Chapter 5
Fuzzy Image Authentication with Error Localization and Correction

Obaid Ur-Rehman and Nataša Živić

5.1 Introduction

During the transmission of multimedia data from source to sink, multiple elements may contribute to data alteration. These elements typically include quantization mechanisms inside the source encoder (or decoder), compression mechanisms, such as lossy compression, and the channel noise, such as the one induced by a wireless medium. In order to cope with such forms of data modifications, communication systems typically employ error correcting codes, which add protection (parity) bits to the original data in such a manner that some or all of the modifications (or errors) in the data can be corrected. Some of the most widely used error correcting codes include Reed–Solomon (RS) codes [1], turbo codes [2] and low density parity check (LDPC) codes [3]. These error correcting codes are good at correcting a certain predefined number of bit modifications. However, if the number of errors exceeds the error correction capability of the codes, they are not correctable. In such a situation, different mechanisms are employed by the data transmission protocols for data recovery. Automatic Repeat reQuest (ARQ) [4] (or its variant called Hybrid-ARQ) is an example of such protocols, where the erroneous data is retransmitted up to a few times, until either the data is received as intended or a threshold number of retransmissions have been done. Most of these communication protocols use error detection codes, such as cyclic redundancy code (CRC) [5] or a message authentication code (MAC) [6, 7], to verify the integrity of data. Hashed message authentication code (HMAC) [8], such as SHA-2, is used as a means to authenticate the origin of the data in addition to its integrity.

Standard MAC and HMAC algorithms are designed such that even a single bit modification in the data changes its MAC by almost 50 %. These authentication algorithms are "hard authentication" algorithms, where the authentication decision is a strict "NO" when the data or its MAC has changed even by a single bit. In such

O. Ur-Rehman (✉) · N. Živić
Chair for Data Communications Systems, University of Siegen, 57076 Siegen, Germany
e-mail: obaid.ur-rehman@uni-siegen.de; natasa.zivic@uni-siegen.de

N. Živić (ed.), *Robust Image Authentication in the Presence of Noise*,
DOI 10.1007/978-3-319-13156-6_5

a case, the transmitted image (or a videoframe) will not be accepted as authentic at the receiver. In case of multimedia, such as an image or a videoframe, a few bit modifications might not have any visual impact on the content of the multimedia due to the nature of multimedia and the human visual perceptual system. For example, a few modifications in the least significant bits of an image data might not change the content and a bare human eye might not be able to differentiate between the original and modified image. If it is not possible to perform retransmissions using (H-)ARQ, e.g., either there is no feedback channel or a real-time communication is intended, or if the number of errors exceeds the error correction capability of the channel codes, the erroneous multimedia content will usually be discarded if hard authentication algorithms are used. In such a case, the receiver might interpolate or extrapolate the data, e.g., in a video stream the previous frame might be re-played or a blank frame might be inserted between the previous and the next frame. This is due to the fact that in some applications, it is be better to have partial data than no data at all.

In order to solve this issue, fuzzy authentication algorithms [9] have been introduced in literature. Primarily, these algorithms include approximate message authentication code (AMAC) [10], image message authentication code (IMAC) [11] and noise tolerant message authentication code (NTMAC) [12]. Some fuzzy as well as standard hard authentication algorithms, tailored specifically for multimedia data, have also been proposed [13–16]. In most of the cases, these algorithms work in little or no coordination with the other modules of the communication systems. This means normally the source coding module has little to do with the cryptographic module and the cryptographic module performs in little coordination with the channel coding. Recently, it has been shown that a better coordination between the different components of communication system improves the channel coding results as well as the authentication results [17]. Some recent fuzzy authentication algorithms, where different components of a communication system interact for authentication as well as error localization and recovery from some errors, are weighted noise tolerant authentication code (WNTMAC) [15], error correcting noise tolerant message authentication code (EC-NTMAC) [18, 19]. In WNTMAC, a different weight is assigned to important and non-important parts of a message. Important parts are provided more protection than the non-important parts. More efforts are spent on the important blocks to identify and correct errors, whereas lesser efforts are spent on the data parts of lesser importance. In EC-NTMAC, the NTMAC algorithm for fuzzy authentication is extended and error correction capability is integrated into the fuzzy authentication algorithm to do error correction in addition to fuzzy authentication. Applications in image authentication together with simulation results are given in [18, 19].

This chapter is organized as follows. In Sect. 5.2, a brief overview of the building blocks used in the fuzzy image authentication algorithms is given. This includes the frequency domain transformation, such as discrete cosine transform and error correcting codes such as turbo codes. In Sect. 5.3, applications of error correcting codes in image authentication are discussed. In Sect. 5.4, two fuzzy authentication algorithms for image authentication based on the integration of error correcting codes are discussed. The performance and security analysis of these algorithms is given in Sect. 5.5. Finally, simulation results are shown in Sect. 5.6 to show the performance.

5.2 Building Blocks of the Fuzzy Image Authentication Algorithms

5.2.1 Content Based Authentication

In order to have the capability to authenticate multimedia data, such as images or videoframes (this chapter discusses only image authentication from this point onward), despite certain changes in the original data, it is advantageous to base the authentication on the content rather than the actual data. The actual data of an image, such as the bit or byte values might change due to quantization, compression, etc. The features of an image are, however, not changed by small modifications. Image authentication based on its content or distinct features, rather than the actual data, is called content-based authentication. Different feature extraction techniques have been proposed in literature by the image processing community. The features can be as simple as the image contour, edges, Fourier, wavelet, or cosine transform coefficients or more complex features such as those based on the image textures. The fuzzy authentication algorithms discussed in this chapter are based on the frequency domain features obtained using the discrete cosine transform (DCT) [22] of an image.

5.2.2 Discrete Cosine Transform

DCT [22] is one of the most widely used techniques in image processing. Other widely used transforms are the discrete Fourier transform (DFT) and the discrete Wavelet transform (DWT). Chen and Pratt [20] pioneered the application of DCT in image processing. DCT removes correlation from image data, after which each transform coefficient can be encoded independently without devitalizing compression efficiency [21]. Since DCT combines most of the energy in a few transform elements, it is preferred as compared to DFT.

DCT of a matrix (called a block in image processing terminology) is defined as follows [22, 23]:

$$X(l,k) = \alpha(l)\alpha(k)\frac{2}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)\cos\left[\frac{\pi(2m+1)}{2N}\right]\cos\left[\frac{\pi(2n+1)}{2N}\right]. \quad (5.1)$$

where,

$$\alpha(i) = \begin{cases} 1/\sqrt{2}\,, & i = 0 \\ 1\,, & 1 \le i \le N-1 \end{cases}$$

It can be observed from (5.1) that the element at the index (0, 0) of the DCT (called DC element) represents the average intensity of the corresponding block and contains most of the energy and perceptual information. Elements at the indexes other than

(0, 0) are known as AC elements. AC elements in the upper minor diagonals have more information as compared to the AC elements in the lower minor diagonals. If the DC components together with a reasonable number of AC components are retained, e.g., by passing the DCT elements through a low-pass filter, the inverse transform of the DCT (IDCT) can reconstruct the original block with some or no loss in quality. By increasing the number of AC components, the loss in the quality of reconstruction decreases.

The inverse of a 2-D DCT for l, k = 0, 1, . . . , N − 1 can be calculated as follows,

$$x(l,k) = \alpha(l)\alpha(k)\frac{2}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} X(m,n)\cos\left[\frac{\pi(2m+1)}{2N}\right]\cos\left[\frac{\pi(2n+1)}{2N}\right] \quad (5.2)$$

5.2.3 Error Correcting Codes

5.2.3.1 Error Correcting Codes in Image Authentication

Error correcting codes, such as RS codes, turbo codes, or LDPC codes, are used to detect and correct the errors introduced in the data by different means, such as by the transmission noise over a communication channel or during storage over a storage medium. The choice of error correcting codes depends on the noise environment and the desired error correction capability. RS codes are good at correcting burst errors in addition to the sporadic errors and work more naturally on symbols. LDPC codes and turbo codes operate on large data sets such as multimedia data and their error correction capability has been demonstrated to be very closer to the Shannon limit [24]. Error correcting codes, when combined with image authentication, increase the tolerance of the authentication algorithm. They also help in identification of the error location and correcting the errors if they are within the error correction capability. Mostly such errors are due to unintentional modifications and therefore in a small number. In this section, a short introduction of some of most widely used error correcting codes is given. These codes are also used in the algorithms discussed next and therefore, they are introduced here before their application is discussed.

5.2.3.2 Reed–Solomon Codes

RS codes were proposed by Reed and Solomon in their legendry paper [1]. RS codes are nonbinary cyclic codes. RS codes work on symbols rather than bits, where each symbols is a Galois Field (GF) element. Each symbol is made up of m-bits, such that m is an integer and usually $2 \leq m \leq 8$. The m-bit elements are defined over the Galois Field GF(q), where $q = p^m$ and p is a prime number typically chosen to be 2. RS codes are linear block codes, which mean that they encode data in blocks. An RS(n, k) code takes a data block of k symbols and protect it by appending n − k parity symbols to produce a block of n symbols.

Here,

$$0 < k < n < p^m + 2 \tag{5.3}$$

The error correcting capability of RS codes is measured in symbols instead of bits. Thus, a burst of bit errors can be grouped in to fewer symbols and then corrected as symbols. The choice of the number of parity symbols dictates how many errors should be corrected by the RS(n, k) codes. If t symbol errors are desired to be corrected by the RS(n, k) code, then $n - k$ must be chosen such that $2t \leq n - k$.

RS codes are also good at correcting erasures, where erasures are the errors whose positions are known in advance via some a priori knowledge. The joint error and erasure correction capability of RS codes is expressed as,

$$2N_e + N_r \leq n - k \tag{5.4}$$

where N_e is the number of errors and N_r is the number of erasures. (5.4) means that if t symbol errors are to be corrected (i.e., $N_e = t$ and $N_r = 0$), then 2t parity symbols are needed. If only erasure correction is desired, then erasures equal to twice the number of errors are correctable, i.e., up to 2t erasures can be corrected.

The decoding algorithm for RS codes can either be a hard or a soft decision decoder. Hard decision decoders are the ones that operate on discrete values. Most widely used hard decision decoding algorithms for RS codes are Berlekamp–Massey [25] algorithm and the Euclidean algorithm [26]. Soft decision decoders, on the other hand, operate on continuous values or floating point numbers. They usually take soft values at the input and generate soft values at the output (soft input soft output (SISO) decoders). However, in general they can also generate only hard output. Well-known soft decision decoding algorithms for RS codes are Guruswami–Sudan [27] decoding algorithm and the Koetter-Vardy [28] decoding algorithm. A more generalized approach is to output a list of codewords when it is not possible to decide on one codeword, e.g., multiple codewords are equiprobable to have been transmitted, given the received word. This approach is called list decoding [29].

Since the hard decision decoders operate on the quantized data, where some information is lost during quantization, their decoding performance is typically not as good as that of the corresponding soft decision decoder [9].

5.2.3.3 Turbo Codes

Turbo codes, proposed by Berrou, Glavieux, and Thitimajshima in 1993, were the first practical codes shown to approach the channel capacity [2]. Turbo codes are theoretically a concatenation of two codes connected by an interleaver. In practice, the constituent codes are usually convolutional codes connected in parallel. For encoding, the input data is interleaved and passed through the encoders. Without loss of generality, it can be said that the input data is passed through an identity interleaver and then through the first encoder and in parallel the input data is passed through another (nonidentity) interleaver before the second encoder. The result is the original data and two encoded sequences or the parity bits.

The decoder for turbo codes is usually a soft decision iterative decoder. The decoder's task is to find the most probable codeword based on the demodulated received sequences and a priori knowledge or probabilities about the messages and their occurrence. The decoding in turbo codes is performed by two decoders, each one capable of working on soft input and producing soft output—or the likelihood information about the output bits. The most common SISO decoding algorithms are soft output viterbi algorithm (SOVA) and the maximum a posteriori probability (MAP) decoding algorithm. The decoding is performed in turns by the constituent decoders. It starts with one decoder and the output of the decoder is an estimate of the first codeword. This estimate is used as a priori information in improving the decoding results of the second decoder, which works on the second encoded sequence and the interleaved output of the first decoder. The used interleaver is the same as the one used in the encoding. This decoding process is repeated iteratively, each time improving the confidence level in the decoded sequence and increasing the reliability of the decoded bits. Repeating the iterations many times would result in a convergence to a final decision about the most probable codeword.

The likelihood of the input and the output bits in the decoders are normally represented as log likelihood ratios (LLRs). The LLR of each bit is based on the a priori probability of the bit and the current observation of the bit. If the sequence to be decoded is $x = (x_1, x_2, x_3, \ldots, x_n)$, then the LLR of each bit x_i can be represented as,

$$LLR(x_i) = \log \frac{\Pr(x_i = 1, observation)}{\Pr(x_i = 0, observation)} \tag{5.5}$$

It can be noticed from (5.5) that the LLR of a bit is based on the a priori probabilities of that bit and the current observation of the bit value.

5.3 Applications of Error Correcting Codes in Image Authentication

In practice, it is important to not only authenticate an image and verify its integrity but also to locate any modifications in the image. Existence of modifications can be detected using cyclic redundancy check (CRC) [5], MACs [6, 7], digital signatures or digital watermarking [30]. Additionally, it is also very important to locate the exact positions of modifications, to as fine grained level as possible, to find out which objects in the image have changed. However, recently it is getting important to look for methods to do error correction in addition to error detection and localization. Error correcting codes, such as RS and turbo codes, have been used in image authentication algorithms to get an additional error localization and correction capability. They have been widely used in detecting errors in the image, isolating the (potentially) erroneous locations and if possible correcting those (potentially) erroneous parts of the image. When combined with the authentication mechanisms, the error correcting codes can help in partial image recovery even if complete image recovery is not possible. This

is particularly helpful in multimedia streams, where a partially authentic image (or part wise authentic image) might be more useful than having no image at all.

In [11], approximate image message authentication codes (A-IMACs) approach for fuzzy image authentication is proposed. The proposed A-IMAC is tolerant to small image modifications but at the same time it is capable of detecting and locating intentional tampering. A-IMAC is based on different composite techniques such as block smoothing, block averaging, parallel AMAC and image histogram enhancement. The performance of A-IMAC in discussed in [11] for three image modification scenarios, i.e., JPEG compression, image forgery, and additive Gaussian noise.

Digital watermarking methods for image authentication with error detection and reconstruction, based on error correcting codes, have also been proposed in literature [31–33]. In [31], RS codes are used to generate parity symbols for each row and column of an image. The resultant parity symbols (as bits) are then embedded as a watermark in the two least significant bit planes of the image. If the watermarked image changes, such that the RS decoder can correct the changes, they are corrected to restore the original image data. A scrambling method is used such that a burst of data modifications is transformed into random noise.

In [32], a watermarking approach is proposed in which the image is divided into blocks and the block hash is encoded with a systematic error correcting code. The parity symbols are then embedded into the blocks. During verification, the hash of each block is recovered with the embedded parity symbols, if the number of tampered blocks does not exceed a threshold value. The size of parity symbols is smaller than the total size of block hashes.

In [33], two techniques for self-embedding an image in itself are introduced with the aim to protect the image content. The self-embedding helps in recovering portions of the image which are cropped, replaced, damaged, or tampered in general. In the first method, the 8×8 blocks of an image are transformed into frequency domain using DCT and after quantizing the coefficients they are embedded in the least significant bits of other, distant blocks. This method provides a good quality of reconstruction but it is very fragile and can be classified as fragile watermarking technique [30]. The second method is based on the principle similar to differential encoding where a circular shift of the original image with decreased color depth is embedded into the original image. It is shown that the quality of the reconstructed image degrades with the increasing level of noise in the tampered image. This method can be classified as a semi-robust watermark [30].

5.4 Fuzzy Image Authentication Codes with Error Localization and Correction

5.4.1 Fuzzy Authentication Based on Image Features

The algorithms introduced in this section (for error correction and soft authentication) are based on image features extracted in the transform domain. The aim is to

authenticate the images in the presence of minor modifications and to be able to locate and correct those modifications before declaring the image authentic or unauthentic. Another aim is to be able to locate major modifications in the images protected with the proposed fuzzy authentication algorithms. The features extracted from images are protected by error correcting codes so that minor changes in the features can be reconstructed in case of noise or distortions. This means that as long as the features are correct or correctable, the image is considered authentic, but if the features are distorted to an extent that the reconstruction is not possible, then the image is declared unauthentic. There are many other ways of capturing the features of an image, such as those based on edge detection to capture the contours of image objects, corner detection, blob detection, affine invariant features, image gradient-based features, etc. [34].

The features based on the frequency domain transform, such as DFT, DWT, or DCT are usually extracted by splitting the image into equal sized smaller blocks. Typical dimensions of a block used in practice are 8×8 pixels, such as in JPEG and JPEG2000 [35, 36]. Then the desired transform is applied on image block by block. DCT is used in this chapter for frequency domain transformation. The high energy components in the DCT of a block capture the most essential features. Thus, if a "reasonable number" of high energy components are retained, the block can be reconstructed through the inverse DCT (IDCT). This reasonable number determines the quality of reconstruction. If the number of retained components is too small, the quality of reconstruction is not satisfactory. If the number of retained components is increased, the quality of reconstruction is improved. This number also determines the quality of compression and decompression and is represented through the different quantization matrices in JPEG2000.

5.4.2 Image Error Correcting Column-Wise Message Authentication Code (IECC-MAC)

The algorithms for error correction and fuzzy authentication of images introduced in this chapter are based on DCT. The first algorithm is called IECC-MAC [18]. It protects the DC coefficients (as they are the most important) of an image by standard MACs and performs fuzzy authentication with error tolerance on the received (noisy) images. The algorithm is able to localize errors to a smaller part of the modified image and to correct a certain number of these modifications. The algorithms for sender and the receiver side are explained below. Definition 1 is essential to understand the concept of the proposed algorithms,

Definition 1 Suppose that a message M is transmitted over a noisy medium along with its n-bit MAC, denoted as H. Let M' be the received message and H' be the received MAC. Let H'' be the MAC recalculated on M' and let d be a small positive integer ($d \ll n$). M' is called d-fuzzy-authenticated if $HD(H', H'') \leq d$, where HD is the Hamming distance.

This method of fuzzy authentication, where the Hamming distance between H' and H'' does not have to be equal to zero for message acceptance, is based on the fact that the MACs of two different messages cannot be the same or even close to each other, in order to avoid near collision attacks [16, 37].

For the sake of simplicity, let's assume that an $N \times N$ pixel image has to be transmitted. For other dimensions, padding or row/column repetitions etc. can be used. Let the image be divided into $m \times m$ pixel blocks such that $m|N$, where both N and m are positive integers and m is typically equal to 8. DCT is calculated for each block and the DC components are chosen for protection, by storing them in a DC matrix corresponding to the DC elements of the whole image. This process is described in Fig. 5.1.

Algorithm: IECC-MAC Tag Generation

Input:

- Image I
- Image width W
- Image height H
- Block length m

Algorithm:

```
blocks = SplitInBlocks(I, W, H, m)

for i = 1 to W/m
    dcs_column_i = []
    for j = 1 to H/m
        dct = ComputeDCTOnBlock(blocks_i,j)
        dcs_column_i = dcs_column_i || dct_1,1
    end
    C-MAC_i = ComputeMAC(k_1, dcs_column_i)
end

IECC-MAC = C-MAC_1 || C-MAC_2 || .... || C-MAC_W/m
```

Output:

$NTMAC_{DC}$

Standard MAC is computed column-wise on the DC matrix (one MAC for each column of DC elements) to get a total of N/m MACs. These MACs are combined together to get a column-wise MAC (C-MAC) and transmitted together with the compressed image. A compression is achieved by keeping the DC coefficient and the first few minor diagonals of each DCT matrix, which can be chosen according

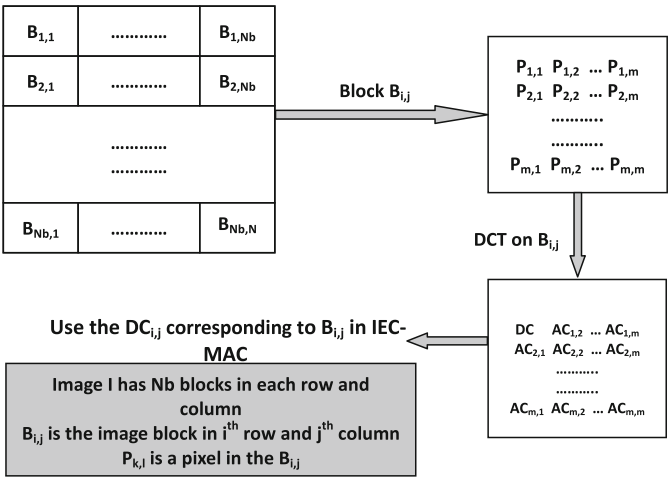

Fig. 5.1 Perform DCT block-wise and keep the DC element.

to the desired image quality as is done in JPEG [35]. An inverse DCT (IDCT) of each DCT matrix (with the DC element and reduced number of AC elements) gives a compressed image. The process of calculation of C-MACs and then combining them to get IECC-MAC is shown in Fig. 5.2.

The receiver receives the (potentially noisy) compressed image I′ and its (potentially noisy) C-MAC′. The receiver recalculates a C-MAC (denoted as C-MAC″) on the image I′using the same steps as above. C-MAC′ is then compared with C-MAC″ for each of the component MACs. If the Hamming distance between the recalculated MAC (denoted as H″) and the received MAC (called H′) of a column of DC elements is lower than d, i.e., if M′ is not d-fuzzy-authenticated according to definition 1, the corresponding column of blocks in the image I′is marked as suspicious, otherwise the column is declared to be authentic. The marked columns are called suspicious as they might potentially have errors. After the suspicious columns are marked, they are tried to be corrected using the bit reliability values or the LLRs of each bit.

The LLRs can be obtained from the SISO channel decoder, for example the MAP decoder for turbo codes. If no channel decoder is used, then the channel measurements are taken as the LLRs of each bit. Based on these LLRs and the received signal values, error correction is attempted. The process of error correction works as follows: The bits of the marked suspicious columns are sorted based on their absolute values of LLRs. A combination of least reliable bits is then flipped, followed by recalculation of the MAC (H″) on the DC components of the (bit-flipped) column and calculating its Hamming distance with H′. If the bit-flipped form of marked column is d-fuzzy-authenticated, the corresponding column of blocks in the received image (I′) is accepted as authentic; otherwise, the iterations are continued till a maximum (threshold) number of iterations have been performed. The error localization and correction is shown in Fig. 5.3.

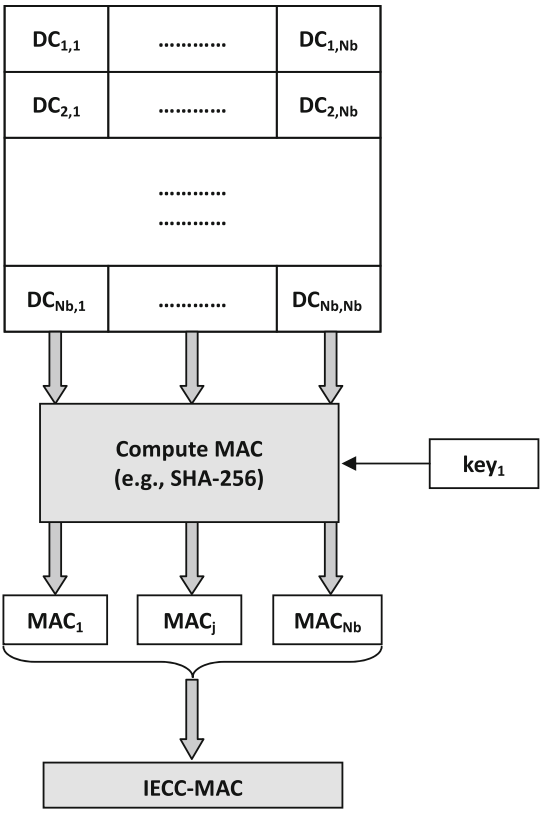

Fig. 5.2 IECC-MAC tag calculation on the DC elements of columns of blocks.

The threshold on the maximum number of iterations (T_{itr}) is predefined and is given by,

$$T_{itr} = 2^{\eta} \tag{5.6}$$

where η is a small positive integer (typically, $0 \leq \eta \leq 24$). T_{itr} can be made adaptive, by choosing the value of η depending on the bit error rate (BER) and also based on the application demand.

The error localization property of IEC-MAC algorithm needs to be further optimized because IEC-MAC localizes errors to the column level. It would be better to localize errors, correct them, and perform fuzzy authentication at the block level. Due to the errors being localized at the column level, extra processing for iterative error correction is required. To improve the error localization, another algorithm called the image error correcting-noise tolerant message authentication code (IEC-NTMAC) is introduced. IEC-NTMAC is based on the EC-NTMAC algorithm [15] and marks the image blocks as suspicious, as opposed to the columns of image blocks, which is done by IEC-MAC.

Algorithm: IECC-MAC Tag Verification

Input:

- Received Compressed Image I´
- Received IECC-MAC´
- Image width W and height H
- Block length m
- Block LLRs: blockLLRs

Algorithm:

```
blocks' = SplitInBlocks(I', W, H, m)
unauthentic_columns = []
dcs_columns'_i_LLRs = BlockLLRsToDCColumnLLRs(blockLLRs)
authentic = true

for i = 1 to W/m
    dcs_column_i = []
    for j = 1 to H/m
        dct = ComputeDCTOnBlock(blocks'_i,j)
        dcs_column'_i = dcs_column'_i || dct_1,1
    end
    C-MAC''_i = ComputeMAC(k_1, dcs_column'_i)
        if( HD(C-MAC´_i, C-MAC''_i) ≤ d )
            status = PerformErrorCorrection(i, j, dcs_column'_i,
        dcs_columns'_i_LLRs)
            if(status == decoding_failure)
                authentic = false
                unauthentic_columns = unauthentic_columns || i
            end
        end
end
```

Output:

```
If (authentic == true)
        verification_status = true
else
        verification_status = false
        return unauthentic_columns
end
```

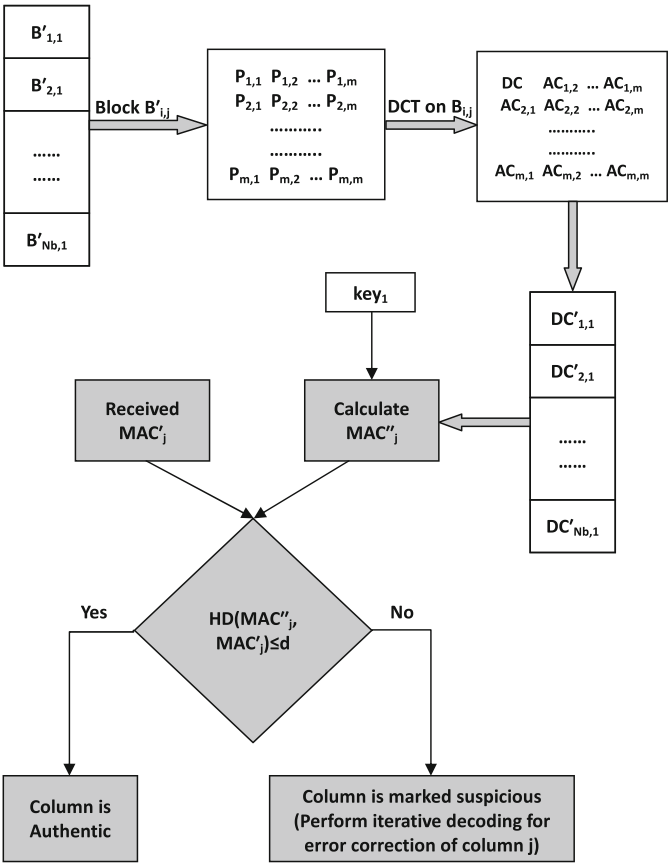

Fig. 5.3 Error localization and correction using IECC-MAC

5.4.3 Image Error Correction Noise Tolerant Message Authentication Code (IEC-NTMAC)

For error localization to a more fine grained level, an algorithm based on the EC-NTMAC [15] is proposed. It has an enhanced error localization and image recovery capability as compared to IEC-MAC.

The IEC-NTMAC algorithm works by calculating an NTMAC for each of the DC component in the DCT sub-matrix. Although a complete standard MAC is calculated for each DC component, only a small portion (called sub-MAC) is retained, e.g., the last s-bits of the MAC (see NTMAC and W-NTMAC [15]). All of these sub-MACs are appended together to constitute a complete NTMAC for each column of the DC matrix. Now the same step is repeated for all the columns, giving N/m NTMACs. Using an NTMAC not only improves the error localization to block level but also

improves the error correction. The reason is that smaller blocks can be corrected more efficiently (in terms of processing power and memory consumption) as compared to the whole column of blocks.

IEC-NTMAC generation for an image I is shown in Fig. 5.4.

Algorithm: IEC-NTMAC Tag Generation

Input:

- Image I
- Image width W
- Image height H
- Block length m

Algorithm:

blocks = SplitInBlocks(I, W, H, m)

for i = 1 to W/m
    for j = 1 to H/m
        dct = ComputeDCTOnBlock($blocks_{i,j}$)
        dc = $dct_{1,1}$
        $subMAC_{DCj}$ = ComputeSubMACOnDC(k_2, dc)
    end
    $NTMAC_{i,DC} = subMAC_{DC1} || subMAC_{DC2} || .... || subMAC_{DCH/m}$
end
$NTMAC_{DC} = subMAC_{1,DC} || subMAC_{2,DC} || .... || subMAC_{W/m,DC}$

Output:

$NTMAC_{DC}$

The receiver receives a compressed image (denoted as I′) as well as its IEC-NTMAC (denoted as IEC-NTMAC′). The receiver recalculates the IEC-NTMAC on I′ by using the same algorithm as explained above (let the recalculated IEC-NTMAC be denoted as IEC-NTMAC″). IEC-NTMAC′ is compared with IEC-NTMAC″ through a comparison of the corresponding sub-MACs. If the sub-MACs are d-fuzzy-authenticated according to definition 1, then the DC component is accepted as authentic and the image block corresponding to the DC component is declared authentic. Otherwise, if the sub-MACs are not d-fuzzy-authenticated, the block is marked as suspicious. All the blocks marked as suspicious identify the potential modifications in the image and are tried to be corrected using the iterative error correction algorithm explained in Sect. 2.2 on IEC-MAC. The major difference is that in Sect. 2.2, the whole suspicious column of image blocks is considered. Thus, it is good to identify the error locations to a column level but depending on the image resolution, it might be computationally very expensive to do the error recovery. In IEC-NTMAC, only the suspicious

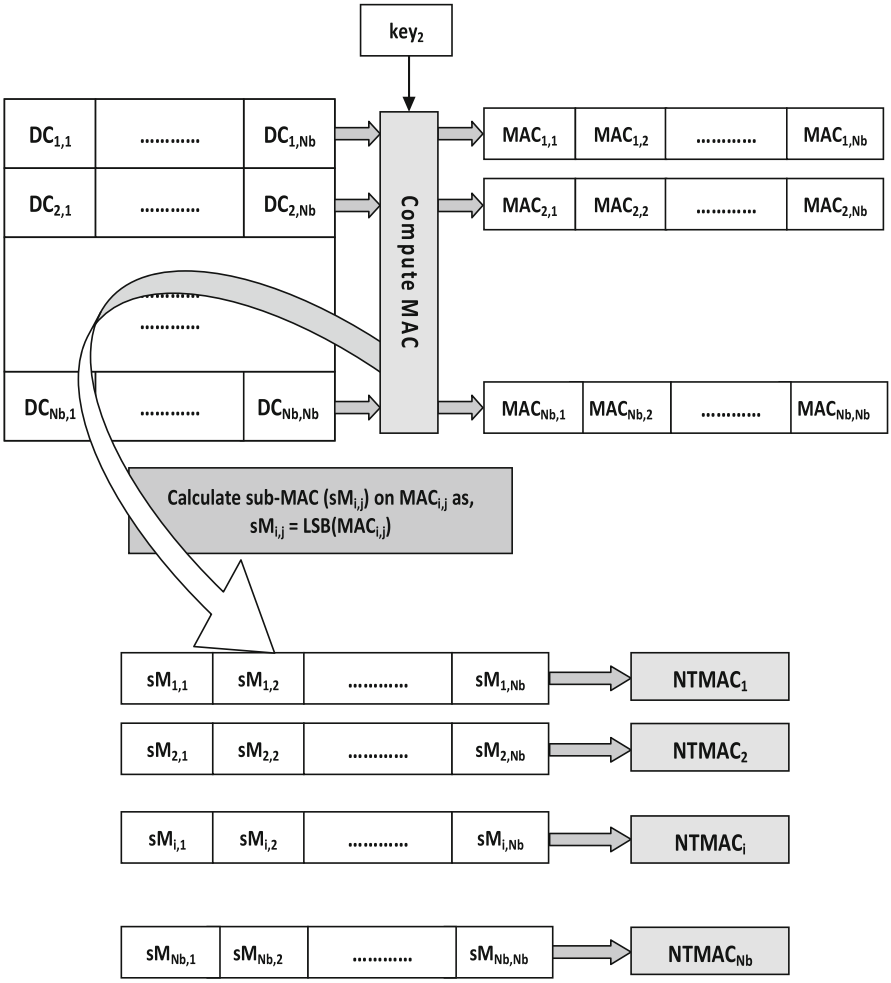

Fig. 5.4 Calculation of IEC-NTMAC at the sender side.

blocks are tried for error recovery, therefore the error recovery is computationally less expensive as compared to IECC-MAC. The pseudo-code of the IEC-NTMAC verification and error localization with error recovery at the receiver is given below and also depicted in Fig. 5.5.

Algorithm: IEC-NTMAC Tag Verification

Input:

- Received Compressed Image I´
- Received IEC-NTMAC´
- Image width W and height H
- Block length m
- Block LLRs: blockLLRs

Algorithm:

```
I´´ = DecompressImage(I´)
subMAC´DC = MakeSubMACsForDCs(NTMAC´)
dc_LLRs = BlockLLRsToDCLLRs(blockLLRs)
blocks = SplitInBlocks(I´´, W, H, m)
authentic = true
unauthentic_blocks = []

for i=1 to W/m
    for j=1 to H/m
        dct = blocks_{i,j}
        dc = dct_{1,1}
        subMAC_{DCi,j} = ComputeSubMAC(k_2, dc)
        if( HD(subMAC´_{DCi,j}, subMAC_{DCi,j}) ≤ d )
            status = PerformErrorCorrection(i, j, dc, dc_LLRs_{i,j})
            if(status == decoding_failure)
                authentic = false
                unauthentic_blocks = unauthentic_blocks || block_{i,j}
            end
        end
    end
end
```

Output:

```
If (authentic == true)
        verification_status = true
else
        verification_status = false
        return unauthentic_blocks
end
```

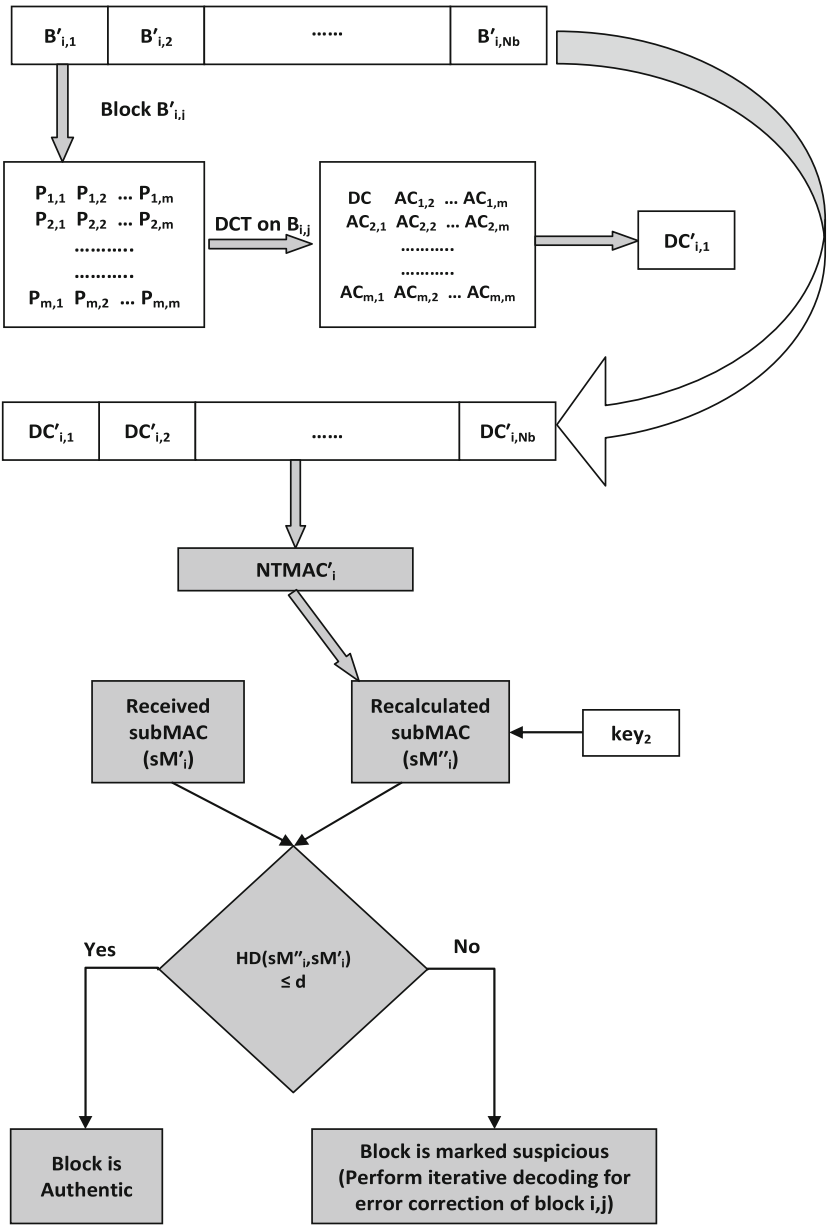

Fig. 5.5 Error localization and correction using IEC-NTMAC at the receiver side

5.5 Performance and Security Analysis of the Proposed Fuzzy Authentication Algorithms

5.5.1 Performance Study

To analyze the algorithms, it is assumed that an ideal n-bit MAC (n ≥ 256) is used for each column of the DC coefficients. The image is divided into $m \times$ m nonoverlapping blocks to get a total of m^2, k-bit DC coefficients.

At the receiver, the fuzzy authentication is considered to be successful if the Hamming distance between the received and the recalculated MACs does not exceed a threshold value. For such a fuzzy authentication algorithm, there is a possibility of the following two types of errors,

False Rejection A correct image (or a complete column of blocks or individual blocks) is discarded, though it should have been declared authentic.

False Acceptance An incorrect image (or a complete column of blocks or individual blocks) is accepted, though it should have been declared unauthentic.

If the BER is too high, false rejection will reduce the efficiency of the proposed schemes. When channel is in good state, i.e., the BER is low, false rejection will happen rarely. The probability of a false rejection for hard verification, i.e., with d = 0, is given by,

$$1 - (1 - BER)^n \tag{5.7}$$

The false rejection probability for fuzzy authentication is given by,

$$\begin{aligned} &\Pr(\textit{FALSE REJECTION}) \\ &= (1 - BER)^{mk} \sum_{i=d+1}^{mn} \binom{mn}{i} BER^i (1 - BER)^{mn-i} \end{aligned} \tag{5.8}$$

which is much smaller than that for hard authentication.

To calculate the probability of a false acceptance, suppose that R is the number of non-authentic column-wise MACs and E is the number of erroneous bits. The probability of a false acceptance in the presence of e erroneous bits in DC coefficient is given by,

$$\begin{aligned} &\Pr(\textit{FALSE ACCEPTANCE}) \\ &= \Pr(R = 0 \mid E = e) \\ &= \sum_{i=1}^{e} p_i q_i \end{aligned} \tag{5.9}$$

where p_i is the probability of a false acceptance in each column when i DC coefficients are erroneous and q_i is the conditional probability of i erroneous DC

coefficients when e bits are in error. p_i is approximated by binomial distribution while q_i can be estimated using classical occupancy problem [12–15]:

$$\Pr(\textit{FALSE ACCEPTANCE}) = \sum_{i=0}^{e} \left(\frac{\sum_{k=0}^{d} \binom{m}{k}}{2^n} \right)^i \binom{m}{i} \sum_{j=0}^{i} (-1)^j \frac{\binom{i}{j} (i-j)^e}{m^e} \tag{5.10}$$

5.5.2 Security Analysis

Algorithms introduced in this chapter are based on the ideal standard MAC, so the generic attacks on the standard MACs are considered as potential threats. In the given approaches, MACs may tolerate a modest number of errors; therefore, the security strength is reduced in general as compared to hard authentication MACs. In IECC-MAC, each DC element is protected by one MAC and the attacker has to forge the DC coefficients in such a way that column MACs can be d-fuzzy-authenticated. In IEC-NTMAC, the attacker even has more difficult task due to random partitioning [15].

A common approach to approximate the required complexity (data/time) for forgery attack on the MACs is given by "birthday paradox" which is based on finding collisions. For the fuzzy authentication, an attacker has to perform a near collision attack [37]. Near collision refers to a message pair, such that their MACs differ a little from each other. By extending the ordinary birthday paradox to the introduced fuzzy authentication scheme with a threshold d, it is expected to have a near collision with at most d-bit differences with the data complexity of,

$$\sqrt{\frac{2^n}{\sum_{d=0}^{d} \binom{n}{d}}} \tag{5.11}$$

The value of n is usually chosen to be 256 (bits). The threshold value is set in such a way that the false acceptance and false rejection rates are minimized. There are other experimental methods to find a conveniently safe threshold zone by image processing techniques [10]. It can be observed that with the smaller threshold values, the security strength can be compensated by selecting longer MAC lengths [38]. The security of IEC-NTMAC is even higher due to secret partitioning. The attacker requires knowledge of partitioning methods before launching any attack.

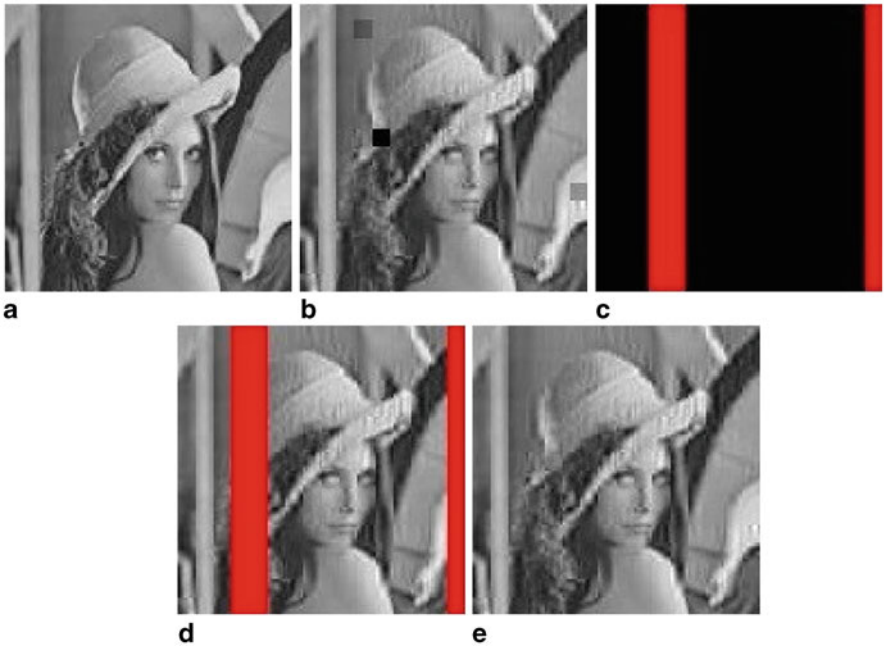

Fig. 5.6 IECC-MAC at 12.5 dB without channel coding.

5.6 Simulation Results

5.6.1 Simulation Parameters

Simulation results, for both of the proposed algorithms, are based on image transmission over additive white Gaussian noise (AWGN) channel with binary phase shift keying (BPSK) modulation. Results are given in the presence of as well as in the absence of channel coding (Turbo codes of rate-1/3). The LLRs produced by the decoder for convolutional turbo codes (CTC) are used for bit reliabilities, whereas the channel measurements are used as bit reliabilities in the absence of channel coding.

The source image is chosen as a grayscale image of 128 $\times$ 128 pixels, where each pixel requires 1 byte. The image is split into 8 $\times$ 8 pixel nonoverlapping blocks resulting in a total of 16 $\times$ 16 blocks for the chosen image resolution. DCT for each of these blocks is calculated and the DC components of the DCT matrices are protected using the algorithms explained earlier in Sect. 5.2. The DC and AC elements in the DCT matrix are of 16 bits each (and therefore need 2 bytes).

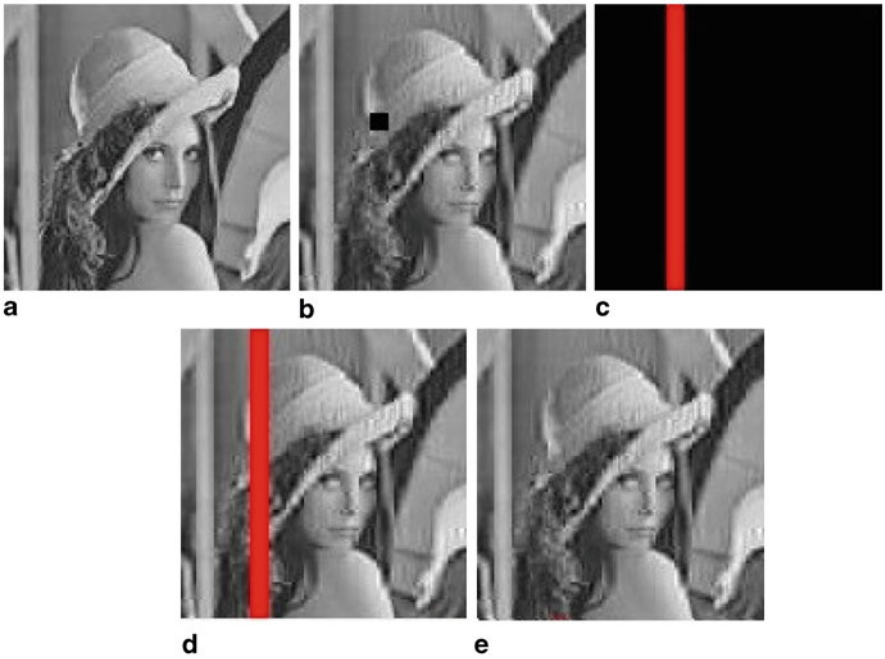

Fig. 5.7 IECC-MAC at 2.0 dB with CTC of rate-1/3

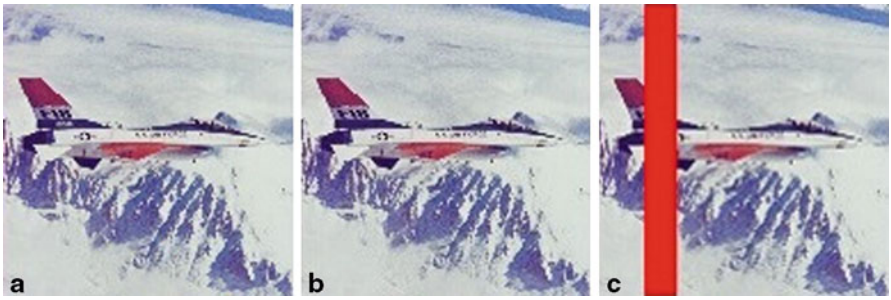

Fig. 5.8 **a** An airplane. **b** The number removed. **c** Modifications identified using IECC-MAC

5.6.2 Data Rate Analysis

SHA-256 is used as the standard MAC in the simulations. For a grayscale 128 × 128 pixels image protected using SHA-256, in total 128 × 128 × 8 bits of image data plus 256 bits of MAC (SHA-256) need to be transmitted. This is equal to 131,328 bits.

By protecting the DC elements using an IECC-MAC, 16 × 256 bits (= 4096 bits) of MACs are computed column-wise. The first 5 minor diagonals of the

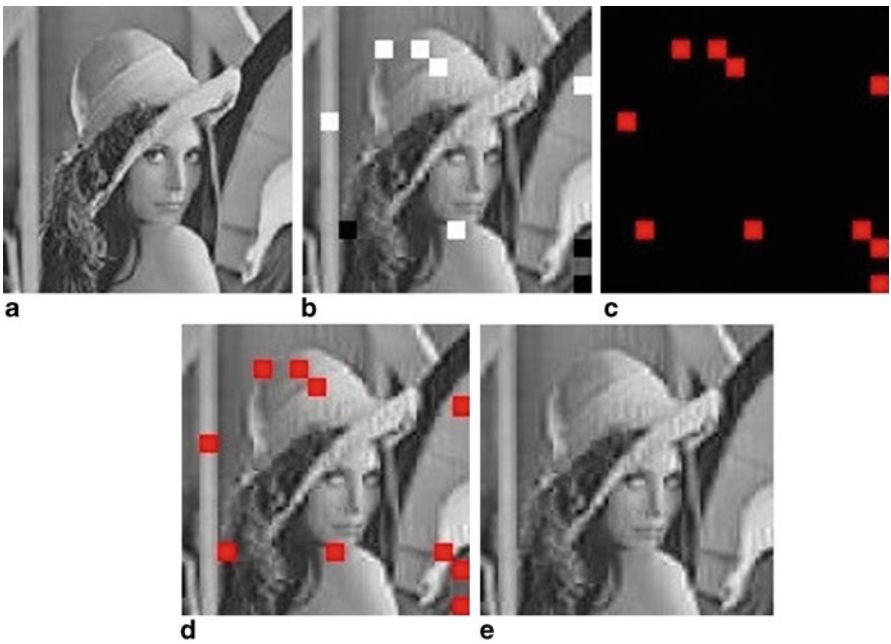

Fig. 5.9 IEC-NTMAC at 12 dB without channel coding

DCT matrices are transmitted (as compressed image in the simulations), which is $15 \times 16 \times 16 \times 16 = 61{,}440$ bits of data, where 15 is the number of elements from the 5 diagonals of each DCT matrix, 16×16 is the number of DCT matrices and the last 16 is the size (in bits) of each element in the DCT matrix. Thus, $61440 + 4096 = 65536$ bits are transmitted in total, which is approximately 50 % of the data that was transmitted by protecting the image using a standard MAC. Using IEC-NTMAC based error protection, the number of data bits transmitted is the same as in IECC-MAC. Each EC-NTMAC is 256 bits long. Thus, again approximately 50 % of the whole data is transmitted as compared to the conventional data transmission.

5.6.3 Simulation Results for IECC-MAC

Figure 5.6 shows the simulations results for IECC-MAC in the absence of channel coding at E_b/N_0 of 12.5 dB. Figure 5.6a shows the source image that is to be transmitted after protection with IECC-MAC. Figure 5.6b shows the image received at the receiver side. Figure 5.6c shows how the suspicious block positions are highlighted in red (columns of suspicious blocks) followed by the suspicious blocks mapped into the received image. Finally, the resultant image is shown, which is obtained by

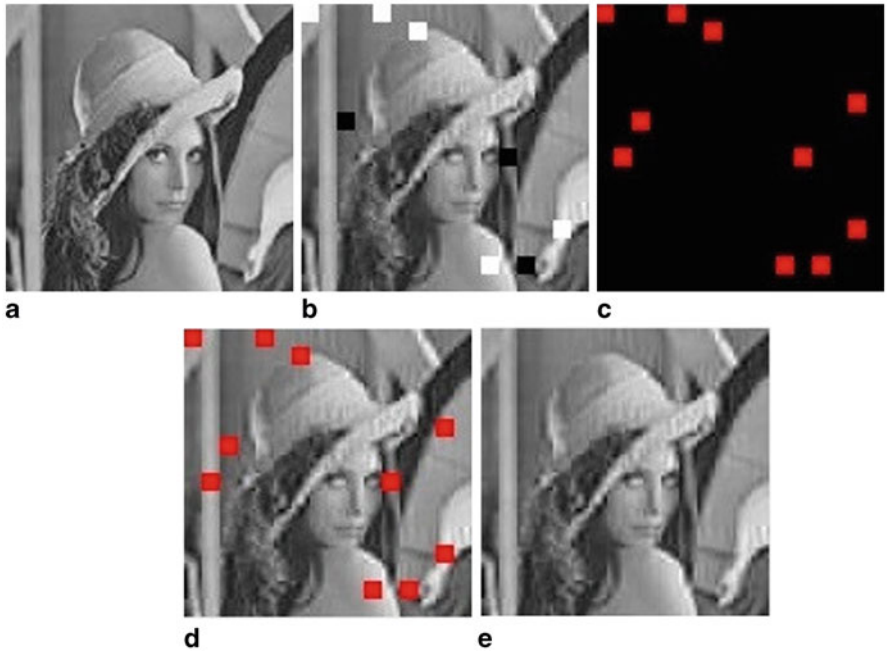

Fig. 5.10 IEC-NTMAC at 2.75 dB with CTC of rate-1/3

Fig. 5.11 **a** An airplane. **b** The number removed (forgery). **c** Modifications identified using IEC-NTMAC

performing the error correction of IECC-MAC algorithms over the suspicious image based on the localized errors.

Figure 5.7 shows the simulation results for IECC-MAC in the presence of channel coding (Turbo codes of rate-1/3) at E_b/N_0 of 2.0 dB. It can be observed that due to the presence of turbo codes, lesser number of erroneous blocks is obtained at a much lower E_b/N_0.

Figure 5.8 shows the results of IECC-MAC after forgery attack. Figure 5.8a shows an airplane image. Figure 5.8b shows a forged image, where the serial number

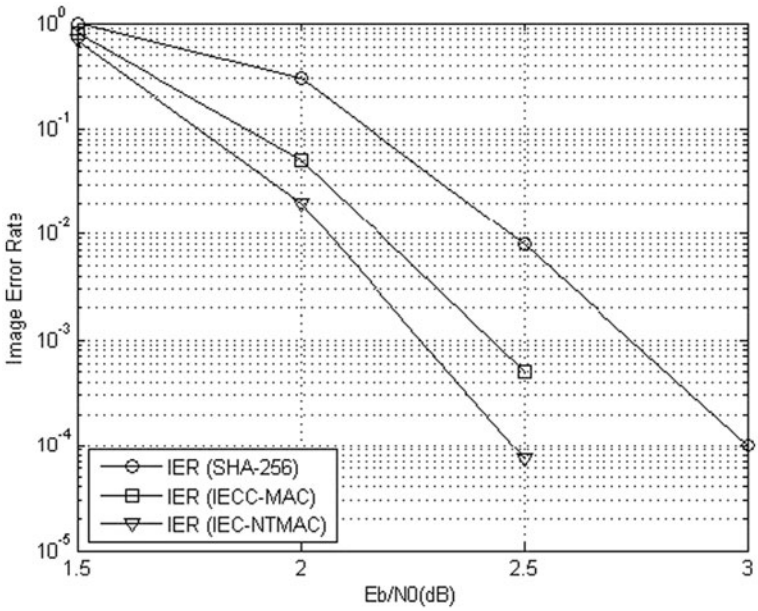

Fig. 5.12 IER over the AWGN channel with BPSK modulation and turbo codes of rate-1/3.

"01568" of the airplane has been removed. IECC-NTMAC is able to localize these modifications, as shown in Fig. 5.8c but is not able to correct them.

5.6.4 Simulation Results Using IEC-NTMAC

Images protected using IEC-NTMACs have better error localization capabilities and so they can be reconstructed more efficiently as compared to IECC-MAC. Simulation results in the presence and absence of channel coding, similar to the previous subsection, are presented in Fig. 5.9 and 5.10. Figure 5.11 shows the forgery attack on the airplane image. The error localization of IEC-NTMAC is refined to the block level, so it can be seen that the exact area of forgery has been marked by the IEC-NTMAC.

5.6.5 Image Error Rate (IER)

IER is defined as the number of times the whole image is declared as unauthentic divided by the number of times the image was received at the receiver, i.e.,

$$IER = \frac{\text{Nunmer of times the image is declared unauthentic}}{\text{Number of times the image is received}} \tag{5.12}$$

IER for both the algorithms is shown in Fig. 5.5 at different values of E_b/N_0. Three different curves are plotted for comparison. These curves show the IER in the presence of a standard MAC based protection, then in the presence of IECC-MAC and finally in the presence of IEC-NTMAC. Figure 5.5 shows that IEC-NTMAC provides the lowest IER and performs the best amongst the proposed algorithms (Fig. 5.12).

References

1. Reed IS, Solomon G. Polynomial codes over certain finite fields. SIAM J Appl Math. 1960;8:300–4.
2. Berrou C, Glavieux A, Thitimajshima P. Near shannon limit error-correcting coding and decoding: turbo-codes. Proceedings of the IEEE International Conference on Communications (ICC'93); Geneva, Switzerland; May 1993. pp. 1064–70.
3. Gallager RG. Low density parity check codes. Monograph, Cambridge: M.I.T. Press; 1963.
4. Peterson LL, Davie BS. Computer networks: a systems approach. 5th edn. Burlington: Morgan Kaufman; 2011.
5. Peterson WW, Brown DT. Cyclic codes for error detection. Proc IRE. 1961;49(1):228–35; (January 1961). doi:10.1109/JRPROC.1961.287814.
6. ISO/IEC 9797-1. Information technology—security techniques—Message Authentication Codes (MACs)—Part 1: mechanisms using a block cipher; 2011.
7. ISO/IEC 9797-2. Information technology—security techniques—Message Authentication Codes (MACs)—Part 2: mechanisms using a dedicated hash-function; 2011.
8. Bellare M, Canetti R, Krawczyk H. Keying hash functions for message authentication. Advance in Cryptology-CRYPTO'96, Lecture Notes in Computer Science, vol. 1109, N. Koblitz ed., Springer-Verlag, 1996. pp. 1–15.
9. Ur-Rehman O. Applications of iterative soft decision decoding. Aachen: Shaker Verlag; 2013. ISBN:978-3-8440-1641-3.
10. Graveman R, Fu K. Approximate message authentication codes. Proceedings of 3rd Annual Fed lab Symposium on Advanced Telecommunications/Information Distribution, vol. 1, College Park, MD, Feb. 1999.
11. Gravemen R, Xie L, Arce GR. Approximate image message authentication codes. Proceedings of 4th Annual Symposium on Advanced Telecommunications and Information Distribution Research Program, College Park, MD; 2000.
12. Boncelet C. The NTMAC for authentication of noisy messages. IEEE Trans Inf Forensics Secur. 2006;1(1):35–42.
13. Onien D, Safavi-Naini R, Nickolas P, Desmedt Y. Unconditionally secure approximate message authentication. Proceedings of the Second International Workshop on Coding and Cryptology, Springer, 2009.
14. Ge R, Arce GR, Crescenzo GD. Approximate message authentication codes for N-ary alphabets. IEEE Trans Inf Forensics Secur. 2006;1(1):56–67.
15. Ur-Rehman O, Zivic N, Amir Hossein S, Tabatabaei AE, Ruland C. Error correcting and weighted noise tolerant message authentication codes. 5th International Conference on Signal Processing and Communication Systems (ICSPCS)/IEEE Conference, Hawaii, USA, December; 2011.
16. Zivic N, Flanagan M. On joint cryptographic verification and channel decoding via the maximum likelihood criterion. IEEE Commun Lett. 2012;6(5):717–9.
17. Zivic N. Joint channel coding and cryptography. Aachen: Shaker Verlag; 2008. ISBN:978-3-8322-7180-0.
18. Ur-Rehman O, Tabatabaei AE, Amir Hossein S, Zivic N, Ruland C. Soft authentication and correction of images. Systems, Communication and Coding (SCC), Proceedings of 2013 9th International ITG Conference on, 2013; pp. 1–6.

19. Ur-Rehman O, Zivic N. Noise tolerant image authentication with error localization and correction. Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on, 2012; pp. 2081–7.
20. Chen WH, Pratt WK. Scene adaptive coder. IEEE Trans Commun. 1984;COM-32:225–32.
21. Watson A. Image compression using discrete cosine transform. Math J. 1994;1(4):81–8.
22. Ahmed N, Natarajan T, Rao KR. Discrete cosine transform. IEEE Trans Comput. 1974;C-23:90–3.
23. Yip P, Rao KR. Fast decimation-in-time algorithms for a family of discrete sine and cosine transforms. Circuits Syst Signal Process. 1984;3:387–408.
24. Chung SY, Forney GD Jr, Richardson TJ, Urbanke R. On the design of low-density parity-check codes within 0.0045 dB of the shannon limit. IEEE Commun Lett. 2001;5(2):58–60.
25. Berlekamp ER. Algebraic coding theory. New York: McGraw-Hill; 1968. (Revised edition, Laguna Hills: Aegean Park Press, 1984).
26. Sugiyama Y, Kasahara Y, Hirasawa S, Namekawa T. A method for solving key equation for goppa codes. Inf Control. 1975;27:87–99.
27. Guruswami V, Sudan M. Improved decoding of reed-solomon and algebraic-geometry codes. IEEE Trans Inf Theory. 1999;45(6):1757–67.
28. Koetter R, Vardy A. Algebraic soft-decision decoding of reed-solomon codes. Proceedings of the 2000 IEEE International Symposium on Information Theory, p. 61, Sorrento, Italy, Jun. 25–30, 2000.
29. Elias P. List decoding for noisy channels. Technical Report 335, Research Laboratory of Electronics, MIT; 1957.
30. Cox I, Miller M, Bloom J, Fridrich J, Kalker T. Digital watermarking and steganography. Berlington: Morgan Kaufmann; 2007.
31. Lee J, Won CS. A watermarking sequence using parities of error control coding for image authentication and correction. IEEE Trans Consumer Electron. 2000;46(2):313–7.
32. Wu Y. Tamper-localization watermarking with systematic error correcting code. Proceedings of IEEE International Conference on Image Processing (ICIP). 2006; pp. 1965–8, Atlanta, GA.
33. Fridrich J, Goljan M. Images with self-correcting capabilities. Proceedings of the IEEE International Conference on Image Processing (ICIP). 1999; pp. 792–6, Kobe.
34. Umbaugh SE. Digital image processing and analysis: human and computer vision applications with CVIPtools. 2nd edn, CRC Press, Nov. 2010. ISBN:978-1-4398-0205-2.
35. International Organization for Standardization. ISO/IEC JTC 1/SC 29 (2009-05-07). ISO/IEC JTC 1/SC 29/WG 1—coding of still pictures (SC 29/WG 1 Structure); 2009.
36. International Organization for Standardization. ISO/IEC 15444-1:2004—information technology—JPEG 2000 image coding system: core coding system; 2004.
37. Preneel B, van Oorschot PC. MDx-MAC and building fast MACs, from hash functions. Proceeding of CRYPTO 1995, LNCS 963, Springer Verlag. 1995; pp. 1–14.
38. Zivic N. Coding and cryptography: synergy for robust communication. Munich: Oldenbourg Verlag; 2013. ISBN:978-3-486-75212-0.