ROBUSTNESS INVISIBLE WATERMARKING RIJNDAEL CODE AGAINST ATTACKING METHODS

Dr. Imad H. Al-Hussaini Dr. Saleh M. Al-Qaraawy Dr. Khalid F. Shubair

Receipted: 3 /4 /2005
Accepted:9 /3 /2008

Abstract

Digital image watermarking has become a popular technique for authentication and copyright protection. For verifying the security and robustness of watermarking algorithms, specific attacks have to be applied to test them. This paper presents different attacks approaches which are based on the estimation of the parameters used. The exploitation of the properties of Back propagation neural network gives good results for detection.

الخلاصـــــــة

أصبحت العلامة المائية للصور الرقمية من التقنيات المألوفة لحماية النشر والترخيص. للتأكد من ســرية ومتانة خوارزمية العلامة المائية ، تستخدم وسائل الهجوم المعروفة لاختبار هذه الخوارزميات. يــستعرض هــذا البحث طرق الهجوم المختلفة والتي تستخدم مبدأ تخمينات العوامل المستخدمة. إن استغلال خــصائص الــشبكات العصبية بالانتشار الراجع أعطى نتائج جيدة في الكشف.

1. Introduction

Noise is somewhat hard to define, and many people argue about its exact definition. In general consider the signal to be the information- carrying part of the input and noise to be any additional information-less part. In general, though, noise is a random process. As such, it is usually characterized using the language of statistics. Often, in image noise is additive, simply causing the resulting image to be pixel-by-pixel higher or lower than it should be. Such noise can be modeled by [1]:

$$\tilde{h}(x) = \bar{h}(x) + \tilde{n}(x) \qquad \ldots(1)$$

h(x):original image

n(x):noise distribution in image

There are four kinds of noise used in this paper:

1-Salt and Pepper noise
2-Poisson noise
3-Gaussian noise
4-Speckle noise

These noises are applied to an image to get the artificial noise images. The noise signals in an image are usually high-frequency signals so back propagation neural network is used to detect watermark Rijndael code from this noises, which are good for training and good for giving low mean square error. A watermark Rijndael code does not affected by this attacking. The proposed attack is based on a stochastic

Control and Systems Engineering Department University of Technology

formulation of the watermark Rijndael code image, considering many attacks embedded in watermark image as additive noise with some probability distribution of other kinds of attacks.

The attack scheme consists of three main stages:

a) Noise attack.
b) Clipping attack.
c) Compression attack.

In this paper, the new detection technique by back propagation neural network give high efficiency against attacks .The watermark is completely appeared in all tested images without effecting original image quality. The approach can be used against watermark embedding schemes that operate in wavelet form.

2. Effect Of Poisson Noise Against Watermark Rijndael Code

In addition to the mean and variance, noise in terms of the shape of the distribution for each pixel can also be discussed. One common distribution for the values of each pixel is determined by the nature of image and the discrete arrivals over a period of time are modeled statistically by a Poisson distribution A Poisson distribution is similar to a Normal or Gaussian distribution with the following exceptions/properties:

1- A Poisson distribution is for discrete values, not continuous ones.
2- A Poisson distribution has the prc that its variance is equal to its mean.

3. Effect Of Gaussian Noise Against Watermark Rijndael Code

There are other sources of noise, however, that are unrelated to the noise above. There are many possible distributions for these random variables, but many of them can be modeled by a normal (Gaussian) distribution [2].

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\mu)^2/2\sigma^2} \quad \ldots(2)$$

Where z represents (noise) varies with the image intensity such Gaussian-distributed. Noise is usually uniform over the image. Noise n(x) that is Gaussian-distributed, zero-mean (doesn't change the average intensity level), uncorrelated, and additive is called white noise. Just as white light includes all parts of the visual spectrum, white noise includes all parts of the frequency spectrum. Just like a *delta* function, white noise has a uniform frequency spectrum. However, unlike the *delta* function (for which all of the frequencies are in phase), the phase part of the transform is random. Some forms of noise, has light low-frequency content and large high-frequency content.

4. Effect Of Salt And Pepper Noise Against Watermark Rijndael Code

Another common form of noise is *data drop-out* noise (commonly referred to as *intensity spikes*, *speckle* or *salt and pepper noise*). Here, the noise is caused by errors in the data transmission. The corrupted pixels are either set to the maximum value (which looks like snow in the image) or have single bits flipped over. In some cases, single pixels are set alternatively to zero or to the maximum value, giving the image a `salt and pepper' like appearance. Unaffected pixels always remain unchanged. The noise is usually quantified by the percentage of pixels which are corrupted [4].

$$p(z)=\begin{cases} P_a & \text{for } z=a \\ P_b=1-P_a & \text{for } z=b \\ 0 & \text{otherwise} \end{cases} \quad \ldots(3)$$

where, $a > 0$, b is an integer.

This enables users to choose what the types of noise they will apply to the image and what kind of noise they will use to apply in the watermark image. Users can set the density of the distribution of salt & pepper noise. The density to be added in this work is of random choice, between 0.01 and 0.09 percents of the pixels in the watermark Rijndael code image will be transformed to noise points. The parameters of the function *"imnoise"* are the watermark Rijndael image, the type of noise, and the density for salt & pepper noise respectively. The output of this function is the watermark Rijndael noise image.

5. Quick Review of Statistics

In this paper, three statistical quantizes will be used [2,3]:

a) Mean:

The average or *expected value*.

$$\sigma^2 = E\{(x-\mu)^2\} = E\{x^2\} - \mu^2 \quad \ldots(4)$$

b) Variance.

The expected value of the squared error.

$$\mu = E\{x\} = \frac{\sum x}{n} \quad \ldots(5)$$

c) Standard Deviation.

The square root of the variance.

$$\sigma = \sqrt{\sigma^2} \quad \ldots(6)$$

Since the units of measure may be arbitrary, we typically normalize the standard deviation by comparing it to the mean.

6. Effect of Speckle Noise

The coherent imaging systems are widely used in many practical applications: remote sensing (synthetic aperture radars), biomedical imaging, acoustic holography, etc[5]. A high level of speckles is typical for obtained images, especially if they are formed using one-look systems. One more peculiarity consists of the fact that the multiplicative noise distribution occurs to be non-Gaussian and, moreover, it is non-symmetrical with respect to the mean value. Considered images can be simultaneously corrupted by spikes and speckles. The reasons of spike occurrence are various, for example, bit errors. The problem is that the probability of spikes and their characteristics are often unknown (unpredictable). That is why it is desirable to apply robust filters for image processing. The corresponding algorithms should satisfy a set of contradictory requirements: to provide appropriate noise suppression efficiency, reliable spike removal and good edge/detail preservation simultaneously. We have proposed the robust locally adaptive algorithms ensuring rather good trade-off of required properties. The proposed approach assumes that for processing and detect watermark Rijndael code from speckle noise. The decision on method selection is based on local activity indicator analysis. This approach permits to formulate different requirements to

achieve any attack occur. The main items for the latter ones are to ensure an effective neural network to detect code from speckle noise combined with appropriate robustness with respect to spikes. The properties of providing the unbiased mean estimate and ensuring its minimal mean square error (MSE) have been analyzed.

This happens exactly when the relative variance of the speckle noise is approximately zero. This model assumes the speckle noise as contaminated distribution and the salt-and-pepper noise as contaminating factor. The probabilities and relative amplitudes of negative and positive spikes were varied in wide limits while performing simulations. The final conclusions are established by generalizing analysis of the obtained results.

7. Effect of Image Compression Against Watermark Rijndael Code

The goal of image compression is to create smaller files that use less space to store and less time to send. A DVD movie is compressed to about 1/60, if it is not compressed, you'll need 60 DVDs to watch one movie! But how can a digital photo or movie be compressed to a smaller file-size without loss of image quality? The answer: there are lots of repetitions hidden in a digital photo!. Compressing an image is significantly different than compressing raw binary data [6]. Of course, general purpose compression programs can be used to compress images, but the result is less than optimal. This is because images have certain statistical properties which can be exploited by encoders specifically designed for them. Also, some of the finer details in the image can be sacrificed for the sake of saving a little more bandwidth or storage space. This also means that lossy compression techniques can be used in this area. Lossless compression involves with compressing data which, when decompressed, will be an exact replica of the original data. This is the case when binary data such as executables, documents etc. are compressed. They need to be exactly reproduced when decompressed. On the other hand, images (and music too) need not be reproduced 'exactly'. An approximation of the original image is enough for most purposes, as long as the error between the original and the compressed image is tolerable. The first step in compressing an image is to segregate the image data into different classes. Depending on the importance of the data it contains, each class is allocated a portion of the total bit budget, such that the compressed image has the minimum possible distortion.

8. Effect Of Clipping Image In Watermarking Rijndael

Image clipping is used to cut a box-shaped section through the data currently being rendered. Only the image contents that lie within the box are then rendered. Image clipping is analogous to starting with a block containing embedded tissue, trimming the block and then cutting a specified cross-section of the block. Once image clipping has been specified, the following steps can be considered:

1- Specify two types of fixed size clip and random size clip for the clipping box which is determent by number of blocks and percentage size for each clip the default is chosen by designer.

2- Specify the percentage size clip for the left, right, upper and lower side of the clipping side.

3- Appear these clipping boxes in the watermark Rijndael code image. The clipping path must appear around the region of watermark Rijndael code image and restricts any parts of the drawing that lie outside of the region bounded by the currently active clipping path are not drawn in the watermark Rijndael image.

9. Rijndael Algorithm

Rijndael is an iterated block cipher with a variable block length and a variable key length. The block length and the key length can be independently specified to 128, 192 or 256 bits with the constraint that the input and the output have the same length [7]. Figure (1) shows Rijndael ciphertext algorithm.

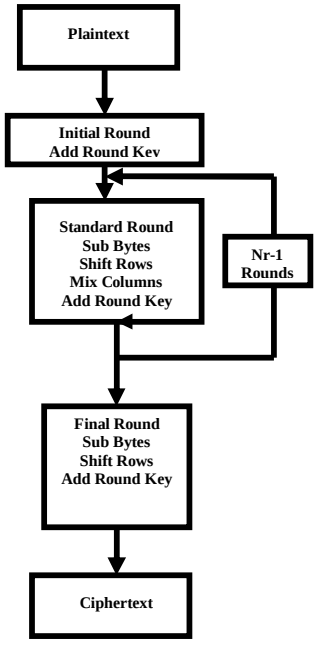

Figure (1) Flowchart for converting plaintext to ciphertext

The cipher transformation can be inverted and then implemented in reverse order to produce a straightforward inverse cipher for the Rijndael algorithm. The inverse roundtransformation. The

individual transformations used in the inverse round transformation {inv shift rows, inv sub bytes, inv mix columns and add round key} process the state to retrieve the data block [8]. The flowchart for inverse transform is shown in Figure (2).

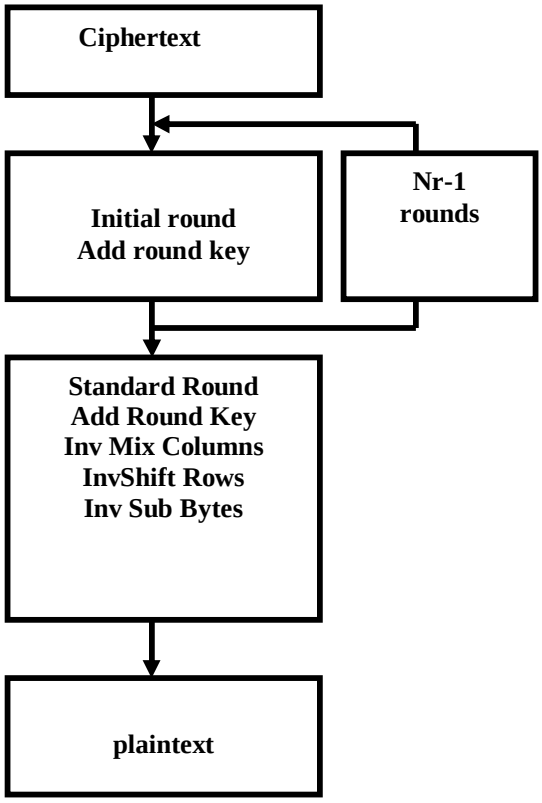

Figure (2) Flowchart for deciphering using Rijndael algorithm

9. Neural Networks

Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output [9]. Such a situation is shown in Figure (3). There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are used, in this type it is called *supervised learning*, to train network.

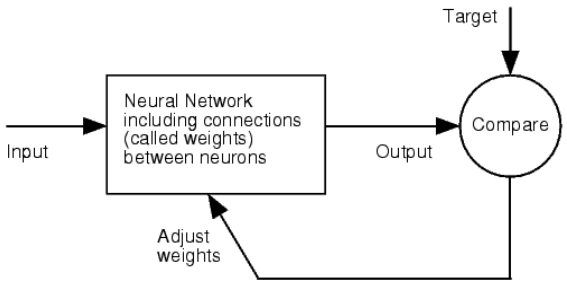

Figure (3) Neural Network Model

Batch training of a network proceeds by making weight and bias changes based on an entire set (batch) of input vectors. Incremental training changes the weights and biases of a network as needed after presentation of each individual input vector. Incremental training is sometimes referred to as "on line" or "adaptive" training.

Back Propagation algorithm has been used to recognize and detect watermarking Rijndael cipher code. Binary digit for the watermarking string Rijndael code has been decoded. One of the critical issues in watermarking Rijndael code is the feature selection for the back propagation layers construction and is dependent on the choice of the features for each case used [9,10].

The aim was to recognize all bits for watermark Rijndael cipher code and to create a network that could recognize the inverse cipher to obtain the watermarking characters correctly though there were some attacks like noise and others which make errors in any bit but increasing the number of copy for watermarking Rijndael code will defeat any attacking appear in image.

10. Requirements For Detecting Watermarking Rijndael Code

It is important to define the requirements for detecting watermarking Rijndael. There are two key aspects of detecting a watermarking system code:

1- Detection the watermark code by identifying the Rijndael secret key and pseudo random key for watermarking location within the host watermarking image.

2- Applying one of the important intelligent system called back propagation neural network and a good result obtained for watermark security against common operations or attacks.

11. Back Propagation Neural Network Training Steps

The main steps of training on a pattern may now be expanded into the following steps:

1. Present randomly pattern at the input layer for BPNN.

2. Let the hidden units evaluate their output using the weights between nodes.
3. Apply the target pattern to the output layer.
4. Calculate the actual output on the output nodes according to training procedure.
5. Train each output node using gradient descent.
6. For each hidden node, calculate its adjusted weights according to difference between desired and actual. This step is collectively known as the *backward pass.*
7. Let the output units evaluate their actual output and compared with desired output (1,0)
8. Learn (500) iterations for desired output one, and learn (500) iterations again for desired output zero, and then save each case for each randomly inputs. The reason for choosing 500 to take more cases for randomly inputs and to achieve watermark string without any errors.
9. After back propagation neural network learned about these inputs, enter the neighborhood pixels around watermark location to this network.
10. The simulate results generate actual output directly according to compare randomly input with watermark input.

11. Applying the average values for each actual output. If the average greater than zero the actual location becomes one, and if the average values less than zero the actual output becomes zero, and so on for all Rijndael ciphering appeared directly from (1 to 128 bit). After that applying inverse Rijndael algorithm to obtain watermark string.

Number of copy:	**30**
Watermark key:	**5**
Rijndael key password:	**iraq**
Watermark string:	**baghdad**
Wavelet kind:	**Db3**

12. Simulation Results

Let us consider the woman image shown below to test the proposed algorithm.

i) At sender

The DWT for woman image is shown in Figure (4).

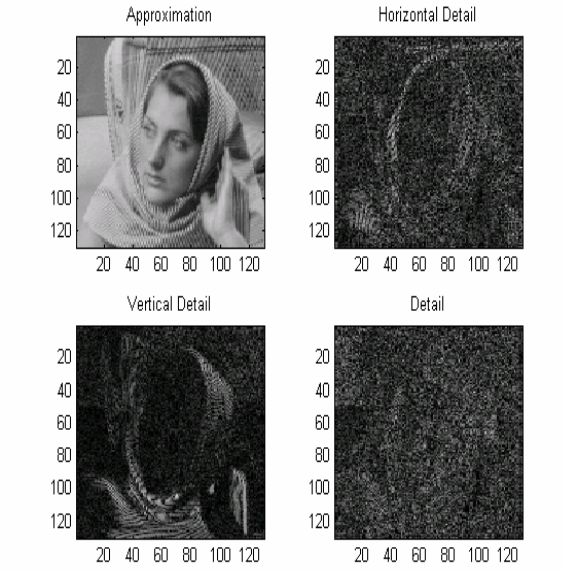

Figure (4) Applying DWT for 2-D woman image

The original image and invisible watermark image are shown in Figures (5) and (6) respectively.

Figure (5) two dimensional original image

Figure (6) (Invisible watermarking) with Rijndael cipher code

ii) <u>At Receiver:</u>

Number of copy:	**30**
Watermark key:	**5**
Rijndael key password:	**iraq**
Wavelet kind:	**Db3**
Number of first hidden layer:	**10**
Number of second hidden layer:	**4**
Number of output:	**1**

The received image with watermarking Rijndael cipher technique is shown in Figure (7), while Table (1) shows the calculated MSE.

Figure (7) Received image with Invisible watermarking Rijndael code

Table (1): Calculation results for epoch and MSE

EPOCH	MSE
0/100	1.40431/0
22/100	9.93849e-017/0

Figure (8) represents the plot of the calculated MSE against number of epochs of the back propagation neural network training at receiver.

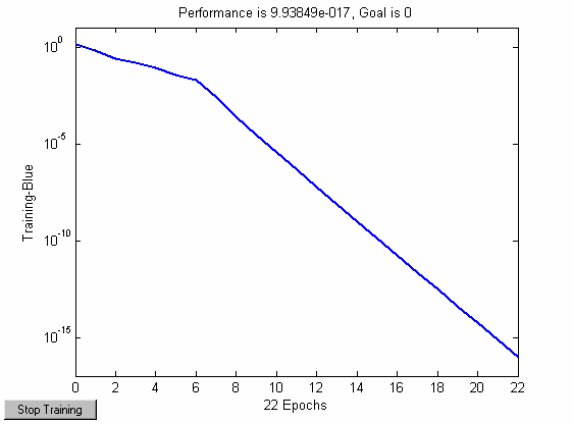

Figure (8) The MSE for training network

Watermarking String Answer: Baghdad

Let us now applying different types of noise to the received image.

a) Applying both Salt – Pepper noise and Speckle noise for the received image

Figure (9 shows the received image after applying this type of noise, assuming that the Density = **0.05**.

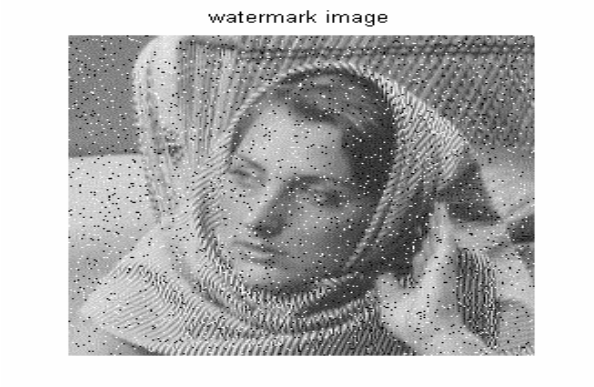

Figure (9) Watermark image after applying salt - pepper noise and speckle noise with salt and pepper density=0.05

The BPNN is used to detect watermarking Rijndael Code with Least Mean Square error as shown in Figure(10).

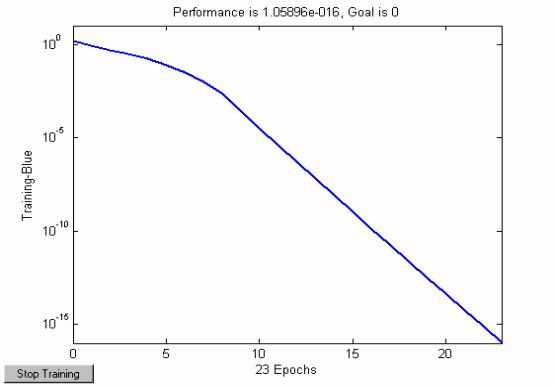

Figure (10): MSE for detection watermark Rijndael code after applying both salt and pepper noise and speckle noise.

The result obtained is the same as input and as follows:

Result Obtained(Watermark Rijndael Code After (Applying both Salt and pepper noise and Speckle noise):

b) Applying Both Fixed Block Size And Left Clip Side.

*** Number of blocks = 10**

*** Percentage clip=9%**

*** Left clip=10%**

Figure(11) shows the attacking watermarking Rijndael code for this image.

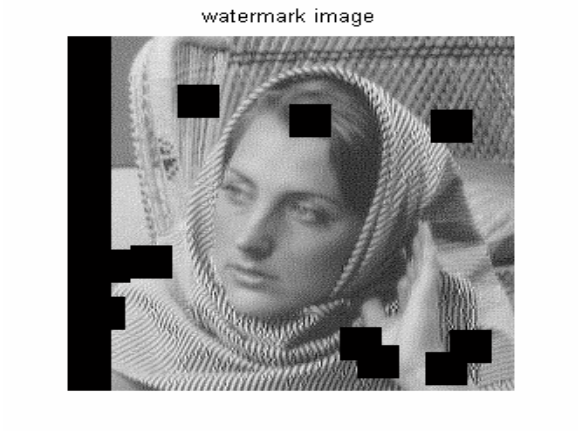

Figure (11) Watermark image after applying fixed size block and left clip side

The BPNN is used to detect watermarking Rijndael code with least

mean square error as shown in Figure(12).

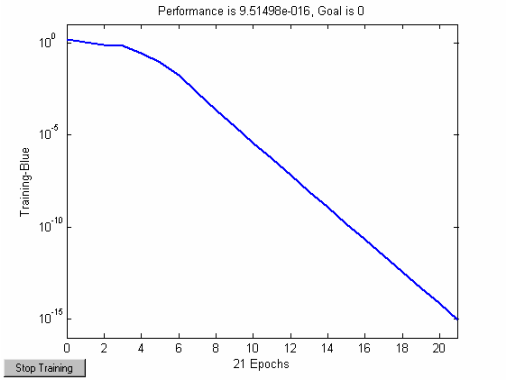

Figure (12) MSE for detection watermark Rijndael code after applying both fixed block size and left clip side

Result Obtained Watermark Rijndael Code (After Applying both fixed block size clip and left clip side): baghdad

c) Applying both Random Size Block and Upper Clip Side for the received image.

*** Number of blocks =10**

*** Percentage clip size = 8%**

*** Percentage upper clip=9%**

Figure(13) shows the attacking watermarking Rijndael code for this state.

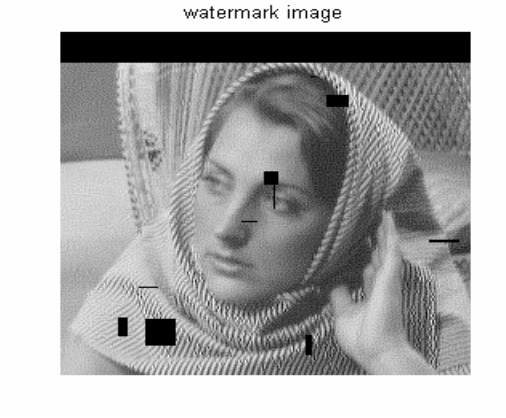

Figure (13) Watermark image after applying random size block and upper clip side

The BPNN is used to detect watermarking Rijndael code with least mean square error as shown in Figure (14).

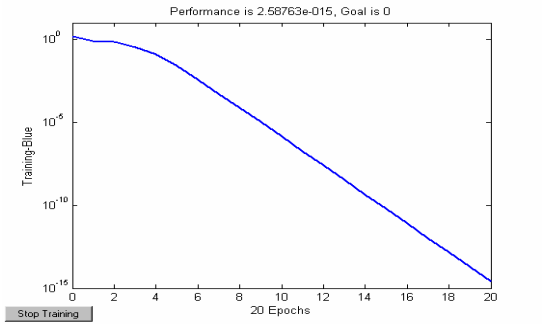

Figure (14): MSE for detection watermark Rijndael code after applying both random block size and upper clip side

Result Obtained (Watermark Rijndael Code After Applying both random block size clip and upper clip side :baghdad

d) Applying both Salt - Pepper Noise and Compression Technique for the received image.

*** Density=0.08**

*** Compression by 15%**

*** Image before compression=66614**

*** Image after compression=36807**

See Figure(15) for attacking watermarking Rijndael code**.**

Figure(15):Watermark image after applying salt and pepper noise and compression technique

The BPNN is used to detect watermarking Rijndael code with least mean square error see Figure(16).

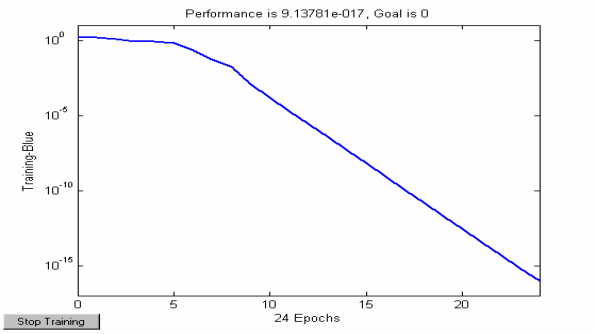

Figure (16) MSE for detection watermark Rijndael code after applying both salt and pepper noise and compression technique.

Result Obtained(Watermark Rijndael Code After Applying both salt-pepper and compression technique :baghdad

Conclusions

In this paper, controlling for attacking technique depends on the parameters used like density, variance and other parameters was considered. Applying different attacks like Gaussian noise, Poisson noise, Speckle noise , Salt-Pepper noise, clipping attack and compression attack in received image was considered too. Good performance obtained by BPNN that helps the designer. BPNN gives good results to achieve watermarking string code. The BPNN learned and simulated results with least mean square error and watermark string for each case were achieved after applying strong attack. The mixing attack applied to test and a good performance achieved by using back propagation neural network. The Rijndael algorithm achieved good security in two dimensional images and deciphering achieved by applying inverse Rijndael algorithm with many mathematical operations were used to achieve watermark string.

References

[1] Iwan, Setyawan., "Attacks on Watermarking Systems ", Information and Communication Theory Group, Delft University of Technology, 2000

[2] Karl, Friedrich, Gauss., " Gaussian Noise :Gaussian Probability Distribution", Internet Report, 2000. Website:http://www.sfu.ca/sonic-tudio/handbook/Gaussian_Noise.html

[3] Bryan, S., " Image Enhancement: Introduction and point Processing", Lecture from Internet, 1995. Website: http://iu1.cs.byu.edu/morse/550-F95/node15.html

[4] Bob, Fisher., " Noise Generation", Lecture from Internet, 1997 Website: http://marathon.csee.usf.edu/hiprjava/node11.html

[5] Syno, S.," Speckle Noise: Model Noise " , Report from Internet, 2001 Website: http://www.atis.org/tg2k/_speckle_noise.html

[6] Satish, Kumar., " An Introduction to Image Compression " Report from Internet, 2001.

Website:
http://www.debugmode.com/imagecmp/

[7] Joen, Daemen., and Vincent, Rijmen., " AES Proposal :Rijndael " Document Version2, Internet Paper, 1999.
Website:
http://csrc.nist.gov/encryption/aes/rijndael/Rijndael

[8] Fedral, et al.," Announcing the Advanced Encryption Standard (AES)", Internet Paper from Information Processing Standards Publication, 2001.
Website:
http://www.cs.technion.ac.il/~cs236506/online/fips197

[9] Lippman, R., "An Introduction to Computing with Neural Networks", IEEE Assp Magazine, pp4-22, April 1987.

[10] Chen, D. S., " A Robust Back Propagation Learning Algorithm for Function Approximation ", IEEE Trans .Neural Network,Vol.5, May 1994.