Master's Thesis
Electrical Engineering with Emphasis on Signal Processing
Thesis no:
June. 2016

Robust Image Hash Spoofing

Azadeh Amir Asgari

Department of Applied Signal Processing
Blekinge Institute of Technology
SE – 37179, Karlskrona, Sweden
www.bth.se

Fraunhofer-Institut für Sichere Informationstechnologie
Rheinstraße 75, 64295 Darmstadt, Germany
Phone: +49 6151 8690
www.sit.fraunhofer.de

This thesis is submitted to the Department of Applied Signal Processing at Blekinge Institute of Technology of Sweden in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering with Emphasis on Signal Processing.

Contact Information:
Author:
Azadeh Amir Asgari
E-mail: aziamirasgari@gmail.com

Advisors:
Dr. .-Ing. Martin Steinebach
Fraunhofer-Institut für Sichere Informationstechnologie
Media Security and IT Forensics
Phone: +49-6151-869-349
E-mail: martin.steinebach @ sit.fraunhofer.de
Address: CASED building, Mornewegstrasse 32, 5th floor

Prof. Dr. Stefan Katzenbeisser
CASED
Security Engineering Group
Computer Science Department
Technische Universität Darmstadt
Phone: +49-6151-16-5016
E-mail: katzenbeisser@seceng.informatik.tu-darmstadt.de
Address: CASED building, Mornewegstrasse 32, 4th floor, room 4.3.25

Dr. Huajian Liu
Fraunhofer-Institut für Sichere Informationstechnologie
E-mail: huajian.liu@sit.fraunhofer.de
Address: CASED building, Mornewegstrasse 32, 5th floor

Supervisor:
Dr. Benny Lövström
Blekinge Institute of Technology
Senior Lecturer
Phone: +46-455-385704
E-mail: benny.lovstrom@bth.se
Address: Room J3612- Blekinge Tekniska Högskola, 371 79 Karlskrona

ABSTRACT

With the intensively increasing of digital media new challenges has been created for authentication and protection of digital intellectual property. A hash function extracts certain features of a multimedia object e.g. an image and maps it to a fixed string of bits. A perceptual hash function unlike normal cryptographic hash is change tolerant for image processing techniques. Perceptual hash function also referred to as robust hash, like any other algorithm is prone to errors. These errors are false negative and false positive, of which false positive error is neglected compared to false negative errors. False positive occurs when an unknown object is identified as known. In this work a new method for raising false alarms in robust hash function is devised for evaluation purposes i.e. this algorithm modifies hash key of a target image to resemble a different image's hash key without any significant loss of quality to the modified image. This algorithm is implemented in MATLAB using block mean value based hash function and successfully reduces hamming distance between target image and modified image with a good result and without significant loss to attacked imaged quality.

Keywords: Robust hash function, Hamming distance, Block mean value, Spoofing attack.

ACKNOWLEDGMENT

I would like to express my sincere thanks to Dr.-Ing. Martin Steinebach who gave me the opportunity to be involved his research team and provided all the necessary facilities and the valuable guidance and encouragement. I sincerely wish to express my gratitude and appreciation to Prof. Stefan Katzenbeisser who is guided me from the commencement to the successful completion of my study. I also thank my supervisor Prof. Benny Lövström at Blekinge Institute of Technology for his guidance and support through this thesis. I would like also to thank Dr. Huajian for his comments and suggestions. Last but not least, I would like to state my sincere thanks to my parents and my family especially Arash Hassanzadeh who without their unconditional support, I would not have been able to reach this point.

Table of Contents

List Of Figures

List Of Tables

1. Introduction

Throughout the last two decades, improving the technology of digital media imposes new problems for managing large multimedia databases in terms of authentication and managing intellectual properties as well as broadcast monitoring and network filtering. Robust Hashing is a technology as a change tolerant alternative to cryptographic hashes since normal cryptographic hashing methods are error prone to image processing techniques. As any other field of technology, perceptual hash functions have two major drawbacks to be named “false negatives” and “false positives”. False negatives occur if a known object is not recognized and false positives occur if an unknown object is recognized as known content. False positive errors are neglected compared to false negative errors in the field of cryptography.
This thesis work is intended to devise a new method in order to raise false positive alarms in robust image hashing for evaluation purposes through global and local modification of another image to have a hash similar to a target image. This method is implemented in MATLAB by block mean value based Hashing algorithm.

In the chapter 2 basic concepts such as perceptual hash function, hamming distance, comparison between several hashing methods have been discussed. Block mean value based hashing method is completely described in the latter part of the chapter since it has been used through this work.
In chapter 3, implementation of the developed method is discussed in full depth and at chapter 4 the devised algorithm is verified and results have been discussed. In the last chapter we have a conclusion on this job.

2. Background

2.1. Introduction

Due to rapid growth in digital media, content authentication of multimedia is a major concern and is used extensively. It has been said that hiding the information or protecting the privacy has a history as long as the writing itself. Mankind has been trying many ways to hide information during its history which is called steganography, i.e., the art or practice of concealing a message, image, or file within another message, image, or file by replacing fixed symbols, and cryptology or cipher. By changing technology from handwritten words on paper sent by courier to the communication of the information via both local and worldwide communication networks and the saving and processing in the form of digital data on computers definitely has enhanced the risk of exposing of information to eavesdropping. Cryptography was the only solution which has been borrowed from the secret world of army commanders and politicians into the global commercial applications. Beside the concealing or privacy protection, it should be noted that both the contents and the originator of the information are not changed. Both of these conditions are expressed in the term "authentication." A hacker who attempts to change contents or origin of information is called an active attacker. The increasing of relative importance of this threat might be captured by the advent of malicious software programs. These digital threats are best exemplified in the form of computer viruses. Others include worms, Trojan horses, and logical bombs.

2.2. Authentication and related history

The origin of the authentication is from the Greek word αὐθεντικός which means real or genuine and in network security discourse it means confirming the identity of multimedia object. Authentication is an approach to protect communicating parties from a third party attack. But, when communicating parties are distrustful to each other and try to refuse their authorities, it is likely to emerge different threat. It means that sender or receiver try to change a message or deny to have sent or received data.

The protection of authenticity could include two aspects:

- The protection of the originator of the information, or in ISO terminology data origin authentication.

- The fact that the information has not been modified, or in ISO terminology the integrity of the information. There are two basic methods for protecting the authenticity of information.

2.3. Cryptographic Hash Function

As it is described below cryptographic hash functions are used to authentication process extensively and first the definition of hash functions is provided. Cryptographic hash function generates hash value from an arbitrary data array while keyed hash function uses a secret key as well [1].

Def. 2-1**:** *A hash function is [. . .] a function H which has, as a minimum, the following two properties:*

- *Compression - H maps an input x of arbitrary finite bit length, to an output H(x) of fixed bit length n.*
- *Ease of computation - given H and an input x, H(x) is easy to compute.*

Hash functions are mostly used to accelerate table lookup or data comparison tasks such as finding items in a database, detecting duplicated or similar records in a large file, finding similar stretches in DNA sequences [7]. The term hash functions has its historic roots in computer science where a hash function or a cryptographic hash function is denoted to any algorithm or subroutine that maps large data sets of variable length, called keys, to smaller data sets of a fixed length as uniformly as possible. The value which is returned by a hash function in cryptographic literature is called hash total, hash result, hash sums, hash code, imprint, (cryptographic) checksum, compression, compressed, encoding, seal, authenticator, authentication tag, fingerprint, test key, condensation, Message Integrity Code (MIC), message digest or simply hashes [15].

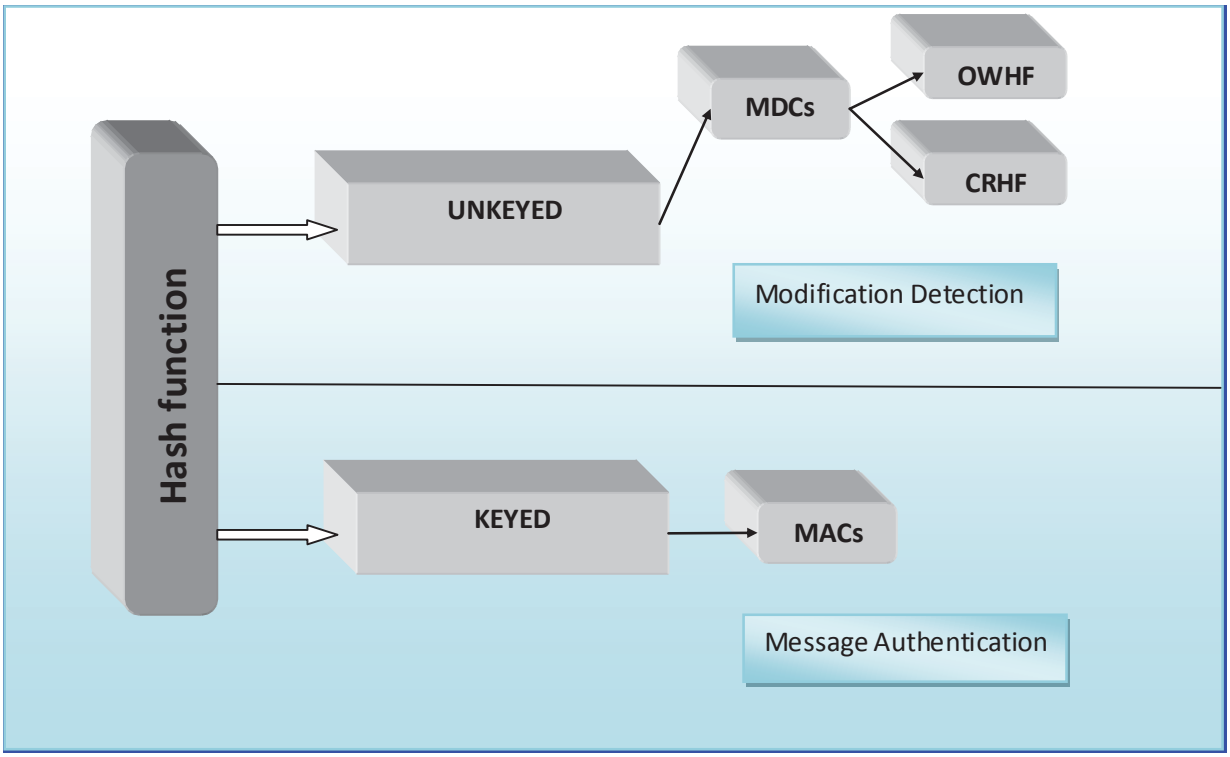
Cryptographic hash functions may be divided in two categories [1]:

1- Unkeyed Hash Functions
2- Keyed Hash Function

These two major approaches to protect authenticity of information are of a great importance and it deserves to have a more detailed explanation. The first approach is similar to the approach of a symmetric or asymmetric cipher in which the concealment of a large amount of data is according to the concealment and authenticity of a short key. Here, the authentication of the information would be based on the concealment and authenticity of a key. To do so, the information is compressed to the amount in length which is called a “hash code”. Consequently, the hash code is annexed to the information. The operation that accomplishes this process is called a hash function. The main concept of the protection of the integrity is to add redundancy to the information. The presence of this redundancy provides the receiver the ability to make the distinction between authentic information and bogus information. In order to assure the origin of

the data, a secret key associated to the origin should intervene in the process. The secret key could be included in the compression process or can be employed to protect the hash code and/or the information. In the first case the hash code is called a Message Authentication Code or MAC, while in the latter one it is called a Manipulation Detection Code or MDC.

The second approach includes making the authenticity (both integrity and origin authentication) of the information based on the authenticity of a Manipulation Detection Code or MDC. A well-known example of this method is a computer user calculating an MDC for all its important files. An individual may perform storage process in two ways: whether storing this collection of MDC's on a hard disk, which is locked in his/her safe, or writing them down on a piece of paper. If he/she is to transfer the files to a distant friend, they can be sent easily and MDC's can be communicated via telephone. Here, the authentication of the telephone channel is performed by voice identification. The second application for cryptographically secure hash functions is manifested through optimizing the digital signature schemes and the building up the digital signature schemes which are not based on a trapdoor one-way permutation. The optimization is performed via signing the MDC of a message rather than each bit or block. The description of the hash function might be public and it does not depend on any secret parameter. The advantages of this method are: the signature includes a fixed short length which minimizes the computational operation. Sometimes, it is possible to increase the security level of the signature scheme. In some cases, the hash function is even an embedded part of a scheme. In general, digital signature schemes based on one-way functions practically are less potent, but can be used as a substitution if one is not permitted or wants to employ a scheme according to a trapdoor one-way permutation. In the following the hash function will be denoted with h, and its argument, i.e. the information to be protected, with X. The image of X under the hash function h will be denoted with h(X) and the secret key with K [15].



Hash function structure

One-way hash function (OWHF)

The first rough definition of an OWHF was evidently suggested by R. Merkle [10, 11] and M. Rabin [12].

A one-way hash function is a function h satisfying the following conditions:

1. The description of h must be publicly known and should not need any secret information for its operation (extension of Kerckhoffs's principle1).
2. The argument X can be of arbitrary length and the result $h(X)$ has a fixed length of n bits.
3. Given h and X, the computation of $h(X)$ must be "easy".
4. The hash function must be one-way in the sense that given a Y in the image of h, it is "hard" to find a message X such that $h(X) = Y$ and given X and $h(X)$ it is "hard" to find a message $X' \neq X$ such that $h(X') = h(X)$.

Collision resistant hash function (CRHF)

The first formal definition of a CRHF was evidently suggested by I. Damgård [13, 14]. A rough definition was suggested by R. Merkle in [11].

A collision resistant hash function is a function h satisfying the following conditions:

1. The description of h must be publicly known and should not require any secret information for its operation (extension of Kerckhoffs' principle).
2. The argument X can be of arbitrary length and the result $h(X)$ has a fixed length of n bits.
3. Given h and X, the computation of $h(X)$ must be "easy".
4. The hash function must be one-way in the sense that given a Y in the image of h, it is "hard" to find a message X such that $h(X) = Y$ and given X and $h(X)$ it is "hard" to find a message $X' \neq X$ such that $h(X') = h(X)$.
5. The hash function must be collision resistant: this means that it is "hard" to find two distinct messages that hash to the same result.

Message Authentication Code (MAC)

Message Authentication Codes have been used for a long time in the banking community and are thus older than the open research in cryptology that started in the mid-seventies. However, MAC's with good cryptographic properties were only introduced after the start of open crypto logic research.
A MAC is a function satisfying the following conditions:
1. The description of h must be publicly known and the only secret information lies in the key (extension of Kerckhoffs's principle).
2. The argument X can be of arbitrary length and the result $h(K, X)$ has a fixed length of n bits.
3. Given h, X and K, the computation of $h(K, X)$ should be "easy".
4. Given h and X, it is "hard" to determine $h(K, X)$ with a probability of success "significantly higher" than $1/2^n$ with n the number of bits of hash code. Even when a large set of pairs $\{X_i, h(K, X_i)\}$ is known, where the X_i have been selected by the opponent, it is "hard" to determine the key K or to compute $h(K, X')$ for any $X' \neq X_i$. This last attack is called an adaptive chosen text attack.

2.4. Hash Functions and Multimedia

A multimedia object e.g. an image can have different digital forms which from a human perception point of view, all are the same. These different forms are the consequence of wide range of image processing techniques such as cropping, compression, flipping, compression and equalization which each changes the binary form of the image. Due to this problem normal cryptographic hash function does not work for multimedia applications. On the other hand image identification methods, such as semantic models or face detection algorithms, although show good performance in identifying illegal multimedia objects have big drawbacks such as high computational complexity and high false alarm rates.
Therefore perceptual hash functions have been introduced in order to circumvent the problem of distinguishing perceptual equality of multimedia content. Recently due to extensive demand in the industry new perceptual hash functions have been introduced by scientific researchers. These

perceptual hash functions extract certain properties form the multimedia content and produce hash values based on these features.
In order to have a measure for comparison between two perceptual hash values, there are defined functions for their distance/similarity scoring such as hamming distance, Bit Error Rate (BER) and Peak of Cross Correlation (PCC).
Perceptual hash functions are interdisciplinary field of research which includes cryptography, digital watermarking and signal processing. These perceptual hash functions because a lack of standard or uniform nomenclature may be addressed with different terms such as [2]:

- Fingerprint
- Passive fingerprint
- Perceptual checksum
- Robust hash
- Soft hash

In passive fingerprint the content of multimedia is unchanged but in active fingerprint the content differs from the original.

2.5. Perceptual Hash Functions

Discrete Cosine Transform (DCT) based Hash function

Discrete cosine transform (DCT) based functions utilize Fourier analysis theory in order to produce hash keys and like any other Fourier transform, it represents the finite sequence of data points as the weighted composition of sinusoids (cosine function) with different frequencies. DCT is similar to DFT operating on real data with even symmetry but on the other hand DCT in contrast with DFT uses real values. There are eight types of DCT and most common is type-II which simply is referred as DCT and type-III that is called inverse DCT.

Def. 2.2: *let x[n], n=0,..,. N-1, denote an N point real sequence the DCT type-II is defined:*

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

The scaling factor $\sqrt{\boldsymbol{2/N}}$ makes DCT matrix orthogonal but breaks the direct correspondence with a real-even DFT of half-shifted input. DCT is separable operation and may be computed along the axes separately.
In practice DCT of a time limited sequence may be computed by use of DCT matrix as below:

$$X(k) = \sum_{n=0}^{N-1} c(k,n).x(n) \ , \quad k = 0, \dots, N-1$$

And $c(k,n)$ is defined as:

$$c(k,n) = \sqrt{\frac{2}{N}} \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \ , \quad k,n = 0, \dots, N-1$$

For a square image I, The two dimensional is equal to one dimensional DCT while a single dimensional followed by the one dimensional DCT and it may be computed by using of DCT matrix D as below:

$$DCT(I) = D.I.D^{T}$$

Various properties of the DCT can be utilized to create perceptual image hash functions. Low-frequency DCT coe_cients of an image are mostly stable under image manipulations. [30]
That is because most of the signal information tends to be concentrated in a few low-frequency components of the DCT. This property is also utilized by the JPEG image compression standard. [31] There, the two-dimensional type-II DCTs of NxN pixel blocks are computed and the results are quantized. [3] N is typically 8 and elements closer to the top left corner represents lower frequencies in the horizontal and vertical direction in the image. DCT coefficients and their corresponding frequencies may be used as hash value of the image. Interest in DCT is because of its strong energy compaction property and this is important particularly in image processing applications like lossy image compression (JPEG is the simplest example). This lies in the fact that normally most of image energy is concentrated in lower frequency component of DCT. Performance of this method is discussed later.

Mar-Hildreth operator based Hash functions

As mentioned before a perceptual hash function extracts certain feature of image and uses them to produce fixed length string namely hash key. Mar-Hildreth operator algorithm core is extracting edges of image and use them to produce the hash key. Edge definition is depending on the context of application but it can be defined as contours or boundaries that separate different

regions of image and regions can be classified base on texture, color and luminance. Normally luminance is used and the result of edge detection algorithm is named edge map.
Edge map contains information about image classification feature as well as amplitude and orientation information and for luminance case, first and second derivatives with respect to spatial location are used for edge detection. So first derivative (gradient based) approach is to locate the positions that first derivative of luminance (gray scale level) are at local extremum and second derivative approach (Laplacian based method) is to identify zero crossing points of luminance function. Since these methods are used for two dimensional images some point must be considered. Discrete nature of digital image implies approximation of derivation. Image has additional property of direction so directionally sensitive edge detector is used for some applications.
Since edge detection is a high pass filter, image noise is a problem and wide range of algorithms have been proposed to deal with this effect. Generally detector error increases with noise
There is a trade-off between correct detection of edges and their location. The reason lies in the fact that good localization needs small spatial filter and conversely better noise suppression is obtained by spatially large filter.
Considering these points one simple approach is the one invented by David Marr and Ellen C. Hildreth which is convolving image with the Laplacian of the Gaussian function i.e. if we define Laplacian filter as provided by definition 2.3 then estimate of given image's Laplacian can be computed by convolution of filter kernel by image. The Laplacian-of-Gaussian image operator is sometimes also referred to as the Mexican hat wavelet due to its visual shape when turned upside-down.

Def. 2.3: *The Laplacian of Gaussian (LoG) ,denoted as* $h_c(x,y)$ *filter can be defined as*

$$h_c(x,y) = \nabla^2 g_c(x,y) = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where $\boldsymbol{g_c(x,y)}$ is Gaussian filter

$$g_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Then the LoG estimate of an image is obtained by convolving the Log with image $\boldsymbol{f(x,y)}$.

$$\nabla^2 f(n1,n2) = f(n1,n2) * g_c(n1,n2)$$

The implementation of LoG filter in digital domain may be achieved by sampling of kernel in spatial domain after choosing a value for σ. Computation cost can be reduced by utilizing the fact

that LoG filter is a separable filter and using 1-D convolution instead of 2-D counterpart. The Marr–Hildreth operator, however, has its drawbacks. It produces estimates that do not correspond to edges, so-called "false edges", and the localization error may be severe at curved edges.

Radial Variance based Hash functions

The Radon transform is calculated by taking the integrals of a two-dimensional image $\boldsymbol{f(x,y)}$ along a set of lines with different directions. The line integral along a particular direction with angle $\boldsymbol{\theta}$ is called a projection. The line integral of the function $\boldsymbol{f(x,y)}$ along the line L defined by direction $\boldsymbol{\theta}$ and the distance $\boldsymbol{x'}$ from the origin in the coordinates $(\boldsymbol{x'}, \boldsymbol{y'})$ is given by

$$R_\theta(x') = \int_L f(x'\cos\theta - y'\sin\theta\,, x'\sin\theta + y'\cos\theta)\,dy'$$

To apply radon transform to discrete images the line integral along the $d = x\cos\theta + y\sin\theta$ can be approximated by variance of pixels along the line projections. Luminance discontinuities caused by the edges are orthogonal to the projection line so we can define "radial variance vector, $R[\theta]$" as follow.

Def. 2.4: *Let* $\Gamma(\theta)$ *denotes the set of pixels* (x,y) *on the projection line corresponding angle* θ *and let* $I(x,y)$ *denote the luminance value of the pixel* (x,y)*, the radial variance vector* $R[\theta]$*where* θ*=0,1,..,179 is then defined by:*

$$R[\theta] = \frac{\sum_{(x,y)\in\Gamma(\theta)} I^2(x,y)}{\#\Gamma(\theta)} - \left(\frac{\sum_{(x,y)\in\Gamma(\theta)} I(x,y)}{\#\Gamma(\theta)}\right)^2$$

Since radon transform is symmetric it is sufficient to extract 180 points instead of 360 and by application of DCT to radial variance vector result will be improved.

Block Mean Value Based Hash Function

In 2006, Yang and colleagues devised a block mean value based perceptual image hash function in four variations with slight difference. The first method is described below as indicated by Zauner [2] since this method is used in our work.

Method 1

Normalize the original image into a preset sizes;

Divide the size-normalized image I into non-overlapped blocks $I_1, I_2, \ldots, I_N$, in which N is the block number equal to length of the final hash bit string;

Encrypt the indices of the block sequence $\{I_1, I_2, .., I_N\}$ using a secret key K to obtain a block sequence with a new scanning order $\{I'_1, I'_2, \ldots, I'_N\}$;

Calculate the mean value sequence $\{M_1, M_2, \ldots, M_N\}$ from corresponding block sequence $\{I'_1, I'_2, \ldots, I'_N\}$ and obtain the median value M_d of this sequence as
$M_d = \text{median}(M_i)$, $i = 1, 2 \ldots N$
Normalize the mean value sequence into a binary form and obtain the hash values h as:

$$h(i) = \begin{cases} 0, & M_i < M_d \\ 1, & M_i \geq M_d \end{cases}$$

Methods 2, 3, 4

In the second variation for increasing the robustness 50% overlapping is used between segments and in the third variation for robustness against any flipping attack a rotation is added to the first method. The fourth method is a combination of method two and three with 50% overlap between blocks and rotation and it is of a great concern that rotation increases the complexity of algorithm [2].

2.6. Comparison Perceptual Hash Functions

Since robust hashing is an approved method for analysis of image sets due to their low false alarm rates and reasonable computational complexity their evaluation and validation is desired for their integration in forensic analysis tools [4]. For this purpose Rihamark is used as a benchmarking framework for perceptual image hash functions in terms of speed, inter score distribution (discriminative capabilities) and intra score distribution (robustness) [3]. Several hash functions performance is evaluated by this benchmarking tool namely:

- DCT (Discrete Cosine Transform)
- Marr-Hildreth operator
- Radial variance
- Block mean value based perceptual image hash function

In terms of speed, Block mean value method is fastest and DCT is the slowest by far. Although Marr-Hildreth is the most discriminative method and DCT the second most, test values can be improved by combining different functions.

From robustness point of view several image manipulation techniques are considered:

- Horizontal flipping
- Resizing
- Jpeg Compression
- Rotation

According to the results none of the methods may be considered robust in case of horizontal flipping and against resizing radial variance based method shows poor performance. In case of resizing all methods except Marr-Hildreth give satisfactory results and for rotation block mean value based method is the best.

It is a matter of application that which method may be chosen since it depends on which characteristic is desirable although block mean based perceptual hash function is the fastest and it is either most robust or approximately equal with other functions [3].

Steinbach and colleagues [4] have evaluated an optimized block based hash function which is optimized by segmentation of hash to four sub areas. The segments' mean value is used for decision and introducing automatic mirroring of the image during the hash calculation in such a manner that darkest part of image is on upper left. This leads to resilience of algorithm to any type of mirroring. Also they have used a weighted distance in addition to hamming distance in order to thresholding and decision making. This method may be a suitable replacement for alternative cryptographic hashes [4].
In this thesis work, block mean value based hash function is used and it is proper to have a detailed review of this function.

2.7. Survey of related works

By reviewing several works which has used different approaches for perceptual hashing, it can be summarized in four different categories:

- Statistic approach
- Relation based approach
- The coarse image represents
- Image feature extraction approach

Statistic approach

The calculation of the parameters such as mean, variance and the intensity of the images blocks were obtained by using the statistic computation.
The results of such a statistic approach must have good properties by small perturbations of the images. The main disadvantage of this method is easy to modify an image without the change of the intensity of its histogram. It caused the weakness security problems of any scheme which comply upon the intensity statistics. Venkatesan et al. [1] develop an image hash based on an image statistics vector extracted from the various sub-bands in a wavelet decomposition of the image. This paper is based on the observing the statistic such as averages and variance of the sub-bands would be remain stable under a significant modification of the content preserving to the image[16,17,18].

Relation based approach

Lin and Chang [19] proposed a typical relation-based method to image authenticate which is relies on JPEG compression. The digital signature extracted by using the invariant relation between each two discrete cosine transform (DCT) coefficients, which has the same position of the two different 8x8 blocks. The result shows that the invariance properties would preserved before and after JPEG compression during the perceptually lossless.
This method is also robust against of JPEG compression, and it would remain vulnerable to various other trivial perceptually modifications. (It must be considered that the nature distortion of statistical is different with the blur which caused by compression).
Lately, Lu and Liao [20] proposed a “structural digital signature” by observing the sub-band wavelet decomposition which parent and the child node are uncorrelated but they are statistically dependent.
The result of their observation shows that for the several content preserving manipulations the difference of the magnitude of wavelet coefficients at their consecutive scales remain largely preserved. (i.e., a parent node and their four child nodes)
It produced such a robust digital signature while the identifying the parent-child pairs and subsequently encoding the pairs form.
The achievement of the method [20], however is very sensitive to global insignificant rotation and furthermore local geometric attacks.

The coarse image represents

This method proposed a robust hash based on preserving selected (low frequency) discrete cosine Transform (DCT) coefficients [21]. The method of Fridrich and Goljan based on the observing the significant modification to the low-frequency DCT coefficients of the image made change the appearance of the image dramatically.
Mihcak and Venkatesan [22] proposed another image hashing algorithm which used an iterative approach to make binary the DC sub-band (lowest resolution wavelet coefficients) in a wavelete decomposition of the image.
Swaminathan et al. propose an image hash [23] based on selecting rotation-invariant Fourier–Mellin coefficients.
Although their approach was down well under large amounts of global rotation, meantime the robustness of Fourier- Mellin coefficients must be desired under many other classical signal processing distortions.

Image feature extraction approach

The hypothesis of this method is based on the use of vision based feature points for perceptual image hashing [24]-[26]. The aim of the robustness of the schemes in [24] and [25] is far as unsatisfactory for robustness of application. Although the corner-based image features in [26] is robust comply under a large class of attacks, beside an expensive search is necessary to handle the geometric manipulations.

The results obtained from the above methods shows that a common weakness of the methods [16]-[26] is poor robustness against of the geometric attacks, particularly the most wasted ones is as cropping.
Recently Kozat *et al.* [27] proposed using low-rank matrix approximations obtained via the well-known singular value decomposition (SVD) for image hashing. Since the SVD-based hashing scheme in [27] exhibits good geometric attack robustness, which this method described as such a different images mapping to the same hash value.
Lately, Lee *et al* proposed non-negative matrix factorization (NMF) [28] and Vishal Monga and M. Kıvanç Mıhçak[14] developed the robust image hashing algorithms based on recently proposed dimensionality reduction technique by using the NMF which is distinguished from the traditional matrix approximation method such as QR and SVD, while it used the non-negativity constraints. They proposed [14] that the geometric distortions on images result in approximately additive and independent, identically distributed noise on NMF vectors, Mean time they exploit

this mentioned purpose to obtain pseudorandom linear statistics of NMF vectors which is significantly enhanced hash robustness since the hash allowed to use the small length.

2.8. Metrics

As mentioned earlier there should be measures for comparison of hash values. The most common functions which are used for similarity/ distance scoring are hamming distance, BER and PCC. In this sequel the measure may indicate similarity between two strings as it is the case for PCC or may specify distance between them such as hamming distance.

Hamming Distance

Hamming distance may be defined as below:

Def. 2.2: *Let A denote an alphabet of finite length.* $x = (x_1, \dots, x_n)$ *denotes an even length-string, whereas* $x \in A$. *The same holds true for* $y = (y_1, \dots, y_n)$. *Then the hamming distance* Δ *between x and y is defined as:*

$$\Delta(x, y) = \sum_{x_i \neq y_i} 1 \quad , i = 1, .., n.$$

Normalized Hamming Distance

For comparison different hamming distances regardless of their correspondent string length normalized hamming distance may be defined as below:

Def. 2.3: *Hamming distance can be normalized by with respect to length n of string as:*

$$\Delta_n(x, y) = \frac{1}{n} \sum_{x_i \neq y_i} 1 \quad , i = 1, .., n.$$

Calculation of hamming distance for binary strings is possible with XOR operation. Two examples for hamming distance is provided in table 1 for binary and Latin alphabet.

String 1	String 2	Hamming Distance
11011001	01010101	3
fake	Take	1

Table 1: Hamming distance examples

Equality Percentage (EP)

Equality percentage may be computed as:

$$EP = 100 \times \Delta_n$$

EP has a range between 0 and 100 percent. Higher value indicates more similarity and lower value means more distinction between two perceptual hash values.

Bit Error Rate (BER)

Def. 2.4: *ρ is BER and defines as number of i bit errors of the perceptual hash normalized by length of the perceptual hash value k.*

$$\rho = \frac{i}{k} \quad , \quad i \in \{0,1, \dots, k\}$$

i is equal to hamming distance of perceptual hash values and obviously $0 \leq \rho \leq 1$. Lower ρ yields perceptually similar images with a minimum of 0 which indicates similar hashes.

Peak of Cross Correlation (PCC)

Cross correlation is a measure of similarity between two sequences i.e. higher cross correlation indicates more similarity between two discrete sequences. Definition of cross correlation for two discrete signals is presented below:

Def. 2.5: *For two discrete finite, with length of N, sequences x(n) and y(n) the cross correlation is defined as:*

$$r_{xy}(k) = \sum_{n} x(n)y(n+k) \quad , \quad n = 0,1,..,N-1$$

x(n) and y(n) are deterministic real valued sequences and k is the time shift.

Cross correlation can be normalized with respect to mean values of x(n) and y(n) which are denoted as m_x and m_y.

Def. 2.6: *For two discrete finite, with length of N, sequences x(n) and y(n) the normalized cross correlation is defined as:*

$$\frac{\sum_n [x(n) - m_x][y(n+k) - m_y]}{\sqrt{\sum_n [x(n) - m_x]^2 \sum_n [y(n) - m_y]^2}} \quad , \quad n = 0,1,..,N-1$$

PCC is the maximum value of cross correlation between these two sequences.

2.9. False Negative/False Positive

As briefly discussed in introduction this work's intention is to raise false positive errors for evaluation purposes. According to Sheskin [5]:
In statistics, a type I error (or error of the first kind) is the incorrect rejection of a true null hypothesis. A type II error (or error of the second kind) is the failure to reject a false null hypothesis. A type I error is a false positive and a type II error is a false negative. This means false positive occurs when an unknown object is identified as known one and false negative occurs when and a known object is identified as unknown.

2.10. Image hash Spoofing

As mentioned before perceptual hash algorithms offer certain degree of robustness by extracting some perceptual features from multimedia object. They are different from generic hashing algorithms in two important points. First they are sensitive only to significant content modification and tolerant to medium level of content preserving image processing techniques. Second difference lies in the fact that they usually utilize a secret key for hashing process in order to increase resilience to malicious manipulations and the length of the secret key

determines the degree of protection without any compression. The latter case lies in the fact that algorithms are publicly known and using them solely is not secure.

According to Li Weng and Perneel two types of attack is possible [9].

- Counterfeiting both hash and content
- Gradually introduction of changes until the content is severely distorted

As suggested by the authors incorporation of a secret key in order to generate hash key protects perceptual hash algorithms against malicious manipulations. The main drawback is that it would not be easy to establish information protection protocol easily. Another problem is that the hashing process would not be efficient.

According to Wikipedia "in the context of network security, a spoofing attack is a situation in which one person or program successfully masquerades as another by falsifying data and thereby gaining an illegitimate advantage". Since here our objective is to develop a method of producing an image hash which closely resembles another image hash code, based on robust image hashing, for evaluation false positive alarms, our developing algorithm is named image hash spoofing. This algorithm must be robust in the sense of showing resilience to different image manipulation techniques such as cropping, compression and scaling. Since our approach consists of modifications to an image in order to imitate another image which causes negative false alarms the terms spoofing and attacking is used interchangeably.

3. Implementation

3.1. Requirements

As previously mentioned our goal is to develop a mechanism that an image masquerades another image and this process must be done automatic and robust in the sense of that it has no limitation against any normal image processing techniques. In order to achieve this, MATLAB® is used as a powerful interactive programming tool for implementation. All images are of 256×256 size in JPEG format and hash key length of 256 bits. As a measure of similarity between hash keys, computed by block mean based algorithm, hamming distance is used.

3.2. Preliminary Work

As first step a database of approximately 150 000 hashed images is analyzed and the task was choosing 30 random images and comparing each to the all other pictures. This has been done in MATLAB by uniformly random selection of 30 images and calculation of hamming distance in order to compare with the entire of database. Regarding the threshold of 16 is imposed on hamming distances and the images with hamming distance below the threshold is considered similar. The result is shown in table 2 and a total of 10 collisions have occurred according the thresholding criterion.

In order to have a better understanding, histogram of hamming distances for selection number 14 has been shown in figure 2 and a magnified version for distances under threshold in figure 3. It must be noticed that corresponding normal distribution have been superimposed on the histogram. In figure 4, boxplot of hamming distances for selected all images has been depicted. In the statistical approach it is a graphical way to represent a set of data through quartiles as shown in figure 1 provided by www.wellbeingatschool.org. In figure 5, boxplot of selected image No.14 is shown.

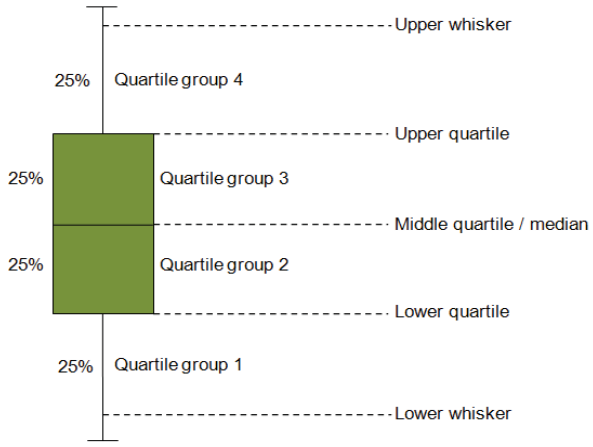


Figure 1: Boxplot

Image No	Report
121323	Number of similar images for selection No 1 is: 1
134885	Number of similar images for selection No 2 is: 1
18910	Number of similar images for selection No 3 is: 0
136014	Number of similar images for selection No 4 is: 3
94167	Number of similar images for selection No 5 is: 0
14526	Number of similar images for selection No 6 is: 0

41473	Number of similar images for selection No 7 is: 1
81438	Number of similar images for selection No 8 is: 0
142586	Number of similar images for selection No 9 is: 0
143685	Number of similar images for selection No 10 is: 0
23471	Number of similar images for selection No 11 is: 0
144534	Number of similar images for selection No 12 is: 0
142535	Number of similar images for selection No 13 is: 1
72279	Number of similar images for selection No 14 is: 2
119173	Number of similar images for selection No 15 is: 0
21129	Number of similar images for selection No 16 is: 0
62806	Number of similar images for selection No 17 is: 0
136365	Number of similar images for selection No 18 is: 0
117970	Number of similar images for selection No 19 is: 0
142881	Number of similar images for selection No 20 is: 0
97649	Number of similar images for selection No 21 is: 0
5318	Number of similar images for selection No 22 is: 0
126447	Number of similar images for selection No 23 is: 0
139084	Number of similar images for selection No 24 is: 1
101073	Number of similar images for selection No 25 is: 0
112838	Number of similar images for selection No 26 is: 0
110663	Number of similar images for selection No 27 is: 0
58408	Number of similar images for selection No 28 is: 0
97610	Number of similar images for selection No 29 is: 0
25492	Number of similar images for selection No 30 is: 0

Table 2: 30 selected images hamming distance to other images

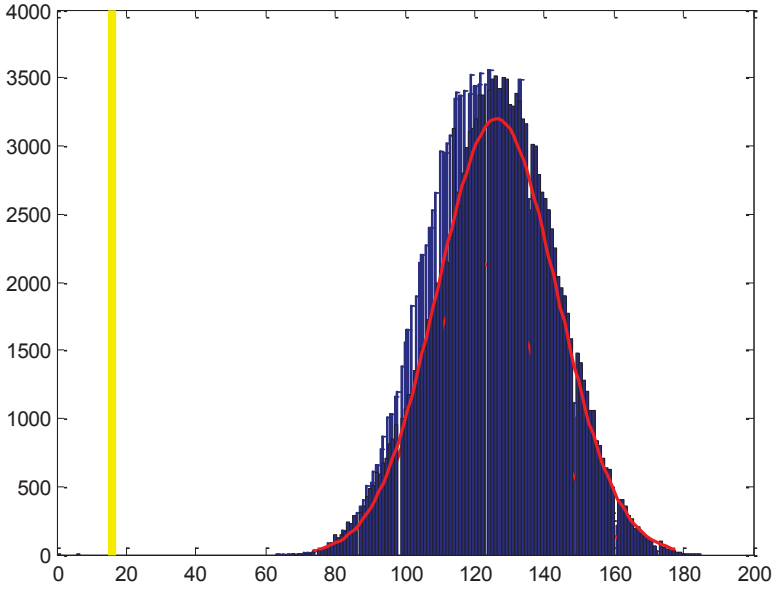


Figure 2: Hamming distance histogram for selection No.14. Yellow line is the threshold imposed

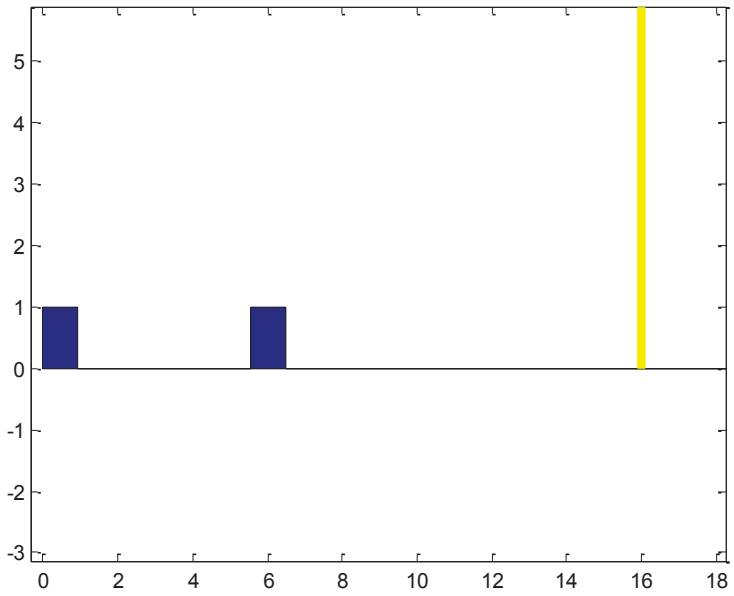


Figure 3: Hamming distance histogram in selection No.14 for values below the threshold

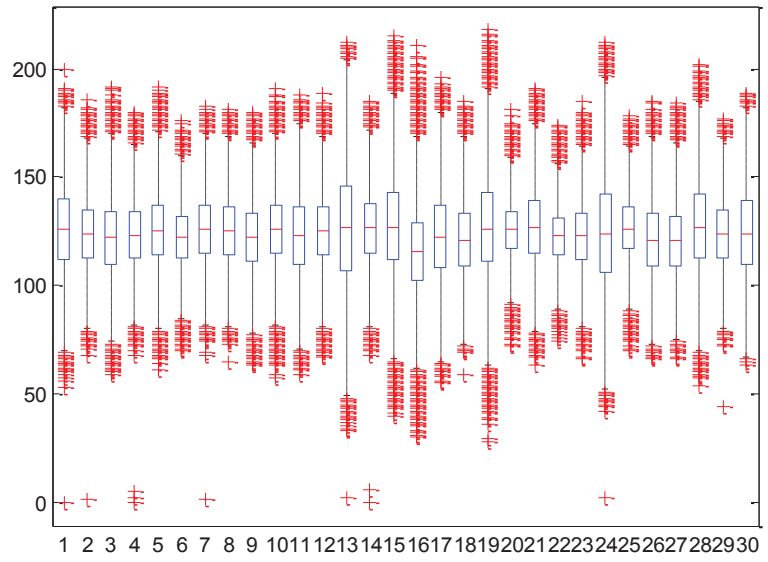


Figure 4: Hamming distance boxplot for all selections

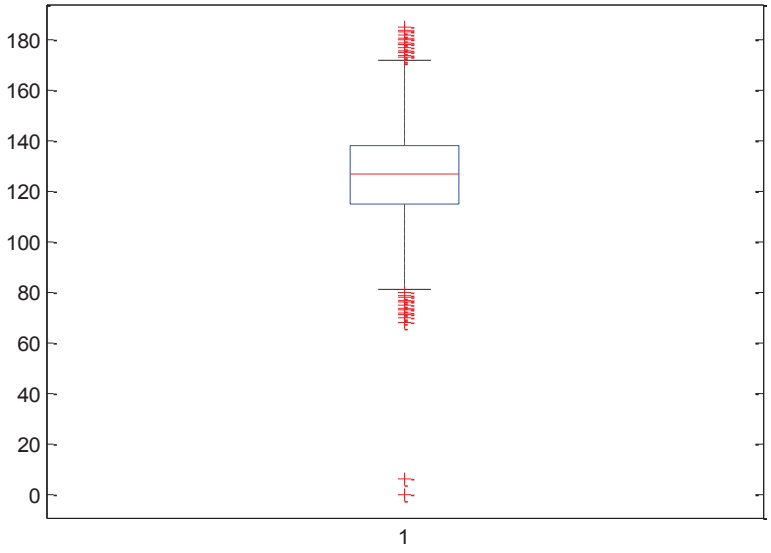


Figure 5: Hamming distance boxplot for selection No.14

After extracting the original random images from the database and checking them visually it is clear that only one true collision has been occurred which indicates a negative false alarm. The other collisions were the same image with different sizes. Collided images have not been shown because the pornographic nature of images.

3.3. Design

In this section we are going to describe how an image, let's call it an attacked image, masquerades another image that to be called favorite image. This spoofing attack must be done in a way that loss of perceptual quality of attacked image would not be significant. In order to achieve this, black mean based hash will be used to produce a similar hash bit to favorite image. In general the whole process can be divided in two stages:

3.3.1. Stage1:

All image are 256×256 pixels and gray scale and if not they will be resized and converted to 256×256.Then hash key attacked image (figure 6) is calculated by block mean based method as below:

- First Favorite image is segmented to 16×16 grid as figure 7.

Figure 6: Target image

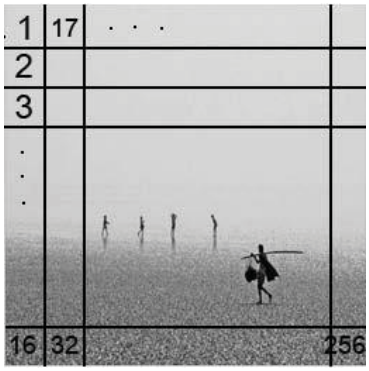


Figure 7: Segmented target image

- Now mean value of each block is calculated. M_i , $i = 1, \ldots, 256$
- Median of M_is is calculated so, M_d is median of M_is.
- Now using following criteria the hash key for the image is calculated as below:

$$h(i) = \begin{cases} 0, M_i < M_d \\ 1, M_i \geq M_d \end{cases}$$

- Now we convert this hash key to an image mask as shown in figure 8.

Figure 8: Target image hash

- All the above steps will be done on attacked image too as shown in figures 9 and 10.

Figure 9: Attacked image

Figure 10: Attacked image hash

- As first modification on attacked image we use mask which is obtained from favorite image to manipulate intensities of attacked image according to favorite mask. This is done by multiplication of pixel intensities on attacked image to numbers higher than 1 where mask is white and lower than 1 in locations that mask is black. It has to be mentioned that hash key is 16×16 so before this operation it must be resized to 256×256. After this operation since intensities have changed then picture will appear unnatural as it can be seen in figures 11 and 12 along its new hash key then some compensation is needed.

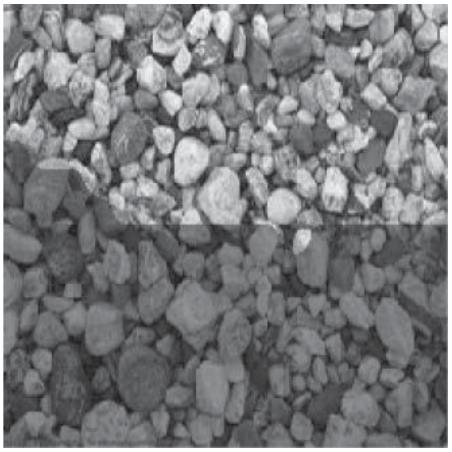


Figure 11: Masked attacked image

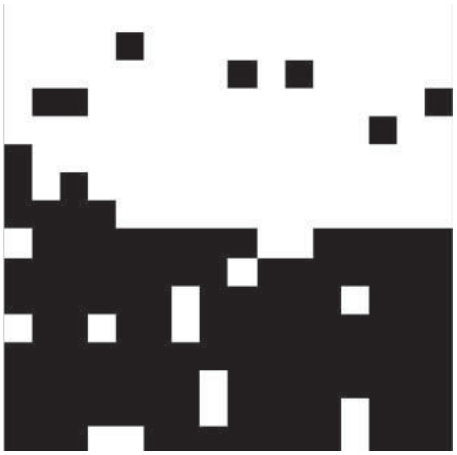


Figure 12: Masked attacked image hash

- Now by use of Gaussian filter the FAVORITE image's mask will be blurred to overcome this problem but higher degree of blurring means more hamming distance at the end so we have a tradeoff here. In our implementation degree of blurring will be controlled by BD parameter. After blurring the attacked image mask and its corresponding masked image is shown at figures 13 and 14.

Figure 13: Filtered attacked image hash

Figure 14: Filtered attacked image

Now the first stage has finished an in this stage we will get lower hamming distance. Further operation in order to obtain hamming distance will be done in the next stage. Here another option is utilized which is instead of using extracted mask from target image, it is possible to use predefined masks although the resulting hamming distance is not as good as the extracted mask method. Some of these masks are depicted in figure 15. Software searches for nearest mask to target image's mask in terms of hamming distance and use it for manipulating attacked image.

-

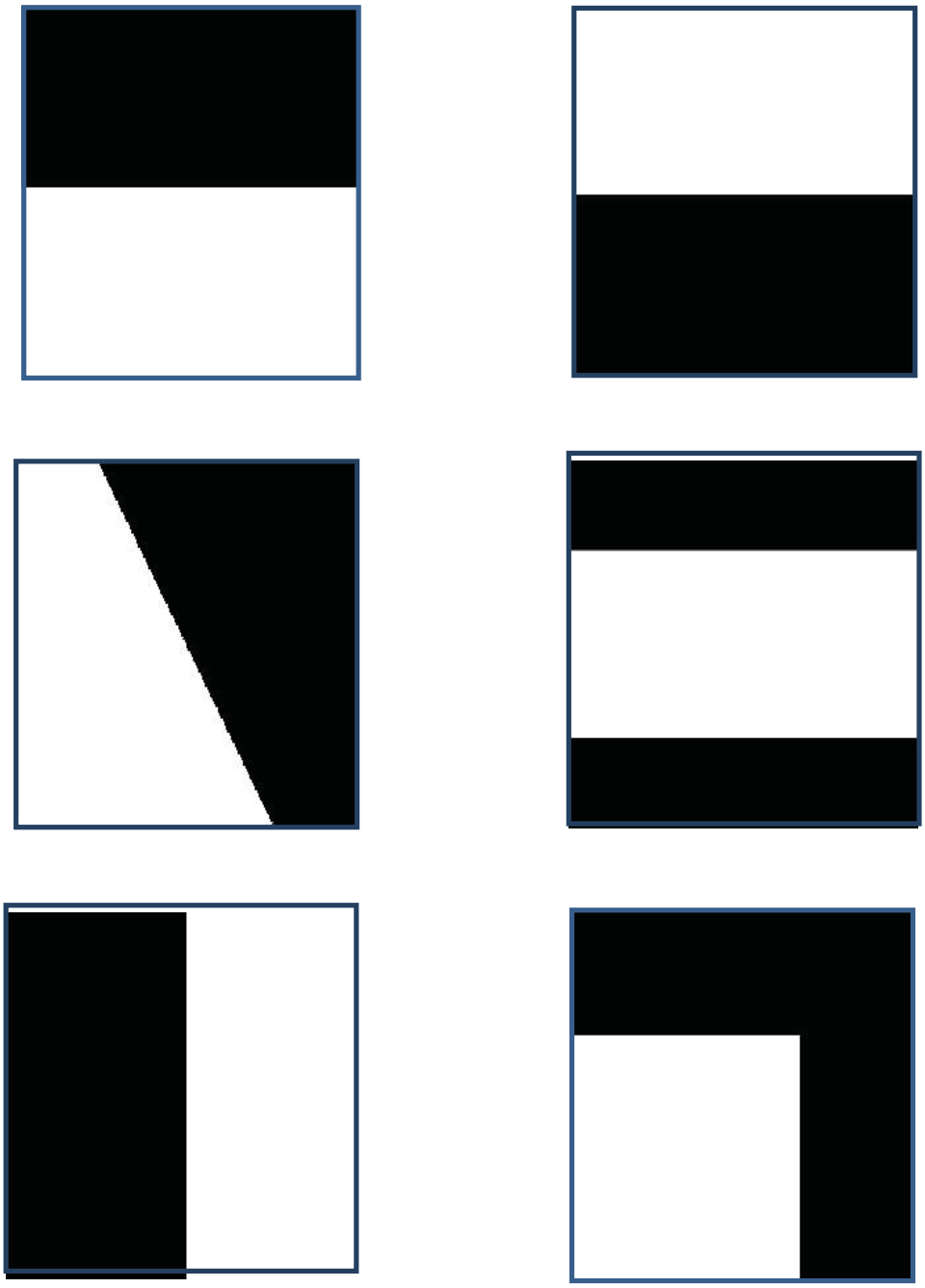


Figure 15: Samples of predefined hash masks

- There is also an option that we can define the minimum hamming distance required to enter the stage one i.e. if an image's hash key has a hamming distance below this threshold to target image software directly goes to stage two and no operation is done in terms of intensity manipulation and filtering.

3.3.2. Stage2:

In this stage the produced attacked image from the previous stage is subjected to changing intensities in those particular blocks that their mean value is different from mean value of target (favorite) image. This process is done by adding 1, 2, … to intensities of those blocks that their mean value will exceed the median if they are lower than median or fell below the median if they are above the median. There are two limitations for these operations i.e.:

- Changing is done in a way that median remains unchanged. It means if a changing of value for a particular block changes the median then another change is done in the reverse direction in order to reach a target value in hamming distance.

- Variance of each block is used as a measure of quality which limits number of change for each block change. This works as a feature in the software i.e. if keeping the quality of the attacked image is important for us then regardless of what is the target hamming distance is, a particular block is subjected to predefined number of intensity changes according the value of the block variance as shown in table 4. These values have obtained by testing on many images. The reason behind is higher the number of variance, changes in the block mean value has lesser effect in human perception.

Values in table 4 have been obtained by several runs of the program. For more information refer to appendix A.

Block Variance	Maximum allowed distance to change
0-100	2
100-1000	4
1000-20000	5
20000-30000	8
30000-2500000	15 (varies from 10 to 20)
≥2500000	24

Table 3: Variance criterion for quality control

3.4. MATLAB® Code

In this Master thesis entire program has been written in MATLAB®. About the code there are some important notes that must be noticed. First is that in the code the burden of the job is on two functions which constructs core of the software namely *Stage1M.m* and *Stage2M.m*. In *Stage1M.m* software calculates the hash bits for original and target image and it uses some input parameters such as *HammingDistanceLimit1*, *BD*, *wh*, *bl* and *method*. *HammingDistanceLimit1* is threshold for attacked image i.e. if hamming distance of attacked image fell below this value then stage 1 is discarded and attacked image after computation of hash bits directly will be send to stage 2. *BD* is controlling parameter for blurring and higher blurring means higher final hamming distance and a typical value for this parameter in between 20 and 30. *wh* and *bl* are the intensity multiplication coefficients for light and dark areas of hash key respectively and value equal to unity means image is remained unchanged and typically for *wl* value is above one and for the *bl* is below one. *method* defines whether target image or predefined masks is used for masking the image and value equal to 1 means that target image will be used for masking the attacked image and *method* equal to 2 means software will use one of predefined masks with lowest hamming distance to target image.
There are other functions that are used by mentioned main functions as below:

- *HashDist.m*:
  Calculates hamming distance for two images' hash codes.

- *ChangeMean.m*:
  Changes the mean value of the desired block in positive or negative direction.

```
function
[I1,H2,Dist2,Dist1]=Stage1M(OriginalImageName,cond,BD,wh,bl,method,ShRe,SaRe)
%Input Arguments:
%OriginalImageName: Attacked Image name
%cond: Hamming distance limit1
%BD: Blurring Degree (Gaussian LPF)
%wh: White area multiplication coefficient
%bl: Black area multiplication coefficient
%method: 1 (Uses target image hash key)-2 (Uses predefined hash key)
%ShRe: Show result (1) Dont show (0)
%SaRe: Save Result (1) Dont Save (0)
%Output Arguments:
%I1: Modified attack image after stage1
%H2: Hash bit of target image
%Dist2: Hamming Distance after modifocation
%Dist1: Hamming Distance before modification
```

```
function [I,Dist1]=Stage2M(I,H0,limit,BEST,name,ShRe,SaRe)
%Input Arguments:
%I: Attacked Image name after stage1
%limit: Hamming distance limit2
%BEST: 'yes'quality is preserved regardless of hamming distance limit
%      'no' hamming distance is reduced till reaching hamming distance
%      limit 2
%name: Target image name
%ShRe: Show result (1) Dont show (0)
%SaRe: Save Result (1) Dont Save (0)
%Output Arguments:
%I: Modified attack image after stage2
%Dist1: Hamming Distance after modification
```

4. Verification and Results

4.1. Method 1

In this section the implemented spoofing program is tested for a set of 163 images. Here the favorite (target image) has been used in order to obtain the mask. In figures 17-24 and tables 4-7 step by step outcome can be observed for four cases. The target and attacking image is different perceptually and from hash key point of view.

For the first case hash pattern of original image is significantly different from attacking image in the top and bottom of image as it can be observed in figure 17 but the variance of intensities in the attacked image is significantly high and this allows high number of block mean changes. In order to mask attacking image by original image mask its hash mask is smoothed by a Gaussian low pass filter and by using scaling factors for black and white regions the final mask is obtained as titled in figure 17 as final mask of original image by offset. Applying this mask to target image, we reach the end of stage one and by changing the mean value of the selected blocks consequently the overall result can be in figure 18. The visual effect of process can be observed as shadows on the image but there is no artifact and damage to image texture and blurring effect of smoothing filter is not significant. At the same time hamming distance between original image and attacking image is reduced significantly by 103 units at the expense of appearing shadows.

In second and third case again the hash keys of target image and attacking image are significantly different and shadows appear on particular areas that original image has low intensity. The hamming distance reduction is high with 96 and 93 points for second and third example respectively. For the fourth case hash key of target and attacking image is more similar compare to previous examples and original image hash has a dark pattern approximately in all areas. As it is expected the final image appears a darker version of the image before attack with hamming distance reduction of 77.

The dark regions on the final regions may be explained by high scaling factors for black and white region which are respectively are chosen 2 and 0.5 which means in the attacking image is intensities according to target image hash key are two times brighter in white regions and one half in black regions. These values are chosen to achieve high hamming distance reduction with a target of 16. For pictures with similar objects such as face the result would be better in terms of intensity distortions.

In order to have a better understanding in figures 25 and 26 histogram of hamming distance reduction after first, second and first plus second stages as well as their boxplot is provided. Histogram and boxplot graphs indicate that most of hamming distance reduction occurs at the first stage with mean of 74.36 whereas in stage 2 is only about 7 degree reduction. Maximum reduction of hamming distance in stage 1 is 106 while in stage 2 is equal to 33. Total hamming distance reduction based on block mean value has a mean of 81.41, maximum value of 106 and

median equal to 85. This significant reduction in hamming distance can be described by choosing the proper attack image set. Images with higher block variance, in other words with many small objects, is a proper choice in order to be chosen as attacking image. As it can be observed in provided figures, images perceptual quality in terms of texture has been preserved. In order to have a measure of contaminated noise in the images peak signal to noise ratio (PSNR) have been calculated and its respective histogram is shown in figure 16.

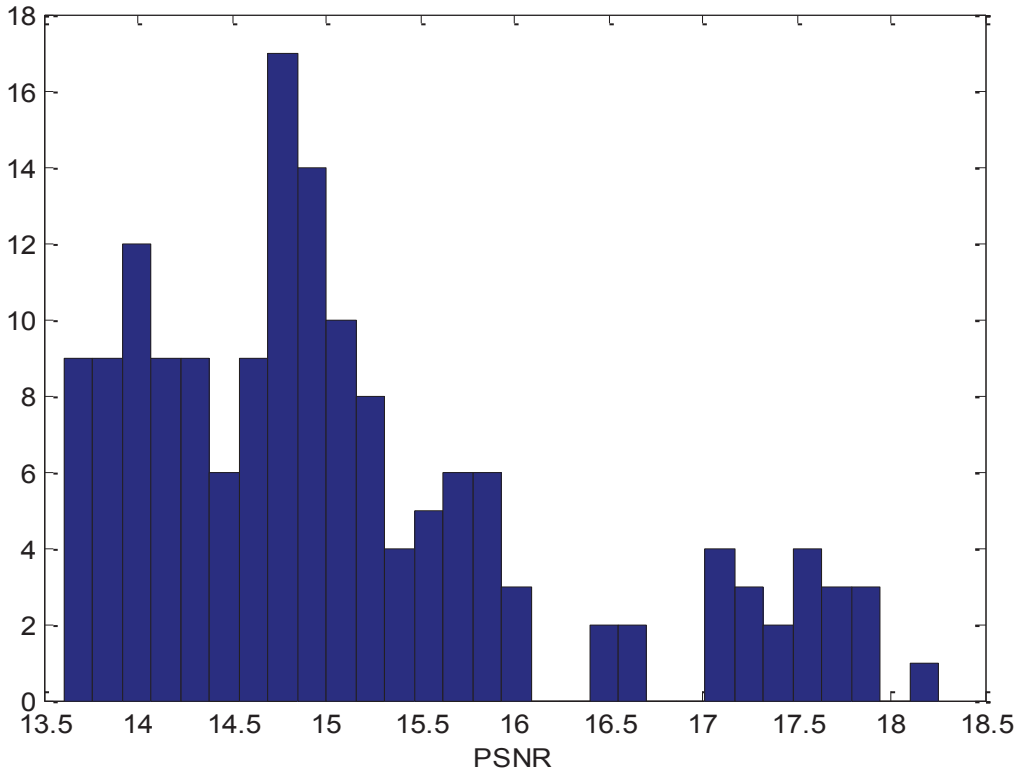


Figure 16: PSNR histogram for 163 attacked image

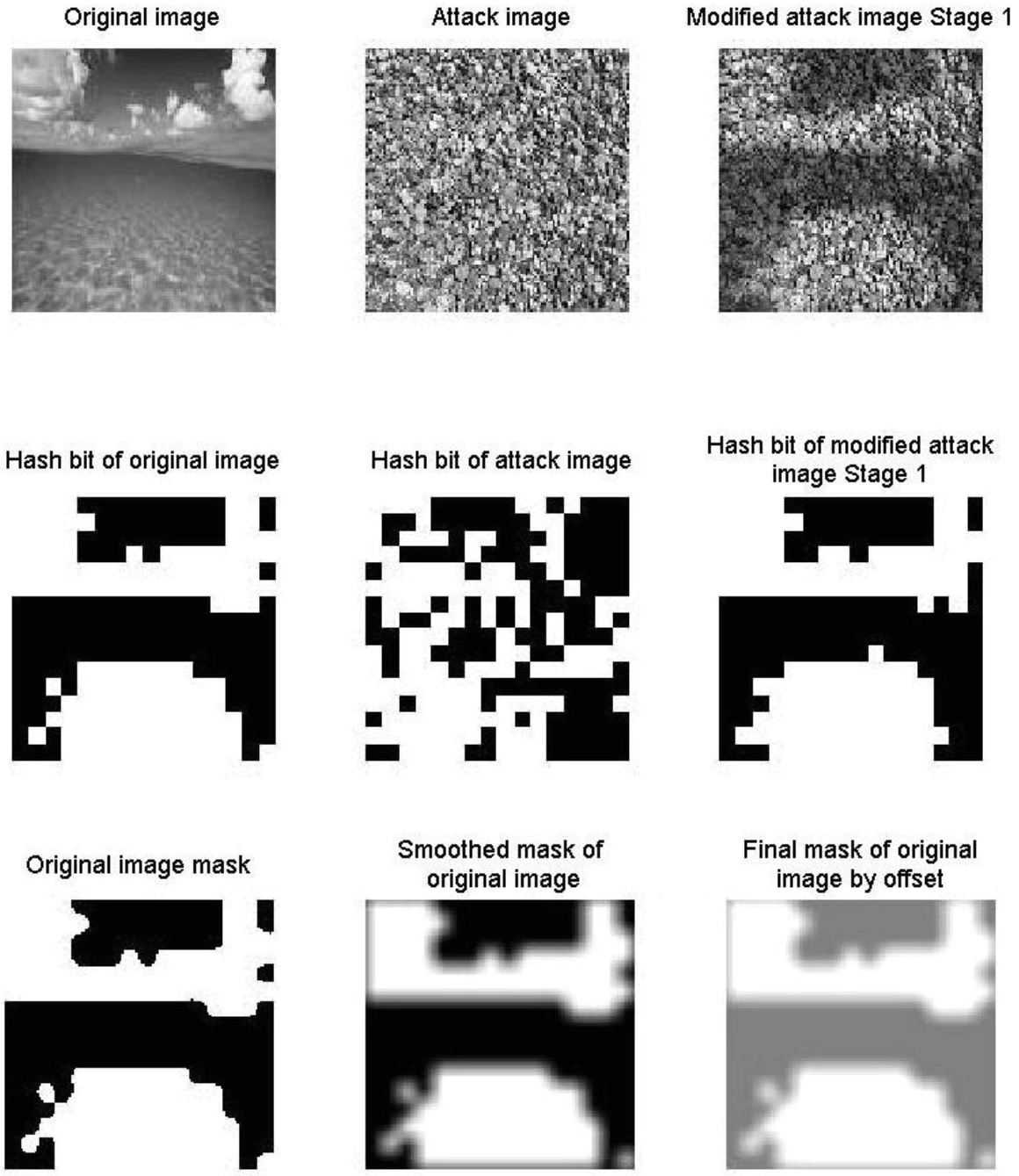


Figure 17: Method 1-image 1

Figure 18:Method 1- Image 1

Hamming Distance Before Attack	Hamming Distance After First Stage	Hamming Distance After Second Stage
113	10	10

Table 4: Hamming distance reduction for Method 1- Image 1

In the program following quantities have been used:

HammingDistanceLimit1 = 20 (Limit for stage 1 and below this value SW discards stage 1)
HammingDistanceLimit2 = 16 (Target Hamming Distance)
BD = 20 (Blurring Coefficient)
Wh =2 (Intensity Multiplication Coefficient in bright parts of the mask)
Bl =0.5 (Intensity Multiplication Coefficient in dark parts of the mask)
Quality = on

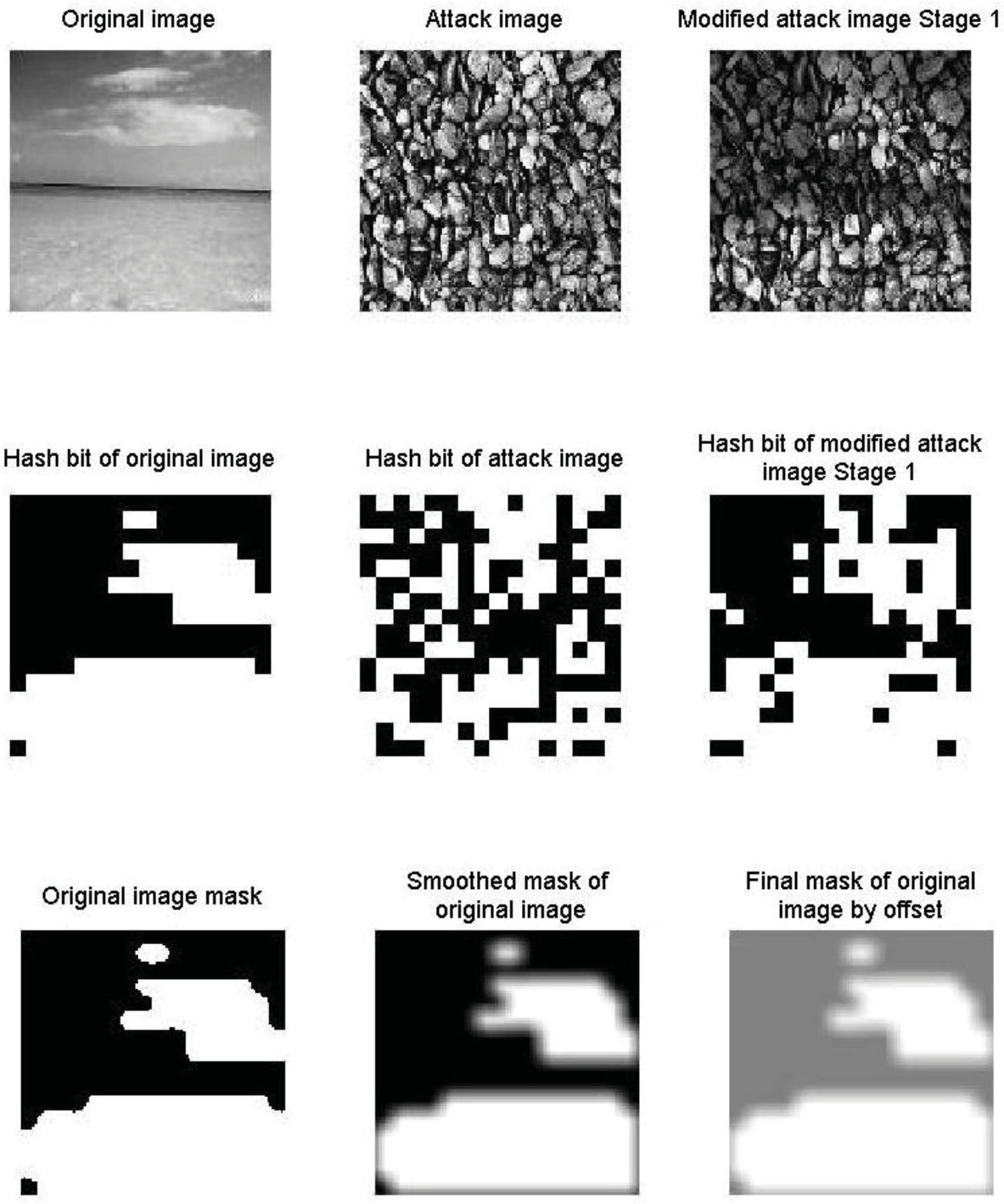


Figure 19: Method 1- Image 2

Figure 20: Method 1 – Image 2

Hamming Distance Before Attack	Hamming Distance After First Stage	Hamming Distance After Second Stage
113	40	17

Table 5: Hamming distance reduction for Method 1- Image 2

In the program following quantities have been used:

HammingDistanceLimit1 = 20 (Limit for stage 1 and below this value SW discards stage 1)
HammingDistanceLimit2 = 16 (Target Hamming Distance)
BD = 20 (Blurring Coefficient)
Wh =2 (Intensity Multiplication Coefficient in bright parts of the mask)
Bl =0.5 (Intensity Multiplication Coefficient in dark parts of the mask)
Quality = on

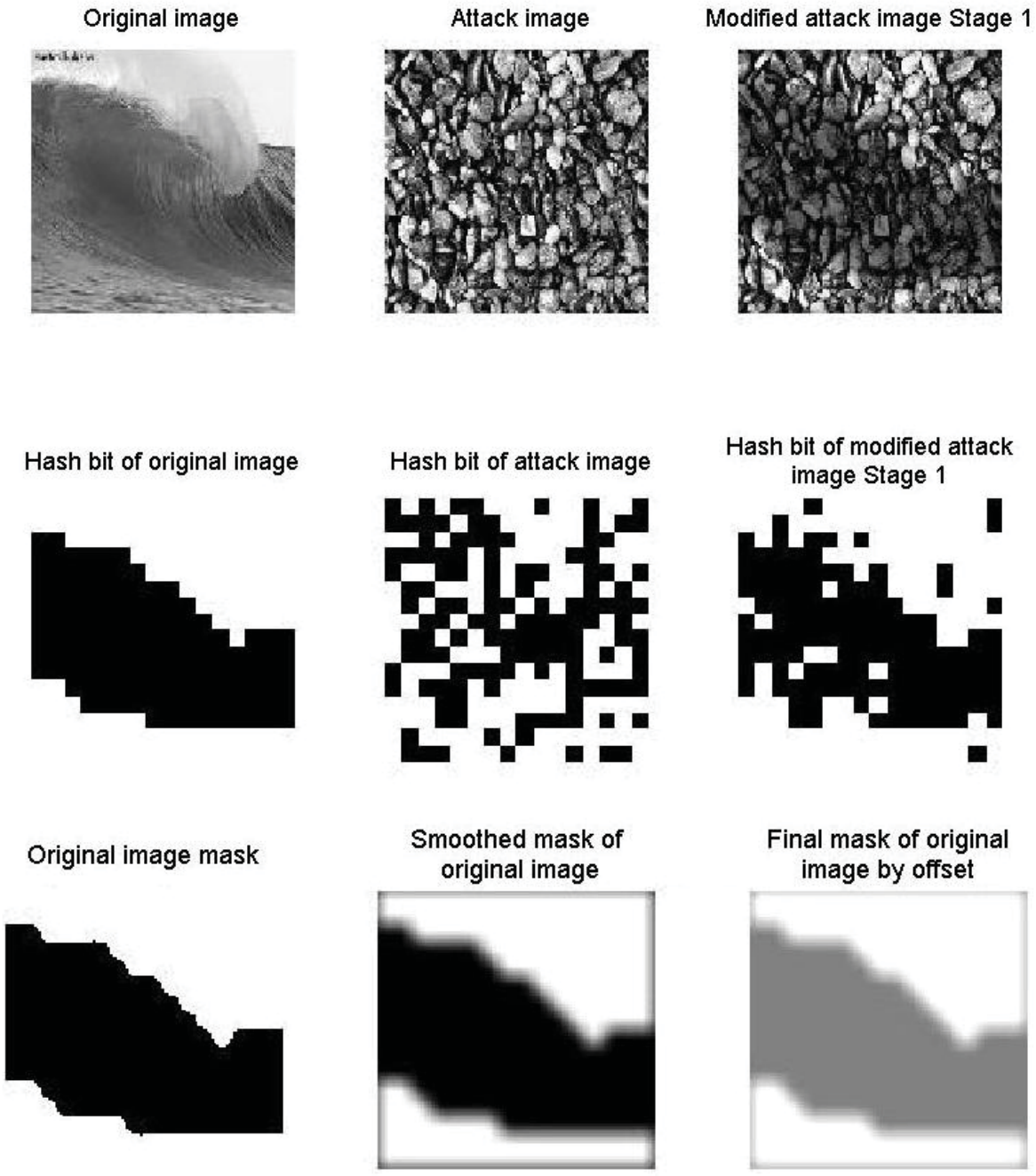


Figure 21: Method 1 – Image 3

Figure 22: Method 1 – Image 3

Hamming Distance Before Attack	Hamming Distance After First Stage	Hamming Distance After Second Stage
109	33	16

Table 6: Hamming distance reduction for Method 1- Image 3

In the program following quantities have been used:

HammingDistanceLimit1 = 20 (Limit for stage 1 and below this value SW discards stage 1)
HammingDistanceLimit2 = 16 (Target Hamming Distance)
BD = 20 (Blurring Coefficient)
Wh =2 (Intensity Multiplication Coefficient in bright parts of the mask)
Bl =0.5 (Intensity Multiplication Coefficient in dark parts of the mask)
Quality = on

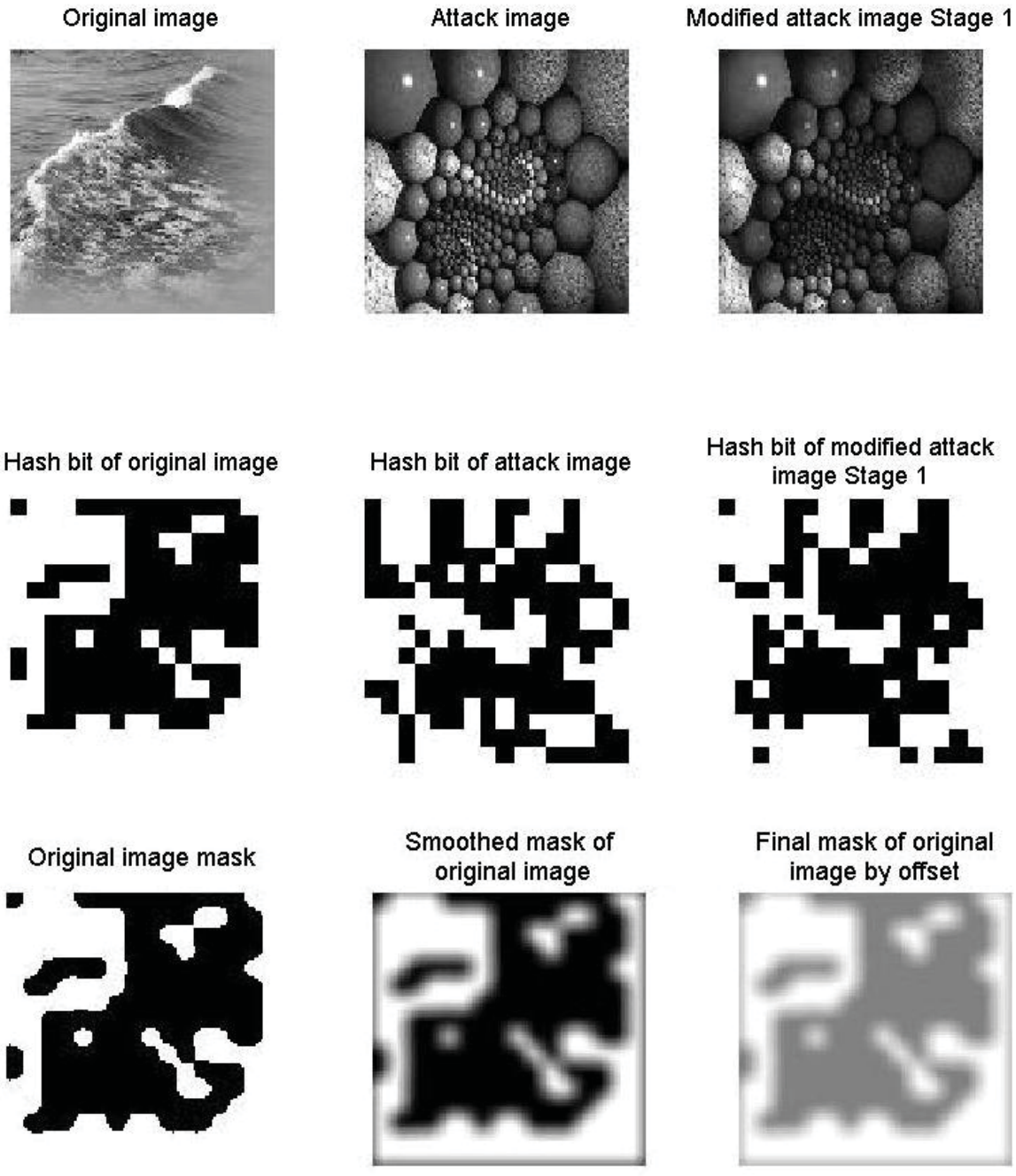


Figure 23: Method 1 – Image 4

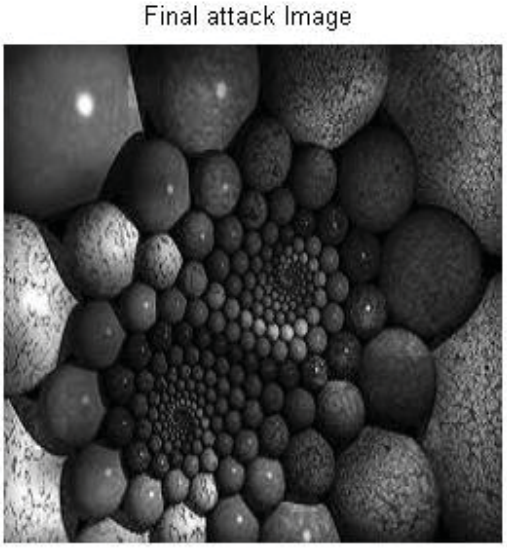


Figure 24: Method 1 – Image 4

Hamming Distance Before Attack	Hamming Distance After First Stage	Hamming Distance After Second Stage
93	33	16

Table 7: Hamming distance reduction for Method 1- Image 4

In the program following quantities have been used:

HammingDistanceLimit1 = 20 (Limit for stage 1 and below this value SW discards stage 1)
HammingDistanceLimit2 = 16 (Target Hamming Distance)
BD = 20 (Blurring Coefficient)
Wh =2 (Intensity Multiplication Coefficient in bright parts of the mask)
Bl =0.5 (Intensity Multiplication Coefficient in dark parts of the mask)
Quality = on

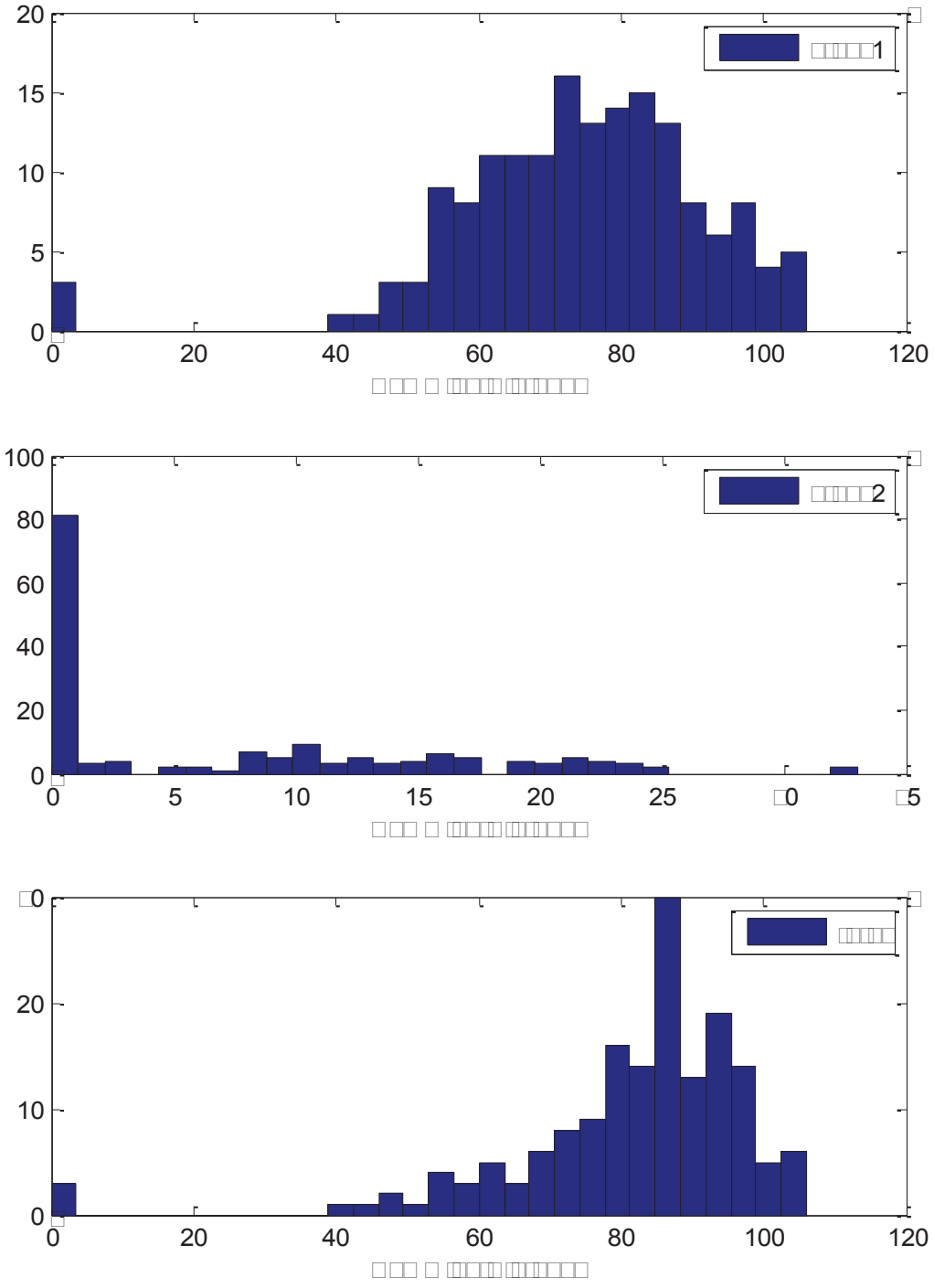


Figure 25: Histogram of hamming distance reduction after stage1 (up), stage2 (middle) and stages1+2 (bottom)

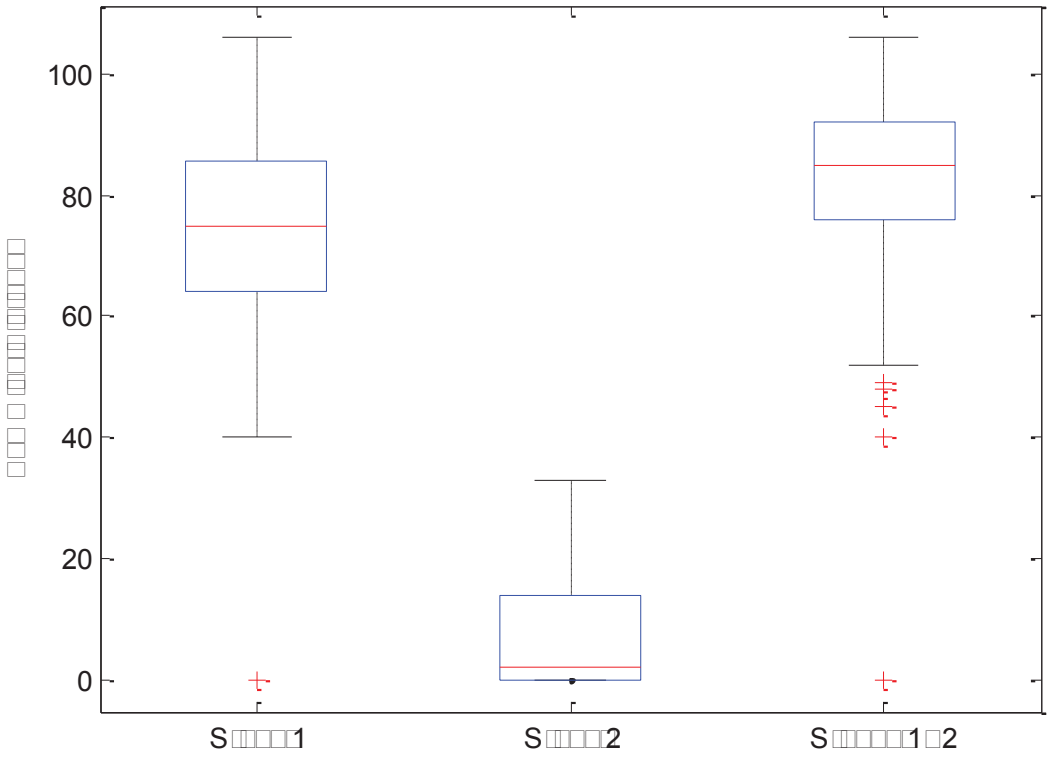


Figure 26: boxplot of hamming distance reduction

4.2.Method 2

Here the program implemented based in predefined masks and is tested on a set of 50 images. Its operation on three of the attacked images has been depicted on in pictures 27 to 31 and tables 8 to 10. In figures 32 and 33 histogram of hamming distance reduction for stage one, stage two, stage one plus stage two as well as corresponding boxplot is depicted. Here hamming distance reduction is declined drastically. For instance at stage one and stage two mean values of hamming distance reduction are 30.14 and 23.24 respectively. Mean of total hamming distance reduction declines from 81 to 53 compared to first method.

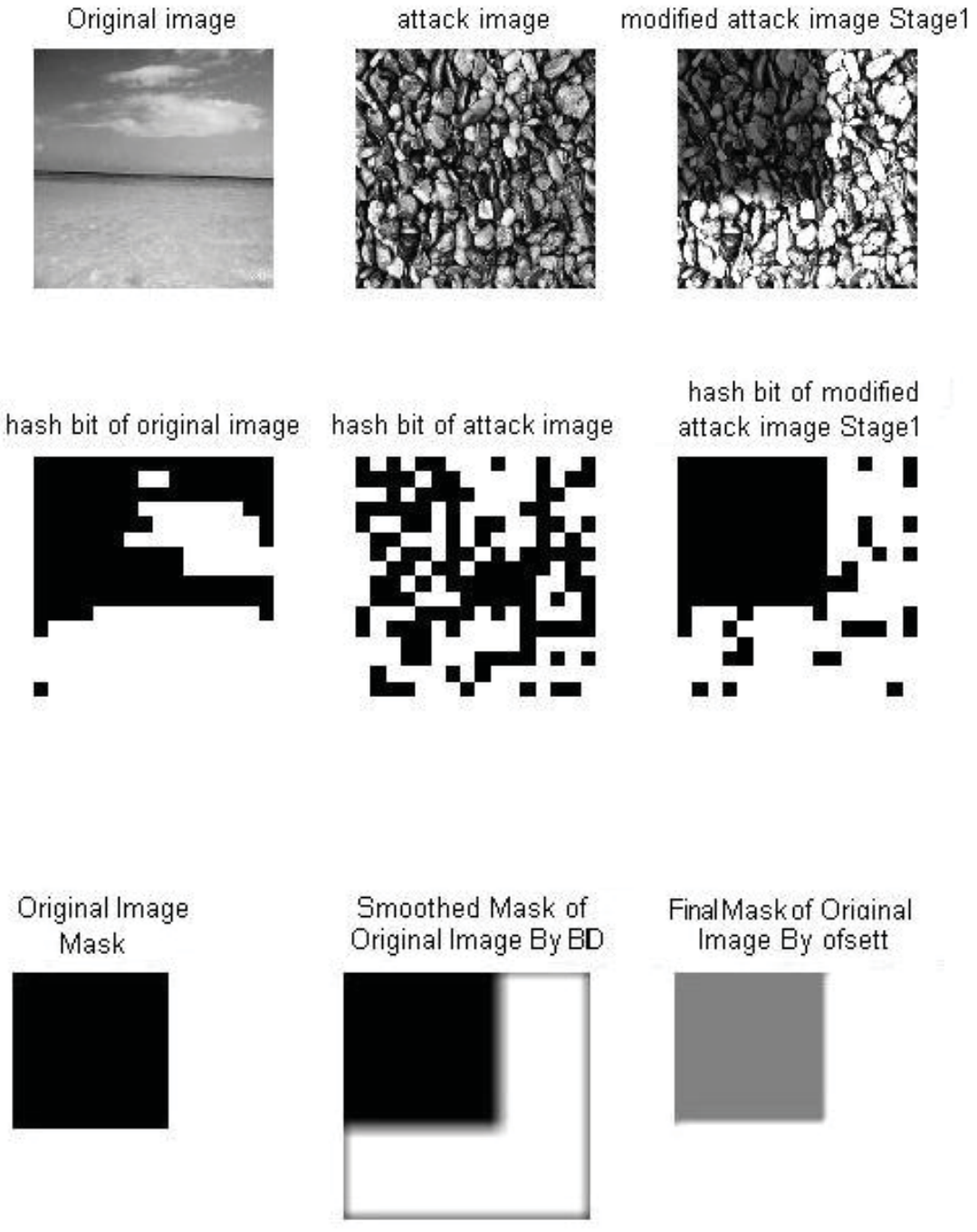


Figure 27: Method 2 – Image 1

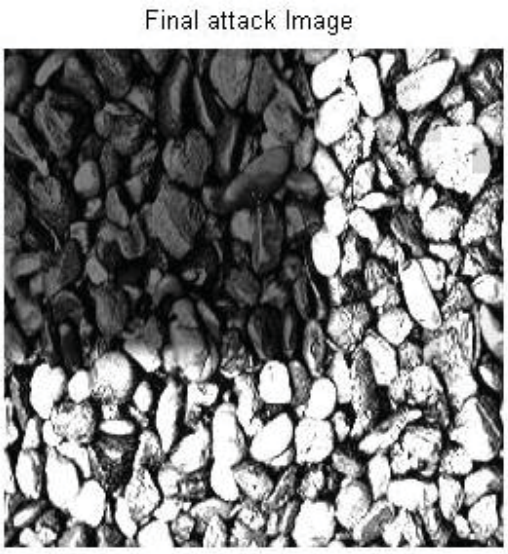


Figure 28: Method 2 – Image 1

Hamming Distance Before Attack	Hamming Distance After First Stage	Hamming Distance After Second Stage
113	63	41

Table 8: Hamming distance reduction for Method 2- Image 1

In the program following quantities have been used:

HammingDistanceLimit1 = 20 (Limit for stage 1 and below this value SW discards stage 1)
HammingDistanceLimit2 = 16 (Target Hamming Distance)
BD = 20 (Blurring Coefficient)
Wh =2 (Intensity Multiplication Coefficient in bright parts of the mask)
Bl =0.5 (Intensity Multiplication Coefficient in dark parts of the mask)
Quality = on

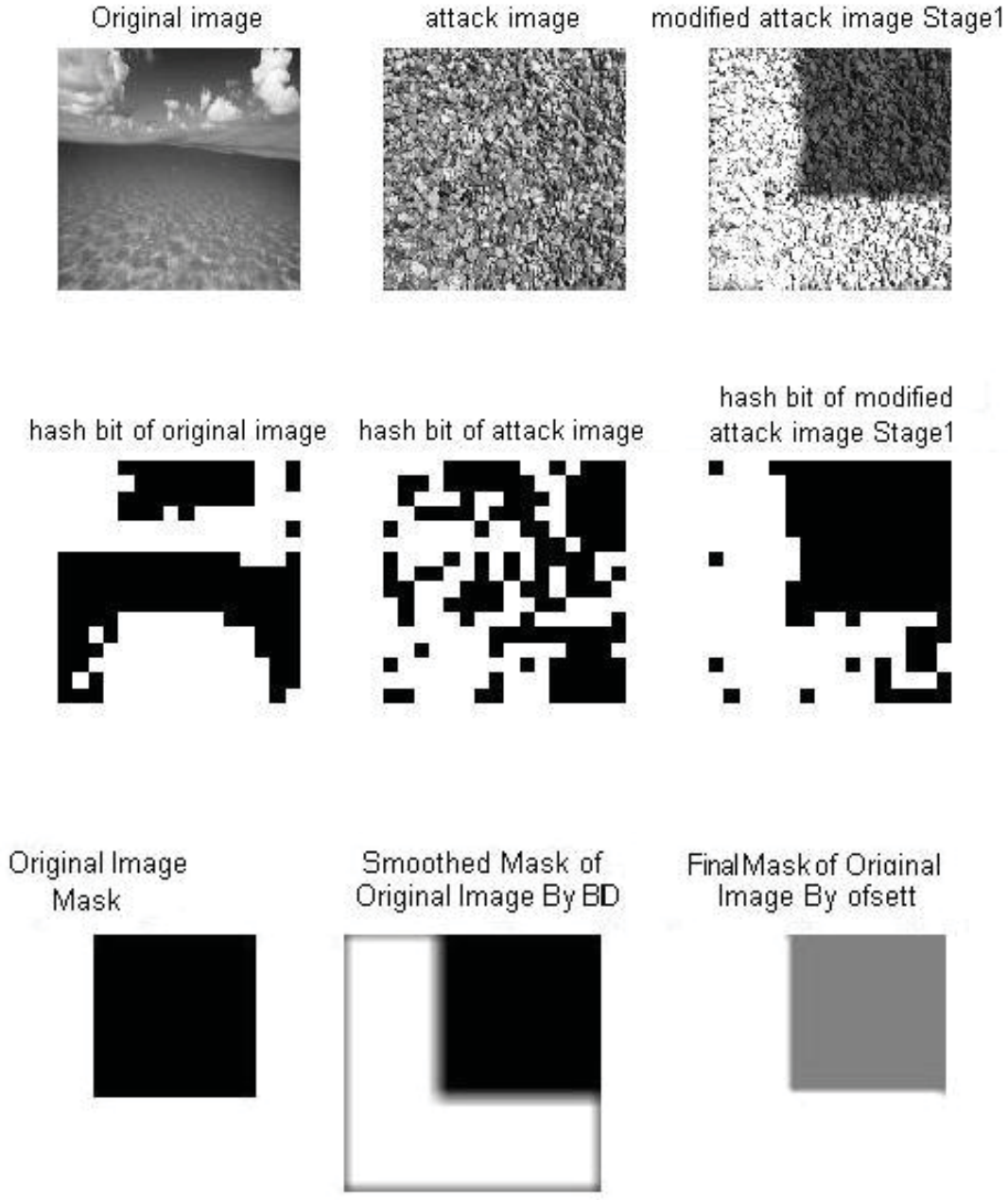


Figure 29: Method 2 – Image 2

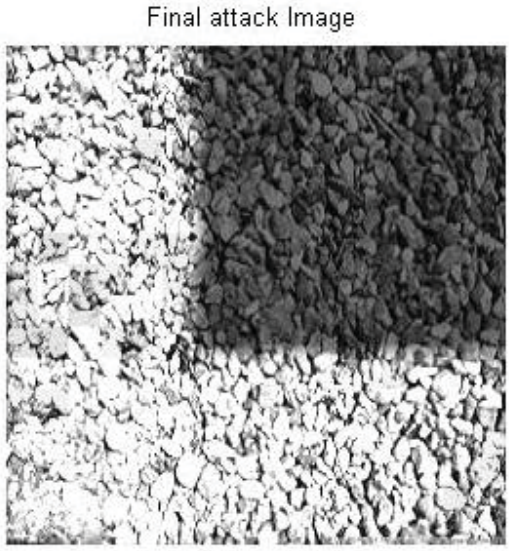


Figure 30: Method 2 – Image 2

Hamming Distance Before Attack	Hamming Distance After First Stage	Hamming Distance After Second Stage
113	96	60

Table 9: Hamming distance reduction for Method 2- Image 2

In the program following quantities have been used:

HammingDistanceLimit1 = 20 (Limit for stage 1 and below this value SW discards stage 1)
HammingDistanceLimit2 = 16 (Target Hamming Distance)
BD = 20 (Blurring Coefficient)
Wh =2 (Intensity Multiplication Coefficient in bright parts of the mask)
Bl =0.5 (Intensity Multiplication Coefficient in dark parts of the mask)
Quality = on

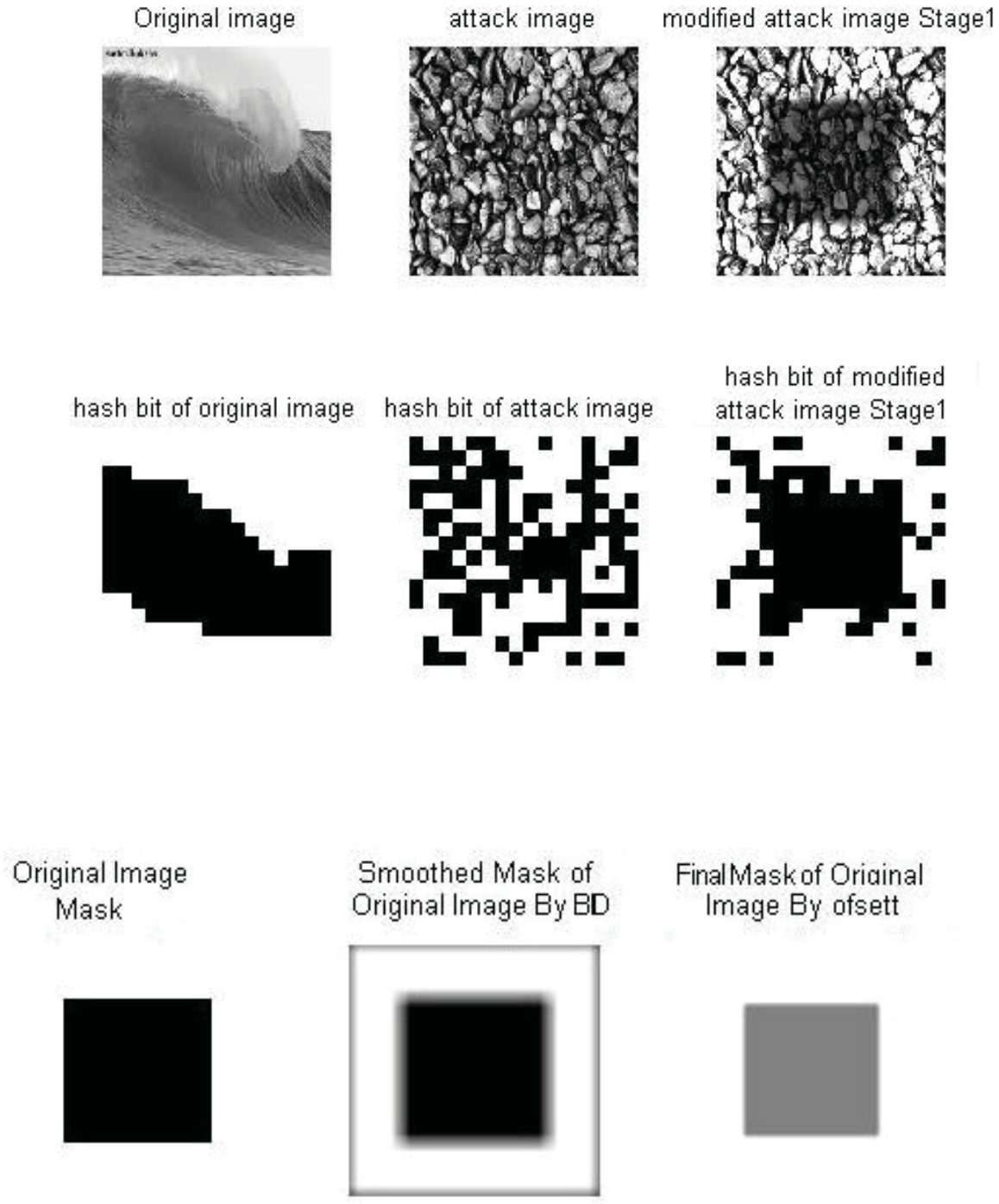


Figure 31: Method 2 – Image 3

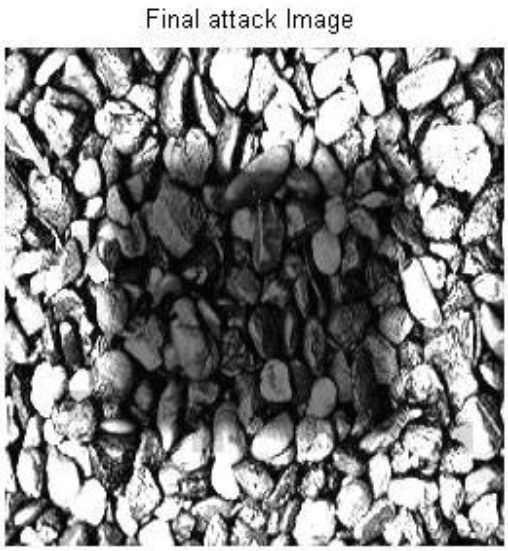


Figure 32: Method 2 – Image 3

Hamming Distance Before Attack	Hamming Distance After First Stage	Hamming Distance After Second Stage
109	92	52

Table 10: Hamming distance reduction for Method 2- Image 3

In the program following quantities have been used:

HammingDistanceLimit1 = 20 (Limit for stage 1 and below this value SW discards stage 1)
HammingDistanceLimit2 = 16 (Target Hamming Distance)
BD = 20 (Blurring Coefficient)
Wh =2 (Intensity Multiplication Coefficient in bright parts of the mask)
Bl =0.5 (Intensity Multiplication Coefficient in dark parts of the mask)
Quality = o

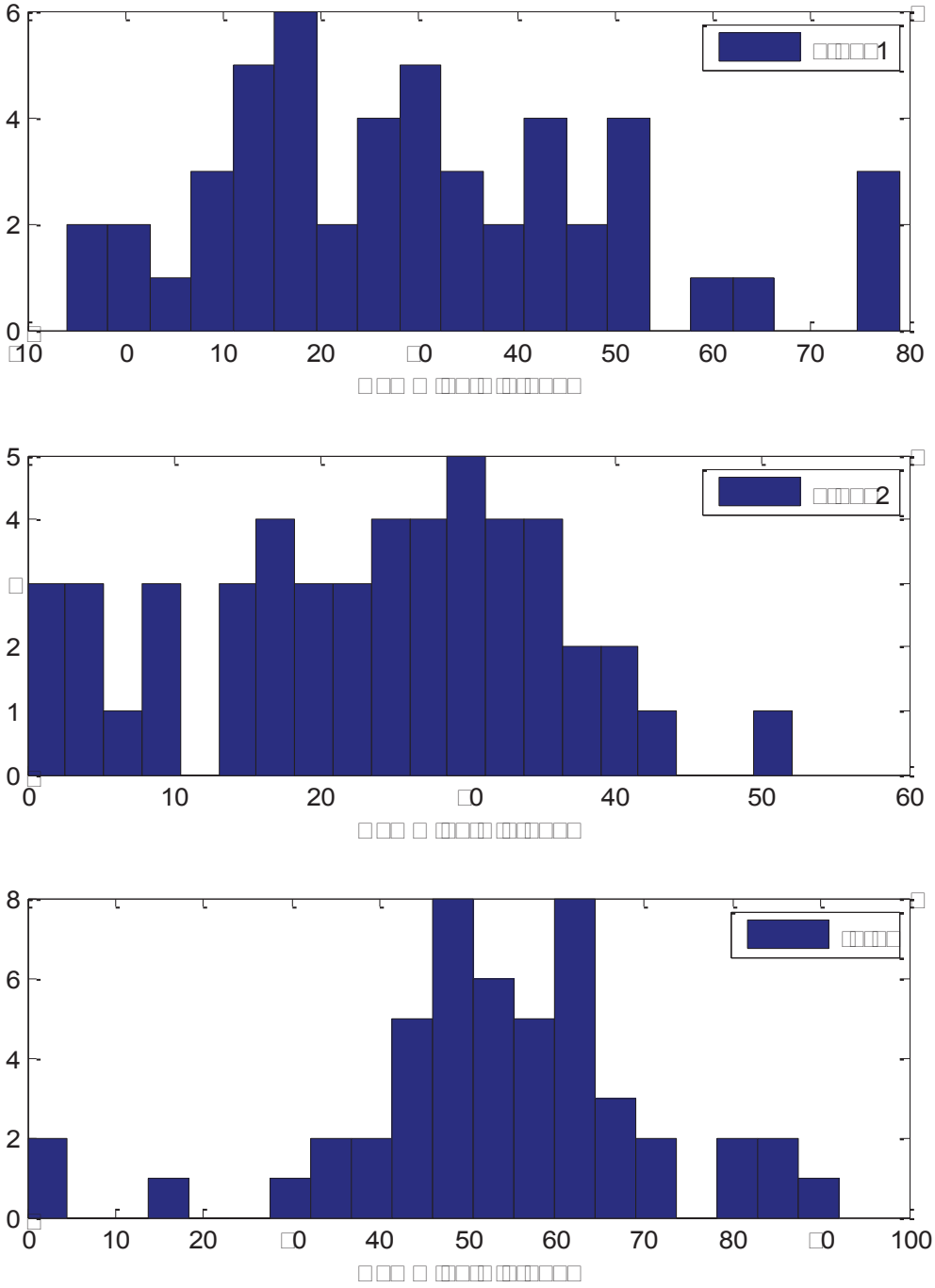


Figure 33: Histogram of hamming distance reduction after stage1 (up), stage2 (middle) and stages1+2 (bottom)

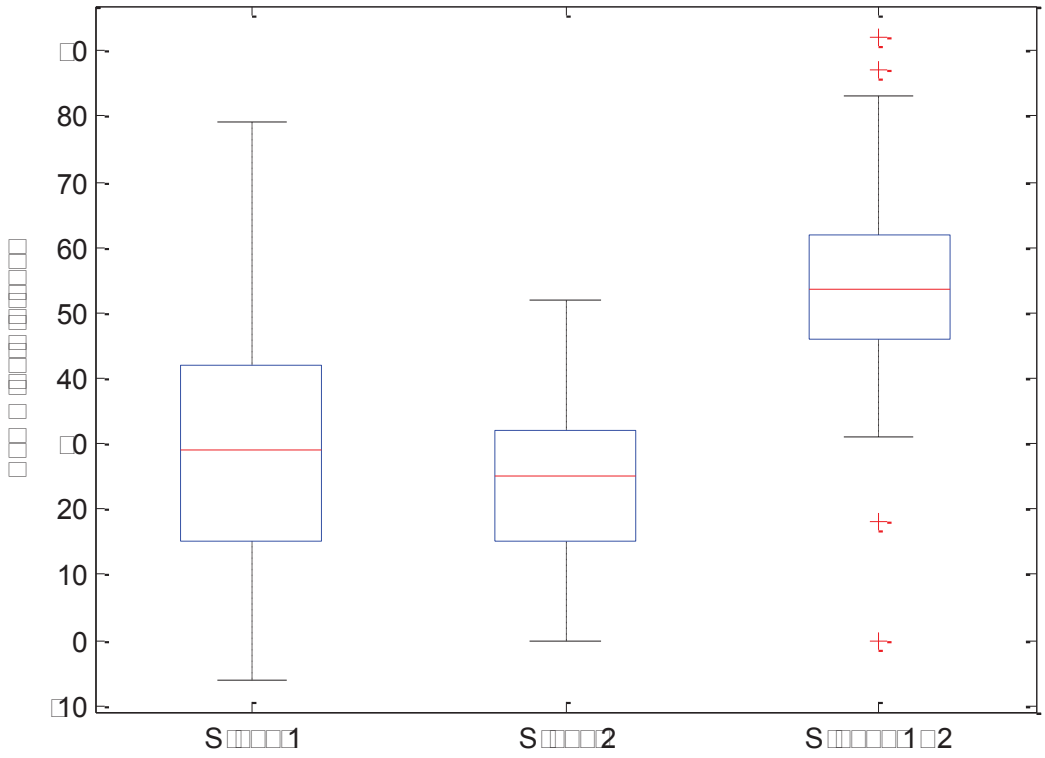


Figure 34: Boxplot of hamming distance reduction

5. Conclusion and Future Work

5.1 Conclusion

This thesis work is undertaken to design and implement a novel method for imitating hash key of a target image by another image. This hash spoofing method implemented based on block mean value hash algorithm by MATLAB and verified. Prior to local modification of specific blocks mean value in the image, a global modification of image is achieved by use of a picture mask and weighting intensities of the image to reach a lower hamming distance. This increase in the similarity of images' hashes had to be done without any visual quality loss of image. For this purpose a thresholding criterion for maximum modifiable distance to mean is obtained by any runs on the program. This thresholding criterion is based on the variance of each block and higher variance means more allowable block's mean modification. Testing the software with a set of images proved that this method of hash spoofing is more effective if target image's hash key is used as an intensity modification mask with a mean of hamming distance reduction equal to 81 while with predefined mask method it is 51. In overall this methods shows acceptable performance especially with the first approach. Since this algorithm uses simplest block mean value based algorithms its cost and complexity is relatively low.

5.2 Future Work

As proposed by Yang, Gu and Niu instead of simple block mean value based algorithm other variations with 50% overlap and rotation may be used. As discussed in [2], it is anticipated better reduction in hamming distance at the expense of complexity and computational cost. For second method better set of predefined masks may lead to improve in hamming distance reduction. Also it is anticipated that in order to improve the algorithm in stage 2, more precise in variance thresholding may lead to better performance of the algorithm.

References:

[1] Menezes, A.J., Vanstone, S.A., and Oorschot, P.C.V. ” *Handbook of Applied Cryptography*”. CRC Press, Inc., Boca Raton, FL, USA, 1996, ISBN 0849385237.

[2] Christoph Zauner, “*Implementation and Benchmarking of Perceptual Image Hash Functions*”, 2011

[3] C Zauner, M Steinebach, E Hermann, “*Rihamark: perceptual image hash benchmarking*” in Proc. SPIE 7880, Media Watermarking, Security and Forensics III, 78800X, 2011

[4] Martin Steinebach, Huajian Liu, York Yannikos “*Efficient Robust Image Hashing*” Proc. SPIE 8303, Media Watermarking, Security, and Forensics, 2012

[5] Sheskin, David ,”*Handbook of Parametric and Nonparametric Statistical Procedures*”. CRC Press. p. 59. ISBN 1584884401.

[6] Yang, B., Gu, F., and Niu, X., “*Block mean value based image perceptual hashing*.”In Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Multimedia Signal Processing (IIH- MSP), pp. 167{172. IEEE, 2006, ISBN 0-7695-2745-0.

[7] Wikipedia contributors, “*Hash function*” Wikipedia, the free encyclopedia. Wikimedia Foundation, Inc., 21-Jun-2012.

[8] Lefebvre, F., Macq, B., Legat, J.: Rash, “*Radon soft hash algorithm*”. In Proceedings of the European Signal Processing Conference, Toulouse, France. (Sep. 2002)

[9] Li Weng, Preneel B,” *Attacking Some Perceptual Image Hash Algorithms*”, In Multimedia and Expo, 2007 IEEE International Conference, Beijing, 2007

[10] R. Merkle, “*Secrecy, Authentication, and Public Key Systems*” UMI Research Press, 1979.

[11] R. Merkle, “*One way hash functions and DES*” Advances in Cryptology, Proc. Crypto’89, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428–446.

[12] M.O. Rabin, “*Digitalized signatures*” in “*Foundations of Secure Computation*” R. Lipton and R. DeMillo, Eds., Academic Press, New York, 1978, pp. 155-166.

[13] I.B. Damgard, “*Collision free hash functions and public key signature schemes*”, Advances in Cryptology, Proc. Eurocrypt’87, LNCS 304, D. Chaum andW.L. Price, Eds., Springer-Verlag, 1988, pp. 203–216.

[14] I.B. Damgard, “*The application of claw free functions in cryptography*”, PhD Thesis, Aarhus University, Mathematical Institute, 1988.

[15] B. Preneel, “*Analysis and Design of Cryptographic Hash Functions*”, Phd Thesis, Katholieke Universiteit Leuven, 1993.

[16] M. Schneider and S. F. Chang, “*Arobust content based digital signature for image authentication*,” in Proc. IEEE Conf. Image Processing, Sep. 1996, vol. 3, pp. 227–230.

[17] C. Kailasanathan and R. S. Naini, “*Image authentication surviving acceptable modifications using statistical measures and k-mean segmentation*,” presented at the Proc. IEEE-EURASIP Work. Nonlinear Sig. Image, Jun. 2001.

[18] R. Venkatesan, S. M. Koon, M. H. Jakubowski, and P. Moulin, “*Robust image hashing*,” in Proc. IEEE Conf. on Image Processing, Sep. 2000, pp. 664–666.

[19 C. Y. Lin and S. F. Chang, "*A robust image authentication system distingushing JPEG compression from malicious manipulation*," IEEE Trans. Circuits Syst. Video Technol., vol. 11, no. 2, pp. 153–168, Feb. 2001.

[20] C.-S. Lu and H.-Y. M. Liao, "*Structural digital signature for image authentication*," IEEE Trans. Multimedia, vol. 5, no. 2, pp. 161–173, Jun. 2003.

[21] J. Fridrich and M. Goljan, "*Robust hash functions for digital watermarking,"* in Proc. IEEE Int. Conf. Information Technology: Coding and Computing, Mar. 2000, pp. 178–183.

[22] K. Mihcak and R. Venkatesan, "*New iterative geometric techniques for robust image hashing,*" in Proc. ACM Workshop on Security and Privacy in Digital Rights Management Workshop, Nov. 2001, pp. 13–21.

[23] A. Swaminathan, Y. Mao, and M. Wu, "*Robust and secure image hashing*," IEEE Trans. Inf. Forensics Security, vol. 1, no. 2, pp. 215–230, Jun. 2006.

[24] S. Bhatacherjee and M. Kutter, "*Compression tolerant image authentication,*" in Proc. IEEE Int. Conf. Image Processing, Chicago, IL, Oct. 1998, vol. 1, pp. 435–439.

[25] J. Dittman, A. Steinmetz, and R. Steinmetz, "*Content based digital signature for motion picture authentication and content-fragile watermarking,*" in Proc. IEEE Int. Conf. Multimedia Computing and Systems, 1999, pp. 209–213.

[26] V. Monga and B. L. Evans, "*Robust perceptual image hashing using feature points,*" in Proc. IEEE Conf. Image Processing, Singapore, Oct. 004, vol. 1, pp. 677–680.

[27] S. S. Kozat, K. Mihcak, and R. Venkatesan, "Robust perceptual image hashing via matrix invariances," in *Proc. IEEE Conf. Image Processing*,Oct. 2004, pp. 3443–3446.

[28] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances Neural Information Process. Syst.*, vol. 13, pp. 556–562, 2000.

[29] Vishal Monga and M. Kıvanç Mıhçak. "*Robust and Secure Image Hashing via Non-Negative Matrix Factorizations*". IEEE Transactions on information forensics and security, Vol. 2, No. 3, Sep

[30] Fridrich, J.: Robust bit extraction from images. In Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS), vol. 2, pp. 536{540. IEEE, June 1999.

[31] Bovik, A. (ed.): The Essential Guide to Image Processing. Academic Press, 2009.

Appendices:

Appendix A: Variance Thresholding

Name	Mean	Variance	Block number	Mean of Block	Variance of Block	Max Pass intensity for +	Max Pass intensity for -
B	120.69	4.9053e+005	26	122.0469	1.4942e+006	3	5
			97	196.8867	2.6184e+005	3	5
			256	140.2500	2.3523e+006	5	10
			170	202.1367	18.9699	3	3
			87	53.4453	1.8760e+006	6	8
			34	124.9883	6.9623e+005	10	18
			120	146.6328	1.2084e+006	15	18
			59	100.5195	1.0453e+006	30	40
			193	129.1523	1.4958e+006	7	10
			148	72.0508	4.2283e+005	8	10
			201	163.1406	2.0434e+006	5	8
			16	108.5977	2.1323e+006	15	20

Name	Mean	Variance	Block number	Mean of Block	Variance of Block	Max Pass intensity for +	Max Pass intensity for -
desert (1)	129.8907	4.1868e+004	16	144.7070	7.5226	2	3
			45	99.6367	3.5061	2	2
			25	115.5977	0.4933	3	4
			38	115.3828	220.3143	3	4
			54	112.3008	1.4261e+003	3	5
			69	101.5586	0.3592	3	5
			78	139.2734	3.3485e+003	3	4
			135	130.7813	626.4725	4	6
			166	128.3008	9.6057e+003	5	6
			183	135.7695	7.9358e+003	7	5
			194	155.9453	6.1390e+003	7	5
			245	120.0547	2.5116e+004	8	9
			255	139.8750	0.1422	2	3

Variance thresholding (Continued)

Name	Mean	Variance	Block number	Mean of Block	Variance of Block	Max Pass intensity for +	Max Pass intensity for -
river (6)	120.0362	3.5216e+005	28	140.3086	3.7625e+006	26	23
			39	112.2695	6.4191e+005	24	15
			56	99.8203	2.1623e+006	25	20
			64	181.4609	1.5694e+005	20	24
			79	107.4805	1.0396e+007	27	24
			85	95.3438	6.4626e+006	18	20
			94	104.7344	2.6235e+005	22	29
			134	112.5938	1.0286e+006	20	25
			155	157.6250	4.5382e+006	24	21
			198	111.1172	4.1756e+006	24	21
			206	123.9453	3.2216e+006	28	25
			245	96.3789	3.1393e+006	28	26

Name	Mean	Variance	Block number	Mean of Block	Variance of Block	Max Pass intensity for +	Max Pass intensity for -
sea (2)	130.1862	2.6821e+005	28	90.1133	5.3403e+003	8	3
			35	143.2695	4.0912e+004	18	11
			48	91.4414	1.5365e+003	7	5
			53	135.6875	4.9744e+004	19	12
			66	159.8281	2.0161e+004	15	19
			76	122.4492	3.1277e+004	7	10
			89	102.9414	419.3829	5	4
			111	80.8633	564.4349	5	4
			134	151.2422	5.9005e+004	8	10
			155	140.1289	5.0154e+003	5	6
			198	142.4219	2.3915e+004	8	7
			236	86.1367	1.0651e+004	5	4
			255	75.3633	35.1890	3	2

Appendix B: Tables of Hamming Distance Reduction by stage

Table1: Results from method 1 using the modified mask from the original image.

Name of Image	Original Hamming Distance	1st Stage Hamming Distance	2nd Stage (Final) Hamming Distance	PSNR in dB
1	113	40	17	15.6338
2	113	10	10	14.5839
3	109	37	16	15.6103
4	111	27	16	14.8646
5	92	37	16	17.3132
6	103	29	16	15.1918
7	92	11	11	13.944
8	99	15	15	14.0747
9	88	7	7	14.029
10	91	7	7	14.7925
11	95	10	10	13.9382
12	86	3	3	13.9965
13	96	24	16	14.7465
14	97	33	16	17.1763
15	93	33	16	17.8165
16	102	32	16	14.7695
17	87	6	6	13.816
18	112	9	9	13.6047
19	107	10	10	13.7455
20	105	13	13	13.951
21	108	25	16	15.1693
22	102	14	14	14.8231
23	113	10	10	14.7227
24	89	10	10	14.1853
25	109	12	12	13.6674
26	104	28	16	15.3215
27	110	4	4	14.4032
28	106	10	10	15.0094
29	111	19	16	14.938
30	96	7	7	14.8179
31	113	32	16	15.0189
32	0	0	0	Inf
33	110	26	16	17.5909
34	107	24	16	14.4356
35	1	1	1	Inf

36	106	11	11	15.2288
37	79	19	16	14.9858
38	97	10	10	13.8789
39	98	37	16	14.7288
40	97	13	13	14.419
41	118	38	15	14.4965
42	110	4	4	14.4032
43	91	14	4	14.2107
44	116	15	15	14.9863
45	87	12	12	14.1973
46	112	49	16	15.7424
47	99	8	8	14.5423
48	89	25	16	14.7929
49	94	9	9	13.8939
50	72	16	16	14.887
51	102	33	16	14.6393
52	98	7	7	13.7488
53	100	35	16	15.7411
54	100	5	5	13.8375
55	95	25	16	17.1012
56	111	14	14	14.3483
57	92	13	13	14.0084
58	102	41	17	16.4607
59	92	7	7	13.9503
60	105	33	16	14.6503
61	95	8	8	14.6173
62	94	11	11	14.1256
63	103	43	18	17.2599
64	63	14	14	18.2579
65	100	30	15	15.7774
66	105	22	16	14.2337
67	109	30	17	14.3314
68	97	9	9	13.8481
69	96	16	16	14.1644
70	98	16	16	15.0636
71	81	14	14	14.0775
72	104	16	16	15.0201
73	70	17	16	15.0369
74	90	22	16	14.354
75	75	15	15	14.8666
76	107	10	10	13.7906
77	92	24	16	14.8782
78	97	17	15	15.1509

79	111	17	16	15.7483
80	99	10	10	13.8598
81	94	33	17	17.5301
82	65	11	11	14.9404
83	103	5	5	15.4496
84	80	9	9	14.0569
85	83	13	13	14.3751
86	116	39	16	15.1156
87	85	12	12	14.227
88	81	26	16	17.8452
89	100	11	11	13.739
90	110	40	15	14.8077
91	101	15	15	15.5761
92	95	30	16	17.0793
93	99	36	16	17.1459
94	87	26	16	17.643
95	103	24	16	15.987
96	83	27	15	14.7262
97	95	32	16	17.7549
98	99	31	16	14.8433
99	112	48	16	15.9446
100	91	9	9	13.9388
101	78	24	15	14.992
102	106	5	5	14.5612
103	101	19	16	14.2475
104	89	27	16	17.6654
105	58	13	13	15.2376
106	102	15	15	15.6779
107	0	0	0	Inf
108	102	16	16	15.619
109	98	12	12	15.6092
110	100	26	16	14.3222
111	103	21	16	15.9963
112	106	33	17	15.2393
113	113	39	16	15.8501
114	87	18	15	16.7025
115	79	15	15	14.9198
116	104	29	16	14.6201
117	99	40	16	17.1178
118	99	28	14	14.7046
119	97	29	16	15.9227
120	87	12	12	16.5258
121	84	27	16	17.4953

122	83	11	11	16.5854
123	104	29	16	17.3527
124	86	21	16	15.285
125	97	26	16	14.601
126	110	38	16	15.8131
127	97	35	16	14.8189
128	78	24	16	14.8608
129	68	16	16	15.2846
130	85	33	16	17.5661
131	85	23	16	17.9415
132	109	12	12	13.7381
133	108	11	11	14.7103
134	107	36	15	13.8569
135	106	40	16	15.8845
136	62	14	14	14.9294
137	81	6	6	14.0135
138	96	36	16	17.3357
139	104	31	16	14.8357
140	70	16	16	15.0719
141	105	18	16	15.2511
142	113	35	16	13.6782
143	108	17	16	14.1683
144	100	7	7	13.8936
145	77	14	14	14.8329
146	107	8	8	13.6847
147	110	28	16	14.6913
148	55	15	15	15.1551
149	108	32	16	15.3651
150	73	25	16	14.9298
151	82	5	5	14.0697
152	108	30	16	15.4116
153	103	13	13	13.6848
154	95	16	16	14.2216
155	91	2	2	13.9672
156	75	18	16	15.0965
157	78	24	16	14.9534
158	101	24	16	15.5052
159	107	7	7	14.5022
160	108	31	16	14.5392
161	100	34	15	15.8422
162	85	8	8	14.3405
163	103	36	16	15.6947

Table 2: Results from method 2 using the predefined mask

Name of Image	Original Hamming Distance	1st Stage Hamming Distance	2nd Stage (Final) Hamming Distance
1	113	63	41
2	113	96	60
3	109	92	52
4	111	64	55
5	92	77	61
6	103	92	61
7	92	74	39
8	115	82	68
9	99	67	35
10	109	57	48
11	88	68	39
12	91	77	37
13	95	34	30
14	86	49	47
15	96	99	47
16	97	34	27
17	93	76	60
18	102	60	46
19	87	55	26
20	112	35	20
21	107	73	45
22	105	75	43
23	108	29	25
24	102	89	56
25	107	91	74
26	113	75	51
27	89	48	44
28	109	83	55
29	104	81	55
30	110	57	29
31	106	72	42
32	111	83	54
33	96	78	35
34	113	67	44
35	0	0	0
36	94	66	43
37	110	101	63

38	107	113	89
39	112	80	60
40	0	0	0
41	106	94	61
42	79	52	32
43	101	97	60
44	98	87	53
45	97	67	48
46	107	29	20
47	118	76	60
48	110	57	29
49	91	77	43
50	116	74	48

Appendix C: MATLAB Codes

Manual Test to find the Threshold

```
%%%%%%%%%%Manual Test to find the threshold for quality saving%%%%%%%%%%
%--------------------------------------------------------------------------
name='B';
k1=20;
v=20;
%--------------------------------------------------------------------------

clc
addr=sprintf('%s.jpg',name);
A=imread(addr);
I=rgb2gray(A);
x=1;
y=1;
x2=16;
y2=16;
ylim=((y2)-1).*(256./y2)+1;
hb=(x2.*y2);
imdim=256;
k=1;
while(x<=imdim)
for j=1:1:x2
    for i=1:1:y2
        F(i,j,k)=I(i+y-1,j+x-1);
    end
end
if(y<ylim)
    y=y+y2;
else
    y=1;
    x=x+x2;
end
k=k+1;
end


%--------------------------------------------------------------------------
I1=I;
I2=I;
%Main Block-------------------------------------------------------------------
    if(rem(k1,16)==0)
        k2=k1-1;
    else
        k2=k1;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k1-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                Q(p-y3+1,o-x3+1)=I(p,o);
            end
        end
```

```
Q=double(Q);
m1=mean(mean(Q));
v1=var(var(Q));
Q1=Q;
m2=m1;
while(abs(m2-m1)<v)
    for o=1:16
        for p=1:16
            Q1(p,o)=Q1(p,o)./1.001;
        end
    end
    m2=mean(mean(Q1));
end
 if(rem(k1,16)==0)
        k2=k1-1;
    else
        k2=k1;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k1-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                I1(p,o)=Q1(p-y3+1,o-x3+1);
            end
        end

Q=double(Q);
Q2=Q;
m3=m1;
while(abs(m3-m1)<v)
    for o=1:16
        for p=1:16
            Q2(p,o)=Q2(p,o).*1.001;
        end
    end
    m3=mean(mean(Q2));
end

 if(rem(k1,16)==0)
        k2=k1-1;
    else
        k2=k1;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k1-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                I2(p,o)=Q2(p-y3+1,o-x3+1);
            end
        end

I=double(I);
```

```
MI=mean(mean(I));
VI=var(var(I));
disp('Name Of Image');
disp(     name)
disp('Mean Of Image');
disp(     MI)
disp('Variance of Image');
disp(     VI)
disp('Block Number=');
disp(     k1)
disp('Mean of Block=');
disp(     m1)
disp('Variance of Block=');
disp(     v1)


subplot(1,3,1); imshow(I,[]);  title('original image');
rectangle('Position',[x3,y3,16,16])
subplot(1,3,2); imshow(I1); title('-');
subplot(1,3,3); imshow(I2); title('+');
```

Hash Function

```
function SaveHash(NumberOfAttackImages)
for num=1:NumberOfAttackImages
addr=sprintf('%s%d.jpg','attack set\',num);
A=imread(addr);
I=rgb2gray(A);
x=1;
y=1;
x2=16;
y2=16;
for k=1:256
if(rem(k,16)==0)
        k2=k-1;
    else
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                Q(p-y3+1,o-x3+1)=I(p,o);
            end
        end
Q=double(Q);
M(k)=round(mean(mean(Q)));
end
S=sort(M); %sort averages from min to max
Md=S(127); %find median
H=M>Md;
H=sprintf('%d',H);
for p=1:256
H1(num,p)=H(p);
end
```

```
end
xlswrite('attack set\AH.xls',H1);
end
```

Find Similar Hash

```
%%%%%%%%%%%%%%%%% SIMILAR HASH %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function num=SimilarHash(NameOfOriginalImage)
addr=sprintf('%s.jpg',NameOfOriginalImage);
A=imread(addr);
I=RGB2gray(A);
x=1;
y=1;
x2=16;
y2=16;
for k=1:256
if(rem(k,16)==0)
        k2=k-1;
    else
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                Q(p-y3+1,o-x3+1)=I(p,o);
            end
        end
Q=double(Q);
M(k)=round(mean(mean(Q)));
end
S=sort(M); %sort averages from min to max
Md=S(127); %find median
H0=M>Md;
H0=sprintf('%d',H0);

H1=xlsread('attack set\AH.xls');
x=size(H1);

for i=1:x(1)
H2=sprintf('%d',H1(i,:));
Dist(i)=sum(H2~=H0);
end
[va,num]=min(Dist);
num=sprintf('%d',num);
end
```

Hash Distance

```
function HashDist(I1,S1,I2,S2)
add1=sprintf('%s.%s',I1,S1);
add2=sprintf('%s.%s',I2,S2);
A1=imread(add1);
A2=imread(add2);
[a,b,c]=size(A1);
if(c==3), A1=rgb2gray(A1); end
[a,b,c]=size(A2);
if(c==3), A2=rgb2gray(A2); end

x=1;
y=1;
x2=16;
y2=16;
%HashNumber Of 1st Image ------------------------------------------------
for k=1:256
if(rem(k,16)==0)
        k2=k-1;
    else
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                Q(p-y3+1,o-x3+1)=A1(p,o);
            end
        end
Q=double(Q);
me(k)=round(mean(mean(Q)));
end
So=sort(me);
Md=So(127);
H1=me>Md;
H1=sprintf('%d',H1)
%-------------------------------------------------------------------------
%HashNumber Of 2nd Image ------------------------------------------------
for k=1:256
if(rem(k,16)==0)
        k2=k-1;
    else
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                Q(p-y3+1,o-x3+1)=A2(p,o);
            end
        end
Q=double(Q);
me(k)=round(mean(mean(Q)));
end
```

```
So=sort(me);
Md=So(127);
H2=me>Md;
H2=sprintf('%d',H2)
%-------------------------------------------------------------------------
Dist=sum(H2~=H1)
end
```

Stage 1

```
%%%%%%%%%%%%%%%%%%%%%%%% Stage 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
function
[I1,I2,H2,Dist2,Dist1]=Stage1M(OriginalImageName,cond,BD,wh,bl,method,ShRe,Sa
Re)
%-------------------------------------------------------------------------
name=OriginalImageName;
name2=SimilarHash(name);
%-------------------------------------------------------------------------
addr=sprintf('%s.jpg',name);
A=imread(addr);
I=rgb2gray(A);
x=1;
y=1;
x2=16;
y2=16;

%Find Median & Mean of Blocks (1 to 256)for OriginalImage ----------------
for k=1:256
if(rem(k,16)==0)
        k2=k-1;
    else
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                Q(p-y3+1,o-x3+1)=I(p,o);
            end
        end
Q=double(Q);
me(k)=round(mean(mean(Q)));
end
So=sort(me);
Md=So(127);
%-------------------------------------------------------------------------
%Caclulate Hash Number of OriginalImage-----------------------------------
H=me>Md;
H2=sprintf('%d',H);
%-------------------------------------------------------------------------
%Make Original Image Hash bit as a Picture (MASK)-------------------------
for i=1:16
    for j=1:16
        H00(j,i)=H(j+(16.*(i-1)));
    end
end
```

```
%-------------------------------------------------------------------------

%read attack Image ------------------------------------------------------
addr=sprintf('%s%s.jpg','attack set\',name2);
A=imread(addr);
I2=rgb2gray(A);
I2=uint8(I2);

%Find Median and Mean of Blocks of Original attack Image -----------------
for k=1:256
if(rem(k,16)==0)
        k2=k-1;
    else
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                Q(p-y3+1,o-x3+1)=I2(p,o);
            end
        end
Q=double(Q);
me(k)=round(mean(mean(Q)));
end
So=sort(me);
Md=So(127);
%-------------------------------------------------------------------------
%Calculate Hash number of Original attack number -------------------------
H=me>Md;
H4=sprintf('%d',H);
%Make Modified attack Image Hash bit as a Picture ------------------------
for i=1:16
    for j=1:16
        H11(j,i)=H(j+(16.*(i-1)));
    end
end
%-------------------------------------------------------------------------
Dist1=sum(H2~=H4);% Hamming Distance of Original Image with Original Attack
Image
ad='attack set\';
if((Dist1-2)>cond)
  name2=FavSimilarHash(name);
  ad='attack set\Fav\';
%read attack Image again-------------------------------------------------
addr=sprintf('%s%s.jpg',ad,name2);
A=imread(addr);
I2=rgb2gray(A);
I2=uint8(I2);

%Find Median and Mean of Blocks of Original attack Image -----------------
for k=1:256
if(rem(k,16)==0)
        k2=k-1;
    else
```

```
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                Q(p-y3+1,o-x3+1)=I2(p,o);
            end
        end
Q=double(Q);
me(k)=round(mean(mean(Q)));
end
So=sort(me);
Md=So(127);
%-------------------------------------------------------------------------
%Calculate Hash number of Original attack number -------------------------
H=me>Md;
H4=sprintf('%d',H);
%Make Modified attack Image Hash bit as a Picture ------------------------
for i=1:16
    for j=1:16
        H11(j,i)=H(j+(16.*(i-1)));
    end
end
%-------------------------------------------------------------------------
Dist1=sum(H2~=H4);% Hamming Distance of Original Image with Original Attack
Image
end

%-------------------------------------------------------------------------
BA = double(imresize(H00, 16)); % Resize H00(HashBit Image(Mask) to 256x256)
if(method==2)
for i=1:1:30
    MaB=im2bw(imread(['Masks\(',num2str(i),').jpg']));
    cor(i)=sum(sum(MaB.*BA));
end
[vcor,icor]=max(cor);
BA=im2bw(imread(['Masks\(',num2str(icor),').jpg']));
end

B3 = BA; %imclose(BA,strel('square',28));
%BD=level of bluring as an input for function
B = conv2(double(B3),fspecial('average',BD),'same');%Smooth Mask by BD...
%BD determines the level of bluring, Higher BD -> more blurring ->less
%changes in picture



%method1-----------------------------------------------------------------
B2=(B.*(wh-bl))+bl;%Complete the mask by setting some degree of effects on
Smoothed Mask
%-------------------------------------------------------------------------
if(ShRe==1)
% show result
subplot(1,3,1); imshow(BA); title('OriginalImage Mask');
```

```
subplot(1,3,2); imshow(B); title('Smoothed Mask of OriginalImage By BD');
subplot(1,3,3); imshow(B2); title('Final Mask of OriginalImage By ofset');
figure()
end
%-------------------------------------------------------------------------

% read the attack image again ---------------------------------------------
addr=sprintf('%s%s.jpg',ad,name2);
A=imread(addr);
I1=double(rgb2gray(A));
if((Dist1-2)>cond)
    I1=I1.*B2;% multiply attack image by mask to get a similar hash number
end
I1=uint8(I1);

%Find Median and Mean of Blocks of Modified attack Image ------------------
for k=1:256
if(rem(k,16)==0)
        k2=k-1;
    else
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                Q(p-y3+1,o-x3+1)=I1(p,o);
            end
        end
Q=double(Q);
me(k)=round(mean(mean(Q)));
end
So=sort(me);
Md=So(127);
%-------------------------------------------------------------------------
%Calculate Hash number of Modified attack number -------------------------
H=me>Md;
H3=sprintf('%d',H);
%Make Modified attack Image Hash bit as a Picture ------------------------
for i=1:16
    for j=1:16
        H22(j,i)=H(j+(16.*(i-1)));
    end
end
%-------------------------------------------------------------------------


Dist2=sum(H2~=H3);% Hamming Distance of Original Image with Modified Attack
Image

if(ShRe==1)
% show result
 subplot(2,3,1); imshow(I,[]);title('Original image');
 subplot(2,3,2); imshow(I2,[]); title('attack image');
 subplot(2,3,3); imshow(I1,[]); title('modified attack image Stage1');
```

```
 subplot(2,3,4); imshow(H00,[]); title('hash bit of original image');
 subplot(2,3,5); imshow(H11,[]); title('hash bit of attack image');
 subplot(2,3,6); imshow(H22); title('hash bit of modified attack image
Stage1');
 figure();
end
%-------------------------------------------------------------------------
 if(SaRe==1)
%save result
dead=sprintf('%sName=%s--1stStage--Dist=%d%sBD=%d%swh=%d--
bl=%d%s','result\',name,Dist2,'--',BD,'--',wh,bl,'.bmp');
imwrite(I1,dead);
end
%-------------------------------------------------------------------------
End
```

Stage 2

```
%%%%%%%%%%%%%%%%% Stage 2 %%%%%%%%%%%%%%%%%%%%%%
function [I,Dist1]=Stage2M(I,H0,limit,BEST,name,ShRe,SaRe)

%find the Median, Mean hash bit of stage1 modified attack Image------------
x=1;
y=1;
x2=16;
y2=16;
for k=1:256
if(rem(k,16)==0)
        k2=k-1;
    else
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                Q(p-y3+1,o-x3+1)=I(p,o);
            end
        end
Q=double(Q);
M0(k)=round(mean(mean(Q)));
V(k)=var(var(Q));
end
S=sort(M0); %sort averages from min to max
Md=S(127); %find median
H=M0>Md;
H=sprintf('%d',H);
Dist=sum(H0~=H); % hamming distance of stage1 modified attack image with
original image
%-------------------------------------------------------------------------

M=M0;
%Vn LOOKUP TABLE
```

```
Vn=[0 0 100 100 1000 1000 20000 20000 30000 30000 30000 30000 30000 30000
30000 2500000 2500000 2500000 2500000 2500000 2500000 2500000 2500000
2500000];

% which bits are different? put them in HDi array-------------------------
t=0;
HDi=[];
i=1;
while(i-1~=Dist)
    t=t+1;
    if(H(t)~=H0(t)), HDi(i)=t; i=i+1; end
end
%-------------------------------------------------------------------------
 I1=I;
liD=limit;% Conditin -- if Dist become less than Limit Stop Modifying--as an
input for function

%Start Modifing Stage2-----------------------------------------------------
n=0;
z=0;
while(Dist>liD && n<=(23+z))
    n=n+1;
    if(z~=0), Vn(n)=0; end
    for i=1:numel(HDi)
    if((M(HDi(i))==Md+n)&& (V(HDi(i))>=Vn(n)))
        [I1]=ChangeMean(I1,n,HDi(i),'+');
        M(HDi(i))=Md;
        S=sort(M);
        H(HDi(i))='0';
        HDi(i)=0;
    end
    end
    ei=find(HDi==0);
    for i=1:numel(ei)
        HDi(ei(i)-i+1)=[];
    end
     ei=[];
     Dist=sum(H0~=H);

m=1;
while(Dist>liD && m<=n)
    for i=1:numel(HDi)
    if((M(HDi(i))==Md-m+1)&& (V(HDi(i))>=Vn(n)) && (S(128)==Md))
        I1=ChangeMean(I1,m,HDi(i),'-');
        M(HDi(i))=Md+1;
        S=sort(M);
        H(HDi(i))='1';
        HDi(i)=0;
    end
    end
    ei=find(HDi==0);
    for i=1:numel(ei)
        HDi(ei(i)-i+1)=[];
    end
    ei=[];
    Dist=sum(H0~=H);
```

```
    m=m+1;
end
if((n==24) && (Dist>liD) && strcmp(BEST,'no') && (z==0)),n=0; z1=255-Md;
z2=Md; z=max(z1,z2); end
end
%-------------------------------------------------------------------------

% disp(n) %max change in pixel intensity----------------------------------
if(ShRe==1)
%show result
subplot(1,2,1); imshow(I); title('Stage1 Modified attack Image');
subplot(1,2,2); imshow(I1); title('Stage2 Modified attack Image (Final)');
figure();
end

%verification-----------------------------------------------------------------
I=I1;
x=1;
y=1;
x2=16;
y2=16;
for k=1:256
if(rem(k,16)==0)
        k2=k-1;
    else
        k2=k;
    end
    cor=(fix(k2./y2).*x2);
    x3=cor+1;
    y3=((k-cor-1).*y2)+1;
        for o=x3:1:(x3+x2-1)
            for p=y3:1:(y3+x2-1);
                Q(p-y3+1,o-x3+1)=I(p,o);
            end
        end
Q=double(Q);
M1(k)=round(mean(mean(Q)));
end
S=sort(M1); %sort averages from min to max
Md=S(127); %find median
H1=M1>Md;
H1=sprintf('%d',H1);


Dist1=sum(H0~=H1); % hamming distance of Original Image with Stage2 modified
attack Image
%-------------------------------------------------------------------------
 if(SaRe==1)
%save result
dead=sprintf('%sName=%s--2ndStage--Dist=%d%s','result\',name,Dist1,'.bmp');
imwrite(I,dead);
end
%-------------------------------------------------------------------------

end
```

PSNR

```
%%%%%%%%%%%%%%%%%% PSNR %%%%%%%%%%%%%%%%%%
function p = psnr(x,y, vmax)

if nargin<3
    m1 = max( abs(x(:)) );
    m2 = max( abs(y(:)) );
    vmax = max(m1,m2);
end

d = mean( (double(x(:))-double(y(:))).^2 );

p = 10*log10( vmax^2/d );
end
```

Final Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FINAL RESULT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
HammingDistanceLimit2=16;
                  %=Stage1M(Original
ImagName,HammingDistanceLimit1,BD,wh,bl,method,ShRe if =1 ->show,SaReif =1 -
>save)
[I1,I2,H2,Dist1,Dist0]=Stage1M(OriginalImageName,HammingDistanceLimit1,20,2,0
.5,1,1,0);
        %=Stage2M(I1,H2,Hamming Distance Limit,yes/no-> if no=lose
quality,OIN,ShRe if =1 ->show,SaReif =1 ->save)
[I,Dist2]=Stage2M(I1,H2,HammingDistanceLimit2,'yes',OriginalImageName,0,0);
p=psnr(I2,I,255);

disp('Original Hamming Distance :')
disp(Dist0)
disp('1st Stage Hamming Distance :')
disp(Dist1)
disp('2nd Stage (Final) Hamming Distance :')
disp(Dist2)
disp('PSNR in dB :')
disp(p)
imshow(I); title('Final attack Image');
```