# Reactive Tamper Detection for Image Authentication

Tao Xie, Andrew Sung
Computer Science Department
New Mexico Institute of Mining and Technology
801 Leroy Place, Socorro
U.S.A.
*{xietao,sung}@cs.nmt.edu*

Srinivas Mukkamala,Qingzhong Liu
Computer Science Department
New Mexico Institute of Mining and Technology
801 Leroy Place, Socorro
U.S.A.
*{srinivas,liu}@cs.nmt.edu*

*Abstract* – **In this paper we present a novel image authentication scheme by embedding a reactive content-based cryptographic signature into image compression-domain. We demonstrate that our technique can detect even one compression-domain element's interpolation so that an adversary cannot tamper a watermarked image without being discovered.**

## I. INTRODUCTION

The expeditious growth of Internet distributions of digital information media has created a demanding need for copyright protection because of easy replication and modification. Image authentication is one of the effective ways to protect the rights of image owners. Generally we can divide image authentication techniques into two broad categories. One is fragile watermarking and the other is digital signature system.

A fragile watermark is a mark that is easily changed or ruined when the host image is modified through a linear or nonlinear transformation [1]. Fragile watermarks are not suitable for enforcing copyright ownership of digital images; an attacker would attempt to destroy the embedded mark and fragile watermarks are, by definition, readily destroyed. The sensitivity of fragile watermarks to modification leads to their application in image authentication. That is, it may be of interest for people to verify that an image has not been interpolated, damaged, or touched since it was watermarked. Image authentication systems have capacity in law, commerce, defense, and journalism [2]. Because digital images are easy to modify, a secure authentication system is very useful in demonstrating that no tampering has happened during scenarios where the credibility of an image may be doubted. Usual instances are the marking of images in a database to detect tempering [3][4], the use in a "trustworthy camera" so news agencies can make sure that an image is not fabricated or edited to falsify events [5], and the marking of images in business so a customer can be assured that the images bought are authentic upon receipt [6]. Other scenarios include images used in courtroom evidence, journalistic photography, or images involved in espionage. Another means to verify the authenticity of a digital image is to apply a signature system [7]. In a digital signature system, a digest of the data to be authenticated is obtained by applying cryptographic hash functions [7][8]. The digest is then cryptographically signed to create the signature that is tied to the original data. Subsequently, a recipient verifies the signature by examining the digest of the data and using a verification algorithm determines if the data is authentic. Although the purpose of fragile watermarking and digital signature systems are similar, watermarking systems offer some advantages compared to signature systems [9] at the expense of requiring some modification of the image data. Since a watermark is embedded directly in the host image, no additional information is needed for authenticity verification purpose. On the other hand, in digital signatures systems, signature itself should be bound to the transmitted data. Consequently, in watermarking system, the critical information needed in the authenticity process is discreetly hidden and much harder to remove than a digital signature. In addition, digital signature systems take an image as an arbitrary and meaningless bit stream and do not exploit its unique structure at all. Therefore a signature system may be able to detect that an image had been tampered but cannot describe the alterations [2].

Our method belongs to the first category. Although fragile watermarking has some advantages compared to digital signature system, we can encounter some problems when using fragile watermark schemes. For example, one cannot guarantee that no one has deliberately tampered the image because a third-party can forge a fragile watermark if the embedded pattern is irrelative to the host image. To address this problem, we embed content-based cryptographic signature into the compression-domain of an image. This way even one compression-domain element has been interpolated by a third-party, our tool can detect it for sure. The rest of the paper is organized as follows. In Section 2, we briefly introduce some previous work and point out their drawbacks. In order to fix these problems, we propose our new technique and give a detailed description in Section 3. In Section 4 we present experiments results to show the effectiveness of our method. Finally we conclude our work in Section 5.

## II. RELATED WORK

Former fragile watermarking systems have some undesirable features. They embedded watermarks directly in the pixel-domain which cannot survive after image compression. As we all know, image compression is required by image storage and transmission. In addition, the content they embedded is irrelative to the original image which means the watermark itself could not reflect any feature of the original image. In other words, watermark and image are not coupled closely. For example, S. Walton embedded checksums [10] in the least significant bit (LSB). R. Wolfgang *et al* insert pseudo-random sequences [8,11] in the LSB plane of an image. More recent fragile watermarking systems apply more advanced embedding mechanisms [1,12,13], including the use of cryptographic hash functions [14] to detect changes to a watermarked image. In addition, except for the DCT-domain, wavelet-domain is also used to embed the

watermarks [15,16]. Among these techniques, L.M. Marvel *et al* presented a representative work [17] which embedded a content-based hash value into the compression-domain of an image.

They proposed a compression-compatible fragile tamper detection method which can detect minor changes. The central idea is that a hash value which is computed from an image's DCT-coefficients is embedded into some selected 8×8 DCT coefficient blocks by one bit per block manner. If a malicious party altered the image thus affecting the corresponding 8×8 blocks of DCT coefficients where the hash is hidden, the extracted hash and newly computed hash would not match since approximately one half of the hash bits will be changed for slight changes in input bits[10]. We adopted a similar methodology. However, our approach greatly improves the sensitivity of the embedded watermark while overcomes several shortcomings of their method. First, in their paper, they claimed that their schema can detect even "minor changes" of a watermarked image. However, they didn't provide the definition of "minor changes". Therefore, people cannot clearly know what changes are minor and thus cannot tell how fragile or sensitive their watermark is. In our paper, we define minor change as one compression-domain element's (e.g., DCT coefficient) interpolation. In other words, even one DCT coefficient is modified, our approach can certainly discover it. Second, they selected 128 8×8 DCT coefficient blocks to embed a 128-bit signature. Only one coefficient per selected block is used to hide one bit of the signature. All the other DCT-coefficients in these 128 selected blocks were not send to the hash function as a part of the input. Therefore, these DCT coefficients were not protected by the watermark. This is a big issue since it could allow a third-party to interpolate any of these unprotected DCT-coefficients without being detected. In fact, they left these 128 selected blocks vulnerable to tampering. We fix this problem by hashing all DCT-coefficients except the 128 selected ones which are used to embed the digest. Thus we send these DCT coefficients to the hash function as part of the input. In this scenario, if an adversary tampers even one DCT-coefficient no matter it is one of 128 embeddable ones or not, our method can detect the tamper for sure. Third, we use different strategy to select candidate blocks for embedding. They used all 8×8 DCT-domain blocks of an image as candidate blocks from which 128 blocks are selected to insert watermark. In our views, not all of the 8×8 DCT coefficient blocks are suitable for embedding due to the characteristics of DCT blocks. We only select some 8×8 DCT coefficient blocks which have large value of AC coefficients. In this way minor changes (increasing one or decreasing one) of these AC coefficients cannot bring us obvious image distortion. Lastly, we use an easy method to search for these embeddable AC coefficients so that much computation can be skipped comparing with their method. In short, our approach is more sensitive to modifications than their method while keeping watermarked image in a good quality. This is the reason why we call our method "reactive" tamper detection, meaning one minor change on a watermarked image can cause a big alteration on embedded watermark.

## III. REACTIVE TAMPER DETECTION

### *A. Central Idea*

Before we give detailed information about signature embedding and signature extraction, we would like to briefly describe the overall idea about our method. First of all, our approach is implemented in the context of JPEG gray scale images since well-known image compression standards such as JPEG offer us many opportunities in the encoder that can be exploited to hide secrete information. Meanwhile we need to consider balances between the signature quality and complexity. Our idea is to find all 8×8 DCT blocks which are suitable for embedding. From these candidate blocks, we randomly select H blocks using a pseudo random seed to embed the signature (H is the length of the signature). Within each selected block we pick one AC coefficient to embed one bit of the signature. All the other DCT coefficients except the H embedded ones will be hashed by a hash function. The digest of the hash function is used as signature and will be embedded in the H AC coefficients one bit per AC coefficient. This way we logically divide all DCT coefficients of an image into two groups. One is H AC coefficients used to embed the signature and all the other coefficients consist of another group used to produce the signature. Thus all DCT coefficients are protected which means if a third-party tamper any DCT coefficient in any of the two groups or both our approach will definitely detect it. When a recipient receives a watermarked image he can extract the embedded signature and compare it with the computer signature using the same computation that the sender used. If there is discrepancy between the extracted signature and the computer signature, he knows that the image has been tampered since it was watermarked.

### *B. Signature Embedding Scheme*

For a M×N gray scale JPEG image, we apply DCT and quantization transform on each 8×8 pixel block and thus obtain the corresponding DCT coefficient blocks. Since blocks which is composed of very small AC coefficients represent smooth part of an image hence not suitable for embedding data, only embeddable blocks (e.g., the blocks which have AC coefficients larger than a threshold value) are used as candidate blocks for data embedding. Next, we randomly selected a fixed-number blocks from these candidate blocks to embed the fixed-length signature. To authenticate as more coefficients as possible meanwhile minimizing image distortion, we only insert one bit of the signature in one particular AC coefficient in a selected block. All but the selected coefficients are sent to a cryptographic hash function as an input to compute the hash vale. The computed digest is taken as a secrete signature and embedded in the selected AC coefficients. For simplicity, we use the MD-5 hashing function to produce a secrete signature. So the signature will be 128 bits long no matter how long the input message is. The basic idea of the embedding process is illustrated as Figure.1.

To avoid blocky effect and high frequency distortion, DC coefficients and very small AC coefficients are not used for embedding. In a generic image such as lena.jpg, we found that on average there are only about 15%~20% AC coefficients (larger than a threshold value) which

scatter in about 81% blocks are embeddable. Thus, we only randomly select DCT blocks which are embeddable.

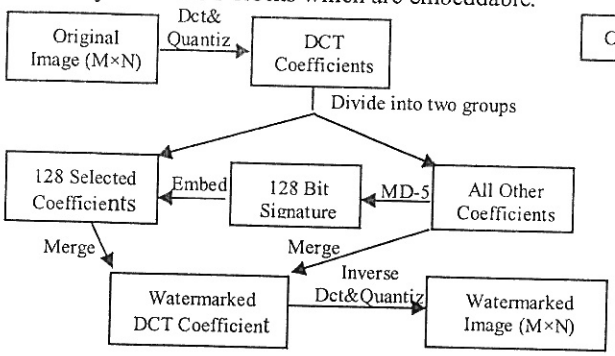


Fig. 1. Embedding Scheme

To skip heavy computation while alleviate image degradation, we search for the first AC coefficient which is not equal to the threshold in a selected DCT block in a zigzag direction. This AC coefficient will be used to embed one bit of the signature. Suppose $AC_i$ is an AC coefficient in the ith selected block which is used to embed one bit of the signature. Also, let's assume that bit "1" means odd and bit "0" means even. Now assume that we want to embed one bit of "1" onto $AC_i$. If $AC_i$ happens to be an odd number, we don't need to change it since its parity tells a recipient that the signature bit it carries is a "1" when he extracts the signature. If $AC_i$ is an even number, we need to adjust it by increasing by one or decreasing by one depending on whether $AC_i$ is smaller or larger than the threshold value. The basic idea of embedding one bit of the signature into one AC coefficient is also illustrated by the examples in Figure.2 and Figure.3. Suppose in example in Figure.2, we need to embed one bit of '1' into the first AC coefficient which is not equal to the threshold value 2. Clearly, this embeddable AC coefficient is 3 if we apply a zigzag search direction. Coincidentally, we need to do nothing in this case since 3 itself meaning an odd number embedded. In Figure.3 if we want to insert a "1" onto the AC coefficient 4, we need to change 4 to 3 since 4 is larger than the threshold value 2 in our experiments. In addition, we only search for the candidate AC coefficients in the up-left triangle area in a selected DCT block since almost all the AC coefficients in the low-right area are zeros (shaded area). Equation (1) shows us the principle of signature embedding as well. $AC_i$ is the ith AC coefficient used to embed $SIG_i$ – the ith bit of signature. $AC_i'$ is the changed value of $AC_i$. TV is the threshold value. On average, only about 50% selected AC coefficients need to be changed to embed one bit of the signature.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 23 | 3 | -1 | 0 | 0 | 0 | 0 | 0 |
| -2 | -1 | -2 | 0 | 0 | 0 | 0 | 0 |
| -1 | -1 | 2 | 0 | 0 | 0 | 0 | 0 |
| -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Fig. 2. No need to change the AC coefficient

Change to 3

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 15 | 2 | -2 | -1 | 0 | 0 | 0 | 0 |
| 4 | -1 | 2 | 0 | 0 | 0 | 0 | 0 |
| -1 | 2 | -2 | 0 | 0 | 0 | 0 | 0 |
| -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Fig. 3. Decrease the AC coefficient by one

$$AC_i' = \begin{cases} ACi, & if \quad parity(ACi) = parity(SIGi) \\ ACi-1, & if \quad parity(ACi) \neq parity(SIGi) and ACi > TV \\ ACi+1, & if \quad parity(ACi) \neq parity(SIGi) and ACi < TV \end{cases} \quad (1)$$

### C. Signature Extracting Scheme

To extract the embedded 128 bit signature, we apply the same DCT and Quantization step on each 8×8 pixel blocks of watermarked image. The same pseudo-random number generator key is used to locate the 128 DCT AC coefficients from which the signature are embedded and send all other DCT coefficients to the MD-5 hash function to produce the computed signature. Meanwhile, we can obtain the extracted signature by recovering the embedded signature. If the computed signature is identical to the extracted signature we know that the image is authentic. Otherwise it has been tampered. Figure.4. shows the principal process of extraction.

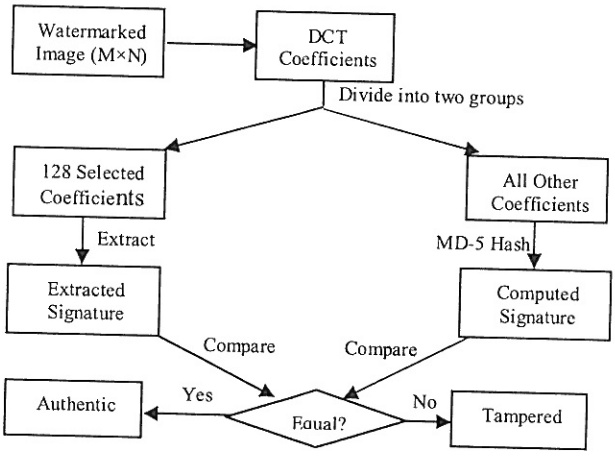


Fig. 4. Extracting Scheme

One issue should be noticed that if a third-party modified a watermarked image which incurs a DCT coefficient change, our method can catch the alteration definitely. However, if he/she only made some minor changes which didn't alter any DCT coefficients finally, our method cannot detect the modification. We don't take this as an incapability of our technique however. After all, digital images are usually stored and transmitted in the compression format which means these minor changes

cannot survive in the compression-domain. In other words, we need not to care about this kind of alteration.

## IV. EXPERIMENTS AND ANALYSIS

In our experiment, we use a 512×512 gray scale JPEG image 'lena.jpg' (Figure.5) which has 4096 8×8 pixel blocks. DCT and Quantization transforms are applied on each block to get a corresponding DCT coefficient block matrix. We set the threshold value to 2 and use this value to verify if a block is embeddable or not. For example, if a block has an AC coefficient which is equal to 2 we think that this block is embeddable. In this manner, there're totally 2072 embeddable blocks in this image. Next, we use a pseudo-random number generator to create 128 random numbers within 1 ~ 2072 and these 128 blocks will be used to embed the 128 bit signature. To alleviate image distortion, we only embed one bit of the signature in one AC coefficient in a selected embeddable DCT coefficient block. All the rest 262016 coefficients are sent to MD-5 hash function to create a 128 bit signature. After we get the signature we embed each bit of it into one embeddable AC coefficient in a selected DCT block. We use the parity of the selected coefficient to embed the signature. If discrepancy between the parity of the selected AC coefficient and the signature bit arises, we decrease the AC coefficient by one if it is larger than the threshold or increase it by one if it is smaller than the threshold.

After embedding, we apply inverse DCT and Quantization transform and then output the watermarked image matrix to a jpg file. The PSNR of the watermarked image is 41.76dB. Also, we did many tests to verify our scheme (see APPENDIX). Below are three typical ones.

*1) Test One:* After watermarking, there is no any alternation (Figure.6). Our algorithm can tell that the image is authentic.

*2) Test Two:* After watermarking, using an image processing software to obviously change a small part of the watermarked image (Figure.7). A completely different signature produced and thus the tamper was detected.

*3) Test Three:* After watermarking, only change one DCT coefficient in the DCT-domain (Figure.8). Also, a different signature is created and the tamper was detected.

Fig. 5. Original Image

Now let's analyze the performance of our reactive tamper detection watermarking system when it is undergone some typical attacks.

*1) Blind Attack:* For the blind attacks which arbitrarily change a watermarked image without any knowledge of the presence of the embedded watermark, the change should be easily discovered by our method since blind attacks most probably alter some DCT coefficients.

Fig. 6. Watermarked Image (no tamper)

Fig. 7. Visible Tamper

Fig. 8. Invisible Tamper

*2) Leave Mark Alone Attack:* This kind of attack tries to modify a watermarked image without interfering the embedded mark. However, this sort of attack cannot cheat our technique since the non-mark part of an image is protected by the signature. Namely, any modification of the non-mark part will incur a big change of the computed signature which will be compared with the intact

embedded signature. Any discrepancy between the two signature will tell us that the image was tampered.

*3) Mark Transfer Attack:* Attackers may use a valid mark from a watermarked image as the mark for another image [3]. However, this attack can be successful only after a attacker deduced how a mark was inserted. This means that he or she must possess the following information: the criteria used to select candidate DCT blocks, the pseudo random key used to randomly select 128 DCT blocks from the candidate blocks, the standard used to pick up a particular AC coefficient in one selected block, the specific insertion method on this selected AC coefficient and the hash function used to produce the signature. It is extremely difficult for a third-party to deduce all the information above by using brute force manner. First, we can see that there are about 2072!/(2072-128)!=$10^{425}$ possible permutations for the selected 128 blocks (128 blocks selected from 2072 blocks). One permutation corresponds to one possible manner to embed 128-bit long signature. The search space is rather huge for an adversary. Even if the attacker accidentally find the 128 randomly selected blocks the probability that the extracted hash value will match the computed hash is still $2^{128}$ since each block can be used to embed a "1" or a "0". Even he knows the 128 selected blocks and the sequence of the signature bits which were embedded in these blocks, he still needs to figure out which one particular coefficient out of $2^6$ coefficients was used to embed data. In short, we think that it is extremely unlikely that a random selected signature matches the modified watermarked image.

*4) Mark Remove Attack:* Attackers may want to remove the mark without leaving any remnants. This also requires an attack to know everything about mark embedding as in the *3) Mark Transfer Attack.*

## V. CONCLUSIONS

We have presented a digital watermarking scheme in JPEG compression domain for image authentication which can detect even one DCT coefficient alternation. The idea has been successfully extended to JPEG color images and can be easily extended to other compression domain such as wavelet-domain. For color images, we can work in RGB coordinates or $YC_rC_b$ coordinates( see APPENDIX). This method can also be applied to MPEG compressed digital video. A straightforward way is to mark I-frames using our scheme. The performance of our system demonstrates that even one jpg image file element alternation can be detected. For future work, intensive tests and better AC coefficient selection which could improve image fidelity should be pursued.

## VI. REFERENCES


[1] M. Yeung and F. C. Mintzer, "Invisible watermarking for image verification," *Journal of Electronic Imaging*, 7(3), July 1998, pp. 578-591.

[2] T. Lin and E. J. Delp, "A Review of Fragile Image Watermarks, " *Multimedia and Security Workshop at ACM Multimedia 99*, Orlando, FL, USA, Oct. 1999.

[3] F. Mintzer, G.B. Braudaway, and M.M. Yeung, "Effective and Ineffective Digital Watermarks," *Proceedings of IEEE ICIP'97*, Santa Barbara, CA, Oct. 1997.

[4] Fred Mintzer, Gordon W. Braudaway and Alan E. *Bel*, "Opportunities for Watermarking Standards," *In Communications of the ACM,* Vol. 41, July 1998.

[5] G. Friedman, "The trustworthy digital camera: Restoring credibility to the photographic image, " *The Fourth National Technology Transfer Conference and Exposition,* Volume 2 p 430-435, February 1994.

[6] P. Wong, "A public key watermark for image verification and authentication, " *Proceedings of the IEEE International Conference on Image Processing,* vol. 1, pp. 455-459, Chicago, October 1998.

[7] D. Stinson, Cryptography Theory and Practice, CRC Press, Boca Raton, 1995.

[8] R. Wolfgang and E. J. Delp, "Fragile Watermarking Using the VW2D Watermark," *Proceedings of the SPIE/IS&T International Conference on Security and Watermarking of Multimedia Contest*, San Jose, January 25-27, 1999, vol. 3657, pp. 204-213.

[9] N. Memon, S. Shende, and P. Wong, "On the security of the Yueng-Mintzer Authentication Watermark, " *Final Program and Proceddings of the IS&T PICS 99,* pp. 301-306, Savanna, Georgia, April 1999.

[10] S. Walton, "Information Authentication for A Slippery New Age," *Dr. Dobbs Journal*, vol. 20, no. 4, April 1995, pp. 18-26.

[11] R. Wolfgang and E. J. Delp, "A Watermark for Digital Images," *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, 1996, pp. 219-222.

[12] M. Wu and B. Liu, "Watermarking for image authentication," *Proceedings of the IEEE International Conference on Image Processing*, Chicago, 1998, pp. 437-441.

[13] C-Y. Lin and S. F. Chang, "Semi-fragile Watermarking for Authenticating JPEG Visual Content," *SPIE Security and Watermarking of Multimedia Content II*, January 2000.

[14] P. Wong, "A watermark for image integrity and ownership verification," *Final Program and Proceedings of the IS&T PICS 99*, Savanna, Georgia, April 1999, pp. 374-379.

[15] A. H. Paquet, R.K. Ward, and I. Pitas, "Wavelet Packets-based Digital Watermarking for Image Authentication and Verification," *Signal Processing*, vol. 83 , issue 10, Oct. 2003, pp. 2117-2132.

[16] D. Kundur and D. Hatzinakos, "Towards a Telltale Watermarking Technique for Tamper-Proofing," *Proc. IEEE Int. Conf. On Image Processing*, Chicago, Illinois, vol. 2, October 1998, pp. 409-413.

[17] L. M. Marvel, G. W. Hartwig, and C. Boncelet, Jr., "Compression-Compatible Fragile and Semi-Fragile Tamper Detection", *Proc. SPIE Security and Watermarking of Multimedia Contents*, San Jose, California, January 2000, pp. 140-151.


## VII. APPENDIX

We present four gray and color JPEG images to show the image quality after watermarking using our method. The average PSNR for these four JPEG images is 42.65dB.

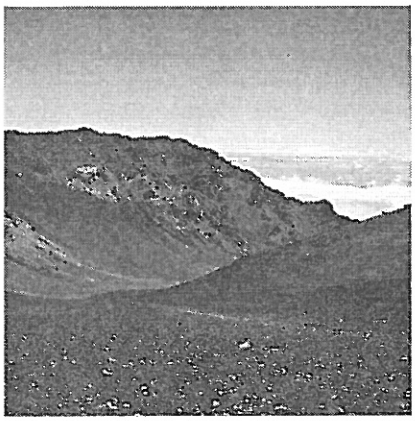

Fig. 9. Original "Moutain.jpg"

Fig. 10. Watermarked "Moutain.jpg"

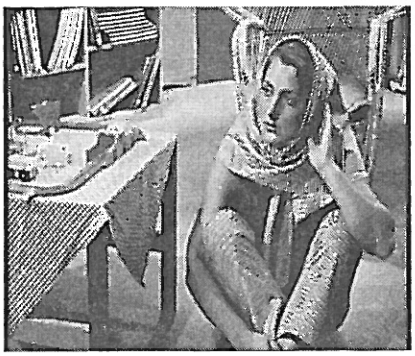

Fig. 11. Original "Barb.jpg"

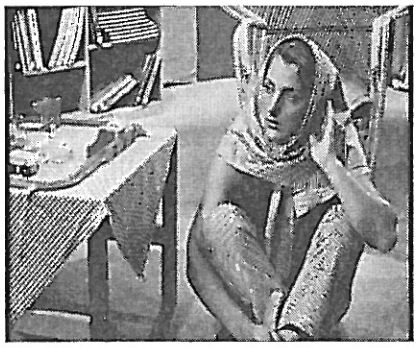

Fig. 12. Watermarked "Barb.jpg"

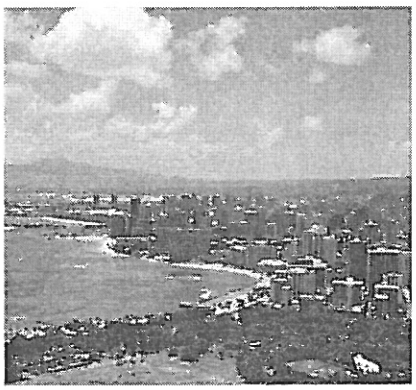

Fig. 13. Original "Honolulu.jpg"

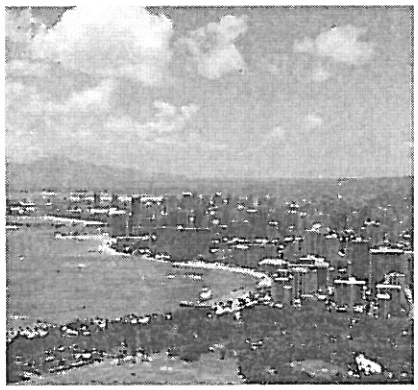

Fig. 14. Watermarked "Honolulu.jpg"

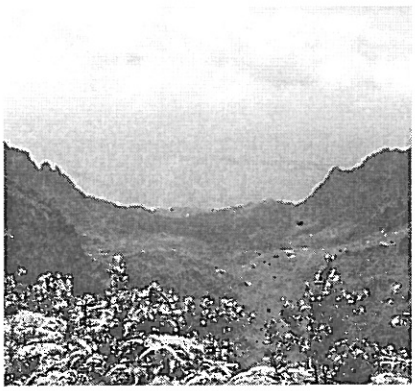

Fig. 15. Original "Kalalau.jpg"

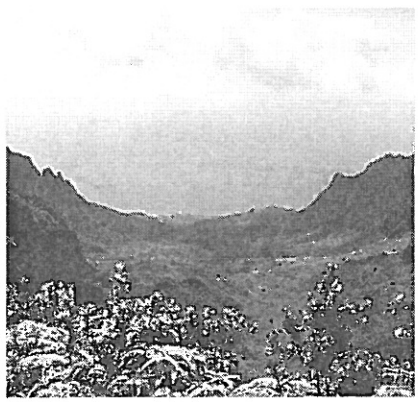

Fig. 16. Watermarked "Kalalau.jpg"