# Approximate Image Message Authentication Codes

Liehua Xie, Gonzalo R. Arce, *Fellow, IEEE,* and Richard F. Graveman

***Abstract*—This paper introduces approximate image message authentication codes (IMACs) for soft image authentication. The proposed approximate IMAC survives small to moderate image compression and it is capable of detecting and locating tampering. Techniques such as block averaging and smoothing, parallel approximate message authentication code (AMAC) computation, and image histogram enhancement are used in the construction of the approximate IMAC. The performance of the approximate IMAC in three image modification scenarios, namely, JPEG compression, deliberate image tampering, and additive Gaussian noise, is studied and compared. Simulation results are presented.**

***Index Terms*—Content-based image authentication, message authentication code (MAC), tampering detection.**

## I. Introduction

CONVENTIONAL countermeasures used against integrity threats to ordinary data (e.g., spreadsheets, word processing files, etc.) rely on cryptographic methods; more precisely, they rely on message authentication codes (MACs). MACs are widely used cryptographic methods for Alice and Bob,[1] who share a secret key, to ensure each other that their messages are authentic and unmodified [1], [2]. The creator of a MAC takes a message and key as inputs and computes a checksum in a way that is hard, given some set of messages and their MACs computed with an unknown key, to forge a MAC for a new message or to discover the key. The usual tools for constructing MACs are cryptographic hash functions [3] and block ciphers [4]. Conventional cryptographic procedures originally crafted to secure ordinary data may be inadequate for multimedia digital sources (images, voice, streaming audio, and video). Ordinary data are intolerant to any changes and the integrity check is based on strict bitwise accuracy. For example, altering a single bit in a binary file can have disastrous consequences. Multimedia, on the other hand, is generally tolerant to minor changes.

Manuscript received March 7, 2001. This work was supported in part by the National Science Foundation under Grant CDA-9703088, by the Dupont Company, and through collaborative participation in the Advanced Telecommunications and Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.  The associate editor coordinating the review of this paper and approving it for publication was Dr. John Aa. Sorensen.

L. Xie and G. R. Arce are with the Department of Electrical and Computer Engineering, University of Delaware, Newark, DE 19716 USA (e-mail: xie@ee.udel.edu; arce@ee.udel.edu).

R. F. Graveman is with the Mathematical Sciences Research Center, Telcordia Technologies, Inc., Morristown, NJ 07960 USA (e-mail: rfg@acm.org).

Publisher Item Identifier S 1520-9210(01)04321-8.

[1]Alice and Bob are the renowned "first couple" of cryptography. Alice is the sender and Bob is the legitimate receiver.

Traditional message digest schemes, such as cipher block chaining message authentication code (CBC MAC) [5] or hashed-based message authentication code (HMAC) [6] with MD5 [3], create a hard authenticator, because they output entirely different message digests, even if only one bit in the original data is changed. A modification of a single message bit is designed to affect the calculation of all of the checksum bits and to change each one in roughly half of the cases. These MACs are deliberately constructed to be as unforgiving as possible, and they fit those applications where Alice and Bob's security requirement is to reject any message that has been altered to the slightest extent. In other applications (e.g., voice or imagery), incidental noise or the effects of lossy downstream compression are at least somewhat acceptable, so long as Alice and Bob can identify and reject all-out forgeries, substantial modifications of content, splicing or "cut-and-paste" attacks, etc. In certain applications, the "acceptable modification" to the message may include the insertion of hidden data: e.g., digital watermarks or fingerprints to signify ownership or to mark a particular copy. One approach to overcome the limitations of traditional MACs for these emerging applications is to pre-compute the expected modifications introduced by the application at hand and compute a MAC based on the result of this expected operation. This may be feasible in some cases, but, in general, it fails to account for unpredictable effects such as channel noise and inhomogeneous multicast where uncertainty in the communication channel is present. Moreover, when message compression is required, it is preferable to use authentication mechanisms prior to compression [7]. One of the reasons for this is that compression algorithms are not deterministic in the sense that various implementations of the algorithm achieve different tradeoffs in speed versus compression ratio and, as a result, produce different compression forms. These different compression implementations, however, are inter-operable because any version of the algorithm can correctly decompress the output of any other version [7]. Applying the hash function and signature after compression would constrain all implementations to be exactly the same.

As a result, existing image authentication schemes fall into two categories: strict and nonstrict authentication. Strict image authentication is used to protect images from the slightest modification therefore traditional MACs are used [8], [9]; Nonstrict authentication is used to ensure the content integrity of images [10]–[12]. The authentication codes are mostly constructed from the prominent features of an image.

Recently, a new cryptographic algorithm referred to as the approximate MAC (AMAC) has been developed [13], [14]. It provides an authentication and data integrity mechanism that *bends* rather than *breaks* when a received message is slightly different

from what was sent. Though AMACs provide a necessary cryptographic primitive that probabilistically estimates the degree of bitwise similarity of two digital messages with only a short checksum, limitations exist if the AMAC is applied directly to the authentication of images. First, the AMAC does not in itself distinguish modifications to an image due to tampering (for instance, adding or removing an object) from those due to some acceptable image manipulations (for instance, image compression). Second, the AMAC cannot spot the location of differences between the messages. Thirdly, the AMAC only measures bitwise differences, not differences in pixel values. Lastly, the fluctuation of the AMAC difference around its expectation is large and it is difficult to set a proper threshold for making the authentication decision. Therefore, we propose to use the AMAC primitive to construct an approximate authentication method tailored to the spatial representation of gray-scale images, namely the approximate IMAC. Techniques such as block averaging and smoothing, parallel AMAC computation, and image histogram enhancement are used in the approximate IMAC to overcome the limitations of the AMAC.

Section II briefly introduces the AMAC algorithm and the statistical properties of the AMAC. Then the limitations of the AMAC for image authentication are presented. Section III details the construction of the approximate IMAC. In Section IV, the performance of the proposed IMAC is described analytically. Simulations illustrating the authentication capabilities of the approximate IMAC are provided and compared with the results from analysis. Ideas for improving the approximate IMAC are discussed in Section V. Section VI concludes the paper. In addition, the simulation results verifying the assumption made in the analysis are presented in the Appendix.

## II. The Approximate Message Authentication Code (AMAC) and Its Limitations

### A. The AMAC Algorithm

The AMAC [13], [14], is a probabilistic checksum calculated using pseudorandom permutations, masking, and the majority function. It is capable of estimating the distance of two binary sequences, since similar messages are likely to have similar AMACs. The closeness or similarity between two sequences is measured by their Hamming distance. The term "approximate" comes from the fact that it is desirable for two sequences that are close, yet not identical, to have the same or only slightly different AMACs.

Let $M$ be an input binary message of length less than or equal to $L \times R \times S$, where $L$ is the AMAC length and $R$ and $S$ are positive integers. Given a shared key $K$ used to secure the process and a pseudorandom bit generator $P$, the algorithm for constructing the AMAC is given below. $L$ should be large enough to resist brute-force forgeries[2] and other such attacks because $L$ determines the number of possible MAC combinations and the amount of time for the attackers to break the code. As with conventional MACs, the length of an AMAC is typically chosen in the range $80 \leq L \leq 400$. It is convenient to pick $R$ and $S$ odd. Choosing the relative sizes of $R$ and $S$ is a matter of fine-tuning the system, so as a matter of practice, one can start with $R = S$.

[2]That is, attacks by trying all possible combinations of binary bits.

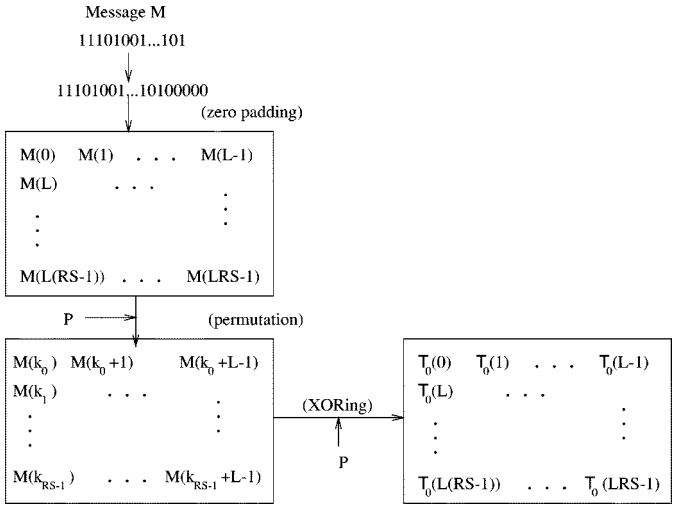


Fig. 1. Reformatting, row permutation, and XORing operations.

Step 1) *Initialization*: The key $K$ and some additional information $I$ are input to the pseudorandom bit generator $P$ as a seed.

Step 2) *Formatting and Randomization*: This stage is shown in Fig. 1. First the binary message $M$ is padded with zeros to the length $L \times R \times S$ if necessary. The padded message is next re-formatted into a binary array of $R \times S$ rows by $L$ columns. Next, $P$ is used to select a pseudorandom permutation on $\{1, 2, \ldots, R \times S\}$ and the rows in the array are permuted accordingly.[3] The purpose of the pseudorandom permutation is to destroy the spatial correlation of rows. If the correlation of positions between columns is also undesired, an optional operation that performs a different circular shift of each row should be added. The permuted array is then masked by XORing all of its bits with a new set of pseudorandom bits generated by $P$. Denote the permuted and masked array as $T_0$.

Step 3) *Two Rounds of Majority Calculation*: This stage is shown in Fig. 2. First we take $R$ rows at a time from $T_0$ and form $S$ subarrays, each of $R$ rows and $L$ columns. Denote the sub-arrays as $T_0^0$, $T_0^1$, ..., $T_0^{S-1}$. For each $T_0^k$, $0 \leq k \leq S-1$, compute the majority bit of each column, i.e., the most frequent bit in the column. The majority bits then make up one new row of length $L$. $S$ new rows of majority bits for $S$ sub-arrays are formed, and they are arranged in a new $S \times L$ array denoted by $T$. The elements of $T$ can be derived as $T(i, j) = \text{MAJORITY}[T_0^i(0, j), \ T_0^i(1, j), \ \ldots, \ T_0^i(R-1, \ j)]$. It is seen that, by this round of majority calculation, the volume of the data is reduced to $S \times L$ bits.

[3]One way to use $P$ to obtain a pseudorandom permutation of the values 1 through $R \times S$ is as follows. An additional key $K_{perm}$ is obtained from $P$. With this key, use either a traditional MAC scheme or a block cipher to generate MACs or encryption of the values 1 through $R \times S$. Sort the results and use the sorted listed to specify the permutation. Note that this can be precomputed from $P$, $K$, and $I$.

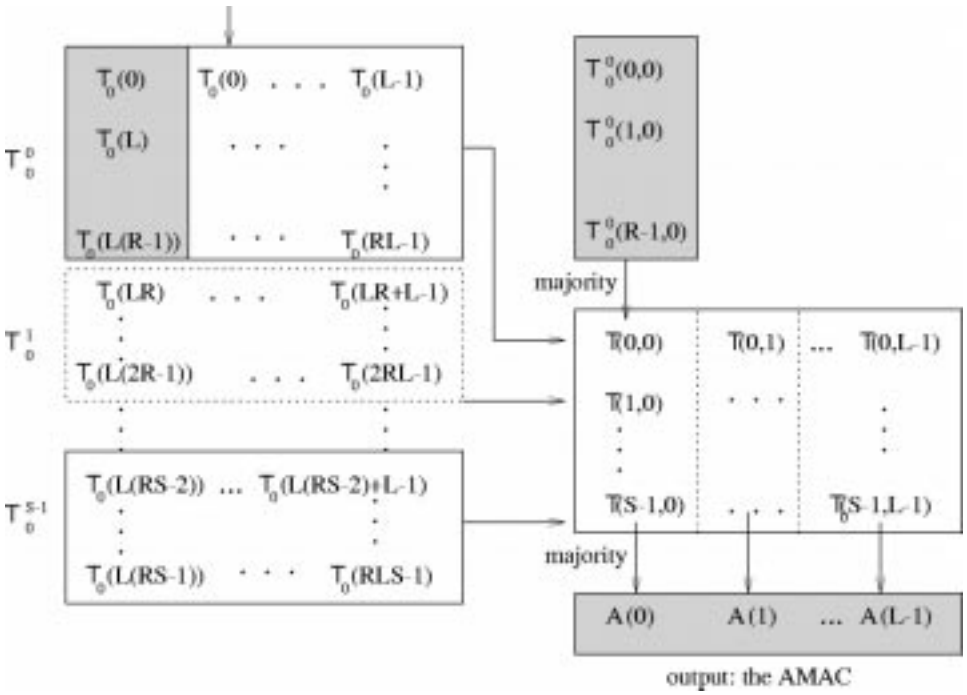


Fig. 2. Two rounds of MAJORITY calculation in the AMAC algorithm.

A second round of majority calculation is carried out for $T$. Compute the majority of each column of $T$ and obtain $L$ bits. These $L$ bits are taken as the AMAC, $A$. If $R$ or $S$ is even, a row of pseudorandom bits is added to the corresponding majority calculation to "break ties."

The constructed AMAC, $A$, which is a binary sequence of $L$ bits, and the initialization data $I$ are sent along with $M$.

### B. Probabilistic Properties of the AMAC

Let $P_A$ be the probability that a given AMAC bit changes, as derived in [14]

$$P_A = f(l_M, L, d_M) \tag{1}$$

where

$l_M$ length of the binary message;
$L$ length of AMAC;
$d_M$ number of modified bits in the message.

For $L$ AMAC bits, the expected number of differences is

$$E\{d_A\} = L \cdot P_A \tag{2}$$

where $d_A$ denotes the number of different bits in the AMAC.

$P_A$ versus the fraction of the differences between two sequences is plotted in Fig. 3. This figure depicts an AMAC of $L = 63$, calculated for sequences with various lengths $l_M = 9L, 81L, 729L, 6561L$. Fig. 3 shows that $P_A$, the probability that an AMAC bit changes, increases when the fraction of bit differences between the original and the modified sequence increases. When all bits are altered in the sequence, $P_A = 1$, meaning every AMAC bit changes. Note also when the fractions of bit differences are the same, the $P_A$ values with different $l_M$ are close to each other.

The AMAC thus provides a means to detect the degree of bitwise similarity of two digital sequences. Since two sequences

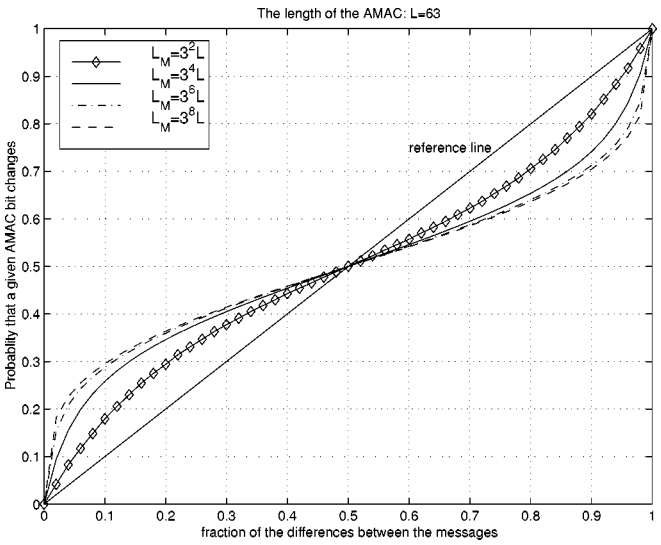


Fig. 3. Probability that a given AMAC bit changes versus the fraction of differences between two sequences $L = 63$.

with small Hamming distance have small probability of having different MAJORITIES, two iterations of MAJORITY calculations should produce a "slowly changing fingerprint" of the input sequence.

### C. Limitations of the AMAC for Image Authentication

The AMAC is a soft message digest for general data, though its construction was originally motivated by approximate image authentication. In order to input an image into the AMAC algorithm, the image data were formatted into a binary array by converting all pixels from their decimal to binary representations and scanning the pixels in a row-by-row manner. Through the analysis and simulation in [13], [14], we conclude that lim-

itations exist if the AMAC is applied directly to the authentication of images.

First, the AMAC does not in itself distinguish modifications to an image due to tampering (for instance, adding or removing an object) from those due to some acceptable image manipulations (for instance, image compression). Both tampering and image compression modify the pixel values, though the semantics of how they modify an image are not the same. Modification by tampering results in large changes to certain coefficients that are most likely concentrated in specific areas. Lossy compression tends to smooth the image, because it is in essence a low-pass filtering process. Compared with tampering, its impact on a coefficient is relatively small, but its overall impact is widely spread across the image. However, the AMAC is "blind" to such differences. The likely difference in the AMACs of two messages is only a function of their Hamming distance. This does not account for the density and distribution of the differences between the messages. In addition, the Hamming distance of the AMACs is the only information available for the receiver to judge the authenticity and integrity of the image. Therefore, if the number of the differences caused by compression is as large as that caused by tampering, no valid threshold can be determined to separate image tampering from compression. A partial remedy to this problem is to constrain the rotations to byte boundaries, so that bit locations within a pixel remain aligned, and the receiver can observe whether a different bit in two AMACs occurs in a low- or high-order position within a pixel.

Second, the AMAC cannot spot the location of differences between the messages. For detecting image tampering, it is highly desirable to expose the location of the tampering. But the permutation and rotations performed in the AMAC algorithm destroy spatial relationships (intentionally), so information based on pixel location is lost.

Third, the AMAC only measures bitwise differences, not differences in pixel values. With pixel values represented in binary, small changes in pixel values do not always produce small Hamming distances. For example, the Hamming distance between 126 and 127 is one, whereas that between 127 and 128 is eight. A partial remedy to this problem is to represent the pixel values with a Gray code[4] [15].

Finally, both the analysis and the experimental data show that the fluctuation of the AMAC difference around its expectation is large. Relying on one criterion, the Hamming weight of the AMAC difference, may not be adequate to determine whether or not an image is tampered, let alone the question of where the image is tampered.

## III. Approximate Image Message Authentication Code (IMAC)

The AMAC successfully addresses the issue of approximate bit-by-bit data integrity but makes no attempt at pattern matching and identifying specific types of distortion. Thus AMACs cannot tell the difference between image compression and tampering, which is a severe limitation. For more powerful approximate image authentication, the capabilities of error detection and localization are needed. Rather than simply patching the AMAC, we constructed a new method tailored to the spatial representation of gray-scale images, namely, the approximate IMAC.

[4]A Gray code is a binary code in which consecutive decimal numbers are represented by binary expressions that differ in the state of one, and only one bit.

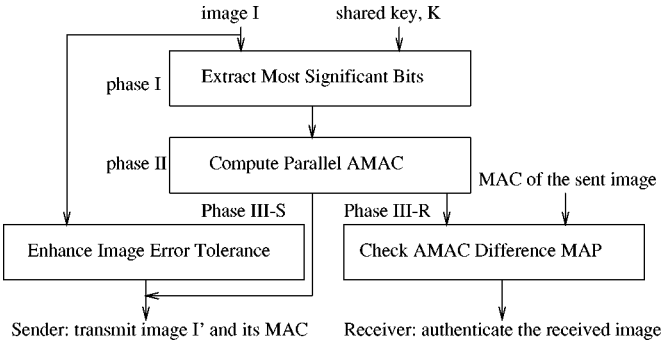


Fig. 4. Data processing flow for the sender and the receiver.

### A. Image Authentication Using the Approximate IMAC

The approximate IMAC is constructed with a three-phase operation, as shown in Fig. 4. The first and second phases are identical for the receiver and the sender, in which the most significant bits are extracted (Section III-B), a parallel AMAC computation is performed, and the IMAC of a given image is created (Section III-C). The third phases are different. At the sender, the third phase enhances the image's error tolerance (Section III-D). At the receiver, it authenticates the image by checking the AMAC difference map (Section III-E). Denote these phases by I, II, III-S, and III-R where III-S and III-R stand for the third phase at the sender and receiver, respectively.

### B. Phase I: Most Significant Bits Extraction

To assure the robustness of the approximate IMAC for images under acceptable manipulations, the least vulnerable binary representation of an image is sought. Since low-frequency components of an image are the least affected during low-pass filtering, and the highest order bit of a coefficient is the least possibly modified bit when the coefficient slightly changes, we decided that the highest order bit of the lowest frequency coefficients is robust and used them as the most significant bits. In our scheme, the highest order bit of each $8 \times 8$ block average is used to build a reliable, compact representation of an image. The scheme is displayed in Fig. 5.

We divide an 8-bit $8w \times 8h$ image $I$ into $w \times h$ nonoverlapping $8 \times 8$ blocks. The $8 \times 8$ blocks are used because they are the most compatible with JPEG compression, in which $8 \times 8$ blocks are used in the DCT. These blocks are labeled as $B_{i,j}$ ($0 \le i \le h-1, 0 \le j \le w-1$). Then the mean of the coefficients in each block is computed. Denote it by $m(i, j)$ ($0 \le m(i, j) \le 255$). The most significant bit of $m(i, j)$, denoted by $b(i, j)$, is output. A binary map $b$ is constructed by $b(i, j)$ ($0 \le i \le h-1$, $0 \le j \le w-1$). In (a) of Fig. 6, the $512 \times 512$ 8-bit test image is presented. The block average $m$, and the binary map $b$ of the test image are depicted in Fig. 7.

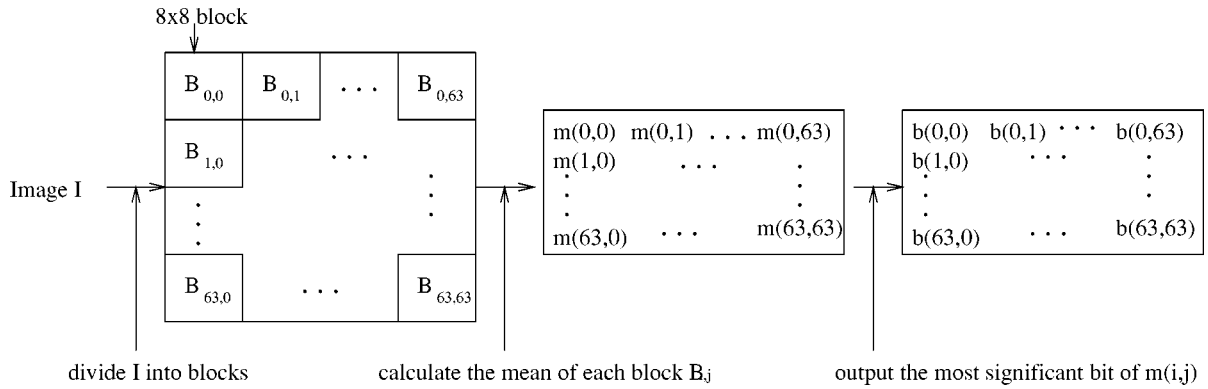


Fig. 5. Extract the most significant bits.

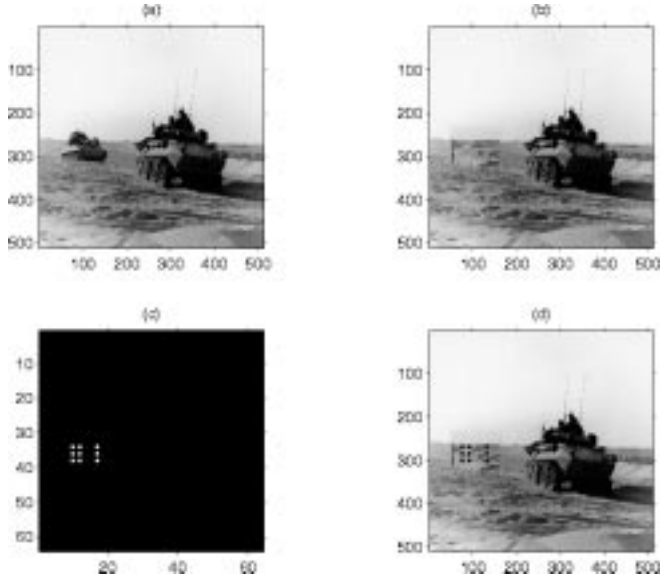

Fig. 6. (a) Test image. (b) Tampered image: the tank is removed. (c) The IMAC difference map obtained by cross-checking the IMAC of rows and column. (d) The IMAC difference mapped in the tampered image.

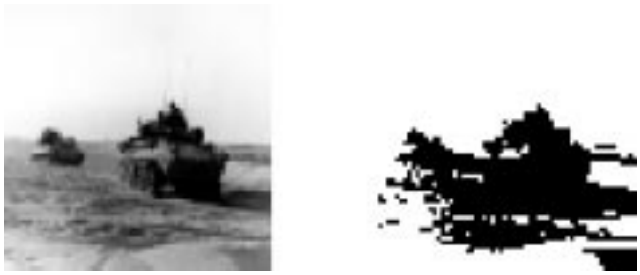

Fig. 7. Left: $m$, the block average of $I$. Right: $b$, $m$ in binary representation.

### C. Phase II: Parallel AMAC Computation

In this phase, one IMAC bit is calculated for each row and column of $b$ using the AMAC algorithm. Hence, an array of $w+h$ IMAC bits is obtained using the procedure shown in Fig. 8. Denote the resulting array by $A'$. Let $A'(k)$ $(0 \leq k \leq h-1)$ be the approximate IMAC bit for $b(k, :)$, the $k$th row of $b$ and $A'(k)$ $(h \leq k \leq w+h-1)$ be the IMAC bit for $b(:, k-h)$, the $(k-h)$th column of $b$. In addition, using the key $K$, $A'$ is permuted[5] and XORed with pseudorandom bits. The final output array $A$ is obtained. The purpose of the randomization and XOR operations is to prevent an attacker from being able to identify any bit of $A$ with a specific row or column of $b$.

At this point, the sender proceeds to send $A$. Receivers get an approximate IMAC with the received image and follow the process described in Section III-E.

### D. Phase III-S: Error Tolerance Enhancement

Though $b(i, j)$ is defined to be the most significant bit of $m(i, j)$, it is possible that $b(i, j)$ will change when $m(i, j)$ is slightly changed. In fact, $b(i, j)$ is vulnerable to small modification of $m(i, j)$ whenever $m(i, j)$ is around the threshold 127.5. To improve the image's capacity for error tolerance, we introduce a guarding zone in the image's histogram by transforming it and creating a gap around the threshold.

Let $T_d$ be the acceptable maximum absolute difference between the original $m(i, j)$ and a modified $m(i, j)$ caused by some image manipulations. We prohibit values in an error tolerance zone around the threshold 127.5 and split the interval of [0, 255] into two separate regions: $[0, 127-T_d]$ and $[128+T_d, 255]$. The minimum distance between two coefficients in two different regions is $2T_d+1$. We then linearly map all $m(i, j)$s from 0 to 127 to the first region and those from 128 to 255 to the second region. In this way, a gap of $2T_d+1$, which is symmetrically positioned around 127.5, is created in the histogram of $m$. If the change of $m(i, j)$ is within $T_d$, the highest bit of the modified $m(i, j)$ remains the same. Suppose $m(i, j)$ is mapped into $m'(i, j)$. Define $d_m = m'(i, j) - m(i, j)$. Since $m(i, j)$ is the mean of $B_{i,j}$, every coefficient in $B_{i,j}$ is changed by the same amount, $d_m$. Define the new image as $I'$. Then, instead of $I$, $I'$ will be sent via the channel.

An image manipulation can be characterized by the parameter $T_m$, the maximum absolute change it causes to the value of $m(i, j)$. If $T_m \leq T_d$, the IMAC, $A$, stays the same. Otherwise, $A$ may be different. Therefore a larger $T_d$ results in a stronger capability of the approximate IMAC to resist modification. On the other hand, the modification forces a gap of width $2T_d+1$ in the histogram of $m$, the block average of the image. In some sense, $m$ is dithered because a smaller scale is used to represent $m$. The larger $T_d$ is, the larger the dithering effect. When $T_d$ increases to the extreme ($T_d = 127$), $m$ becomes binary. So there is a tradeoff between the quality of $I'$ and the robustness of its

[5]The permutation is carried out following the method used in Step 2) of the AMAC algorithm (Section II-A): Formatting and Randomization

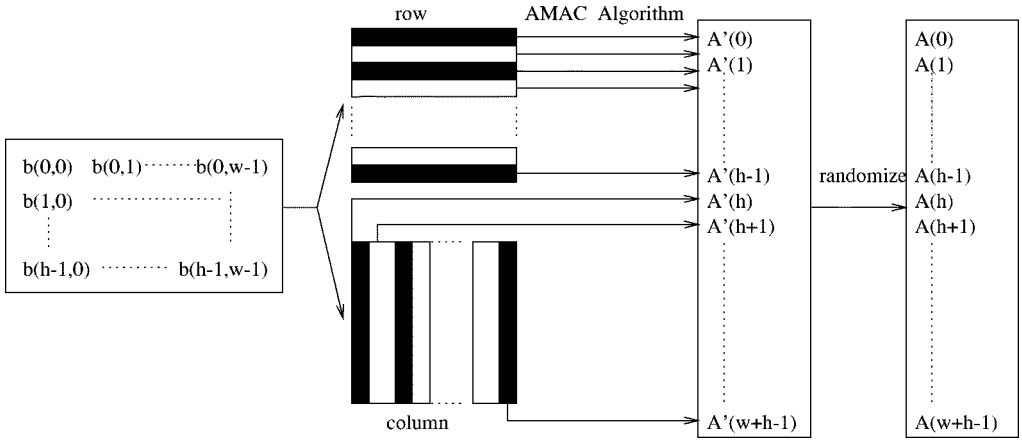


Fig. 8. Parallel AMAC computation on all rows and columns.

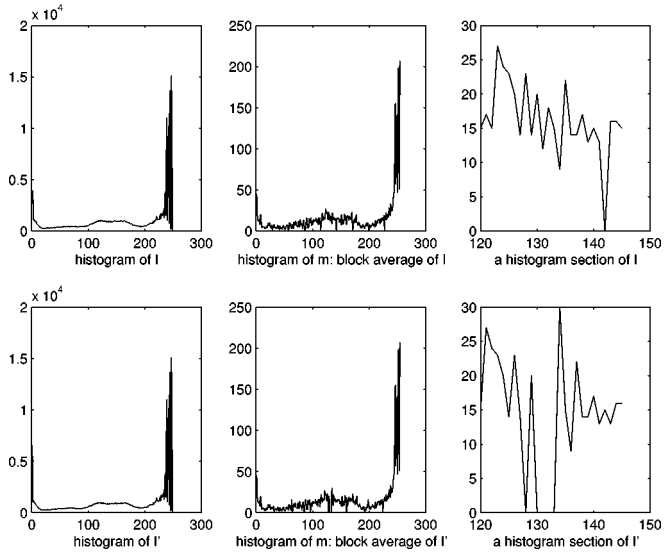


Fig. 9. Comparison between the histograms of the test image before and after histogram transformation.

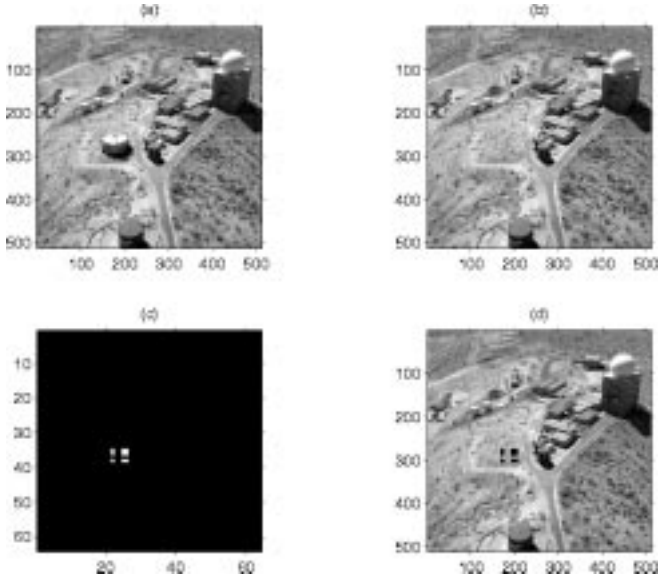

Fig. 10. (a) Test image. (b) Tampered image: the building is removed. (c) The IMAC difference map obtained by cross-checking the IMAC of rows and columns. (d) The IMAC difference mapped in the tampered image.

IMAC. In practice, $T_d = 2$ provides good performance, because it produces both a sufficient guarding zone for moderate compression and a slight distortion to the images. Using $T_d = 2$, we obtained $I'$, the transformed test image. No visible difference is observed between $I'$ and $I$. The histogram change is observed and plotted in Fig. 9. It is seen that the histograms of $I$ and $I'$ look very close to each other, therefore the histogram transformation does not severely distort the image. The histograms of $m$ and $m'$ appear to be slightly different and a closer look at their histograms around the threshold 128 clearly indicates that a gap is created after the transformation.

### *E. Phase III-R: IMAC Difference Map Checking*

Denote the approximate IMAC of the received image by $a$. Note it has been assumed that the receiver will be able to recover an accurate copy of $A$. Using the key $K$, $a'$ and $A'$ are recovered. Let $d = A' \bigoplus a'$, and create a binary map $D$ with $h$ rows and $w$ columns

$$D(i, j) = \begin{cases} 0 & \text{if } d(i) = d(j+h) = 1 \\ 1 & \text{otherwise.} \end{cases} \tag{3}$$

Fig. 6 shows an example of the IMAC difference map. In (b) of Fig. 6, an example of a tampered image is given. The difference map $D$ when an IMAC of the original image is compared with that of the tampered image is shown in (c). In (d), the IMAC difference is highlighted in the tampered image and it clearly indicates the area where tampering has taken place. Another example is provided in Fig. 6, in which a different test image is used and shown in (a) of Fig. 10. A similar set of test results are plotted.

## IV. Performance of the Approximate IMAC

We use an 8-bit $512 \times 512$ image as our example, and the desired IMAC length is 128. Since the block average in the approximate IMAC is equal to one eighth of the DC coefficient in the $8 \times 8$ DCT block, we find that $b$, the binary representation of the block average, can be equivalently constructed by extracting the highest order bits of the DC coefficients. In order for us to analyze the performance for DCT-based compression, this method will be used in the following.

### A. Impact of JPEG Compression

Lossy JPEG compression [16] quantizes the 64 DCT coefficients of each nonoverlapping $8 \times 8$ block using a quantization table. Larger quantization coefficients are used for higher frequencies, since they are less significant to the human visual system. By tuning the quantization table, various image compression ratios can be attained. A user defined "quality" factor can be used to determine the scaling multiplier to adjust a standard quantization table. A number of tuning methods have been proposed. Here we use a quality factor setting scheme which was introduced in the IJG JPEG library [17], in which a quality factor $Q = 75$ can be considered "high" quality, $Q = 50$ "good" quality, and $Q = 30$ "low" quality.

I) *Approximate IMAC Without Image Histogram Transformation:*

In Phase I of the approximate IMAC, the most significant bit of each block is extracted. We get $m$, the block average of an image. The impact of compression on the block average has been analyzed in our study of the AMAC in the minimum distortion model [13], [14]. It was seen that in the case of minimum distortion, the number of different AMAC bits, is generally smaller than in real quantization processes. In other words, the AMAC from the block average is more robust than the AMAC derived directly from the pixels. In the second step, $b$, a binary representation of $m$ is generated by extracting the highest order bits of $m$. Since the highest order bit is the least sensitive to coefficient change, the IMAC from $b$ will be even more robust. In all, the robustness of the approximate IMAC, which uses the binary representations of the image's block average as the input, is improved over the AMAC, which uses the bit representations of all image pixels. Furthermore, the approximate IMAC carries spatial information by computing one IMAC bit for each row and column of $b$.

To analyze the approximate IMACs probabilistic behavior, we make the following assumption.

*Assumption 1:* We approximate the distribution function of DC coefficients, $f(x)$, to be triangular in the range of [0, 2048], so that $f(x) = x/1024^2$, $0 \le x \le 1024$, and $f(x) = f(2048 - x)$ for $1024 < x \le 2048$, i.e., $f(0) = 0$, $f(1) = 1/1024^2, \ldots, f(1024) = 1/1024, \ldots, f(2048) = 0$.

The probabilities at both ends are 0 and the probabilities in the center, 1024, are the highest. In JPEG compression, the DC coefficient is further quantized. If $q$ is the DC quantization coefficient, $x$ the DC coefficient, and $x_q$ the quantized $x$, $x_q = \lceil x/q \rfloor \times q$, where $\lceil \rfloor$ represents the rounding process. The maximum absolute quantization error would be $(1/2)q$ and the minimum error is 0. After quantization, the highest order bit of the DC coefficient might change. For example, when $q = 8$, $x = 1023$, $x_q = 1024$, the highest order bits of $x$ and $x_q$ are different. In fact, the highest order bit of $x$ changes after quantization if $1024 - (q/2) < x < 1024$. In addition, if $q \ll 1024$, $f(x) \approx 1/1024$ for $1024 - (q/2) < x < 1024$. Therefore, the expected number of bit changes in $b$, denoted by $d_b$, is estimated as

$$d_b \approx \frac{q}{2048} \times l_b \tag{4}$$

when $q \ll 1024$. Let $l_b$ be the length of the message, $l_b = 64 \times 64$. Denote the probability that a given IMAC bit changes by $P_{A-I}$. $P_{A-I}$ is computed as a function of $l_b$ and $d_b$ [14]. There are 64 IMAC bits for the rows and 64 for the columns. Denote the expected number of IMAC differences in each direction by $d_{A-I}$, so $d_{A-I} = 64P_{A-I}$. (We assume each IMAC bit is independent). When the IMAC bits in two directions are cross checked, the expected number of differences in the difference map, denoted by $d_{map}$, is $d_{A-I}^2$. The density of the differences, denoted by $D_{map}$, is $d_{map}/(64 \times 64)$. In the quantization scheme, the quality factor $Q$ determines the quantization constant $q$. Table I shows the value of $q$, $P_{A-I}$, $d_{A-I}$, $d_{map}$, and $D_{map}$ when various quality factors $Q$ are used.

TABLE I

| $Q$ | 73 | 50 | 32 | 21 | 16 | 12 |
|---|---|---|---|---|---|---|
| $q$ | 9 | 16 | 25 | 36 | 49 | 64 |
| $d_b$ | 18 | 32 | 50 | 72 | 98 | 128 |
| $P_{A-I}$ | 0.0224 | 0.0384 | 0.0571 | 0.0778 | 0.0994 | 0.1214 |
| $d_{A-I}$ | 1.4336 | 2.4576 | 3.6544 | 4.9792 | 6.3616 | 7.7696 |
| $d_{map}$ | 2.0552 | 6.0398 | 13.3546 | 24.7924 | 40.4700 | 60.3667 |
| $D_{map}$ | 0.0005 | 0.0015 | 0.0033 | 0.0061 | 0.0099 | 0.0147 |

TABLE II

| $Q$ | 73 | 50 | 32 | 21 | 16 | 12 |
|---|---|---|---|---|---|---|
| $q$ | 9 | 16 | 25 | 36 | 49 | 64 |
| $d_b^H$ | 0 | 0 | 0 | 8 | 34 | 64 |
| $P_{A-I}^H$ | 0 | 0 | 0 | 0.0103 | 0.0405 | 0.0705 |
| $d_{A-I}^H$ | 0 | 0 | 0 | 0.6572 | 2.5949 | 4.5139 |
| $d_{map}^H$ | 0 | 0 | 0 | 0.4319 | 6.7335 | 20.3753 |
| $D_{map}^H$ | 0 | 0 | 0 | 0.0001 | 0.0016 | 0.0050 |

II) *Approximate IMAC Using Image Histogram Transformation with the Parameter $T_d$:*

When the image histogram is re-mapped using parameter $T_d$, a gap of width $2T_d + 1$ is created around 127.5. In this way, the robustness of the IMAC is further enhanced, because a coefficient changed by no more than $T_d$ will not exceed the boundary, and its binary representation stays the same in spite of the change. Let $T_d = 2$. The 8-bit coefficients of $m$ in the range [0, 127] will be linearly transformed to the range [0, 125], and [128, 255] to [130, 255]. A gap of width 5 is created. Since the DC coefficient is eight times the block average, the maximum tolerated difference is 16 for the DC coefficient. The expected bit difference in $b$, denoted by $d_b^H$, is

$$d_b^H \approx \begin{cases} \dfrac{\frac{q}{2} - 16}{1024} \times 64 \times 64 & \text{if } \dfrac{q}{2} > 16 \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The results for $P_{A-I}$, $d_{A-I}$, and $D_{map}$ are shown in Table II. These values are denoted by superscript "$H$" to distinguish them from the results of the IMAC without histogram transformation. Comparing Tables I and II, we find significant improvement due to histogram transformation. The expected difference barely differs from zero when $Q \ge 21$, which means the IMAC is robust under moderate compression.

TABLE III
PERFORMANCE OF THE APPROXIMATE IMAC WITH IMAGE TAMPERING

| $N_s$ | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|
| $d_b$ | 5 | 6 | 7 | 8 | 9 | 10 |
| $P_{A-I}$ | 0.2159 | 0.2371 | 0.2555 | 0.2716 | 0.2861 | 0.2993 |
| $d_{A-I}$ | 1.0795 | 1.4226 | 1.7885 | 2.1728 | 2.5749 | 2.9930 |
| $d_{map}$ | 1.1653 | 2.0238 | 3.1987 | 4.7211 | 6.6301 | 8.9580 |
| $D_{map}$ | 0.0233 | 0.0281 | 0.0327 | 0.0369 | 0.0420 | 0.0448 |

### B. Impact of Image Tampering

For the purpose of mathematical tractability, we assume that image tampering takes place in a square. The modification to each pixel in the square is random. Suppose the size of the square is $8N_t \times 8N_t$. The model is then further simplified so that the tampering takes place in a square of size $N_s \times N_s$, where $N_s \approx \sqrt{(1/2)}N_t$ so that $N_s \times N_s \approx (1/2)N_t \times N_t$. Furthermore, we assume that every bit of the coefficients in this $N_s \times N_s$ square changes because of tampering.

The performance of the approximate IMAC with tampering is shown in Table III. Note there is no difference in the results for the IMAC with and without image histogram transformation, because the effects of tampering are the same for both. $P_{A-I}$ is computed as a function of $N_s$ [14]. In Table III, $d_{A-I} = 64P_{A-I}$, $d_{map} = d_{A-I}^2$ and $D_{map} = d_{map}/(2N_s \times N_s)$.

To compare the impact of tampering with that of JPEG compression on the approximate IMAC without Image Histogram Transformation, we find the value of $d_{A-I}$ in Table III, when $N_s = 8$, is 2.1728, which is close to that of the compressed image with quality factor 50 in Table I. It shows that the approximate IMAC successfully suppresses the IMAC errors resulting from image compression, so it is now possible to set a threshold in the number of IMAC differences to separate image tampering and moderate image compression.

Compared with the values of $d_{A-I}$ in Table II, $d_{A-I}$ in Table III, when $N_s = 8$, is close to that of a compressed image with quality factor 16, so the performance of the IMAC with histogram transformation is further improved. A clear distinction between the compressed and tampered images is revealed.

The presence of one or more IMAC bit errors can be used as the indication of image tampering. For such an image authenticator, there are two types of error: "false alarm" and "miss" [18]. A "false alarm" occurs when a compressed image is reported as "tampered." Denote the values of $P_{A-I}$ and $P_{A-I}^H$ with JPEG compression using quality factors $Q$ by $P_{A-I}|_Q$ and $P_{A-I}^H|_Q$, respectively. For the IMAC without histogram transformation, the probability of "false alarm" is $1-(1-P_{A-I}|_Q)^L$, where $L$ is the IMAC length. $L = 128$ in the example. For the IMAC with histogram transformation, since $P_{A-I}^H|_{Q\geq 32}$ equal zero, the expected IMAC errors resulting from compression with $Q \geq 32$ is 0, the probability of false alarm is 0. When $Q < 32$, $P_{A-I}^H|_{Q<32} > 0$, and the probability of a "false alarm" is $1-(1-P_{A-I}^H|_Q)^L$. Fig. 11(a) compares the probability of "false alarm" versus different $Q$s between the IMAC without and with histogram transformation. The probability is obtained using the results in Tables I and II. Clearly, when $Q$ decreases, the probability of "false alarm" increases. The probability of a "false alarm" is greatly reduced by using histogram transformation.

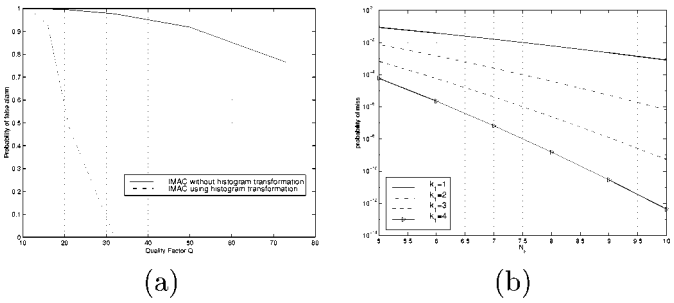


Fig. 11. (a) Probability of "false alarm" when an image is compressed by JPEG using different quality factor $Q$; (b) Probability of "miss" when the size of the tampered area varies.

On the other hand, if a tampered image is reported as acceptable, there is a "miss." Denote the value of $P_{A-I}$ by $P_{A-I}|_{N_s}$ when the tampered area is described by $N_s$. The probability of a "miss" is $(1-P_{A-I}|_{N_s})^{2N_s}$. The probability versus different values of $N_s$ is presented by the solid line in Fig. 11(b). (A more detailed discussion on Fig. 11(b) appears in the next section). In general, when $N_s$ increases, the probability of a "miss" decreases. For example, when $N_s = 8$, the probability of "miss" is 0.0063. $N_s = 8$ represents a tampered area that is $1/32$ of the total image size. The number of bit differences between the test image in Fig. 6(a) and tampered image in Fig. 6(b) fits the $N_s = 8$ model approximately.

### C. Impact of Gaussian Noise

Suppose Gaussian noise of zero mean and $v^2$ variance is added into the image. The expected absolute coefficient change is $\sqrt{2/\pi}\,v$. Denote the sum of the coefficient changes by SUM$d_B$. Since the DC coefficient is one eighth of the sum of the coefficients in the block, the absolute DC coefficient change is one eighth of $|\text{SUM}d_B|$. Using the central limit theorem, the distribution of SUM$d_B$ is approximated as a Gaussian distribution with zero mean and variance $64v^2$. The expected absolute value of SUM$d_B$ is therefore $8\sqrt{2/\pi}\,v$ and the DC coefficient change $\sqrt{2/\pi}\,v$ is expected. Given the expected absolute DC coefficient change, we derive the values of $d_b$ and $d_b^H$ when the variance of the Gaussian noise is varied. We found $d_b$ ranged from 2.3937 to 14.3622 and $d_b^H$ is zero when $3 \leq v \leq 18$.

### D. Simulation Results

The performance of the approximate IMAC without and with histogram transformation was examined by observing the number of IMAC differences in the rows and columns. The results shown in Tables IV and V were computed by averaging the data from 5000 times experiments. Results are reported for the test image in Fig. 6(a) with JPEG compression using different quality factors $Q$, the tampered image in Fig. 6(b), the tampered image with compression, and the test image with Gaussian noise of variance $v^2$. The observed differences in the rows and columns are denoted by row-d$_{A-I}$ and col-d$_{A-I}$, respectively. The superscripts "$c$," "$c/t$," "$t$," "$n$" are used

TABLE IV
OBSERVED BIT DIFFERENCES IN THE ROWS AND COLUMNS OF THE APPROXIMATE IMACS WITHOUT HISTOGRAM TRANSFORMATION

| $Q$ | 73 | 50 | 32 | 21 | 16 | 12 |
|---|---|---|---|---|---|---|
| row-d$^c_{A-I}$ | 0.8 | 1.5 | 1.8 | 2.3 | 2.6 | 3.6 |
| col-d$^c_{A-I}$ | 0.9 | 1.6 | 2.0 | 2.4 | 3.0 | 3.9 |
| row-d$^{c/t}_{A-I}$ | 2.5 | 3.1 | 3.2 | 3.5 | 4.1 | 4.8 |
| col-d$^{c/t}_{A-I}$ | 2.4 | 3.2 | 3.5 | 3.9 | 4.6 | 5.3 |
| row-d$^t_{A-I}$ | 2.1 | | | | | |
| col-d$^t_{A-I}$ | 2.3 | | | | | |
| $v$ | 3 | 6 | 9 | 12 | 15 | 18 |
| row-d$^n_{A-I}$ | 0.27 | 0.72 | 1.08 | 1.49 | 1.94 | 2.25 |
| col-d$^n_{A-I}$ | 0.31 | 0.75 | 1.14 | 1.59 | 2.15 | 2.53 |

TABLE V
OBSERVED BIT DIFFERENCES IN THE ROWS AND COLUMNS OF THE APPROXIMATE IMACS WITH HISTOGRAM TRANSFORMATION

| $Q$ | 73 | 50 | 32 | 21 | 16 | 12 |
|---|---|---|---|---|---|---|
| row-d$^c_{A-I}$ | 0 | 0 | 0 | 0.3 | 1.1 | 2.0 |
| col-d$^c_{A-I}$ | 0 | 0 | 0 | 0.3 | 1.3 | 2.1 |
| row-d$^t_{A-I}$ | 2.1 | | | | | |
| col-d$^t_{A-I}$ | 2.4 | | | | | |
| row-d$^{c/t}_{A-I}$ | 2.1 | 2.1 | 2.1 | 2.4 | 3.3 | 3.8 |
| col-d$^{c/t}_{A-I}$ | 2.2 | 2.3 | 2.4 | 2.6 | 3.5 | 4.1 |
| $v$ | 3 | 6 | 9 | 12 | 15 | 18 |
| row-d$^n_{A-I}$ | 0 | 0.02 | 0.17 | 0.42 | 0.77 | 1.12 |
| col-d$^n_{A-I}$ | 0 | 0.02 | .19 | 0.45 | 0.83 | 1.24 |

to denote the results for compressed images, tampered and compressed images, tampered images, and images with added noise, respectively. In Figs. 12 and 13, the performance of the IMAC without and with histogram transformation is presented, respectively. The values of row-d and col-d are plotted versus the quality factor $Q$.

In Fig. 14, the performance of the approximate IMAC without and with histogram transformation is compared when Gaussian noise is added to the image, where row-d and col-d are plotted versus the standard deviation of Gaussian noise. We see the approximate IMAC is robust when $v$ is small ($v < 6$). The number of IMAC differences increases when $v$ increases, and the numbers do not follow our prediction well. The reason may be because using the absolute mean of Gaussian noise alone is not sufficient to estimate the bit differences in the message resulting from Gaussian noise.

## V. IMAC GENERALIZATION

This section discusses several methods designed to improve the security and the performance of the approximate IMAC.

First, considering the probabilistic nature of the AMAC, we can increase the number of AMAC bits for each row (column) of $b$. Instead of one AMAC bit, we output $k_1$ ($k_1 > 1$) AMAC bits for each row (column) by repeatedly calculating one AMAC bit for each row (column) $k_1$ times. The length of the combined IMAC is $k_1(w + h)$. Given $k_1$ and $N_s$, the probability that we miss detecting image tampering in the rows or columns is $(1 - P_{A-I})^{2k_1 N_s}$, where $P_{A-I}$ is the probability that a given IMAC bit changes. (We assume the AMAC bits are independent from each other.) When $k_1 = 4$ and $N_s = 8$, $P_{A-I} = 0.2716$

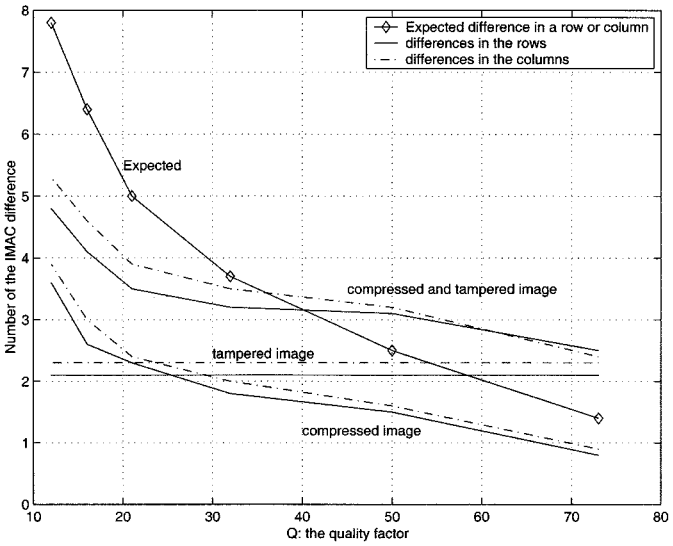


Fig. 12. Performance of the approximate IMAC without histogram transformation.

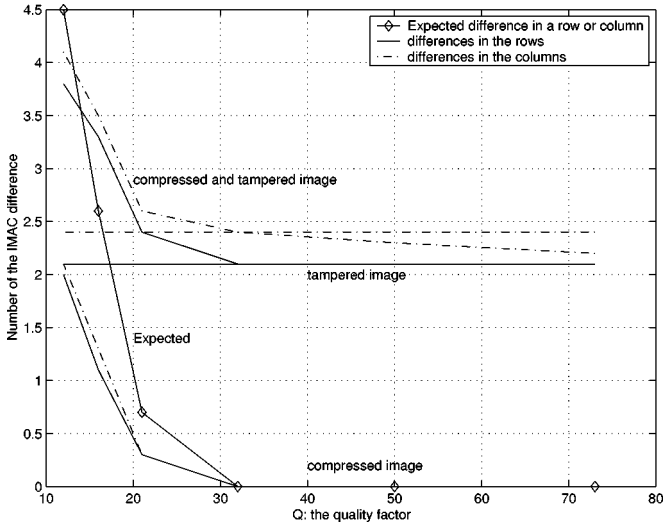


Fig. 13. Performance of the approximate IMAC with histogram transformation.

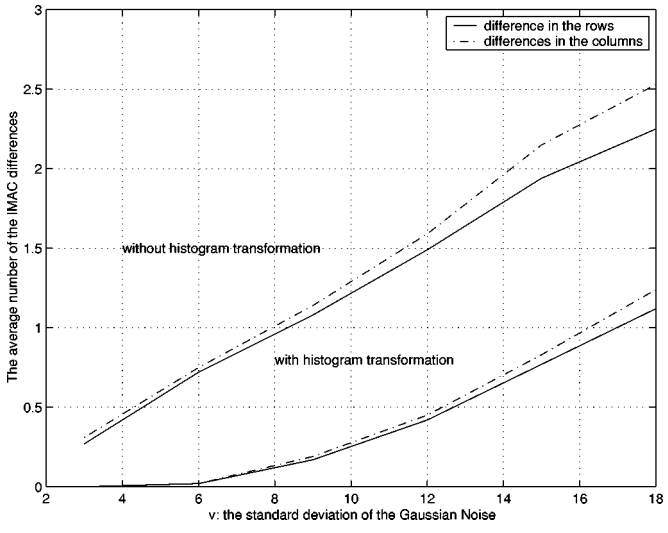


Fig. 14. Performance of the approximate IMAC with Gaussian noise.

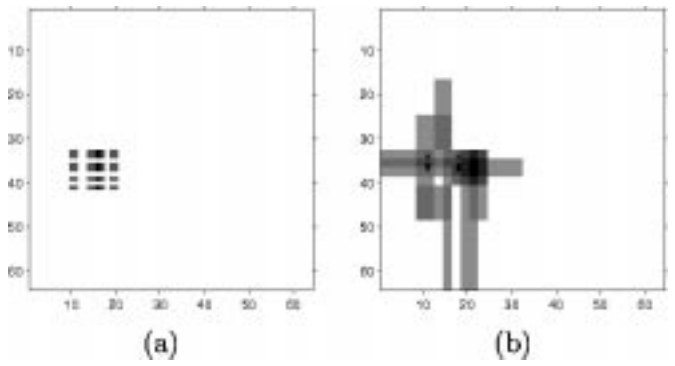


Fig. 15. Approximate IMACs of the original image in (a) of Fig. 6 and the tampered image in (b) of Fig. 6 are compared and a grayscale difference map is obtained. (a) For each row and column, four AMAC bits are computed. (b) Different block shapes, $1 \times 64$, $2 \times 32$, $4 \times 16$, $8 \times 8$, $16 \times 4$, $32 \times 2$, and $64 \times 1$, are used to compute the IMAC.

in Table III, and the probability of "miss" is $10^{-9}$. When $k_1$ increases, the approximate IMAC is more capable of detecting tampering. This is shown in Fig. 11(b), in which the probability of a "miss" for different values of $k_1$ is presented. In addition, because the probability that a given IMAC bit changes is almost zero when the image is compressed with $Q \geq 21$, we expect these IMACs will tolerate moderate compression well.

In addition, a grayscale difference map $D_{i,j}$ rather than a binary image can be constructed from $d = A' \bigoplus a'$. We set each pixel of $D_{i,j} = 255$, and then decrement each pixel of $D_{i,j}$ by $\lfloor 255/k_1 \rfloor$ for each corresponding 1 bit in $d$. The result is that the pixels of $D_{i,j}$ that correspond to altered bits of the IMAC are increasingly shaded according to the number of altered IMAC bits to which they are associated. Fig. 15(a) shows the difference map we obtained for the test images when four AMAC bits were calculated for each row and column.

Secondly, in the construction of the IMAC, we calculate one AMAC bit for each row and column in $b$. In other words, size $1 \times w$ and $h \times 1$ blocks are used. In this way, the intersection of one altered row bit and one altered column bit in the IMAC $A$ identifies one specific $8 \times 8$ block in $B_{i,j}$. Other shapes are also feasible, and they may intersect in different ways or not at all. In our example, size $1 \times 64$ and $64 \times 1$ blocks were used for a $64 \times 64$ binary map. In fact, the IMAC $A$ can be constructed, for example, from seven sets of 64, where each set consists of nonoverlapping blocks of shapes $1 \times 64$, $2 \times 32$, $4 \times 16$, $8 \times 8$, $16 \times 4$, $32 \times 2$, and $64 \times 1$, respectively. The combined approximate IMAC $A$ then has a length of $7 \times 64 = 448$ bits. When the IMAC is constructed in this manner, the values of $P_{A-I}$ are the same as before with respect to JPEG compression. When the image is tampered, the values of $P_{A-I}$ for block types other than $1 \times 64$ and $64 \times 1$ are likely to be higher than those of block types $1 \times 64$ and $64 \times 1$ in places where tampering takes place, because the fraction of the tampered coefficients in one block may be higher. As a result, the overall performance of the approximate IMAC is enhanced. Similarly to the last idea, a grayscale difference map can be constructed as shown in Fig. 15(b).

Thirdly, as the approximate IMAC is described above, an attacker with an image and its associated approximate IMAC can substitute any other image, so long as all $8 \times 8$ blocks $B_{i,j}$ with mean pixel value brighter (darker) than the threshold $T_{i,j} = 255/2$ remain brighter (darker). By introducing some uncertainty about where $T_{i,j}$ is for each $B_{i,j}$, the attacker's task appears to become much more difficult. Threshold values too close to 0 or 255 may not be helpful, so one possibility is, instead of fixing $T_{i,j} = 255/2$, pick each $T_{i,j}$ uniformly from the set of values $T/2$, where $T$ is an odd integer and, for example, $63 \leq T \leq 447$. Alternatively, each $T_{i,j}$ can be chosen from a suitable Gaussian distribution. The other concern on $T_{i,j}$ is to reduce $|T_{i,j} - B_{i,j}|$. When $|T_{i,j} - B_{i,j}|$ is smaller, it is more difficult for the attackers to generate a false $B_{i,j}$ that the approximate IMAC will miss to detect. In a more elaborate design, the value of $T_{i,j}$ should be adaptive to the value of $B_{i,j}$. For example, if the area is very bright, i.e., $B_{i,j}$ is close to 255, a large $T_{i,j}$ is preferred. When such an adaptive scheme is used to decide the value of $T_{i,j}$ for each $B_{i,j}$ at the sender, the correct value of $T_{i,j}$ needs to be retrieved based on the modified $B_{i,j}$ in order to locally generate the approximate IMAC at the receiver.

Finally, the latter two ideas just discussed, varying the threshold and using a larger set of nonoverlapping shapes for the IMAC calculations, are not consistent with the construction of the error tolerance zone of size $2T_d + 1$ around the threshold. The intervals around the various thresholds for the set of shapes to which a block $B_{i,j}$ belongs could border on each other in an unfortunate way and force an unnaturally large adjustment to the pixels of $B_{i,j}$ to construct the error tolerance zone around each different threshold. If $T_d$ and the number of shapes to which a block belongs are too large, this may becomes a severe problem, but for small values of $T_d$, for example, $T_d < 6$, the following simple rule ensures that the thresholds are either identical or have sufficient space between them to allow the pixels of $B_{i,j}$ to be adjusted by the minimum amount required by $T_d$. Instead of choosing each $T_{i,j}$ uniformly from the set of values $T/2$, where $T$ is an odd integer and $63 \leq T \leq 447$, constrain $T_{i,j}$ to the set $\{\lfloor (63 + 32i)/2 \rfloor | i = 0, \ldots, 12\}$.

## VI. Conclusion

In this paper, we proposed a new approximate IMAC for soft image authentication. The approximate IMAC is based the AMAC and it was developed for error detection and localization in images. Theoretical analysis was performed on the expected performance of the approximate IMAC when images were modified in several scenarios. Simulations were also carried out to compare and help validate the analysis. It was seen that the approximate IMAC successfully suppresses the impact of JPEG compression but preserves the ability to detect image tampering.

## Appendix
## Validation of Triangular Distribution of DC Coefficients

A group of images was collected to examine the effectiveness of the distribution model that we used to estimate the bit errors of the input sequence for the IMAC algorithm. In the simulations, 1000 nonsynthetic images were used, which included images provided in [19] and some widely available images such as Lena, Babara, Boat, Pepper, etc. The results reported here are the average of the results of all test images.

TABLE VI
OBSERVED NUMBERS OF BIT DIFFERENCES IN $b$ WITHOUT AND WITH IMAGE HISTOGRAM TRANSFORMATION

| $Q$ | 73 | 50 | 32 | 21 | 16 | 12 |
|---|---|---|---|---|---|---|
| $d_b$ | 17.6 | 31.5 | 50.3 | 72.0 | 97.5 | 127.1 |
| $d_b^H$ | 0 | 0 | 0 | 7.8 | 34.5 | 65.2 |

The assumption of triangular distribution of DC coefficients in their range was applied in the derivation of Equations (4) and (5). In Table VI, the observed number of bit differences in $b$ by compression is shown, where $d_b$ and $d_b^H$ are denoted as the observation on images without and with the histogram transformation, respectively. The results match the predicated values in Tables I and II very well. Hence, Equations (4) and (5) are good approximation for estimating the bit differences in $b$ resulting from compression.

## ACKNOWLEDGMENT

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

## REFERENCES

[1] A. Menezes, P. Van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1996.
[2] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code*, 2nd ed. New York: Wiley, 1996.
[3] R. L. Rivest, "The MD5 message-digest algorithm,", Internet Request for comments 1321, April 1992.
[4] NIST, "Data Engcryption Standard,", FIPS PUB 46-3, Nov. 1999.
[5] NIST, "Computer Data Authentication,", FIPS PUB 113, May 1985.
[6] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed hashing for message authentication,", Internet Request For Comments 2104, February 1997.
[7] W. Stallings, *Cryptography and Network Security*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1998.
[8] G. L. Friedman, "The trustworthy digital camera: Restoring credibility to the photographic image," *IEEE Trans. Consumer Electron.*, vol. 39, Nov. 1993.
[9] S. Walton, "Image authentication for a slippery new age," *Dr. Dobb's Journal*, vol. 18–26, pp. PAGE NOS?–, Apr. 1995.
[10] M. Schneider and S.-F. Chang, "A robust content based digital signature for image authentication," in *Proc. IEEE ICIP*, Laussane, Switzerland, Oct. 1996.
[11] J. Fridrich, "Methods for tamper detection in digital images," in *Multimedia and Security Workshop at ACM Multimedia*, Orlando, FL, Oct. 1999.
[12] L. Xie and G. R. Arce, "Image enhancement toward soft image authentication," in *Proc. IEEE ICME*, vol. 1, New York, Aug. 2000.
[13] R. F. Gravemen and K. Fu, "Approximate message authentication codes," in *Proc. 3rd Annual Fedlab Symp. Advanced Telecommunications/Information Distribution*, vol. 1, College Park, MD, Feb. 1999.
[14] G. R. Arce, L. Xie, and R. F. Gravemen, "Approximate image authentication codes," in *Proc. 4th Annual Fedlab Symp. Advanced Telecommunications/Information Distribution*, vol. 1, College Park, MD, Mar. 2000, http://www.ee.udel.edu/~xie/paper/amac.ps.
[15] F. Gray, "Pulse code communication," U. S. Patent 2 632 058, Mar. 1953.
[16] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM 31-44 34*, Apr. 1991.
[17] Independent JPEG Group, , http://www.ijg.org/.
[18] L. L. Scharf, *Statiscal Signal Processing—Detection, Estimation, and Time Series Analysis*. Reading, MA: Addison-Wesley, 1991.
[19] J. Z. Wang, G. Wiederhold, O. Firschein, and S. X. Wei, "Content-based image indexing and searching using Daubechies' wavelets," *Int. J. Digital Libraries*, vol. 1, no. 4, pp. 311–328, 1998.

**Liehua Xie** was born in Hunan, China. She received the B.S. degree in electrical engineering from Beijing University, Beijing, China, in 1995, and the M.S. and Ph.D. degrees in electrical engineering from the University of Delaware, Newark, in 1998 and 2000, respectively.

From September 1996 to June 2000, she was a Research Assistant in the Department of Electrical and Computer Engineering, University of Delaware. She has consulted with industry in the areas of image and video processing. Her research interests include image and video processing, image authentication, and secure multimedia communication.

**Gonzalo R. Arce** ('82–SM'93–F'00) was born in La Paz, Bolivia. He received the B.S.E.E. degree (with highest honors) from the University of Arkansas, Fayetteville, in 1979, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1980 and 1982, respectively.

Since 1982, he has been with the Department of Electrical and Computer Engineering, the University of Delaware, Newark, where he is currently Professor and Chair, and a Fellow in the Center for Advanced Studies. He has consulted for several industrial organizations in the general areas of signal and image processing and digital communications. His research interests include robust signal processing and its applications, communication theory, image processing, and electronic imaging. He holds two U.S. patents.

Dr. Arce has served as Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING, as Guest Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, and Guest Editor for the Optical Society of America's *Optics Express*. He is a Senior Editor of EURASIP's *Applied Signal Processing Journal*. He is a member of the Digital Signal Processing Technical Committee of the Circuits and Systems Society, and is a member of the Board of Nonlinear Signal and Image Processing.

**Richard F. Graveman** received the M.S. degree in computer science from Stevens Institute of Technology, Hoboken, NJ.

He is Director of the Security Research Group, at Telcordia Technologies, Inc., Morristown, NJ, and is currently P.I. for the U.S. Army Research Laboratories Fed Labs project on multimedia security and works on standards for ATM and optical network security. He chairs the ATM Forum Security Working Group. He has lectured on cryptography, digital watermarking, secure e-mail, single sign-on, broadband security, IP security, ATM security, and Internet firewalls. He has authored security and privacy assessments for NREN and OTA, served as IEEE JSAC Guest Editor, and served on Program Committees for ACM CCS and ISOC NDSS Conferences. He was Chair for the Fourth ACM CCS Conference and General Chair of Crypto'96. He has covered security literature and conferences for Security Reviews, SIGSAC Review, and IEEE Cipher, and was an adjunct faculty member in Computer and Information Sciences at New Jersey Institute of Technology from 1980 through 1986.

Dr. Graveman is a member of the ACM and IACR.