite a good robustness with respect to scaling and small amounts of rotation. It appears that the visual hash based on circular patterns will be an extremely useful tool for oblivious watermarking robust to geometrical transformations.

In this report, we also attempted to estimate the information capacity of the visual hash bit-stream. Since the entropy of the visual hash is computationally intractable, we investigated the correlation between visual hash bits. In the case of square smooth

patterns, the correlation between bits obtained from 4500 blocks of 64×64 pixels showed small to moderate correlation. The correlation was mostly due to the correlation between images rather than the correlation between the patterns themselves. The mutual correlation between patterns was very low. As a consequence, we expect the information capacity of the visual hash to be very close to the total number of patterns (50 in our experiments).

Circular patterns, on the other hand, exhibit quite strong correlations, as expected. Consequently, the information capacity of the visual hash for the circular patterns is expected to be significantly smaller than the number of patterns.

In the next sections, we provide details of our approach and present the results of numerical experiments.

1.6.1 Extraction of the edge map and the center of gravity

In the wavelet transform, we perform 1-dimensional transforms in the x and y directions to obtain $W(x)$ and $W(y)$. The wavelet transform amplitude of a given pixel is the square root of $W(x)^2 + W(y)^2$. The amplitude of the wavelet transform at the level 2 is used as the edge map. We also convert the amplitude value proportionally to the range [0, 255]. Color images are transformed to gray scales before applying the wavelet transform.

From the edge map, we calculate its COG (x, y) according to following formulas. The g_{ij} is the pixel value, and M and N are the image width and height respectively

$$x = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} i \times g_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{M} g_{ij}}, \quad y = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} j \times g_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{M} g_{ij}}.$$

1.6.2 Calculating the visual hash

As explained above, after obtaining the center of gravity, we project the gray scale version of the image onto a set of pseudo-random smooth patterns with circular symmetry centered on the center of gravity. The circularly symmetrical pattern is obtained by rotating a pseudo-random, smooth profile around the center of gravity. An example of a pseudo-random smooth profile is shown in Figure 2. The profile is obtained by first generating a pseudo-random number sequence of length 1/10 of what is needed to span a 256×256 image. The diagonal length is 362, so the length of the profile is 362/2=181. To get the full sampled version, we interpolate between each two points using cubic splines. The smoothed circular pattern is made DC-free by subtracting its mean value.

For the visual hash, we generate 50 pseudo-random circularly symmetrical smooth profiles. Then, for every pattern we calculate the distance from the center of gravity to

every pixel. The distance determines the value of the pattern and this value is multiplied by the pixel value. After adding up those numbers for all pixels, the summation is thresholded with zero. If it is greater than zero, the extracted hash bit is 1, otherwise it is 0. This procedure is applied for all 50 pseudo-random profiles to obtain the full visual hash consisting of 50 bits.

1.6.3 Experimental results

The COG can be calculated from the raw image, from its edge map, or from an image preprocessed with histogram equalization followed by the edge map extraction. We tested the performance for all three possibilities. The best overall performance was obtained for the last case when the COG is calculated from an edge map calculated from a histogram-equalized image.

The circular patterns have been projected on the histogram-equalized image and thresholded with zero as before. This scenario provided the best overall robustness. Other combinations tested included projecting the patterns on the edge map, the raw image, and the histogram equalized image.

We tested the stability of the COG and the number of correctly extracted bits for the following combinations:

(1) Calculate the center of gravity from the edge map and project onto the raw image;
(2) Calculate the center of gravity from the raw image and project onto the raw image;
(3) Do histogram equalization first and calculate the center of gravity from the edge map and project onto the image;
(4) Do histogram equalization first and calculate the center of gravity from the image and project onto the image;

For comparison, we also tested the case when the circular patterns are centered at a fixed point (the middle of the image). The results for this case are shown in the last row of Tables 2'–4'.

In Tables 2–4 and 2'–4' below, we test the performance of the visual hash for different image distortions. The second row corresponds to calculating the COG from the edge map. The rows show the change in the coordinates of the COG after certain distortion has been applied to the image. The smaller the change, the more robust the visual hash will be. The third row shows the results when the COG was calculated from the image and not from the edge map. The tables with a prime ' correspond to experiments done with preprocessing the image using histogram equalization. The test image used for the experiments was a true color 256×256 test image Lena.

Robustness to scaling is achieved by rescaling an image (or an image block) to a predetermined fixed size of 256×256 pixels. Because the patterns have rotational symmetry, rotation of the image does not change the visual hash bits. In Tables 5–7 and 5'–7' below, we summarize our experimental results regarding the robustness of the

visual hash bits with respect to scaling, rotation, and small cropping. The second and the third rows show the stability of the COG in the case when it is calculated from the edge map and the raw image, respectively. The tables with a prime correspond to results when the image has been preprocessed with histogram equalization. It also has an additional row for a fixed COG.

COG	Before Distortion	JPEG			Gamma Correction		Random Noise		Uniform Noise	
		30%	60%	99%	1.5	2.2	20%	50%	20%	50%
X,Y Edge	124.030 142.379	-0.0869 -0.0415	-0.2965 0.0995	-4.5529 1.6220	1.2002 -1.7337	1.9118 -2.8962	-1.9506 6.7884	-3.2632 10.1526	-1.0963 3.0181	-2.4651 8.2968
X,Y Image	133.063 123.452	-0.0102 0.0095	0.0026 0.0281	-0.6286 0.1408	1.7287 -0.9623	2.9164 -1.7482	0.1863 -0.1664	0.4009 -0.4228	0.0025 0.0237	0.1125 -0.0839
# Correct Bits		48/50	48/50	41/46	49/49	48/48	47/49	43/47	49/50	44/49

Table 2 Distortion without histogram equalization.

COG	Before Distortion	JPEG			Gamma Correction		Random Noise		Uniform Noise	
		30%	60%	99%	1.5	2.2	20%	50%	20%	50%
X,Y Edge	122.868 140.124	-0.0587 -0.0719	-0.1598 -0.0405	-3.7821 0.7544	0.5498 0.1932	0.7854 0.0662	-2.4465 5.8853	-3.1442 8.3645	-1.1064 3.2177	-2.0117 8.2810
X,Y Image	136.104 121.730	-0.0230 0.0010	0.0092 0.1036	-0.2169 0.7449	-0.0576 -0.2413	-0.1233 -0.3620	0.9472 -0.4457	1.8548 -0.8694	0.1270 -0.1484	1.1202 -0.5992
# Correct Bits		50/50	50/48	43/46	50/50	49/50	45/46	41/47	48/49	43/47
Fixed COG		49	49	45	49	47	49	48	49	49

Table 2' Distortion with histogram equalization.

COG	Before Distortion	Contrast			Blur		Sharpen		Histogram	
		10%	30%	50%	Once	More	Once	More	Equalize	Stretch
X,Y Edge	124.030 142.379	0.0330 -0.0066	0.1714 0.8344	0.1321 1.4477	-2.3823 1.8954	-3.0938 2.6184	-0.0561 -0.1738	0.6239 -0.5303	1.1624 2.2548	0.0866 0.0082
X,Y Image	133.063 123.452	-0.6686 0.4842	-2.5098 1.5601	-4.6438 2.0040	-0.0209 0.0167	-0.0194 0.0164	-0.0967 0.0099	-0.0050 -0.0401	-3.0414 1.7221	-1.1872 0.8665
# Correct Bits		50/50	50/40	47/41	45/50	44/50	49/50	50/50	47/42	50/48

Table 3 Distortion without histogram equalization.

COG	Origin	Contrast			Blur		Sharpen		Histogram	
		10%	30%	50%	Once	More	Once	More	Equalize	Stretch
X,Y Edge	122.868 140.124	-0.0359 -0.0350	-0.1487 -0.4340	-0.0589 -1.2050	1.4278 1.6286	-1.7823 2.3076	0.0790 -0.0668	0.3922 -0.1706	-0.1484 -0.0777	-0.0693 0.0159
X,Y Image	136.104 121.730	-0.0597 -0.0270	-0.3586 -0.0604	-0.6616 0.0033	-0.4056 0.6533	-0.5051 0.8101	0.1645 -0.3292	0.7080 -0.8530	-0.0949 0.0432	-0.0192 0.0161
# Correct Bits		50/49	49/49	50/46	48/47	47/47	49/50	49/50	50/50	50/50
Fixed COG		49	48	47	49	48	49	49	50	49

Table 3’ Distortion with histogram equalization.

COG	Origin	Dilate	Emboss	Erode	Mosaic	Hot Wax Coating
X,Y Edge	124.030 142.379	1.2411 0.5344	4.5998 -4.4453	-3.5432 2.8128	-9.6402 7.8581	-6.1076 9.9639
X,Y Image	133.063 123.452	1.3169 -1.7443	5.7176 -4.1657	-1.2706 1.8803	0.0111 0.0127	-3.0462 4.9173
# Correct Bits		41/38	32/37	47/49	40/49	43/45

Table 4 Distortion without histogram equalization.

COG	Origin	Dilate	Emboss	Erode	Mosaic	Hot Wax Coating
X,Y Edge	122.868 140.124	2.0833 0.7558	-1.6633 3.2454	-3.1255 1.4857	-7.8998 7.8110	-5.1302 5.6961
X,Y Image	136.104 121.730	0.5751 -2.0524	9.7800 -5.8302	-0.2346 1.8682	-1.2573 1.6903	0.5208 3.5390
# Correct Bits		40/43	30/38	45/45	41/47	42/43
Fixed COG		43	30	44	46	42

Table 4' Distortion with histogram equalization.

Scale in Width Scale in Height	256 => 257 256 => 257	256 => 300 256 => 256	256 => 128 256 => 128	256 => 64 256 => 128
X, Y Edge	-0.8425 0.2822	0.2530 -0.1813	-0.9061 0.6691	-2.7275 2.4504
X, Y Image	-0.0019 0.0202	0.0966 -0.0496	-0.0153 0.0125	-0.0210 0.0140
# Correct Bits	46/50	48/49	46/50	44/50

Table 5 Scale distortion without histogram equalization.

Scale in Width Scale in Height	256 => 257 256 => 257	256 => 300 256 => 256	256 => 128 256 => 128	256 => 64 256 => 128
X, Y Edge	-0.6634 0.2867	0.3215 -0.0851	-0.7040 0.5774	-2.0855 2.1830
X, Y Image	-0.1000 0.1718	0.1536 -0.1166	-0.1508 0.2166	-0.2941 0.5189
# Correct Bits	50/48	49/47	50/48	47/47
Fixed COG	49	46	49	48

Table 5' Scale distortion with histogram equalization.

Crop Size	Center 32 × 32	Center 64 × 64	Corner 32 × 32	Corner 64 × 64
X, Y Edge	-0.0029 -0.1772	0.7364 -0.7057	1.8079 2.2216	2.2403 3.1105
X, Y Image	-0.0867 -0.1795	-0.3407 -0.6167	1.3312 1.2480	5.4959 5.0920
# Correct Bits	43/41	26/32	43/40	35/34

Table 6 Crop distortion without histogram equalization.

Crop Size	Center 32 × 32	Center 64 × 64	Corner 32 × 32	Corner 64 × 64
X, Y Edge	0.2173 -0.0558	1.1610 -0.4084	0.5680 0.8946	0.1535 1.6891
X, Y Image	0.0134 -0.2122	0.0657 -1.2181	1.4279 1.2475	6.7785 5.4257
# Correct Bits	46/43	26/40	44/46	37/30
Fixed COG	44	31	42	39

Table 6' Crop distortion with histogram equalization.

Rotate Degree	Origin	0.5(253)	1.0(251)	1.5(249)	2.0(247)
X, Y Edge	124.030 142.379	-1.1176 0.4074	-0.4778 0.8833	-0.7387 0.7244	-0.6774 1.0811
X, Y Image	133.063 123.452	-0.0376 0.0125	0.0040 -0.0224	0.2185 -0.1709	0.1913 -0.1429
# Correct Bits		48/49	48/46	48/45	46/45

Table 7 Rotate and crop without histogram equalization

Rotate Degree	Origin	0.5(253)	1.0(251)	1.5(249)	2.0(247)
X, Y Edge	122.868 140.124	-0.9404 0.2982	0.8909 0.2751	-0.5870 0.5164	-0.5647 0.5975
X, Y Image	136.104 121.730	-0.0544 -0.0007	-0.0906 0.0260	-0.0334 -0.0270	0.1186 -0.1762
# Correct Bits		48/44	46/44	45/41	43/40
Fixed COG		45	45	44	44

Table 7' Rotate and crop with histogram equalization.

1.6.4 Evaluation of experimental results

After a careful analysis of the results reported in the previous sections, we conclude that the best (most robust) performance is achieved when

1. The image is I first normalized for contrast/brightness by applying the histogram equalization $Norm(I)$.
2. The COG is calculated from the edge map $E(Norm(I))$ extracted using wavelet maxima.
3. The circular patterns centered at the COG are projected on the normalized image $Norm(I)$ and thresholded with zero.

1.6.5 Information capacity of the visual hash (square 2d patterns)

An important question for practical applications is the information capacity of the visual hash. Total of 50 bits is extracted from each image. However, because the patterns are not orthogonalized, the information capacity is not equal to 50, but is somewhat smaller. To answer this question, one would have to calculate (or estimate) the entropy E

$$E = -\sum_{i_1,i_2,...,i_N} P(i_1,...,i_N)\log_2 P(i_1,...,i_N),$$

where N=50, and $P(i_1, i_2, \ldots, i_N)$, $i_k \in \{0,1\}$ is the probability that the extracted visual hash is $(i_1, i_2, \ldots, i_N)$. These probabilities are very hard to estimate using statistical simulations because there are too many of them (2^{50}). Because of this large number, it is not possible to run Monte Carlo simulations. Yet, the issue of the information capacity of the visual hash should be addressed. As an alternative, we studied the correlation between hash bits. We used a database consisting of 300 images and divided each image into 15 blocks. The size of the block was 64×64. Then each block was projected onto 50 smooth patterns. Each pattern was generated from a sequence of random numbers and smoothed as before. Projecting the patterns onto the blocks and thresholding with zero, we extract 50 bits from each block. To evaluate the correlations among pixels, we calculate the correlation C_{ij} between bits i and j ($1 \le i, j \le 50$). In the formula below, we used the following notation: c_k^i is the ith bit for the kth block and N=50×300=4500. The correlation C_{ij} is shown in Figure 3.

$$C_{ij} = \frac{\sum_{k=1}^{N} c_k^i * c_k^j}{\sqrt{\sum_{k=1}^{N} \left(c_k^i\right)^2} \sqrt{\sum_{k=1}^{N} \left(c_k^j\right)^2}}$$

We can see that there are quite strong correlations between the hash bits. But as will be seen from further analysis, the correlations are due to correlations that naturally occur between images rather than correlations due to non-orthogonal patterns. To see how

$$V_{ij} = \frac{\sum_{k=1}^{N} p_k^i * p_k^j}{\sqrt{\sum_{k=1}^{N} \left(p_k^i\right)^2} \sqrt{\sum_{k=1}^{N} \left(p_k^j\right)^2}}$$

correlated the patterns themselves are, we calculated the correlation V_{ij} using the following formula, where p_k^i is the kth value of the ith pattern and N=64×64. Geometrically, the correlation V_{ij} represents the angle between the ith pattern vector and the jth pattern vector. Figure 4 shows the mutual correlations V_{ij}. As we can clearly see, the mutual correlation due to non-orthogonality of the patterns is quite small and will most likely not decrease the information capacity of the visual bits in any significant manner.

1.6.6 Information capacity of visual hash (circular patterns)

The disadvantage of the approach based on circular patterns is that the circular patterns will in general be more correlated than two-dimensional random patterns, and as a consequence, the hash bits may exhibit strong correlations. This could be avoided by orthogonalizing the patterns prior to calculating the visual hash. Since it is necessary to do the orthogonalization in a one-dimensional space only, using the simple Gram-Schmidt procedure should produce a reasonably fast code. Another disadvantage of this approach is that the number of hash bits will most likely have to be lower than in the fully two-dimensional case.

Following are the experimental results for the circular patterns. We recapitulate that the center of gravity is calculated from the image. Because the center of gravity is different for different image, the 50 circle

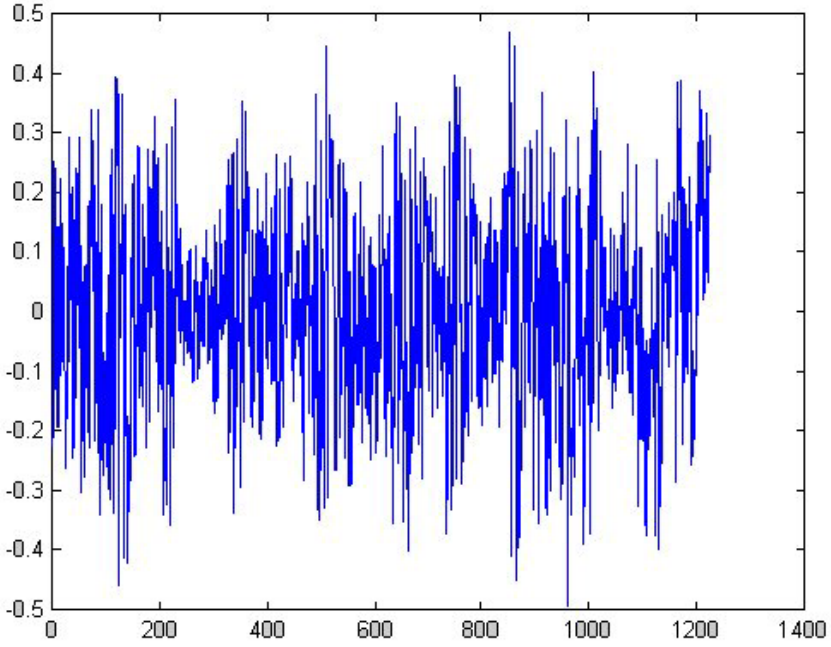


Figure 3 Correlation C_{ij} between visual hash bits for square patterns.

patterns aren't the same for different images although the same key value was used to generate the random smooth profiles. For the circular patterns, we calculate the intersection of the pattern with the image to calculate both the correlations V_{ij} and C_{ij}. Figure 5 and Figure 6 show the corresponding values of the correlation C_{ij} and V_{ij}. As expected, the circular patterns exhibit much stronger correlations and it is quite obvious that the information capacity of the visual hash for the circular patterns is significantly smaller.

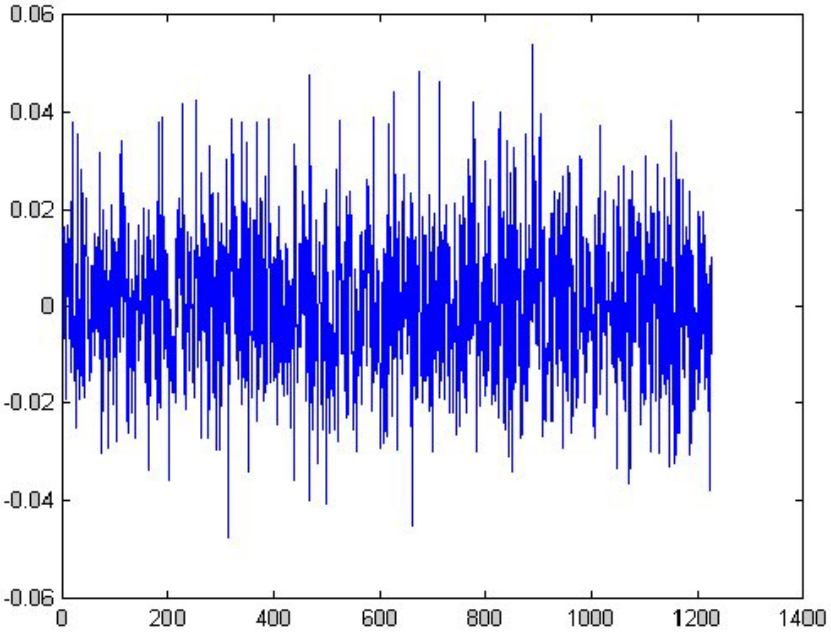


Figure 4 Correlation V_{ij} between square patterns.

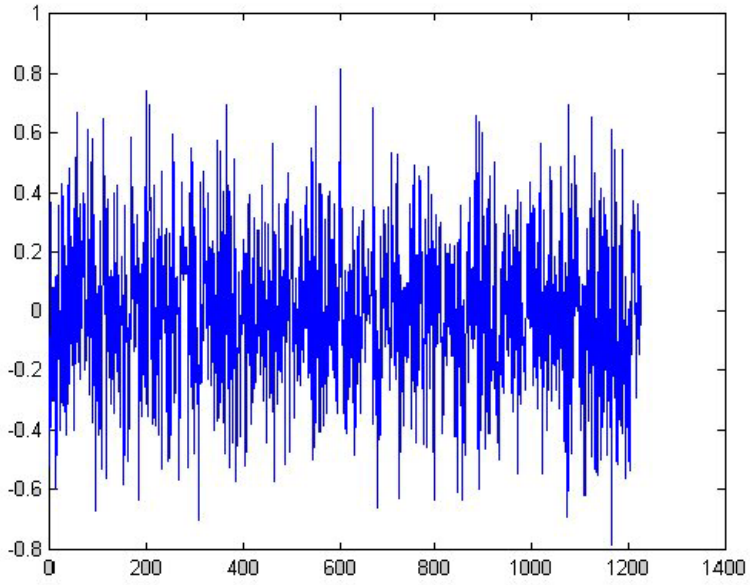


Figure 5 Correlation C_{ij} between hash bits for circular patterns.

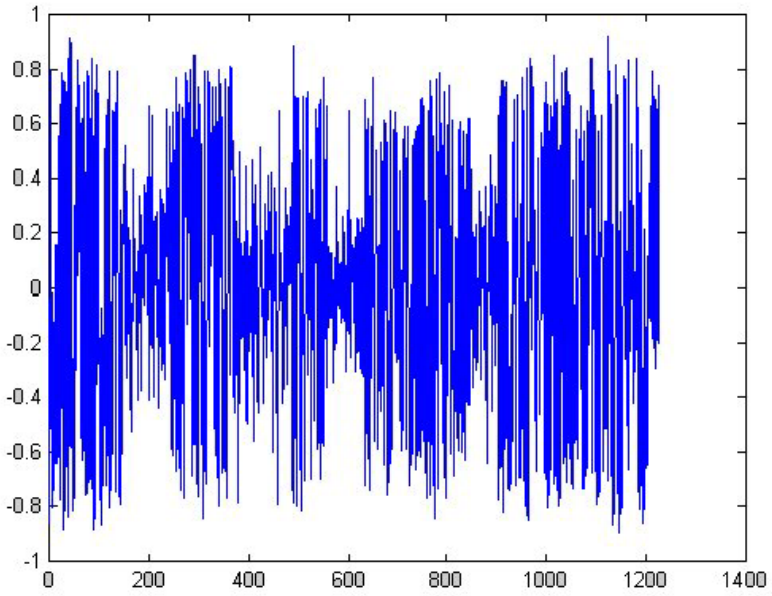


Figure 6 Correlation V_{ij} between circular patterns themselves.

1.6.7 Conclusion and summary

In this report, we describe an algorithm for robust visual hash functions with applications to digital image watermarking for authentication and integrity verification of video data and still images. The robust image digest can also be used as a search index for efficient database searches. The hash function depends on a parameter K (a secret key) in a sensitive manner and on the image in a robust, continuous manner. The hash function always returns the same N bits from any image of arbitrary size. The bits obtained from two different images or for two different keys K will generally be different (uncorrelated). However, for the same key K, two images that can be matched after applying gray scale operations, such as lossy compression, recoloring, filtering, noise adding, gamma correction, and simple geometrical operations including rotation and scaling, the extracted N-tuple will be almost the same. We also explain how the extracted N-tuple can be further utilized for synthesizing a Gaussian sequence that gradually changes with increasing number of errors in the extracted bits. Thus the robust hash function can be used for generating pseudo-random watermark sequences that depend sensitively on a secret key yet continuously on the image. This robustness enables us to construct watermarks that depend on the original unwatermarked image in a non-trivial manner while making it possible to recover the watermark without having to access any information about the original image (oblivious watermarking). Such watermarks play an important role for authenticating videos or still images taken with a digital camera (Figure 7).

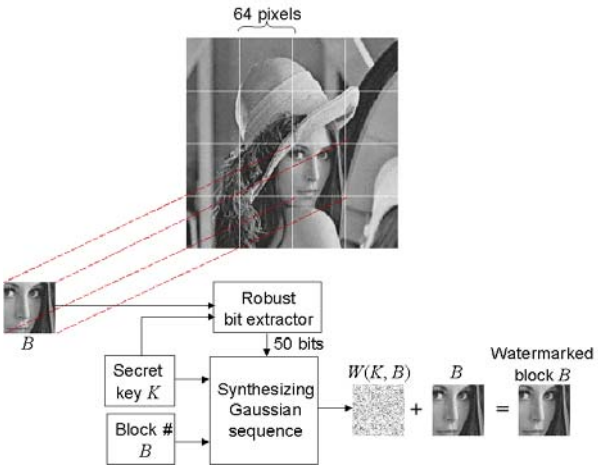


Figure 7 General scheme for watermarking using the proposed bit extraction procedure.

In those watermarking schemes that are robust with respect to rotation and scale, the watermark recovery is typically obtained using special synchronization patterns used to detect the geometric operations applied to the image [9]. The watermark is read after those geometric operations have been undone. Since the accuracy with which the geometrical operations are detected is usually limited, it is necessary to have a robust technique for generating the watermark at the receiving end. For current robust hash functions [6,10], even small rotations and change of scale can produce large differences in the extracted bits. The methods reported here constitute a step towards robust message digest functions that are insensitive to rotation and scale. The technique works with DC-free circularly symmetrical random smooth patterns centered on the center of gravity calculated from the edge map projected onto a histogram equalized image.

The visual hash produces different hash bits for cropped images after a substantial amount of edges have been cropped out. It appears that this problem has no easy solution because a cropped image is virtually a different image, and, therefore, it should produce a different hash. One possibility to overcome this problem would be to calculate the hash bits locally and produce a random spatial pattern as the result, rather than a set of hash bits. This spatial pattern could be the initial input for synthesizing a watermark for a specific watermarking scheme. This approach is attractive because it has the potential to provide a solution to robust oblivious watermarking that could survive non-linear random warping (StirMark). This possibility will be investigated as part of our future research.

As another application of robust hash functions, we mention indices for efficient image database search. There are many quantities that could be derived from images using which one can search a database in an efficient manner. Many indices are based on color information that can be extracted from a histogram. However, such indices are not useful if the image has been processed using histogram equalization, or recolored. The essence of an image can be well captured using its edges. Our method captures the mutual spatial relationship among edges rather than color information. This relationship is independent of the image orientation and size and on typical non-destructive image processing

operations, such as recoloring, brightness adjustment, filtering, lossy compression, or small noise adding. Thus, it is computationally much more efficient to search an extensive image database by matching the extracted bit-string rather than whole images.

1.7 References

1. M. D. Swanson, B. Zhu, and A. H. Tewfik, “Data Hiding for Video in Video”, *Proc. ICIP '97*, vol. II, pp. 676–679, 1997.
2. M. Holliman, N. Memon, and M. M. Yeung, “On the Need for Image Dependent Keys for Watermarking”*, Proc. Content Security and Data Hiding in Digital Media*, Newark, NJ, May 14, 1999.
3. J. Fridrich, “Methods for Tamper Detection in Digital Images”, *Proc. ACM Multimedia 1999, Workshop on Multimedia and Security*, October 30 – November 5, 1999
4. I. J. Cox and J.-P. M. G. Linnartz, “Public watermarks and resistance to tampering”, *ICIP’97*, Santa Barbara, California, October 1997. Paper appears only in CD version of proceedings.
5. I. J. Cox and J.-P. M. G. Linnartz, “Some general methods for tampering with watermarks”, preprint, 1998.
6. J. Fridrich, “Robust Bit Extraction From Images”, *Proc. ICMCS'99*, vol. 2, Florence, Italy, June 7–11, pp. 536–540, 1999.
7. S. Mallat, *A Wavelet Tour of Signal Processing*, Chestnut Hill, MA, Academic Press, 1998.
8. R. D. Brandt and F. Lin, "Representations that uniquely characterize images modulo translation, rotation and scaling", *Pattern Recognition Letters* **17**, pp. 1001–1015, 1996.
9. J. J. K. Ó Ruanaidh and T. Pun, “Rotation, scale and translation invariant digital image watermarking”, *Proc. of the ICIP'97*, vol. 1, pp. 536–539, Santa Barbara, California, 1997.
10. J. Fridrich, "Visual Hash for Oblivious Watermarking", *Proc*. *SPIE Photonic West Electronic Imaging 2000, Security and Watermarking of Multimedia Contents*, San Jose, California, January 24–26, 2000.
11. Holiman M., Macy W., and Yeung M.M., "Robust Frame-Dependent Video Watermarking", *Proc*. *SPIE Photonic West Electronic Imaging 2000, Security and Watermarking of Multimedia Contents*, San Jose, California, January 24–26, 2000.
12. S. Walton, “Information Authentication for a Slippery New Age”, *Dr. Dobbs Journal*, vol. 20, no. 4, pp. 18–26, Apr 1995.
13. P. Wong, "A Watermark for Image Integrity and Ownership Verification", *Proc. IS&T PIC,* Portland, Oregon, 1998.
14. R. B. Wolfgang and E. J. Delp, "Fragile Watermarking Using the VW2D Watermark", *Proc. SPIE, Security and Watermarking of Multimedia Contents*, San Jose, California, Jan 25–27, 1999, pp. 204–213.
15. C. W. Honsinger, P. Jones, M. Rabbani, J. C. Stoffel, "Lossless Recovery of an Original Image Containing Embedded Data", US Patent application, Docket No: 77102/E–D, 1999.

16. D. Kundur and D. Hatzinakos, "Towards a Telltale Watermarking Technique for Tamper Proofing", *Proc. ICIP*, Chicago, Illinois, Oct 4–7, 1998, vol 2.
17. J. Fridrich and M. Goljan, "Protection of Digital images Using Self-Embedding", *Symposium on Content Security and Data Hiding in Digital Media*, New Jersey Institute of Technology, May 14, 1999.
18. J. Fridrich and M. Goljan, "Images with Self-Correcting Capabilities", *ICIP'99*, Kobe, Japan, October 25–28,1999.
19. Ching-Yung Lin and Shih-Fu Chang, "Semi-Fragile Watermarking for Authenticating JPEG Visual Content", *Proc. SPIE, Security and Watermarking of Multimedia Contents*, San Jose, California, Jan 24–26, 2000, pp. 140–151.
20. Ming Kuei-Hu, "Visual Pattern Recognition by Moment Invariants", *IRE Transactions on Information Theory*, Vol. 8, pp. 179–187, February 1962.
21. L. Xie and G. R. Arce, "A Class of Authentication Digital Watermarks for Secure Multimedia Communication", preprint, submitted to *IEEE Transactions on Image Processing*, December 1999.
22. B. Schneier, *Applied Cryptography*. Wiley, New York, 1996.
23. M. Alghoniemy and A. H. Tewfik, "Progressive Quantized Projection Watermarking Scheme", *Proc. ACM Multimedia '99*, Orlando, Florida, November 2–5, 1999, pp. 295–298.

2. Fragile Global Invertible Authentication

Both cryptographic techniques and the new authentication can detect every possible change in the image. They provide the same guarantees as the MD5 function. Actually, the new fragile watermark uses MD5 and this is what guarantees its cryptographic strength. Instead of appending the hash to the image and encrypting, we *embed* the hash in the image itself. The hash stays with the image no matter what we do to the header or how we represent the image (lossless conversion format does not interfere with the authenticity, but can be made sensitive to it if desirable). The new invertible method allows reconstruction of the *exact* original data. The method is global meaning that tampering cannot be cannot localized.

In this report, we present two new methods for authentication of digital images using invertible watermarking. While virtually all watermarking schemes introduce some small amount of non-invertible distortion in the image, the new methods are invertible in the sense that, if the image is deemed authentic, the distortion due to authentication can be removed to obtain the original image data. Two techniques are proposed: one is based on robust spatial additive watermarks combined with modulo addition and the second one on lossless compression and encryption of bit-planes. Both techniques provide cryptographic strength in verifying the image integrity in the sense that the probability of making a modification to the image that will not be detected can be directly related to a secure cryptographic element, such as a hash function. We also explain that invertible authentication can only be achieved at the expense of not being able to authenticate every possible image. However, it is argued that all images that occur in practice can be authenticated. The techniques provide new information assurance tools for integrity protection of sensitive imagery, such as medical images or high-importance military images.

2.1 Introduction and problem statement

In today's world, digital images and video are gradually replacing their classical analog counterparts. This is quite understandable because digital format is easy to edit, modify, and exploit. Digital images and videos can be readily shared via computer networks and conveniently processed for queries in databases. Also, digital storage does not age or degrade with usage. On the other hand, thanks to powerful editing programs, it is very easy even for an amateur to maliciously modify digital media and create "perfect" forgeries. It is usually much more complicated to tamper with analog tapes and images. Tools that help us establish the authenticity and integrity of digital media are thus very essential and can prove vital whenever questions are raised about the origin of an image and its integrity.

The visual redundancy of typical images makes it possible to embed a weak imperceptible signal in the image making it capable of authenticating itself without accessing the original or other auxiliary data derived from the original. The advantage of having the authentication code embedded in the image rather than appended to it is obvious. One can say that watermarks authenticate the content of the image rather than its

particular representation. Lossless format conversion, such as changing the format from PGM to BMP, leads to a different representation of the image data but does not change the visual appearance of the image. Also, if the authentication information is lumped and localized in the image, one can localize the modifications as well as verify the content integrity of image fragments after cropping.

In this paragraph, we give an example of a practical scenario in which integrity protection using watermarks is more appropriate than authentication based on cryptography. Let us assume that we have possibly many digital cameras equipped with a watermarking chip that authenticates every image the camera takes before storing it on its flash card. The authentication watermark could be uniquely tied to the camera's serial number via a secret key hard-wired in the camera itself thus creating a link between the images and the hardware that took them. The watermark can also depend on some key information provided by the person who took the image. In the future, iris or thumbprint scans as well as simple punch-in passwords could be used to facilitate the connection between a digital image and the person who took them. The watermark could later be used to prove the integrity of a specific image and certify that it has been taken with certain hardware by a specific person. This application scenario is graphically explained in Figure 8.

Authentication based on watermarking has not been intended and cannot replace classical cryptographic authentication protocols that protect communication channels. Cryptographic authentication protects the communication link making sure that nobody can impersonate the sender and the data being sent has not been tampered with. This is typically achieved by attaching a hash of the image to it and encrypting the result. Watermarking cannot provide equivalent satisfactory properties for the same scenario in which subjects exchange data using public-key encryption infrastructure. This is intuitively clear because the authenticated image is usually perceptually equivalent to the original, and hence an attacker can simply overwrite the image with his watermark and the resulting image will appear to be authenticated by the attacker. Although this problem can be alleviated to some extent by embedding in the image the sender's ID in a robust manner, the security this mechanism provides is nowhere close to what can be obtained using cryptographic authentication. The authentication watermark can, however, be useful in those cases when the set of authentication keys is well defined and controlled such as keys wired in a digital camera. One should think of authentication watermarks as additional secondary image assurance tools that may come in handy if an origin and/or integrity of an image comes in question.

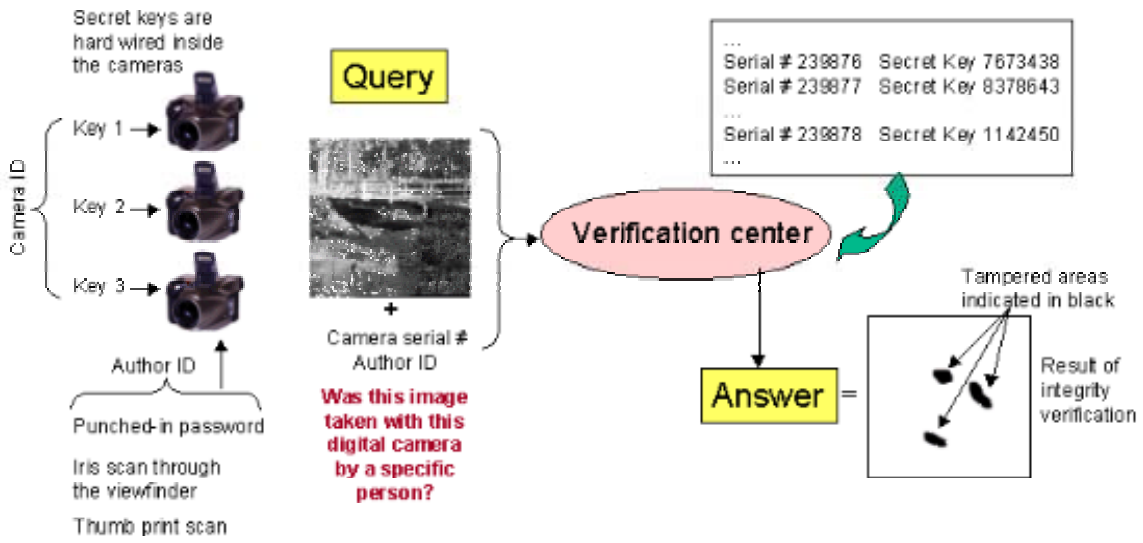


Figure 8 Example of an application scenario in which images are authenticated with a fragile watermark determined by the secret camera key and author's ID. The verification center can perform integrity check on the image and verify that it was taken with a specific camera and author.

Cryptographically secure fragile watermarks [1–4] have been proposed as a means to verify image integrity without encrypting the image. Fragile watermarks typically depend on the hash of the image or an image block and are thus capable of detecting every change that occurred to the image. It can be shown that without the secret key, the probability of a modification that will not be detected can be related to the cryptographic element present in the scheme, such as a hash or a check-sum. Another advantage of digital authentication watermarks is their potential ability to localize the changes and in some cases even provide information about the distortion [5] or even reconstruct tampered/damaged areas in the image [6–8]. Those and similar techniques belong to the category of semi-fragile watermarks that allow authentication "with a degree". The goal of this soft authentication is to distinguish between malicious operations, such as feature adding/removal from non-malicious changes that do not modify the essential features in the image (slight filtering, high quality lossy compression, etc.). In this report, we limit ourselves to fragile authentication watermarking built upon secure cryptographic elements. Such schemes are designed to detect every possible change that occurred to the image with very high probability.

One possible drawback of authentication based on watermarking is the fact that the authenticated image will inevitably be distorted by some small amount of noise due to the authentication itself. In all previously proposed authentication watermarking schemes, this distortion cannot be completely removed even when the image is deemed authentic. Although the distortion is often quite small, it may be unacceptable for medical imagery (for legal reasons) or images with a high strategic importance in certain military applications. In this report, we analyze the conditions under which it is possible to "undo" the changes introduced by authentication if the image is verified as authentic. We present

some new concepts and techniques that make invertible authentication of typical images possible. Finally, we describe two watermarking techniques that embed Message Authentication Code (MAC) in images in an invertible way so that anyone who possesses the authentication key can revert to the exact copy of the original image before authentication occurred.

In Section 2.2, we discuss some fundamental limitations of invertible authentication and present some heuristic arguments that will help us with development of invertible schemes later. In Section 2.3, we describe invertible authentication based on additive robust watermarking schemes that utilize invertible modulo addition. A completely different authentication technique based on lossless compression of bit-planes is presented in Section 2.4. This portion of the research is concluded in the last section 2.5 where we also discuss ideas for future research.

2.2 Invertible authentication – fundamental limitations

There are several different sources of non-invertible distortion that typically occur during watermarking. Techniques in which information is replaced (discarded), such as in the Least Significant Bit (LSB) embedding either directly in the colors or transform coefficients, are obviously lossy. The discarded information is irreversibly replaced with the message. In techniques that use quantization, information is also lost because multiple values are possibly rounded to one quantized value. Another source of information loss in watermarking is rounding of quantities due to their limited range. This includes rounding errors due to truncation to integers and rounding at the boundary grayscale values of 0 and 255. This last issue applies even to additive watermarks in the spatial domain that would otherwise be invertible.

Invertible authentication is not possible if we insist that all possible images, including "random" images, be authenticable. This can be explained as follows. Let G denote the set of all grayscale images formed by all ordered $M \times N$-tuples of integers from the set $\{0, \ldots, 255\}$. The authentication process is described with a mapping F_K: $G \rightarrow G$ and it depends on a secret key K. The set $A_K = F_K(G)$ is the set of all images authenticated with the key K. Since we do not want the authenticated image to contain any disturbing artifacts, we require that the original image I and the authenticated image $F_K(I)$ be perceptually "close". One obvious necessary requirement of a secure authentication scheme is that the probability that an arbitrary image is deemed authenticated with K by pure chance should be very small. This implies that the cardinality of A_K must be significantly smaller that that of G: $|A_K| << |G|$. And this should be true for all keys K. This simple argument indicates that the mapping F_K cannot be one-to-one, but, in fact, must be many-to-one with the cardinality of $F_K^{-1}(I)$, $I \in A_K$ that is very high. For example, in the Wong's scheme [2], the cardinality of $F_K^{-1}(I)$ is at least $2^{M \times N}$ where M and N are the image dimensions. This is because in Wong's scheme the LSBs of the original image are erased and replaced with the XOR of the hash of the 7 Most Significant Bits (MSBs) and a binary logo. Consequently, all images differing only in their LSBs will authenticate to the same image.

This observation seems to suggest that invertible authentication is not possible. This would essentially be true if we insisted that all images in G be authenticable. However, the set of all images that occur "in the real world" is significantly smaller than G. Typical images are highly correlated and form a very small subset of G. So, it makes sense to ask a question if it is possible to design an invertible authentication scheme for typical images at the expense of not being able to authenticate every possible image in G. The answer is positive and we elaborate on this topic in the next two sections.

2.3 Invertible authentication using robust watermarks

The first publication on invertible authentication the authors are aware of is the patent by Honsinger et al. [4] owned by The Eastman Kodak Company. We briefly outline the main idea behind their technique and generalize their approach. We also present some general design comments and discuss the performance, properties, and limitations of invertible schemes that utilize modulo addition and a robust spatial additive watermark.

It was explained in the introduction that the main reason why virtually all watermarking techniques cannot be completely reversed is the loss of information due to discarded (replaced) information, quantization, and integer rounding at the boundaries of the grayscale range (at zero and 255 gray levels). There is little hope that an invertible watermarking scheme could be constructed from schemes in which information is replaced or quantized. On the other hand, a typical spatial additive watermark is almost invertible because the watermark can be exactly subtracted from most pixels with the exception of pixels truncated due to over and underflow. If we were somehow able to guarantee that no truncation occurs, we should be able to subtract an additive watermark pattern from a watermarked image and reverse to the original data. If the watermark payload was the hash of the original image, we could simply check if the extracted hash matches the hash calculated for the image after subtracting the watermark pattern. This idea forms the basis of the first invertible authentication method based on robust additive watermark in the spatial domain:

Algorithm for invertible authentication using spatial robust additive watermarks:

1. Let $I \in G$ be the original image to be authenticated. Calculate its hash $H(I)$.
2. Choose an additive, non-adaptive robust watermarking technique and generate a watermark pattern W from a secret key K, so that the payload of W is $H(I)$. Note that we require that the watermark pattern W be a function of the key K and the payload $H(I)$ only, $W = W(K, H(I))$.
3. Use a special "invertible addition" $\oplus$ to add the watermark pattern W to I to create the authenticated image $Iw = I \oplus W$.

Integrity verification:

1. Extract the watermark bit-string H' (payload) from Iw (read the watermark).
2. Generate the watermark pattern W' from the key K and the extracted bit-string H': $W' = W(K, H')$.

3. Using the inverse operation, subtract W' from Iw to obtain $I' = I - W'$.
4. Compare the hash of I', $H(I')$, with the extracted payload H'. If they match, the image is authentic. If they do not match, the image is deemed not authentic.

2.3.1 Invertible addition

The key element of the invertible authentication is the invertible addition "⊕". As explained above, simple addition and truncation at the boundaries (at 0 and 255) is not invertible because it is impossible to distinguish between the cases when, for example, the grayscale 255 was obtained as 254+1 or due to an overflow and subsequent truncation. Honsinger et al. [4] proposed addition modulo 256 as an invertible operation. While this operation is indeed invertible, it may introduce a large error into the authenticated image when a grayscale value close to 255 is flipped to a grayscale close to zero, or a value close to zero is mapped to a value close to 255. In other words, the authenticated image may contain some "flipped" pixels (black to white and white to black). The visual impact of this resembles a salt and pepper noise and its visibility and extent depends on the number of saturated colors in the original image I and the watermark strength (amplitude). Even though one could easily give examples of images for which the artifacts will be very disturbing, typical images do not contain that many saturated colors and the artifacts due to pixel flipping are typically not that serious. Also, one should keep in mind that if the image is not modified (i.e., if it is authentic), one can invert the authentication and obtain the original image without any distortion. Thus, the presence of the artifacts is not a problem at all for those who posses the right authentication key.

The addition modulo 256 is a special case of a more general invertible addition. We can take any permutation P of the integers $\{0, 1, 2, \ldots, 255\}$ and define an invertible addition as $i \oplus k = P^k(i)$ for all grayscales i and k. The invertible addition from the previous paragraph can be represented using the following permutation: $0 \to 1$, $1 \to 2$, …, $254 \to 255$, $255 \to 0$. One can try to minimize the visual distortion due to flipped pixels by using permutations with shorter cycles. For example, we can cycle the grayscales in 16 cycles of length 16 rather than one cycle of length 256: $0 \to 1$, $1 \to 2$, …, $15 \to 0$, $16 \to 17$, $17 \to 18$, …, $31 \to 16$, …. This will guarantee that the maximal disturbance due to authentication will never be larger than 15 gray scales. The formula for this invertible addition is

$$i \oplus k = C \lfloor i/C \rfloor + \mathrm{mod}(i+k, C),$$

where $\lfloor x \rfloor$ stands for the integer part of x and $C = 16$ is the cycle length. The subtraction is defined as

$$i - k = i \oplus (-k).$$

2.3.2 Watermarking technique

The choice of the watermarking technique has a big influence on the overall performance of the authentication algorithm. First, the capacity of the technique must be at least equal to the length of the hash $H(I)$. Second, because we add the watermark pattern W using the invertible addition rather than the "truncated addition", as it is done in virtually all watermarking techniques, the watermarking algorithm must be robust with respect to the distortion due to "pixel flipping". The flipped pixels can be considered as a special case of a correlated salt-and-pepper noise. They will mostly influence the high frequencies in the image. Therefore, watermarking techniques that are robust with respect to adding small amount of salt-and-pepper noise are good candidates for application in the proposed invertible authentication scheme. Honsinger et al [4], use their spatial watermarking algorithm based on random phase spreading. This algorithm provides sufficient capacity and is extremely robust to a wide range of distortions [10]. According to the authors, their invertible authentication technique was successfully tested on a database of several hundred images.

In this report, we have decided to test the authentication method with a spread-spectrum frequency-based robust watermarking algorithm [11] in order to show the artifacts due to flipped pixels and explain possible problems that can be encountered and strategies for their solutions. We note that the watermarking technique does not have to be a spatial method but could utilize image transformations as an intermediate step as well. The only requirement is that it should be additive and the watermark pattern W must be a function of the key and the payload only (shaping the watermark pattern using a perceptual mask would introduce additional error into the scheme that may prevent us from being able to authenticate some images). The watermarking method starts with calculating the DCT transform of the whole image and arranging the DCT coefficients in a zigzag pattern. Then, the middle 30% of the DCT coefficients D_k are modulated by a Gaussian signal S_k with zero mean and unit standard deviation by simply adding both signals

$$D'_k = D_k + \alpha S_k , \quad k = 1, \ldots, N_m ,$$

where D'_k denotes the modulated DCT coefficients, α is an image independent watermark strength, and N_m is the number of modified coefficients. The watermarked image is obtained by performing the inverse DCT using the modulated coefficients D'_k. In order to use the technique in our context, it was rewritten in a matrix form to the spatial domain

$$X' = X \oplus \mathrm{IDCT}(\alpha S),$$

Where X is the original image, X' is the watermarked image, and S is the Gaussian sequence S_k in the matrix form synthesized so that it carries all hash bits. Similar to the approach proposed by Ruanaidh [11], we represent the hash using M=25 six-bit symbols s_i, $1 \le s_i \le 2^6$. For each i, a sequence $\xi^{(i)}$ of pseudo-random numbers of length N_m+2^6 uniformly distributed in [−1,1] is generated. Each symbol s is represented using the segment $\eta^{(i)} = \xi_s^{(i)}, \ldots, \xi_{s+N_m-1}^{(i)}$ of consecutive N_m pseudo-random numbers. For each

symbol, a new sequence of pseudo-random numbers is generated. The secret key is used as a seed for the PRNG. The hash bit-string is then represented as a summation

$$S = \sqrt{\frac{3}{M}} \sum_{i=1}^{M} \eta^{(i)} \ .$$

The spread spectrum signal S is approximately Gaussian with zero mean and a unit standard deviation.

The detection of the hash proceeds by first transforming the authenticated image X' using a DCT and extracting the middle N_m DCT coefficients. The secret key is used to generate M pseudo-random sequences of length N_m+2^6 needed for coding the symbols s_i. For each sequence, all 2^6 segments of length N_m are correlated with the middle N_m DCT coefficients. The largest value of the correlation determines the encoded symbol s_i. We point out that this detection method provides significantly more robustness when compared to a simple correlation of DCT coefficients with the spread spectrum signal and thresholding with a fixed threshold. Since *relative* values of correlation are compared rather than *absolute* values, the hash can be extracted even after the image has been severely distorted.

We observed that a low amplitude watermark $0.8 \leq \alpha \leq -1$, which corresponds to a maximal spatial watermark amplitude of 2, gave us actually better results than a stronger watermark. This is because a weak watermark will cause fewer flipped pixels thus increasing the chances that an image will be authenticable. We first performed tests on the test image "Lenna" with 256 grayscales and 512×512 pixels. We used the addition modulo 256. The watermarked image had PSNR of 54.1dB and there were no flipped pixels in the image due to modulo addition. As our second experiment, we chose a 512×512 grayscale image with a large area of pixels saturated at gray level 0 (see Figure 9). The PSNR of the watermarked image was 13.7dB due to 11273 flipped pixels that can be seen as white dots on black background in Figure 10. To minimize the effect of flipped pixels on a successful extraction of the hash, we found it necessary to use a preprocessing step before extracting the payload from the authenticated image. Most of the flipped pixels can be easily detected and their values corrected by subtracting/adding the period of the modulo addition before the watermark extraction occurs. We note that the resulting preprocessed image may have a larger dynamic range than 0–255. This preprocessing step significantly increases the reliability of the verification procedure, or, equivalently, more images from G are authenticable. Of course, once the payload has been extracted, the watermark pattern W should be subtracted from the authenticated image and not the preprocessed image.

The authors found it rather difficult to formally analyze this proposed invertible authentication technique in an attempt to define the set of images that will be authenticable. It appears that the only way to determine whether or not an image is authenticable is to run the authentication and then verify if the distortion due to invertible addition enables successful integrity verification. As pointed out above, the reliability of this process can be significantly increased by pre-processing the image under

investigation to minimize the influence of the flipped pixels. We want to point out that it is not necessary for the watermark to be robust against any other distortion than the distortion due to invertible addition. The authors believe that a special watermarking technique with robustness targeted to artifacts due to invertible addition can be designed. We anticipate that for such a technique, the set of authenticable images will be easier to define using a more exact mathematical approach. This problem will be part of our future effort.

Finally, we note that whenever the construction of the watermark pattern $W(K, H)$ from the key and the hash utilizes floating point operations instead of just integer operations, there is a non-zero probability, albeit extremely small, that even an image without saturated colors will not authenticate properly if the verification program is run on a computer with different floating point precision. This is because in very rare cases it may happen that accumulation of errors of the order of the machine precision may give us a slightly different watermark W on different computers even though the watermark payload was extracted correctly.

Figure 9 The original 512×512 grayscale image "Moon" with a large area of pixels with grayscales zero.

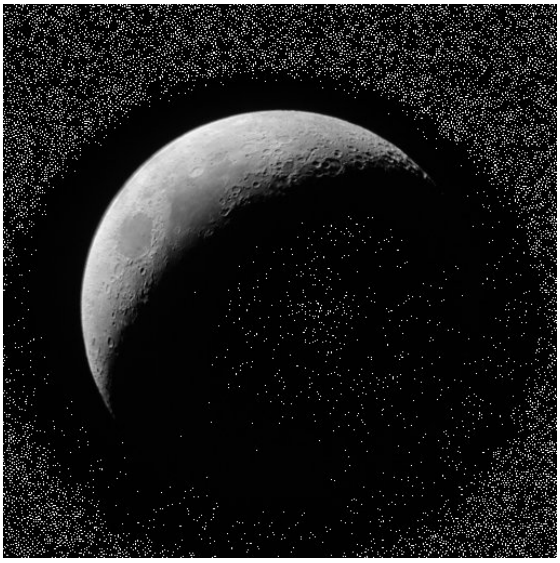


Figure 10 Authenticated image "Moon" with 11273 black pixels flipped to white pixels due to modulo 256 addition.

2.3.3 Practical implementation considerations

Note that it is easy to find out if a given image is authenticable by simply running the verification algorithm for the authenticated, non-tampered image. If it happens that the image is not authenticable, we might be able to change the values of a few pixels by a small amount and rerun the authentication procedure till we obtain an authenticable image. In this case, the scheme could be termed "nearly invertible".

We close this section with a few security considerations. Because a robust watermarking method is used, even if the authenticated image is tampered, it may be possible to recover

the payload, the hash $H(I)$, correctly. However, when the watermark pattern $W(K,H(I))$ is subtracted from the authenticated (tampered) image, the calculated image hash will not match the extracted hash and thus the method will not determine the image as authentic. Any small random change to the authenticated image will be detected with a high probability because the chances of obtaining a match between the calculated image hash and the extracted hash are equal to finding a collision for the hash.

2.4 Invertible authentication using lossless compression

The problem of invertible image authentication by embedding a hash can be interpreted as adding additional information to the image in an invertible manner without increasing the image size. Because natural images are highly correlated it should be possible to insert the hash by increasing the image entropy without causing irreversible damage. At the same time, the embedding must provide security comparable to cryptographic authentication. In this section, we propose to combine lossless compression with encryption to achieve this goal. By losslessly compressing the image (or its part), we can make some space where the image hash could be inserted. This needs to be done without introducing a large distortion to the image. Even though the distortion can be completely removed, it is desirable to preserve as much of the visual content of the image as possible. We propose to use the lowest bit-plane that, after lossless compression, provides enough space for the image hash. Unless all bit-planes in the original image are noisy, which would correspond to a completely noisy image, there will be some bit-planes exhibiting strong correlations. Such bit-planes will be compressible in a lossless manner. At the same time, we hope that randomizing one of those bit-planes will not introduce severe artifacts into the image.

2.4.1 The method for invertible authentication using lossless compression

In this report, we use the JBIG lossless compression scheme [12] for compressing the bit-planes. The algorithm always starts with inspecting the 5^{th} LSB plane. For color images, all three color-channels are compressed separately. We continue by calculating the redundancy for each channel:

$$\text{Redundancy} = \text{Number of pixels} - \text{Compressed data size (bit)}.$$

For channels that can't be compressed (their redundancy is not positive), we set their redundancy to zero. Such channels will not be modified during authentication. If the sum of redundancies for all three channels is greater than or equal to 128, we go to the next lower bit-plane (the 4^{th} LSB plane). If the sum is less than 128, we go to the next higher bit-plane (the 6^{th} LSB plane). This process is repeated until we find a bit-plane for which we cannot go lower. For such a bit-plane, the sum of redundancies is larger than or equal to 128 and the redundancy of the next lower bit-plane is less than 128. This is the lowest bit-plane that provides enough space for lossless embedding of the image hash and it will be called the *key bit-plane*. To minimize the distortion introduced by authentication, we use this key bit-plane for hash embedding.

The authentication process continues with evaluating the 128-bit image hash using the hash function MD5. Starting from the channel with the largest redundancy, we concatenate the hash bits to the compressed channel data so that the length of the compressed data with the length of concatenated hash bits equals to image size. This is repeated for each channel in the order of decreasing redundancy until all hash bits have been concatenated with the compressed data.

The key bit-plane now consists of the compressed stream and the concatenated hash. To prevent somebody from manipulating the image and replacing the hash, the bit-plane should be encrypted to obtain a secure scheme. The encrypted bit-stream is embedded in the corresponding channel(s) of the key bit-plane. Consequently, the authenticated image will have a randomized key bit-plane and thus it will contain some noise. In our implementation, we decided to not use classical encryption engines, such as IDEA or DES. These algorithms have a fixed encryption block size, and, depending on the image dimensions, we would have to pad the compressed bit-plane concatenated with the hash in order to apply the encryption. In the worst case, this padding would cost us additional 63 bits per channel (if the block size is 64 bits), which might force us to use higher bit-plane for authentication and thus further increase the distortion. Instead, we opted for a special symmetric encryption scheme based on two-dimensional chaotic maps described in [14–16]. This scheme fits our needs because its block size is variable and equal to the image size and no padding is needed. The principle of this scheme is briefly described in Appendix A.

The integrity verification proceeds in an inverse order. First, we determine the key bit-plane that was used in the authentication process. Note that the redundancy of the key bit-plane can be both negative and positive. If all three channels are used for hash embedding, the redundancy will be negative. If, however, only one channel was used for hash embedding, the redundancy may still be positive. In either case, the redundancy of the next lower bit-plane below the key bit-plane must always be negative. Based on this observation, in the integrity verification process, we calculate the redundancies of the bit-planes and take the highest bit-plane i with negative redundancy (the LSB plane has i = 1). The key bit-plane is either the i-th bit-plane or the i+1st bit-plane. Both are valid candidates for the key bit-plane used in the authentication. We try both bit-planes to verify the image. For an authentic image, one of them will give us positive integrity check. If none of them leads to successful integrity verification, the image is considered not authentic.

After we obtain the data from the key bit-plane, we perform decryption and the decrypted data of each channel is decompressed with the JBIG method. In our work, we used the JBIG implementation from ImageMagick [13]. For channels that are not modified during embedding, the JBIG decompression process will return an error because there is a fixed 20-byte header used by the JBIG method. In this header, there are 12 bytes that are derived from the image (4 bytes for width, 4 bytes for height, and 4 bytes related to the compressed data size). The remaining 8 bytes are fixed parameters or options used in JBIG. Thus, the probability of considering the data of an unmodified channel as a JBIG compressed data is at most $(1/2)^{64}$. From the JBIG decompression process, we know the

redundancy for each channel and we can concatenate the extracted hash bits together in the correct order.

Finally, we put the decompressed data back to its corresponding bit-plane and channels and calculate the hash of the recovered image. If the calculated hash matches the hash extracted from the key bit-plane, the image is deemed authentic, otherwise it is not.

We note that the performance of the method can be slightly improved by shortening the JBIG header only to information that cannot be fixed or read from the image itself. The only parameter that can not be obtained from the host image is the size of the compressed data. Thus, the JBIG header can be shortened to only 4 bytes. In this way, the possibility of embedding the image hash to lower bit-planes is increased and the distortion introduced by authentication will on average become smaller.

The test image "Moon" shown in Figure 9 is one of the best candidates for this authentication method because its LSB plane has large uniform areas due to the pixels with grayscale 0. Thus, the key bit-plane for the Moon image is the LSB plane and the distortion due to authentication is 51.2dB.

Below, we include a simple pseudo-code for grayscale images:

Algorithm for invertible authentication using lossless bit-plane compression:

1. For the original image I, identify the key bit-plane as the lowest bit-plane with redundancy greater than or equal to 128.
2. Losslessly compress the key bit-plane denoting the compressed bit-stream as C.
3. Append the hash $H(I)$ of the original image to C, padding with random bits if necessary, to form the final bit-stream B of length $M \times N$: $B = C$ & $H(I)$ & random bits.
4. Replace the key bit-plane of I with the encrypted bit-stream $E_K(B)$, where E is an encryption scheme and K is a secret key for the encryption.

Verification:

1. Identify the two candidates for the key bit-plane.
2. Decrypt and decompress the key bit-plane. Extract the hash of the original image I and the original bit-plane.
3. Replace the key bit-plane with the decrypted uncompressed bit-plane and calculate the hash of the image.
4. If the calculated hash matches the extracted hash for one of the two candidate key bit-planes, the image is deemed authentic, otherwise it is not.

2.4.2 Security Considerations

Any change made to the key bit-plane of the authenticated image will change the decrypted bit-plane, and therefore the extracted hash. The chances of obtaining a match

are astronomically small and are directly related to obtaining a collision for the hash. Changes made to other bit-planes will also cause a mismatch because the calculated hash will not match the extracted one. Again, the probability of obtaining a match is comparable to finding a collision for the hash. We conclude that, when it comes to detecting modifications in the image, the security provided by the proposed fragile watermarking scheme is equivalent to the security of cryptographic authentication based on secure hash functions.

2.5 Summary and future directions

In this report, we propose two new invertible authentication techniques based on fragile watermarks for digital images. The fragile watermarks have cryptographic strength and are global in the sense that they can detect every modification made to the image with probability equivalent to finding a collision for a cryptographically secure hash function. Although the watermarking techniques introduce a small amount of distortion to the image, this distortion can be completely removed (inverted) if the image is deemed authentic. The invertible authentication is only possible at the expense of not being able to authenticate every possible image. On the other hand, it is argued that all typical images can be authenticated.

The first technique embeds the hash of the whole image as a payload for a robust watermark. Any watermarking technique that can be used as a non-adaptive additive technique in the spatial domain can be used. To enable a complete removal of the distortion, the operation of addition is replaced with a modified addition based on modulo arithmetic. This operation actually introduces some potentially large distortion (flipped colors of pixels resembling the salt-and-pepper noise) in the authenticated image. In fact, this is the only distortion with respect to which the watermarking technique needs to be robust. To minimize the distortion in the authenticated image, a whole range of invertible addition operators has been proposed. Special preprocessing that eliminates the influence of flipped pixels is also recommended for the integrity verification in order to maximize the set of authenticable images.

It is rather difficult to characterize the set of all images that can be authenticated using this method. One really needs to run the algorithm and attempt to verify its integrity before one can say whether or not a given image is authenticable. Images that contain large areas of saturated colors, such as astronomical images (see Figure 9), are likely to cause problems because the number of flipped pixels may be too large for a reliable extraction of the watermark. On the other hand, because the only distortion with respect to which the watermark must be robust is the distortion introduced due to the invertible addition (pixel flipping), the authors believe that a special technique tailored to this application could be developed. For such a technique, the flipped pixels would not pose a serious problem, and the set of authenticable images would be sufficiently large. One possibility would be to embed the watermark into cyclical variables, such as the hue.

The second method for invertible authentication based on lossless compression of bit-planes and encryption is much more transparent for analysis. Images that cannot be

authenticated must contain noise in all eight bit-planes, which essentially means that they must look like noise. One can probably say with confidence that every image that one can take with a digital camera will be authenticable. For this second technique, images with large areas of saturated colors are actually good images and can often be authenticated in the LSB plane.

Both methods can be localized to blocks rather than applied to the whole image. To be able to detect swapping of blocks among different authenticated images, one would have to concatenate the image index to the hash. The size of the block is limited by the minimal necessary robustness required for watermark extraction in the first method. Depending on the watermarking method, we may be able to afford as small as 64×64 or 128×128 pixel blocks. If too small a block is used in the second method, however, one will be forced to use higher bit-planes and increase the noise to an unacceptable level. Nevertheless, since the distortion is invertible, small block sizes may be used as long as the visual content of the image is still recognizable.

The idea behind the second invertible authentication technique is actually quite general and can be extended to other objects besides images. Let us assume that we have an object X represented in a discrete form using bits. For example, X could be a complex multimedia object, an audio file, a digitized hologram, a representation of a 3D structure, or any other digital object that needs to be authenticated. Let us further assume that it is possible to identify a subset $E \subset X$ that has a structure and that can be randomized without changing the essential properties of X or its semantic meaning. The subset E needs to have enough structure to allow lossless compression by at least 128 bits (the hash of X). One can then authenticate X in an invertible manner by replacing the subset E with an encrypted version of its compressed form concatenated with the hash $H(X)$.

APPENDIX A

The following pseudo-code is used to encrypt a rectangular array of integers g_{ij} in the interval $[0, 2^L-1]$:

For k=1 to 10

Permutation $pg_{t1(i,j),t2(i,j)} = (g_{ij} + i \cdot j) \bmod 2^L$

Diffusion odd iteration $g_{ij} = (pg_{ij} + G[pg_{ij-1} \bmod 2, pg_{ij-2} \bmod 2, ..., pg_{ij-8} \bmod 2]) \bmod 2^L$

even iteration $g_{ij} = (pg_{ij} + G[pg_{ij+1} \bmod 2, pg_{ij+2} \bmod 2, ..., pg_{ij+8} \bmod 2]) \bmod 2^L$

End loop.

The array g_{ij} is the input data for encryption and it is also the final output of encrypted data. The function G is a permutation of integers {0, 1, ..., 255} generated from a secret key. In our implementation above, the argument of G is actually represented in a binary form (8 bits). The arrays $t1$ and $t2$ are called transfer matrices and are derived from a discretized chaotic map using the same secret key.

The following pseudo-code is used for the decryption (the array pg_{ij} denotes both the input and the output bit-stream):

For k=10 to 1

Diffusion odd iteration $g_{ij} = (pg_{ij} - G[g_{ij-1} \bmod 2, g_{ij-2} \bmod 2, ..., g_{ij-8} \bmod 2]) \bmod 2^L$

even iteration $g_{ij} = (pg_{ij} - G[g_{ij+1} \bmod 2, g_{ij+2} \bmod 2, ..., g_{ij+8} \bmod 2]) \bmod 2^L$

Permutation $pg_{ij} = (g_{t1(i,j),t2(i,j)} - i \cdot j) \bmod 2^L$

End loop

More details on the encryption scheme and its analysis can be found in the work by Fridrich [14–16].

2.6 References

1. S. Walton, “Information Authentication for a Slippery New Age”, *Dr. Dobbs Journal*, vol. 20, no. 4, pp. 18–26, Apr 1995.
2. P. Wong, "A Watermark for Image Integrity and Ownership Verification", *Proc. IS&T PIC,* Portland, Oregon, 1998.
3. R. B. Wolfgang and E. J. Delp, "Fragile Watermarking Using the VW2D Watermark", *Proc. SPIE, Security and Watermarking of Multimedia Contents*, San Jose, California, Jan 25–27, 1999, pp. 204–213.
4. C. W. Honsinger, P. Jones, M. Rabbani, J. C. Stoffel, "Lossless Recovery of an Original Image Containing Embedded Data", US Patent application, Docket No: 77102/E–D, 1999.
5. D. Kundur and D. Hatzinakos, "Towards a Telltale Watermarking Technique for Tamper Proofing", *Proc. ICIP*, Chicago, Illinois, Oct 4–7, 1998, vol 2.
6. J. Fridrich and M. Goljan, "Protection of Digital images Using Self-Embedding", *Symposium on Content Security and Data Hiding in Digital Media*, New Jersey Institute of Technology, May 14, 1999.
7. J. Fridrich and M. Goljan*,* "Images with Self-Correcting Capabilities"*, ICIP'99*, Kobe, Japan, October 25–28,1999.
8. Ching-Yung Lin and Shih-Fu Chang, "Semi-Fragile Watermarking for Authenticating JPEG Visual Content", *Proc. SPIE, Security and Watermarking of Multimedia Contents*, San Jose, California, Jan 24–26, 2000, pp. 140–151.
9. B. Zhu, M. D. Swanson, and A. Tewfik, “Transparent Robust Authentication and Distortion Measurement Technique for Images”, preprint, 1997.
10. C. W. Honsinger, "A Robust Data Hiding Technique Based on Convolution with a Randomized Phase Carrier", *Proc. of PICS'00*, March 26–29, 2000, Portland, Oregon, USA.
11. A. Herrigel, J. Ó Ruanaidh, H. Petersen, S. Pereira, T. Pun, “Secure copyright protection techniques for digital images,” *Proc. of the 2nd Int. Information Hiding Workshop*, Portland, Oregon, 1998.
12. K. Sayood, *Introduction to Data Compression*, Morgan Kaufmann, pp. 87–94.
13. Software ImageMagick, http://www.wizards.dupont.com/cristy/www/perl.html.

14. J. Fridrich, “Symmetric Ciphers Based on Two-Dimensional Chaotic maps”, *Int. J. Bifurcation and Chaos*, **8**(6), June 1998, pp. 1259–1284.
15. J. Fridrich, “Secure Image Ciphering Based on Chaos”. Final report for Grant #F30602-96-1-0047 for Air Force Research Laboratory, NY, March 10, 1997.
16. J. Fridrich, "Techniques for Secure Image Processing". Final Report for Grant # F30602-98-C-0009 sponsored by the Air Force Research Laboratory, Rome, NY, October 1999.

3. Secure Image Distribution Using Invertible Fidelity Degradation

3.1 Motivation and problem statement

In order to explain the motivation for the research and the techniques described in this report, we will assume the following model situation. A satellite sensor takes high-resolution images and sends them to a Central Distribution Unit (CDU). The task of the CDU is to forward the imagery to recipients belonging to several security classes. Recipients from the highest security class, such as certain military subjects or government organizations, will be allowed to see the original high-resolution image. Another class of recipients, such as military subcontractors, NATO allies, or certain private companies, will be allowed to see only lower-fidelity images with lower degree of detail obtained as degraded versions of the high-fidelity images. It may be desirable to have another set of images with even more degradation available to recipients in foreign countries. Depending on the application, we may desire that one or more degraded versions exist for recipients from the lowest security class (the public access). Recipients from different security classes will see different level of detail in the images. It is very important that high-resolution imagery is not sent by mistake to the wrong recipient. Also, it is absolutely necessary that there is no mechanism to somehow "invert" the intentionally introduced distortion, otherwise recipients from low security levels could access high-fidelity images.

One can think of several approaches to distribute the imagery to the recipients. For example, the CDU can degrade the original image to several different fidelities, and send corresponding images to their recipients via several channels. Each channel must be either secure or the images must be encrypted with a key shared between the CDU and each security class of recipients. This method is shown in Figure 11.

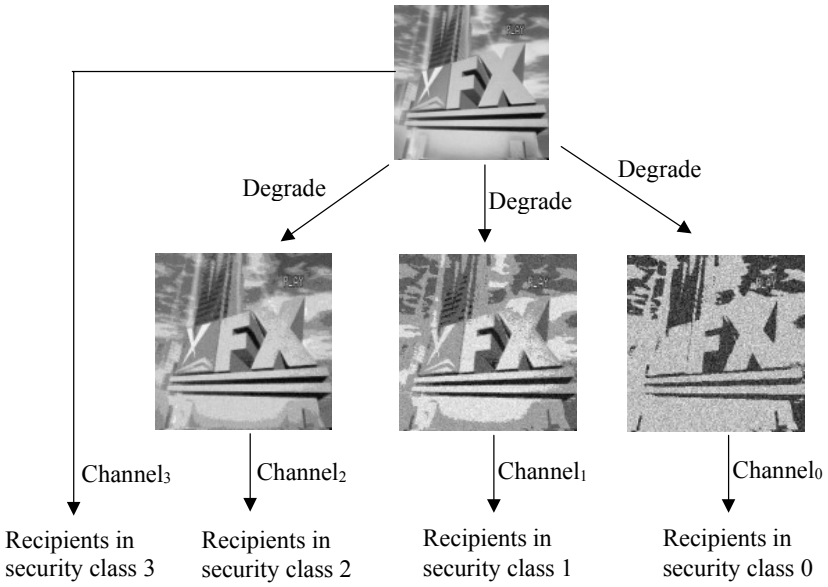


Figure 11 Send degraded images to recipients using separate secure channels.

In this method, the CDU has to degrade the images first and then make sure to send the degraded images to proper recipients. We may need several secure channels to send the images to prevent other recipients from accessing the imagery. Below, we describe a new method for secure distribution of images based on invertible degrading that is more convenient than the method described above. It can use only one public channel that does not have to be secure or encrypted thus providing a considerable simplification of the distribution process while minimizing the error of sending a high-fidelity image to the wrong recipient.

3.2 Proposed solution

In this report, we describe a class of techniques, that degrade image details using visible watermarking, so that high-fidelity image data can be managed and distributed through one network without security constraints. The government potentially can address multiple level security by controlling the apparent quality of the data, effectively making the data unclassified, and giving only authorized users access to the mechanisms that allow retrieval of the data essence. The commercial world can similarly benefit through an ability to use a single dataset to meet the broad fidelity needs of many customers. Only authorized customers will have the access methods that expose only certain level of detail.

The system developed in this effort reduces the apparent fidelity of the data using invertible techniques. In typical use, a high-fidelity image is transformed into a low-fidelity form by inserting a visible watermark into the image while retaining the data essence. The watermark insertion process depends on master key K. A set of access keys Key_1, Key_2, …, Key_R that allow access to image details is derived from the master key K. The low-resolution image is distributed to all recipients, with each recipient possessing exactly one access key. The key is used together with a public algorithm to reveal a specific level of detail that is masked in the low-fidelity image. For example, the Key_1 may correspond to the lowest access level, allowing the user to see only crude features in the image but no details. The Key_R may allow the user to access the original high-resolution image. By design, users with low-level access keys cannot circumvent the fidelity control watermark to either access higher levels of detail or approximate the details with a good accuracy. By making the watermark virtually impenetrable to attack, the only effect of an attack against the watermark is degraded data quality, and the fidelity control watermark prevents unintended access and use of the data.

The principle of this image distribution method is shown in Figure 12.

3.3 Requirements

The requirements for the fidelity control watermark are:

1. Contrary to the most common type of watermarks (the invisible watermarks), the fidelity control watermark has to be visible but it must retain the data essence.
2. The watermark should be robust against unauthorized removal.

3. The watermark should depend on a set of keys, each key allowing a limited, hierarchical access to image details.
4. The watermark must be strongly dependent on the image otherwise a good approximation to the watermark could be extracted from multiple images.
5. The watermark must be invertible.

Watermarks can be based on filtering, adding a controlled amount of noise, encrypting image bit-planes or other image features. In this report, we describe two methods. The first one is based on encrypting image bit-planes using a special chaotic symmetric block encryption algorithm, the second one is based on adding key-and-image dependent spatial noise modulo 256 (the maximal gray level).

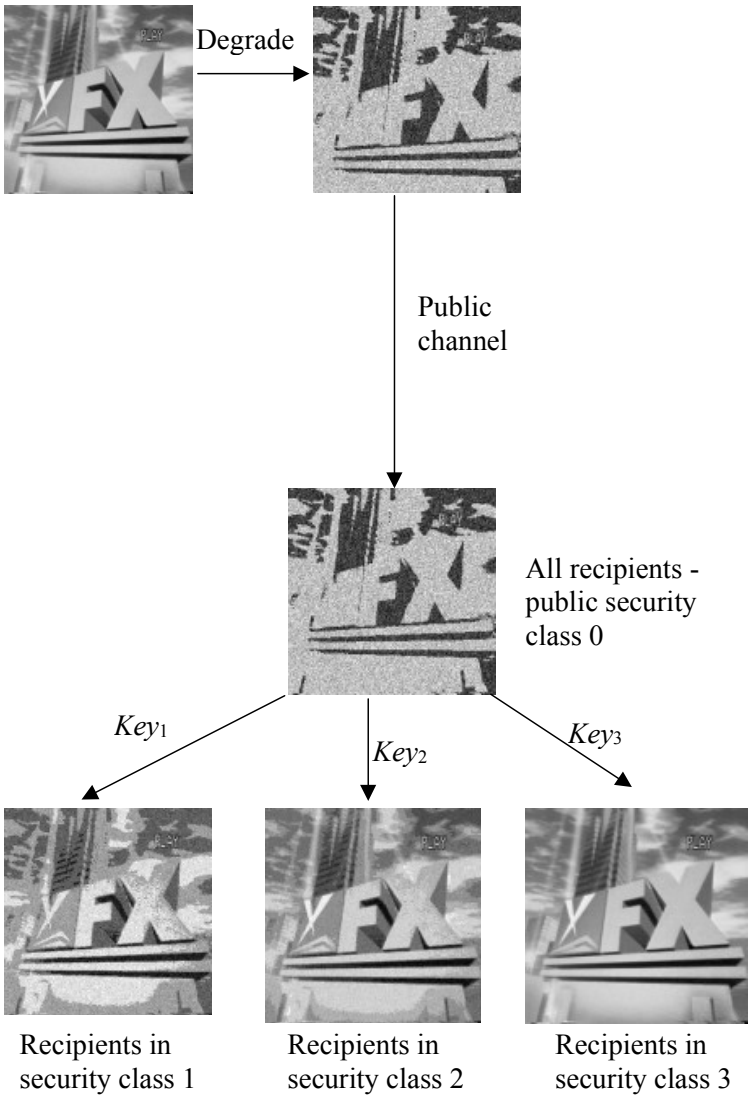


Figure 12 Send degraded images to all recipients using one public channel.

3.4 Encryption: Encrypt the least significant bits of an image.

This method was designed to allow 3 levels of degradation. Altogether, there will be four versions of an image: The original image (security class 3) and three degraded images (security classes 2, 1, and 0). The degrading process starts with a master key K from which three access keys Key_1, Key_2, and Key_3, are derived. This process is described in Section 3 on key management. To produce the degraded image for the security class 2 (the second highest class), we encrypt the 5 Least Significant Bits (LSBs) of the image. This has the effect of introducing a visible amount of noise in the image. Figure 13 shows the basic steps of the encryption. Suppose each pixel of the image has 8 bits (color 24-bit images are treated as three grayscale images). We divide the image bit-planes into two parts. One part will be formed by the 3 Most Significant Bits (MSBs) of each pixel and the other with the 5 LSBs of each pixel. We encrypt the second part using a special chaotic cipher described in [1–3]. Then we combine the encrypted 5 MSBs with the 3 MSBs and obtain a low-resolution image that will be sent to the security class 2 recipients. Figure 14 shows the result of this step for a grayscale 320×240 image.

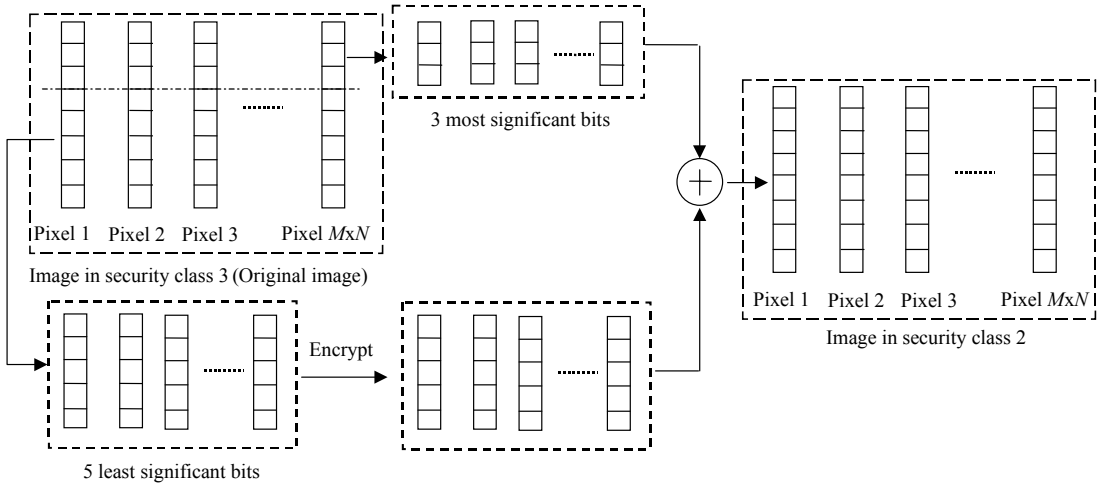


Figure 13 Step1: Encrypt the 5 LSBs of the original image (security class 3) to obtain an image of the security class 2.

The remaining two steps are similar in nature to the first step. In the second step, using the Key_2 we encrypt the 6 LSBs of the degraded image from security class 2 and obtain an image from security class 1. The amount of distortion is increased because one more least significant bit-plane has been encrypted (i.e., randomized). In the third step, we encrypt (using Key_1) the 7 LSBs of the degraded image from security class 1 to obtain the lowest quality image from security class 0. This image is the one that will be distributed to all recipients. Actually, it can be made public by posting it on a web page accessible to everybody. Only the right recipients who possess one of the access keys can access certain level of detail. The recipients in security class 0 (public) have no keys to upgrade this image, so they can only view this image at the lowest fidelity (lowest security class). Recipients in security class 1 can use Key_1 to upgrade the image to security class 1. The recipients in security class 2 will use Key_2 to upgrade the image to security class 2, and

finally the recipients in security class 3 use Key_3 to upgrade the image to the highest fidelity in security class 3.

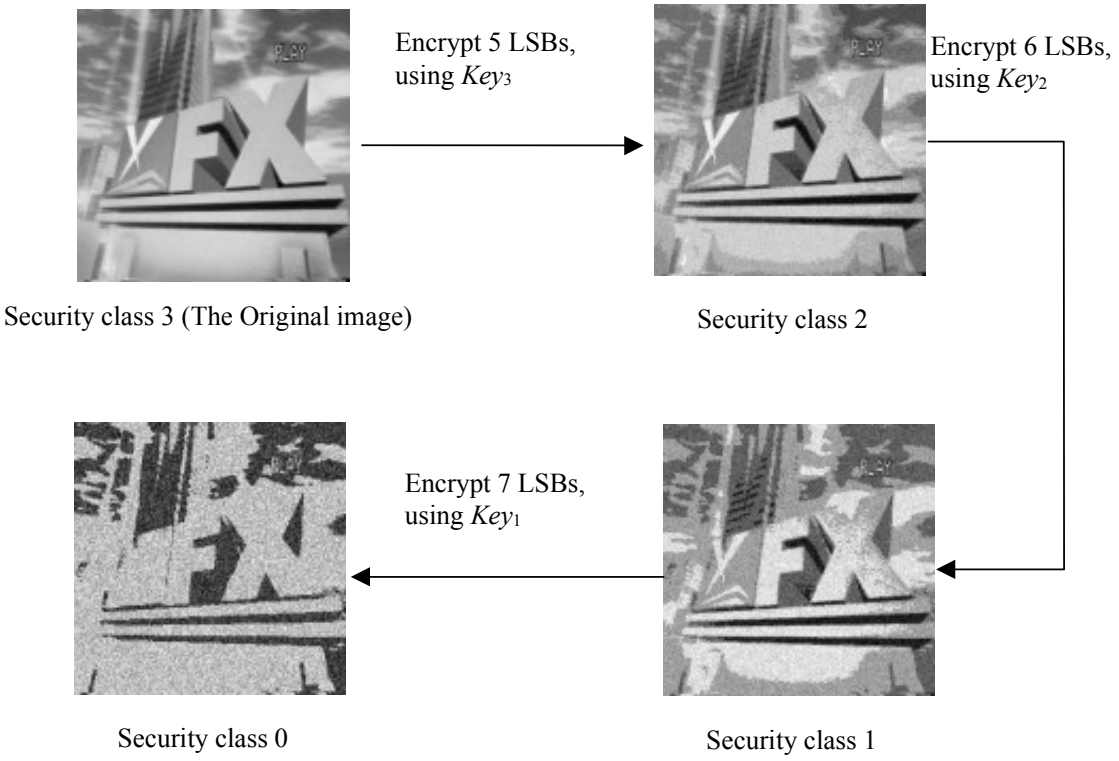


Figure 14 Results after degrading the original image to three security classes.

The Key_3 corresponds to the highest security class. The key generation process involves a non-invertible hash function so that it is possible (and easy) to generate the lower security class keys from higher security class keys to obtain Key_j from Key_i, $i>j$. It is, however, not computationally feasible to generate higher security class keys from lower security keys (e.g., to derive Key_3 from Key_2 or Key_2 from Key_1). Detail description of the key generation is given in Section 3.

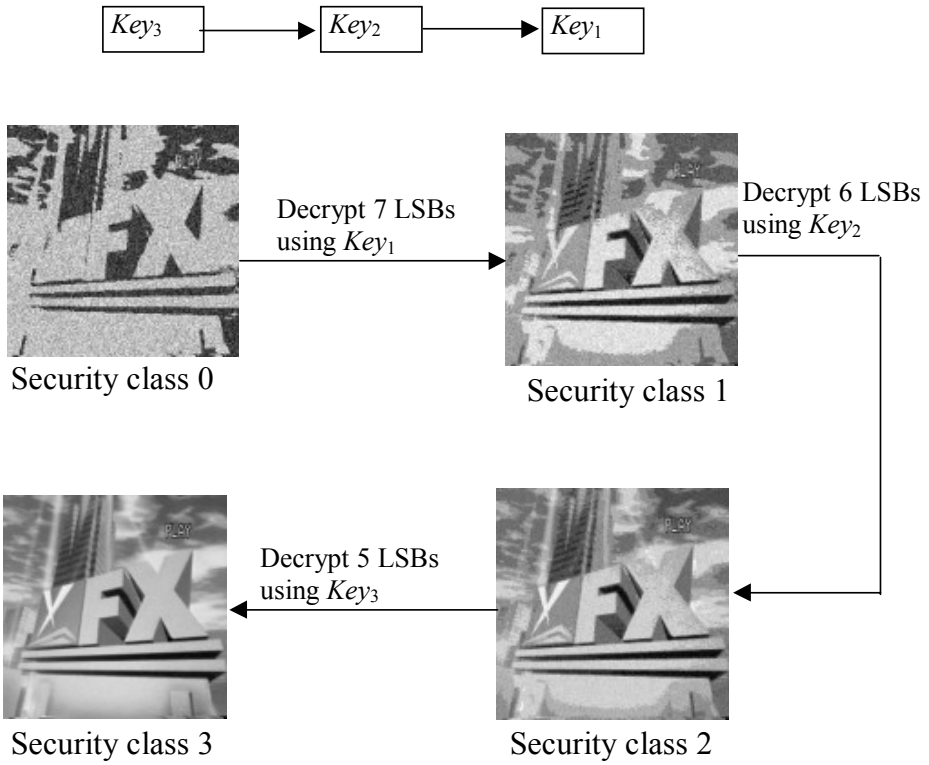


Figure 15 Results after upgrading the degraded image from security class 0.

3.4.1 Description of the encryption technique

The encryption technique used in our work is a slight modification of the chaos-based cipher developed before [1–3]. It consists of several iterative applications of two steps: permutation and diffusion. We opted for this special symmetric encryption scheme because its block size is variable and adjusts automatically to the image size and no padding of the image is needed. The principle of this scheme is briefly described below. The following pseudo-code is used to encrypt a rectangular array of integers g_{ij} in the interval $[0, 2^L - 1]$ (as explained in Section 2.1 we use this scheme for $L = 5$, 6, and 7)

For $k = 1$ to 10
  Permutation $pg_{t1(i,j),t2(i,j)} = (g_{ij} + i \cdot j) \bmod 2^L$
  Diffusion
    odd iteration $g_{ij} = (pg_{ij} + G[pg_{ij-1} \bmod 2, pg_{ij-2} \bmod 2, ..., pg_{ij-8} \bmod 2]) \bmod 2^L$
    even iteration $g_{ij} = (pg_{ij} + G[pg_{ij+1} \bmod 2, pg_{ij+2} \bmod 2, ..., pg_{ij+8} \bmod 2]) \bmod 2^L$
End loop

The array g_{ij} is the input data for encryption and it is also the final output of encrypted data. The function G is a permutation of integers {0, 1, ..., 255} generated from a secret key. In our implementation above, the argument of G is actually represented in a binary

form (8 bits). The arrays $t1$ and $t2$ are called transfer matrices and are derived from a discretized chaotic map using the same secret key. For more details on the encryption scheme, see [1–3].

The decryption proceeds as follows (the array pg_{ij} denotes both the input and the output values):

For $k = 10$ to 1
  Diffusion
    odd iteration $g_{ij} = (pg_{ij} - G[g_{ij-1} \bmod 2, g_{ij-2} \bmod 2, ..., g_{ij-8} \bmod 2]) \bmod 2^L$
    even iteration $g_{ij} = (pg_{ij} - G[g_{ij+1} \bmod 2, g_{ij+2} \bmod 2, ..., g_{ij+8} \bmod 2]) \bmod 2^L$
  Permutation$pg_{ij} = (g_{t1(i,j),t2(i,j)} - i \cdot j) \bmod 2^L$
End loop

Pseudo-code for image downgrading using bit-plane encryption

Input: Original image
Output: Encrypted image

Enter Key_3
From Key_3 generate Key_2, Key_1 // see the algorithm of key generation
Calculate the transfer matrices $t1$, $t2$, and the permutation G
For *security_class* = 1 to 3
For each channel (r, g, b)
Calculate new value of each pixel
Encrypt the image
Display the encrypted image

Pseudo-code for image upgrading using bit-plane encryption

Input: Encrypted image
Output: Higher quality image

Enter *Key*, and *security_class*
If *security_class* = 3
    $Key_3 = Key$
    use Key_3 to generate Key_2, Key_1
else if *security_class* = 2
$Key_2 = Key$
use Key_2 to generate Key_1
else if *security_class* = 1
$Key_1 = Key$
For *security_class* = *security_class* down_to 1
Calculate the transfer matrices $t1$, $t2$, and the permutation G
For each channel (r, g, b)
Get the original value of each pixel

Decrypt the image
Display higher quality image

3.5 Noise adding in the spatial domain

A noise signal obtained as a summation of R individual noise components will be added to the rows in the image. The low-resolution image (the watermarked image) will look noisy. Each noise component will be generated using a pseudo-random number generator (PNRG) with the seed derived from the access key. Given the key Key_i, the seed generation procedure should allow deriving the seeds for all lower levels i–1, …, 1. Therefore, the recipient possessing the key Key_i will be able to remove all noise components up to the level i, however, he will have no information about the remaining noise components that mask the fine details in the image. This method is shown in Figure 16.

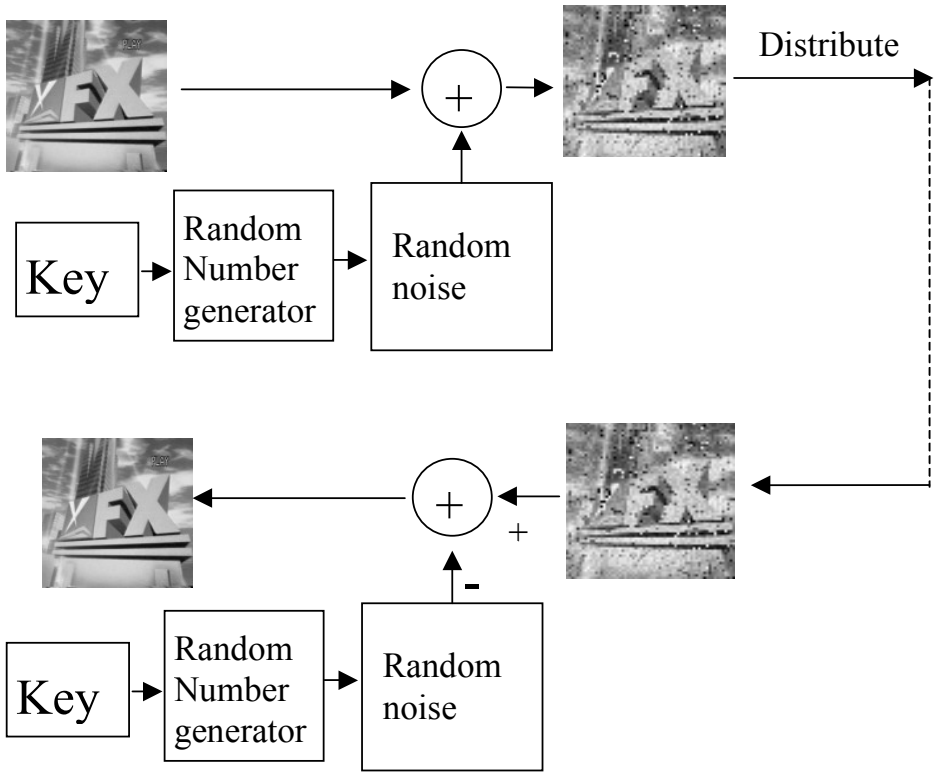


Figure 16 Fidelity degrading method based on noise adding in the spatial domain.

We add random noise to each pixel in the image. If the new pixel value X' is larger than 255 or less than 0, we use the modular addition: $X'' = X' \bmod 256$. Similar to the encryption method, we add three levels of noise thus obtaining three security classes (plus the original image). In the security class 2, the noise is small. In the security class 1, the noise is stronger. And the noise is the strongest in the security class 0 – the public domain. The key generation is the same as in the previous method based on bit-plane encryption. Details can be found in Section 3.

For each security class, the noise added to each image is image dependent, i.e., the seed for generating the noise depends on the image itself. This is very important because if the same noise sequence was added to all images, it would be possible to remove the noise very effectively even using only a small number of images [4].

We divide the image into 32×32 (or in general into $B \times B$) blocks. If the image dimensions are not multiples of 32 (or B), we pad the image with zeros to the next multiple of 32. For each block i, the sum of all gray levels modulo 256 is calculated and denoted s_i. We concatenate the secret key and the string of all values s_i and obtain a new string S. To calculate the seed for the pseudo random number generator, we hash S and obtain a 32-bit seed for the PRNG. The seed is used to generate a zero-mean noise sequence. It is important that the noise has zero mean because the same values s_i can be extracted from the degraded image by the recipients. Figure 17 is the diagram of the image dependent pseudo-random noise generation.

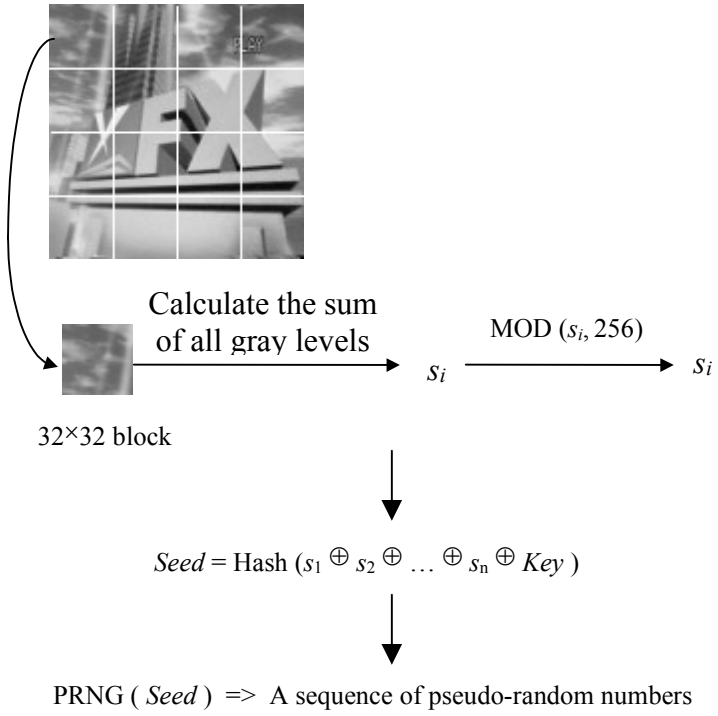


Figure 17 Method for obtaining image dependent noise.

To generate a zero-mean uniformly distributed sequence of integers in the range $[-A, A]$, we first generate a random sequence x_i and then calculate its sum m. If m is not zero, we then randomly select m numbers from the sequence and add 1 (or subtract 1) from them depending on the sign of m (if $m>0$ we subtract 1, if $m<0$ we add one). This simple adjustment will generate a pseudo-random uniformly distributed zero-mean sequence of integers. This adjustment must be done for every block separately and care needs to be taken for boundary blocks that contain some zero padding. Only those portions of the block that correspond to the image should take part in the zero mean calculation.

Pseudo-code for image downgrading using noise adding in the spatial domain

Input: Original image
Output: Noisy image

Enter *Key*3 and the noise strength for all three classes
Use *Key*3 to generate *Key*2, *Key*1 // see the algorithm for key generation
MB = int (*Height*/32)
NB = int (*Width*/32)
Interval = 16777216 // Interval is a fixed parameter for the hash function, see the algorithm for the hash)
For *security_class*=1 to 3
For each channel (r, g, b)
Divide the image into *MB*×*NB* blocks with size of 32×32 pixels
For each block *i,* let s_i = the sum of all gray values in that block modulo 256
$Image\_key = Key_{class} \oplus s_1 \oplus s_2 \oplus \ldots \oplus s_n$ ($n = MB \times NB$)
$Seed_{class}$ = Hash(*Image_key*, *Interval*, *key_length*)
// *key_length* is the length of *Image_key*
For i = 1 to *key_length* / 2
$Image\_key(i) \leftrightarrow Image\_key(key\_length - i)$
$Range_{class}$= Hash(*Image_key*, *Interval*, *key_length*)
If $Seed_{class} > Range_{class}$
$Seed_{class} \leftrightarrow Range_{class}$
Use $Seed_{class}$ and $Range_{class}$ to generate the pseudo random number sequence
Smooth this pseudo random number sequence
Remove the DC component of this pseudo random number sequence
Add the noise to the image
Display the noisy image

Pseudo-code for image upgrading using noise adding in the spatial domain

Input: Noisy image
Output: Higher quality image

Enter *key*, *noise strength* in all three classes, and the *security_class*
If *security_class* = 3
$Key_3 = Key$
use Key_3 to generate Key_2 and Key_1
else if *security_class* = 2
$Key_2 = Key$
use Key_2 to generate Key_1
else if *security_class* = 1
$Key_1 = Key$

MB = int (*Height*/32)
NB = int (*Width*/32)
Interval = 16777216 // *Interval* is a parameter for Hash function, see the algorithm of Hash
For *class* = *security_class* down_to 1
For each channel (r, g, b)
Divide the image into *MB*×*NB* blocks with size of 32×32
For each block *i,* let $s_{i=}$ the sum of all gray classes modulo 256
$Image\_key = Key_{class} \oplus s_1 \oplus s_2 \oplus \ldots \oplus s_n$ ($n = MB \times NB$)
$Seed_{class}$ = Hash(*Image_key*, *Interval*, *key_length*)
// *key_length* is the length of *Image_key*
For i = 1 to *key_length* / 2
$Image\_key(i) \leftrightarrow Image\_key(key\_length - i)$
$Range_{class}$= Hash(*Image_key*, *Interval*, *key_length*)
If $Seed_{class} > Range_{class}$
$Seed_{class} \leftrightarrow Range_{class}$
Use $Seed_{class}$ and $Range_{class}$ to generate the pseudo random number sequence
Smooth this pseudo random number sequence
Remove the DC component of this pseudo random number sequence
Remove the noise from the image
Display the higher fidelity image

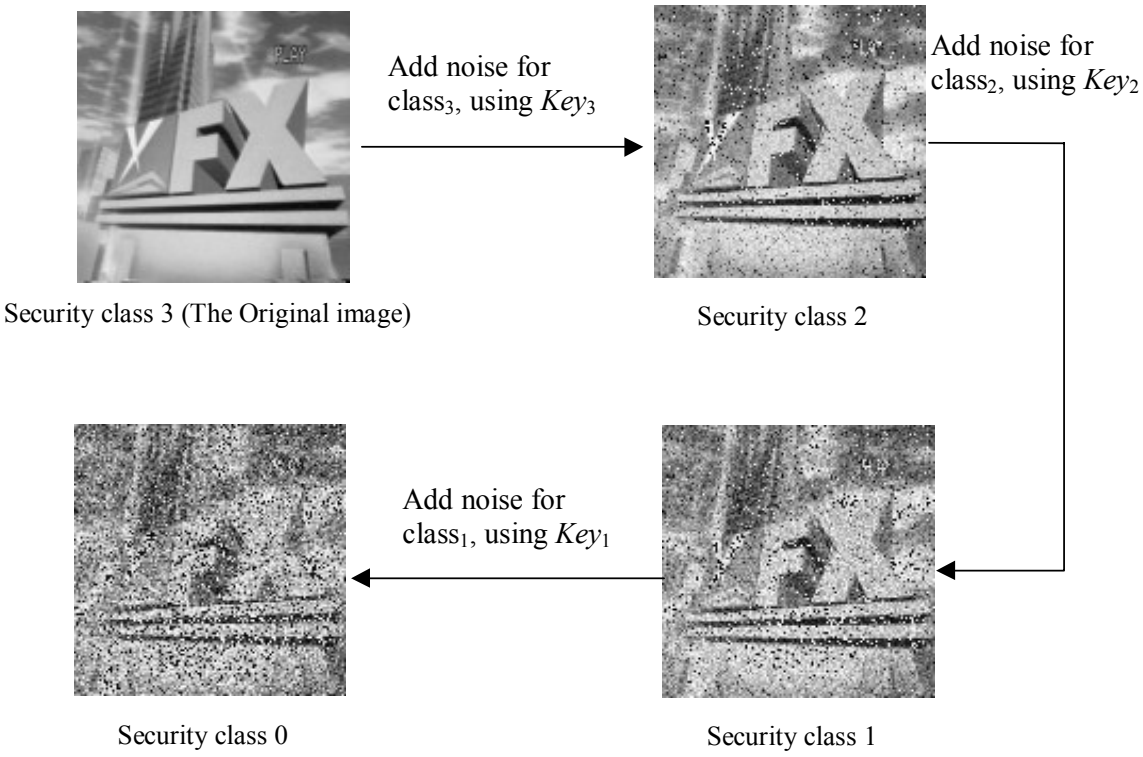


Figure 18 Results after degrading the original image to three security classes.

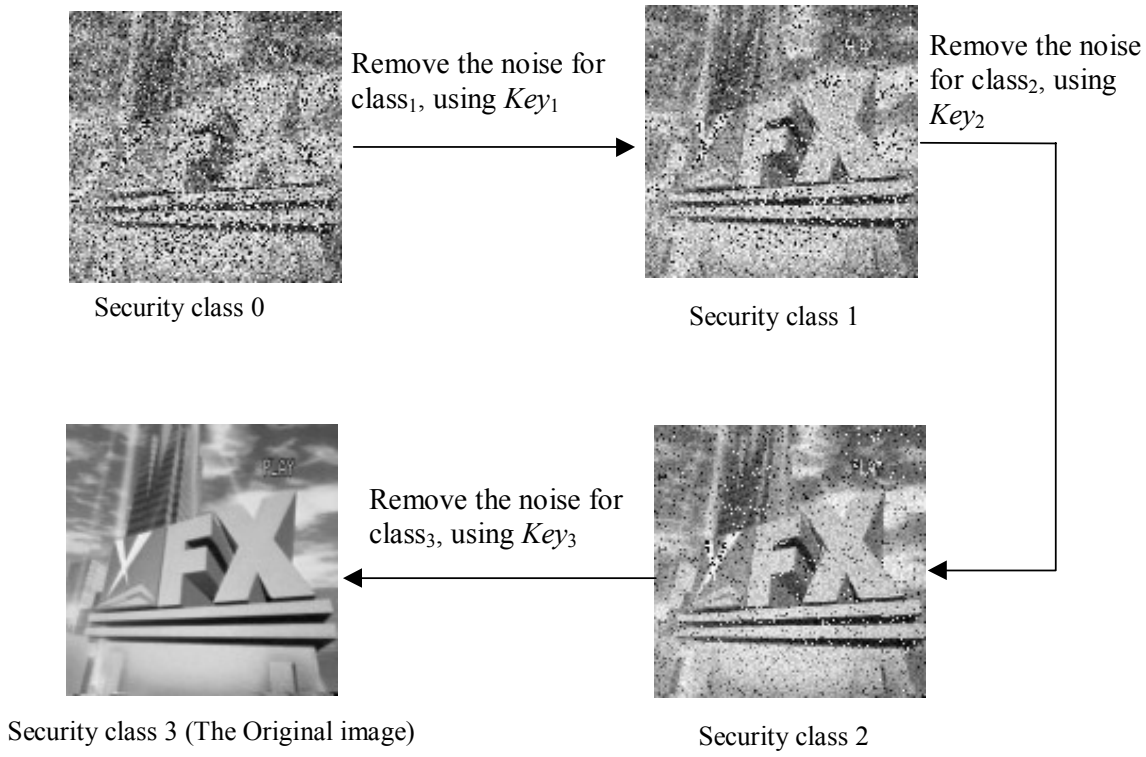


Figure 19 Remove noise from the image in security class 0.

3.6 Key generation

Before we can start with secure image distribution, three access keys need to be generated. The key-generation mechanism should allow fast calculation of all lower-security access keys from higher-security access keys. However, it should not be computationally feasible to infer information about the higher-security access keys from lower-security access keys. Below, we explain how to generate the access keys Key_2 and Key_1 from the highest security class Key_3.

In our implementation, the Key_3 can be up to 20 characters long (it is case sensitive). If the number of characters in Key_3 is less than 20, the algorithm pads it with '0's. Key_3 is hashed to obtain 32 bits (see the hash algorithm in Appendix). These 32 bits are further divided into 4 segments, with 8 bits per each segment. The segments represent four integers I_0, I_1, I_2, I_3 in the range [0, 255]. We continue by calculating 4 digits, $c_0 = I_0 \bmod 10$, $c_1 = I_1 \bmod 10$, $c_2 = I_2 \bmod 10$, and $c_3 = I_3 \bmod 10$. We set the first 4 digits of Key_2 to be c_0, c_1, c_2, and c_3. To obtain the rest of the numbers for Key_2, we replace the first 4 digits of Key_3 with c_0, c_1, c_2, and c_3 thus obtaining a new key Key_3'. The Key_3' is hashed and in a similar process as before and four new digits are obtained. Those will be the next 4 digits of Key_2. This process is repeated to obtain the remaining 12 digits for Key_2. The method is shown schematically in Figure 20. The same method is used to derive Key_1 from Key_2.

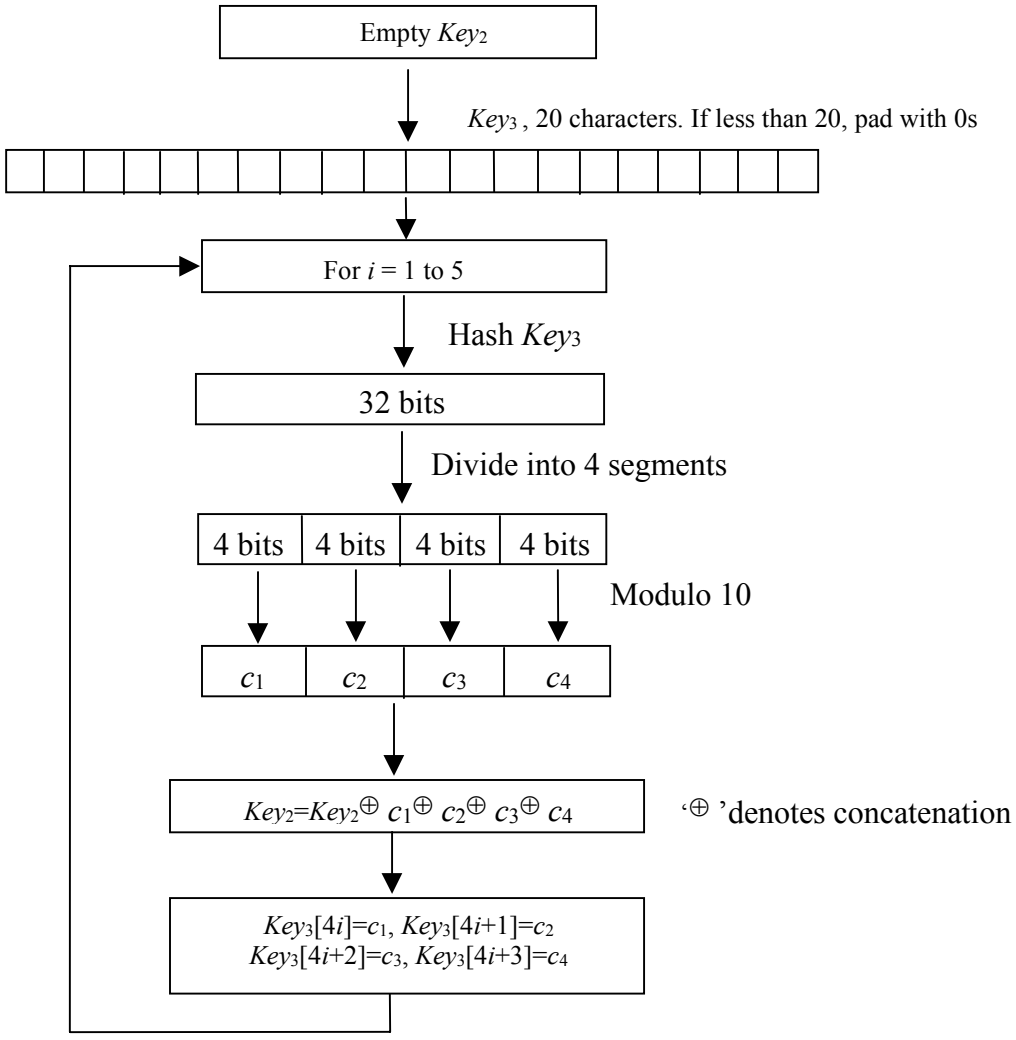


Figure 20 Flow chart for the key generation.

3.7 Security analysis

The security of the encryption scheme depends on the security of the encryption algorithm. Security study of the algorithm has been performed in [1,3]. Should a weakness in this scheme be discovered in the future, other symmetric block encryption private-key algorithms, such as IDEA, Blowfish, or TEA could be used to replace the chaos-based encryption. The chaos cipher is a natural fit in our case because its block size matches the image dimensions exactly and there is no need to pad the image to larger dimensions.

One can attempt to increase the fidelity of the degraded image by applying classical denoising filters, such as the adaptive Wiener filter [6] in an attempt to increase the image fidelity. However, since the five (or more) LSB planes are randomized after encryption, denoising filters will be of little use because the encrypted bit-planes form a special case

of a fixed-amplitude non-additive noise. The encrypted information is essentially lost and cannot be recovered.

The degrading based on encryption provides a pleasing degradation while preserving the image content and removing the image details. The number of security classes is essentially limited by the number of bits allocated per pixel. This is a limitation that may not be acceptable in some applications requiring more refined security classes. Also, this method does not have a set of parameters that could be used to adjust the degradation.

The method based on adding the noise in the spatial domain is probably less secure than the encryption-based technique. The noise is additive and has known statistical properties. This means that application of adaptive denoising filters, such as the Wiener filter, is more meaningful than in the bit-plane encryption technique. However, since the amount of noise in our application is typically much larger than in typical noisy images, we do not expect the denoising filters to dramatically improve the image fidelity. Even after the filters are applied, the denoised image will have some artifacts typical for denoising filters, such as "artificial segmentation" and a little blurring. It is thus not possible to obtain the original image fidelity. Key-dependent noise distribution could be used to lessen the problem with denoising. Smoothened noise sequences could be used to change the visual quality of the degraded imagery. Low-pass filtered zero-mean noise sequences would lead to a qualitatively different degradation in images.

3.8 Usage of Secure Image Distribution in SecureStego

The secure image distribution via invertible fidelity degradation has been implemented in the software application called SecureStego. It includes other capabilities such as steganography, steganalysis, image authentication, watermarking, and some unique special image tools. Figure 21 shows the interface of the SecureStego software. Secure image distribution is represented as **Fidelity** option in the **Authentication** menu.

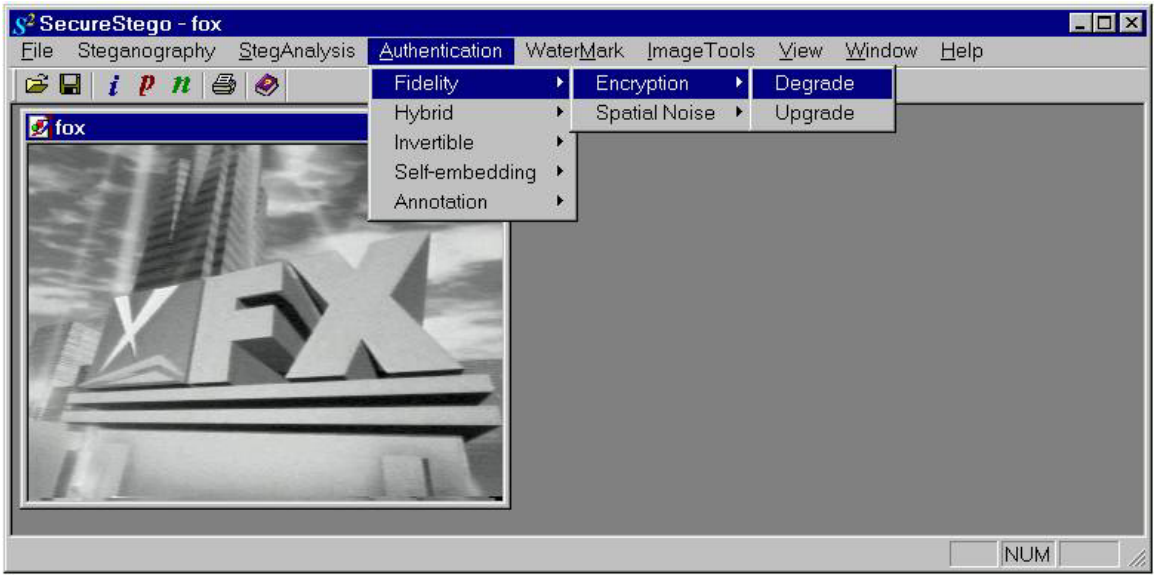


Figure 21 User interface of the SecureStego software.

3.8.1 Method based on encrypting bit-planes

In degrading of encryption method, go to the menu **Authentication : Fidelity : Encryption : Degrade** (Figure 22), a dialog window will be displayed (Figure 23).

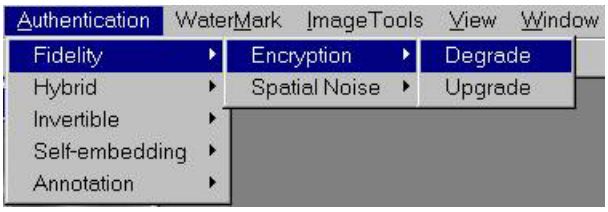


Figure 22 Image degrading menu for the encryption method.

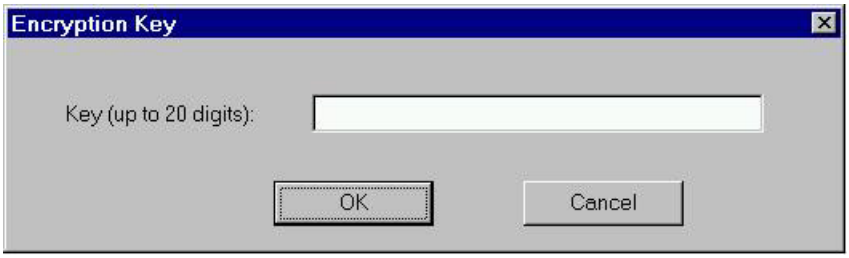


Figure 23 Encryption Key window.

In the dialog box, enter a key (up to 20 digits), and click OK. The software begins the degrading process. When the degrading is finished, three lower quality images are shown, and a message box informs the user that all three secret access keys have been stored in a file named 'encrypt.key' in the directory where the executable file for SecureStego resides. The degraded images and message box are shown in Figure 24. Figure 25 shows the three access keys.

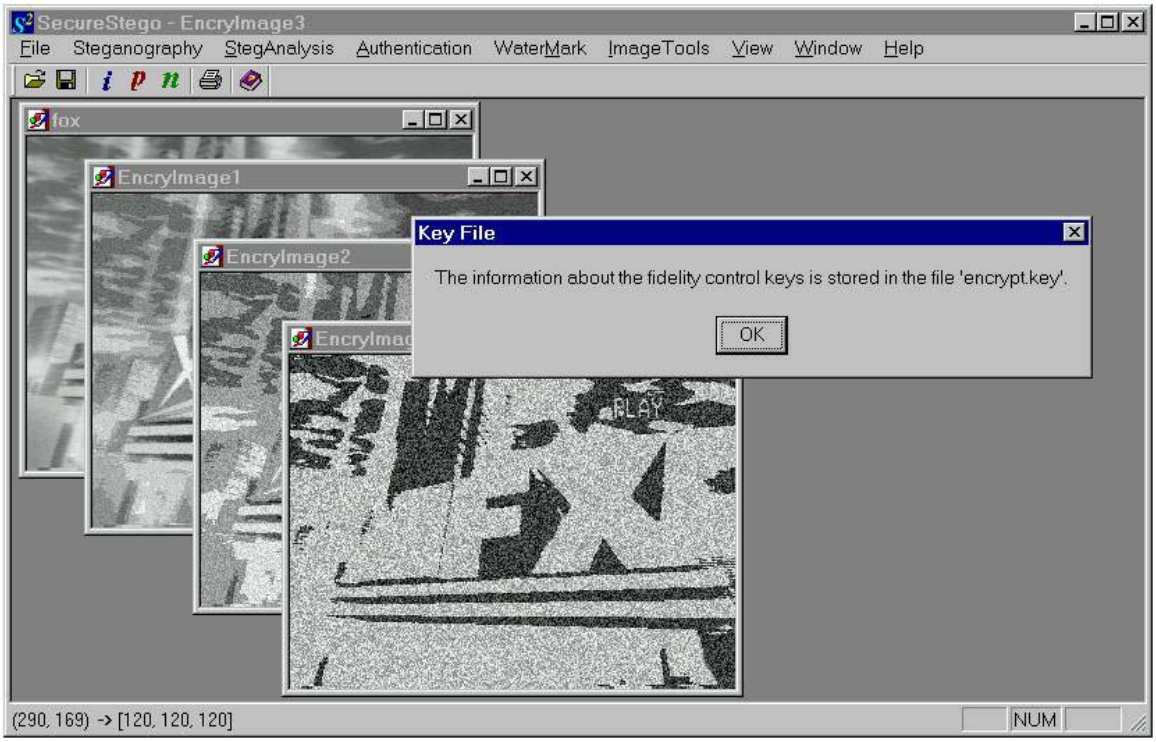


Figure 24 Degraded images using the encryption method.

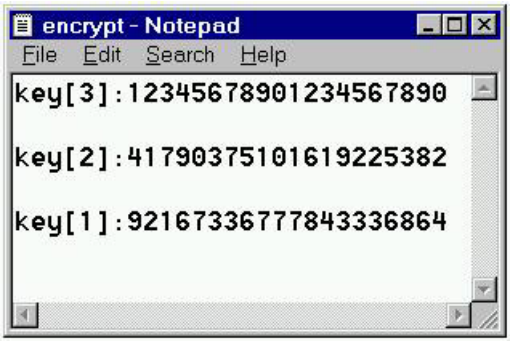


Figure 25 Secret access keys.

The lowest fidelity image is the one that corresponds to the lowest public security class and the one that will be distributed in the plain. We stress that the downgraded image must be stored in a *lossless* image format, such as the BMP format, otherwise the high-fidelity image cannot be accessed by the users. Currently developed image downgrading methods are not compatible with lossy image compression. To upgrade the image quality using the encryption method, go to the menu **Authentication : Fidelity : Encryption : Upgrade** (Figure 26). A dialog window shown in Figure 27 will be displayed.

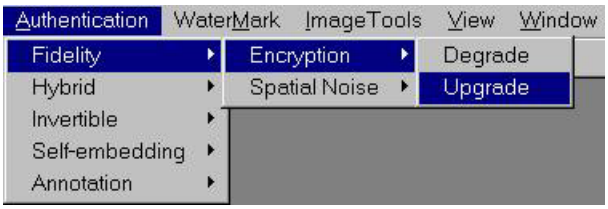


Figure 26 Image upgrading menu for the encryption method.

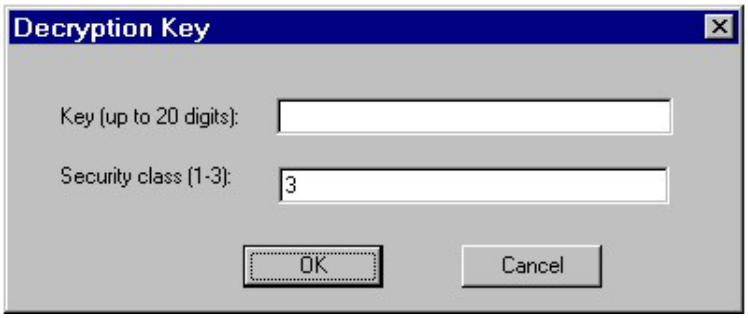


Figure 27 Upgrading key window for decryption.

In the dialog window, enter the key and the security class, and click OK. The software begins the upgrading process. After the process, the high quality image corresponding to the specified security class is shown as a result (Figure 28). We note that the user must enter *both* the right security class and the correct key. Entering the correct key and a non-matching security class will not lead to a higher fidelity image.

Figure 28 Upgraded image using the encryption method.

3.8.2 Method Based on Adding Spatial Noise

To use this method for image fidelity downgrading, go to the menu **Authentication : Fidelity : Spatial Noise : Degrade** (Figure 29). A dialog window will be displayed (Figure 30).

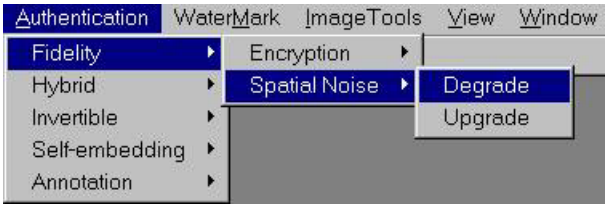


Figure 29 Image degrading menu for the spatial noise method.

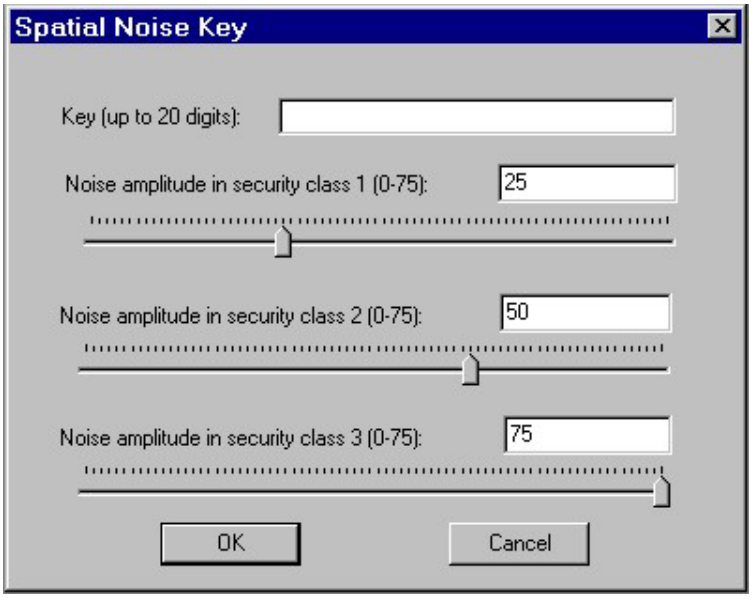


Figure 30 Spatial Noise Key window.

In the dialog window, enter a key (up to 20 digits), and enter the noise amplitudes in the dialog boxes, or use the slider bars to change the noise amplitudes, and then click OK. The software begins the degrading process. After the process, three low quality images are displayed, and a message box again indicates the name of the file which stores all three secret access keys. Similar to the encryption method, the key file is stored in the directory where the SecureStego executable resides. The degraded images and the message box are shown in Figure 31. Figure 32 shows the secret access keys.

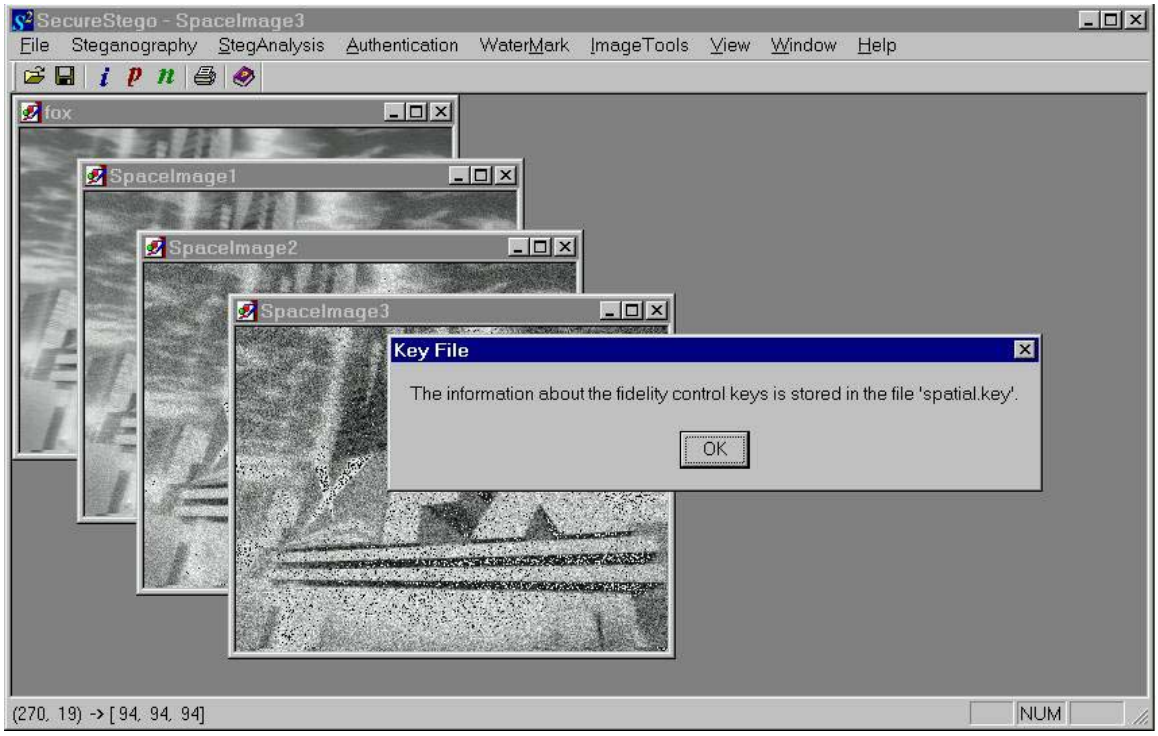


Figure 31 Degraded images using the spatial noise method.

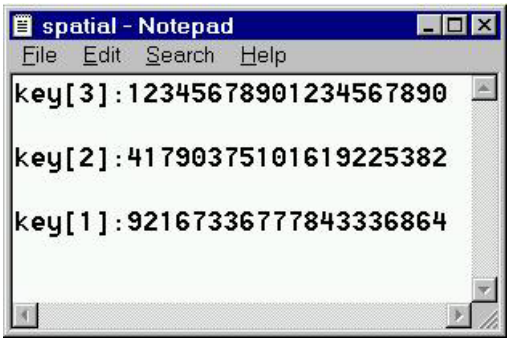


Figure 32 Spatial noise keys.

Similar to the encryption case, we stress that the downgraded image must be stored in a *lossless* image format, such as the BMP format, otherwise the high-fidelity image cannot be accessed by the users. Currently developed image downgrading methods are not compatible with lossy image compression. To obtain the higher fidelity image from the degraded public image, go to the menu **Authentication : Fidelity : Spatial Noise : Upgrade** (Figure 33). A dialog window will be displayed (Figure 34).

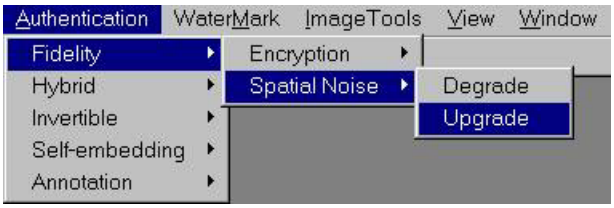


Figure 33 Image upgrading menu for the spatial noise method.

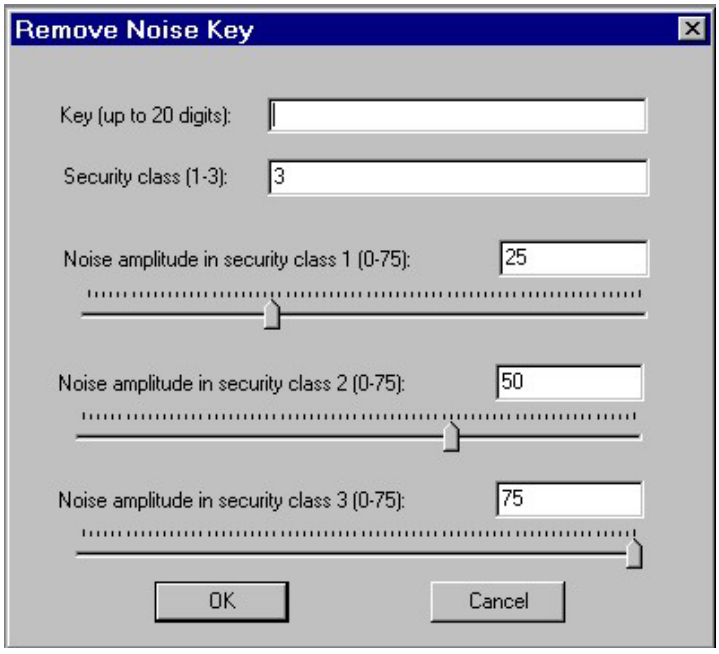


Figure 34 Upgrading key window for the spatial noise method.

In the dialog window, enter the key, the corresponding security class, and all three noise amplitudes, and then click OK. The software begins the upgrading process. At the end, the high quality image corresponding to the entered security class is shown (Figure 35). We note that all entries in the dialog window must be correct. Incorrect security level or noise amplitude will not lead to higher fidelity images.

Figure 35 Upgraded image using the spatial noise method.

3.9 Future research

In this section, we outline some ideas that we believe are worth pursuing in the future as a follow-up research.

3.9.1 Noise Adding in the Transform Domain

Instead of adding noise directly to the image in the spatial domain, we can add noise to the coefficients of some image transform, such as the DCT transform. By adding the noise signal to different bands of the DCT matrix, we can modify the spectral properties of the noise. If we add noise to the top-left corner of the DCT matrix that contains low-frequency DCT modes, we obtain a low frequency noise. If we add noise to the bottom-right corner, a high frequency noise is obtained as a result. The method is schematically shown in Figure 36.

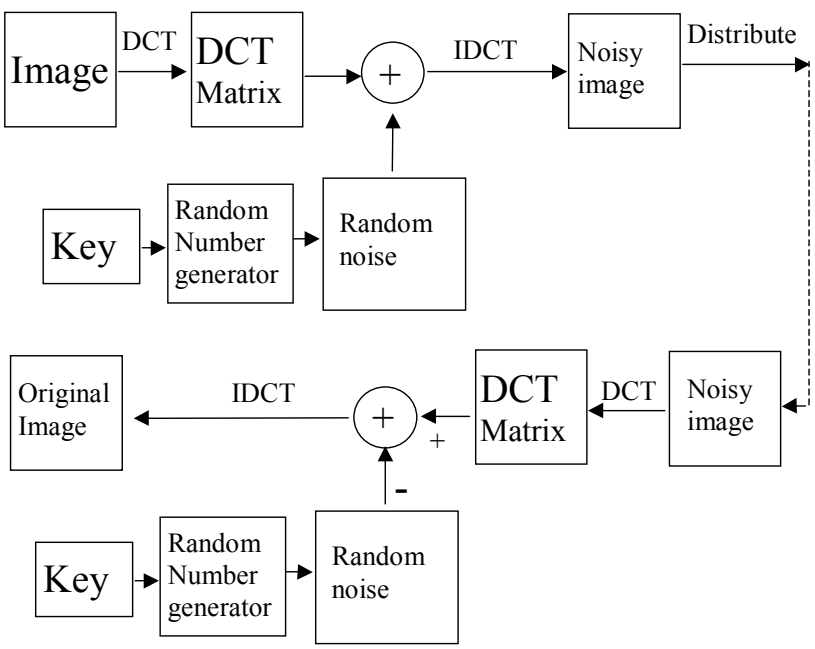


Figure 36 Method based on noise adding in the transform domain.

Figure 37 Original image.

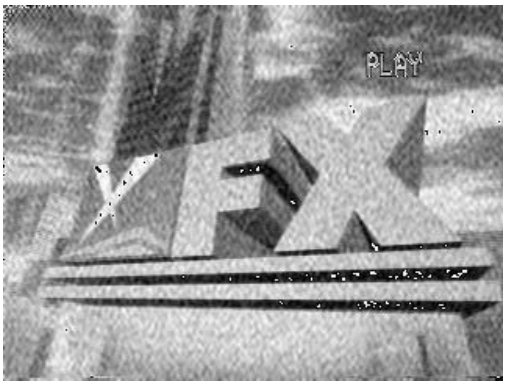


Figure 38 Image with noise in the DCT domain.

The advantage of this method is that the visual appearance of the degraded image can be easily adjusted by changing the spectral properties of the noise. As in the spatial noise case, the noise must be strongly image dependent. We note that spectral shaping of the spatial noise is possible for the spatial noise method as well. A low-pass kernels or band-pass filters could thus be used to obtain more desirable spatial noise properties.

3.9.2 Key-dependent Convolution

In this method, we use randomized convolution to add degradation to the image. Using a secret key, we generate a random convolution kernel that is used to process the image using the convolution operation. Care needs to be taken so that the convolution operation is invertible. We have found out that the following general class of convolution kernels provides invertible degradation, at least in theory when no discretization and rounding at 0 and 255 occurs.

Let us suppose that I is the original image and K is the random kernel (generated from a secret key). The degraded image I_w is obtained using the formula:

$$I_w = \alpha I + \beta I * K = \alpha\delta * I + \beta I * K = (\alpha\delta + \beta K) * I,$$

where '*' denotes convolution, α and β are positive constants, and δ is the unit matrix.

The previous equation could be rewritten using a Fourier Transform:

$$F(I_w) = F(\alpha\delta + \beta K) \; F(I),$$

where F denotes the Fourier transform. Thus

$$I = F^{-1}(F(I_w) / F(\alpha\delta + \beta K)),$$

where F^{-1} denotes the inverse Fourier Transform. This means that the process of convolving an image with the kernel $(\alpha\delta + \beta K)$ is indeed invertible. Since we need to store the degraded image using integers in the range [0, 255], rounding errors and errors due to truncation at 0 and 255 are likely to introduce a certain amount of noise that would be difficult to control. Additional research is necessary to clarify this issue.

Figure 39 Image degraded by convolving the original with a randomized kernel.

3.9.3 Fidelity Control Via Resolution Modification

In some applications, it may be actually desirable to not reveal the original resolution of the image and achieve distribution control by simply changing the resolution of the image. This method, however, can never be done in a completely invertible manner because the information theoretic bounds for images simply do not permit to compress imagery to a significantly smaller resolution form in a lossless manner. As an initial idea, we propose to downsample the image first. This can be done, for example, by blocks and selecting only one pixel from every 2×2 block to form a smaller image. At the same time, we compress the original image (losslessly). Then, we embed the compressed image as noise to the small image using steganographic techniques to get a small noisy image. Given the key, we can retrieve the compressed image from the small noisy image. Since the smaller image is noisier than the original, it has higher entropy. Thus, it may hold enough information to reconstruct the original image with a better quality than what is provided by the lossy compression. This heuristic argument can be used to justify the correctness of our proposed mechanism.

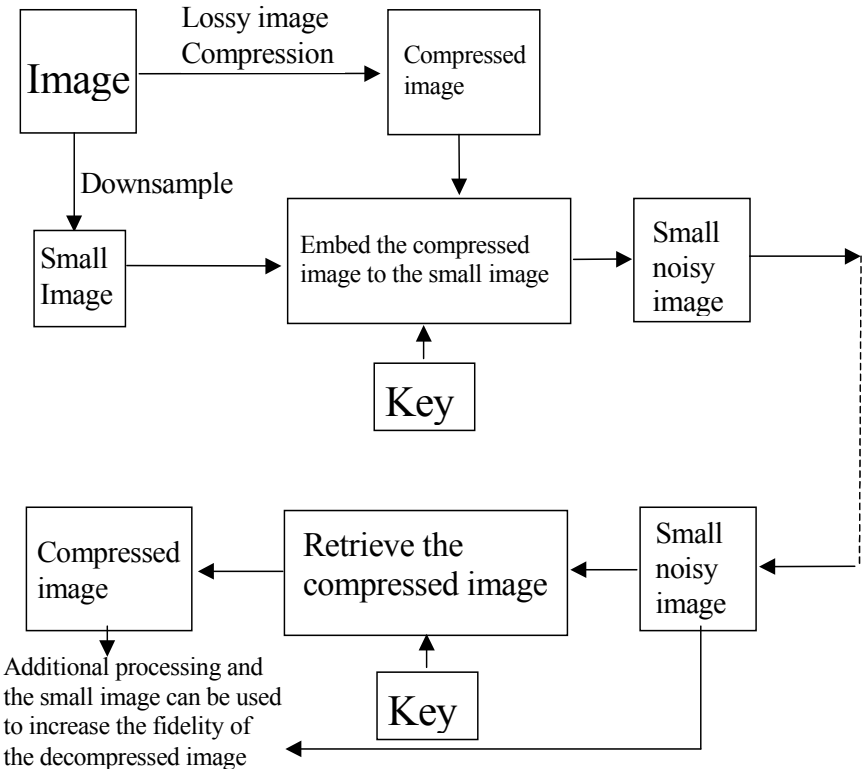


Figure 40 Method based on resolution control.

Figure 41 shows the small noisy image with the embedded compressed form of the original image shown in Figure 37. Figure 42 shows the retrieved image. The retrieved image could be further processed using the information about the downsampled image to arrive at an image with quality higher than the lossy compressed image. In either case, the quality of the retrieved image is better than a simple upsampling of the small image.

Figure 41 Small noisy image.

Figure 42 Retrieved image.

3.10 Conclusion

In this research, we present a method for achieving secure distribution of images to users from different security classes via invertible key-dependent degradation. A high-quality image is degraded to a low quality image without losing its essence. By this we mean that the main objects in the image are still recognizable but the details are scrambled and not easily accessible without a special secret access key. The low quality images can be sent to recipients from different security classes through one insecure public channel. The recipients in different security classes have different access keys, which can be used to upgrade the quality of the images. In our work, we provide three different lower-quality security classes plus the original highest quality class. We presented two methods: image degrading based on bit-plane encryption and on adding spatial noise.

In the encryption method, we encrypt the 5, 6, and 7 LSB planes of each pixel to blur the details in the image. In the spatial noise adding method, image details are also scrambled but more visible and disturbing artifacts are created due to the use of invertible addition modulo 256. Both processes do not blur the edges, which are important for recognizing the image features.

The security of the image distribution based on invertible degradation has been discussed. The encryption method is more transparent and provides better security. Its security is derived from the security of the encryption method. In our implementation we use chaos-based encryption because this ciphering algorithm requires no padding (i.e., enlarging) of the image. The encryption method better preserves the content of the image. However, it does not have parameters that could be adjusted to modify or adjust the image fidelity. Also, the number of security classes is limited by the number of bits allocated per pixel.

The method based on adding spatial noise provides more flexibility because the noise amplitudes can be adjusted by the user to achieve the most appropriate degradation[1]. However, the security of the spatial noise adding method is lower because of the possibility to use denoising filters, such as the adaptive Wiener filter. We designed a special procedure for noise generation so that the added noise depends strongly on the whole image. Thus, it is not possible to filter out the noise by collecting many images and extracting the noise component. We note that the image dependency of the noise on the image is absolutely necessary because the noise present in the degraded images is quite strong and not too many degraded images would be necessary to extract a very good approximation to the noise [4]. Since potentially a large number of images are expected to be sent to different recipients, if an image-independent noise were used, an attacker would have enough information to break the scheme and access the higher fidelity imagery.

The image degrading and upgrading methods were added to a user-friendly software application called SecureStego developed during our previous contracts. Pull-down menu items can be conveniently used to display dialog windows to enter information about the secret key, the security class, and the noise amplitudes (in the spatial noise adding method). The degraded images must be saved in a lossless format otherwise the high-

[1] An image analyst may wish to adjust the noise levels to obtain specific NIIR ratings.

fidelity images cannot be accessed. The generated access keys are stored in a key file in the directory where the executable application resides.

Appendix

The hash function

Hash a variable-length key into a 32-bit value

Inputs:

k : the key (the unaligned variable-length array of bytes)

len : the length of the key, counting by bytes

interval : can be any 4-byte value

Output: A 32-bit value.

Every bit of the key affects every bit of the return value. More details about the hash function and the source code can be found at http://burtleburtle.net/bob/hash/perfect.html.

3.11 References

1. J. Fridrich, "Symmetric Ciphers Based on Two-Dimensional Chaotic maps", *Int. J. Bifurcation and Chaos*, **8**(6), June 1998, pp. 1259–1284.
2. J. Fridrich, "Image Encryption Based on Chaotic Maps", *Proc. IEEE Conf. on Systems, Man, and Cybernetics*, October 12–15, 1997, pp. 1105–1110.
3. J. Fridrich, "Secure Image Ciphering Based on Chaos". Final report for Grant #F30602-96-1-0047 for AFRL, NY, March 10, 1997.
4. M. Holliman, N. Memon, and M. M. Yeung, "On the Need for Image Dependent Keys for Watermarking"*, Proc. Content Security and Data Hiding in Digital Media*, Newark, NJ, May 14, 1999.
5. B. Jenkins, http://burtleburtle.net/bob/hash/perfect.html.
6. A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1989.