Final Year Project Report

Histogram-based counter-forensics of digital images

Vsevolods Caka

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science

Supervisor: Felix Balado

UCD School of Computer Science
University College Dublin

April 7, 2017

Acknowledgments

I am very thankful to my supervisor **Felix Balado** for his active guidance throughout the completion of my final year project.

Also, I would like to extend my appreciation to my fellow classmates who could not be mentioned here but played their role in inspiring and assisting me during this four year period.

Thank You,

Vsevolods Caka

Table of Contents

Chapter 1: **Abstract**

The research field that challenges digital forensics and systematically explores its limitations against intelligent counterfeiters is called counter-forensics, or anti-forensics.

The question of the trustworthiness of digital image forensics arises because most publications still lack rigorous discussions of robustness against counterfeiters. Forensic methods might benefit from research on countermeasures in a similar way that reasoning about attacks in multimedia security in general is useful to improve security. In this sense attacks on image forensic algorithms can be understood as schemes to mislead the detection methods.

Besides the need to assess and improve the reliability of forensic methods, two more reasons motivate research on counter-forensics.

First, many forensic techniques link images to the circumstances of their acquisition. This indirectly reveals information about identities of the author or depicted subjects. This is not always desired. Researchers have studied systems providing unlinkability and anonymity in digital communications for some time . All these efforts are useless if the link to the subject can be re-established by forensic analysis of the message. Hence counter-forensic techniques to suppress traces of their origin in digital images are a relevant building block for anonymous image communication, which can be useful in practice to protect the identity of sources, e.g., in the case of legitimate whistleblowing.

Second, some authors argue that counter-forensics, if implemented in image acquisition devices, can be useful to hide details of the internal imaging pipeline and thus discourage reverse engineering. This motivation is mentioned for completeness, but remain reserved on whether current counter-forensics is ripe enough for this purpose. Our main concern is that counter-forensics often imply a loss of image quality. Camera manufacturers, for instance, compete on quality. It is questionable if they would sacrifice a competitive edge for making reverse engineering a little harder. (Gloe T. et al 2007)

Before proceeding, here are two important definitions of terms used in this work.

- Forgery: an algorithm used for image tampering (histogram stretching, gamma correction, double jpeg compression etc.).
- Decoy: a legitimate histogram that is most similar to the forged image's histogram (in a given domain such as: spatial, discrete cosine, Benford's law). For this project, we are focusing exclusively on the first order statistics i.e. histograms of images.

In this project, the work is focused on first order statistics forensic detectors i.e. a visual inspection of the image's histogram. Firstly, the research conducted, in order to accomplish the project will be discussed. Then, implementation of a decoy selection strategy and actual counter-forensic algorithms will be demonstrated.

Chapter 2: **Introduction**

The field of multimedia counter-forensics (also known as multimedia anti-forensics) deals with techniques aimed at misleading forensic detection tests for multimedia, whose goal is determining the authenticity of assets such as digital images. Many algorithms have been proposed after the counter-forensics concept was first formulated by Kirchner and Bohme 2007. However existing counter-forensic methods have generally been of a heuristic nature. More recently optimum counter-forensic algorithms have been developed for the case in which the forensic detector only relies on first-order statistics (histograms) in the papers proposed by Barni M. et al(2012), Comesana-Alfaro P. et al(2013), Balado F.(2014) . In the case in which the relevant histogram is directly computed from the image or from an energy-preserving unitary transform (i.e. Fourier-type transforms), the optimum counter-forensic algorithm is found by relying on either transportation theory or permutation decoding, leading to closed-form algorithms. However when the histogram is computed in a non-energy-preserving transform the optimum can only be found numerically through the simplex algorithm proposed by Comesana-Alfaro P. et al(2014). An important non-energy-preserving scenario occurs when the forensic examiner uses the Benford's law (distribution of most significant digits) in order to detect content manipulation.

Special attention will be paid to decoy selection strategies. Two scenarios will be discussed:

- A decoy selection from a database of images.
- A handcrafted decoy

Two counter-forensic algorithms were implemented and evaluated:

- The role of permutation coding in minimum-distortion perfect counter-forensics.
- The role of simplex in histogram counter-forensics.

The report is structured as follows: the background research, which is a compulsory part of any project. Then, after obtaining necessary skills, we will talk about the implementation and evaluation of the counter-forensic strategy using permutation decoding before introducing a new way of counter-forensics using simplex linear optimization in spatial domain.

Chapter 3: **Research**

In this section, the necessary research that was conducted in order to complete this project will be discussed. Each section will contain a short summary of the paper, followed by a take-home message i.e. which parts of the paper will/won't be used in the implementation and why.

3.1 Detecting traces of Re-sampling vs Tamper hiding

Before the research and implementation of counter-forensic strategies, the decision was made to look at the origin of anti-forensics. M. Kircher and R. Bohme (2007) in their paper, proposed a technique of tamper hiding against detecting traces of resampling proposed by A.Popescu and H. Farid (2005).

3.1.1 Detecting traces of resampling

The idea is that resampling can be detected by applying the expectation/maximization algorithm to estimate the sample pixels and their correlation to their neighbors (Popescu A.C. et al (2005)). Each pixel belongs to one of two sets. First, if a pixel is correlated to its neighbors, then it belongs to the model M_1, if not - to M_2. The EM algorithm was then executed in two steps:

- E-step, computes the probability of the pixel belonging to each model. This is estimated using Bayes' rule.
- M-step, estimates the specific form of correlations between pixels.

The result of EM algorithm classification is a probability map of two clusters corresponding to two models. While the forged image may be visually indistinguishable from the original, the periodic patterns in the probability maps are distinct. Also, the Fourier transformation of a resampled image will contain several sets of peaks corresponding to the method of resampling. Multiple consecutive re-samplings are as detectable as single ones. For better understanding of what is happening:

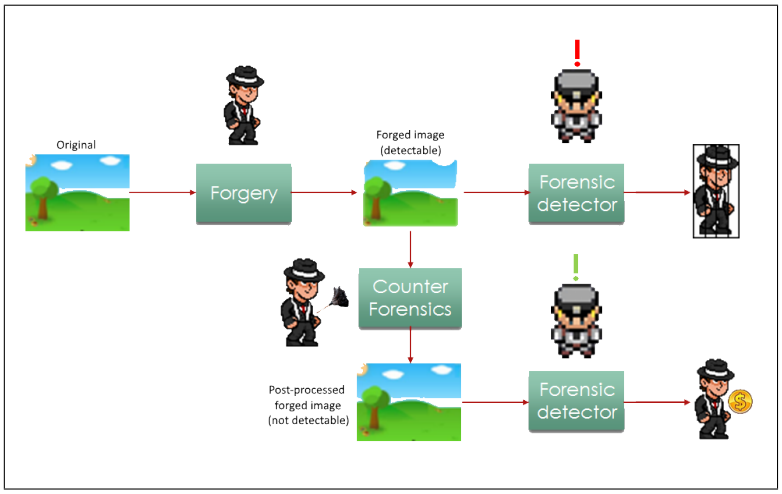
3.1.2 Tamper Hiding

Kirchner and Bohme (2007), proposed three different methods of image tamper hiding.

- *Non-linear filters*: The idea is to apply a median filter, which replaces each pixel by the median of all surrounding pixels. The attack was successful against Popescu A.C. et al(2005), but the quality of a post-processed image was quite poor i.e with this technique there is noticeable blurring.
- *Geometric distortion*: Applying naive geometric distortion to a forged image causes visible image blurring, which is most visually disturbing at straight lines and edges. Also, the Fourier transformation still looks awkward. To avoid such quality loss, they modulated edges using horizontal and vertical Sobel filters - to smoothen the image by emphasizing edges. At the end, with this approach, the resampled image must be transformed twice due to having to apply Sobel filters horizontally and vertically.
- *A dual path approach*: Undetectability can be improved by applying different operations to the high and low frequency components of the image signal. These two components are separated by a median filter. The low frequency component *LFC* is obtained by applying a median filter directly to the source image. The high frequency component *HFC* is then resampled with geometric distortion and edge modulation. Edge modulation is obtained from the resampled image prior the last median filter. The result is computed by adding together *LFC* and *HFC* resulting to a successful attack.

Now, when we explored one of the first counter-forensic technique and got a small bit of a taste of the topic, we can move further. Following sections will focus on attacking histogram based forensics.

Take-home message
Introduction to the origins of counter-forensics. We saw a forensic detector and an attack. Therefore the whole process can be demonstrated like this:



Couter-forensic process Fig(1)

An adversary starts with the original image and tampers with it (forgery step). The result of this is a tampered image which is detectable by the forensic analyzer. But if we cover the traces of tampering in the counter-forensics step, the forensic detector will not detect tampering, which is exactly what is desired. This serves as an introduction to the art of counter-forensics. In general, the application of counter-forensics is the same as ethical hacking. To improve existing forensic detectors, we shall constantly attack them.

3.2 A Universal Technique to Hide Traces of Histogram-based image manipulations

Barni et al. proposed a universal counter-forensic technique for concealing traces left on the image histogram by any processing tool(Barni M. et al(2012)). In this approach, the authors assume that the forensic analyst can only consider first order statistics i.e. histograms, and the adversary must satisfy the requirements of image quality. There are two assumptions to consider:

- The adversary doesn't need to know anything about the forensic analyst's detection algorithms.
- The adversary can use the proposed technique to hide traces produced by the forgery.

The process of hiding the traces of forgery is done in three steps:

- *Histogram retrieval*: among all histograms in the database of legitimate images, find the one (hx) that is most similar to the forged image (hy).
- *Mapping*: a transportation map to modify (hy), while satisfying some constraints of maximum image distortion.
- *Pixel remapping*: from the obtained displacement matrix, change image pixels according to it.

The goal of the mapping step, is to find a transportation map which moves a given distribution into another, minimizing function output. The problem proves to be MINLP, for which the global optimum solution exists. After the transportation map is obtained, the attacker needs to modify the image y into image new image z, which should not be too distant from y. Since the human visual system is known to be less sensitive to highly textured regions, we should add noise primarily to these areas. Furthermore, it makes sense to determine which parts are more sensitive to noise. The pixel remapping process is as follows:

- Set all pixels as admissible
- Compute a map of local variance of y.
- For each couple in the transportation map (i,j)
  - Find admissible pixels location having value i.
  - Scan them selecting the first $n(i¿j)$ with higher variance.
  - Substitute them with j.
  - If no more pixels of value i must be remapped, compute the SSIM between the current image and y.

Take-home message
The part of the most concern in this paper is a decoy selection process (Histogram retrieval). It is possible to make one image's histogram look like another. If a decoy is a legitimate image with a legitimate histogram, we can turn a histogram of a forged image into the undetectable one. It doesn't matter if the actual decoy image looks like the forged, what matters is the similarity of histograms. If two histograms are too distant - visually or mathematically - the output image will become distorted. This is discussed further in the implementation & results sections.

3.3 Counter-forensics for histogram-based forensics

The objective of this work was to prepare a general attacking method, with the distinctive feature of a single target function the optimization of which is pursued in the different steps of the attack design. This paper focuses on histogram based forensic detectors.The strategy proposed by Comesana and Perez-Gonzalez differs from the decoy selection technique proposed by Barni et al. in that rather than scanning and selecting the closest decoy from the database, it constructs both the forgery and the histogram directly from the original taking into consideration the type of forensic detector in order to identify the attack optimized to evade that detector.
Barni M. et al(2012) use different functions for each of the three solution stages. In the paper by Comesana-Alfaro P. et al(2013) instead, a single procedure is employed to prevent possible forensic methods. Since they derive a target image directly from the original, this allows determination of the value of the manipulation distortion in the original domain as a simple function of the original and the target distorted histograms. Therefore, 3-step optimization is reduced to one. Once a cumulative histogram is obtained, they give a closed rather than an iterative procedure to compute the signal in the original domain that produces the required histogram with minimal distortion. Since the iterative steps are now avoided, the computational cost is reduced. This can be represented as a formula:

$$g^H(H(\mathcal{B},\mathbf{y}),H(\mathcal{B},\mathbf{y}')) = \sum_{j=1}^{N}\left[H^{-1}\left(\tfrac{j}{N},\mathbf{y}\right)-H^{-1}\left(\tfrac{j}{N},\mathbf{y}'\right)\right]^2.$$

1-step optimizationComesana-Alfaro P. et al(2013)

,where g is the Euclidean distance, $H(B,y)$ is the cumulative histogram and $\frac{j}{N}$ represents a histogram's bin.
Also, in contrast to Barni M. et al(2012), in the work proposed by Comesana-Alfaro P. et al(2013) if the attacked is aware of the specific detector being used, he can take the advantage of such knowledge to come up with an optimal attack against it.

Take-home message
This solution, can be clearly considered as a decoy selector. The adversary always knows both the original signal and the forgery function. Therefore, he is aware of the type of artefacts that may appear in the forged image by comparing it to the original. The main issue then, is that a synthetic histogram cannot be called 'perfect' since the forger cannot guarantee the legitimacy of the synthetic decoy. For this reason, we are going to be using a decoy selection technique from work proposed by Barni M. et al(2012), since there is still a very high probability of finding a good decoy, by pulling from a database of legitimate images.

3.4 Permutation coding in minimum-distortion perfect counterforensics

The main idea is that any sequence of the processed forged image *(y)* with the same histogram as a given decoy *x* must be a rearrangement of it i.e. h(y) = h(x) iff y = Πx. The only strict condition for a decoy selection, is that it must be the same size as the forged image *z*. There are two scenarios that can lead to the discovery of an appropriate decoy:

- *Blind counter-forensics*: a decoy selection proposed by Barni M. et al(2012). A decoy can be obtained from a database of legitimate images, whose histograms are close to that of the *z*.
- *Non-blind counter-forensics*: Comesana-Alfaro P. and Perez-Gonzalez F.(2013) have shown that in this scenario it is possible to find an ideal decoy, considering the detection algorithm.

The main challenge for permutation decoding is to find a correct permutation. The straight forward idea would be taking *z*, computing all possible permutations for its pixel combinations, and applying them to a decoy while measuring the distortion of the image.

To obtain the number of possible permutations of a set of numbers, we can recall the following rule

$$\frac{n!}{(n-r)!},$$

,where *n* is the size of an image and *r*, is the size of a permutation. Since we require all pixels to participate, $n == r$. We can now shorten this formula to: *n!*. The problem with this approach is that for a regular 512x512 Lena image, there would be *(512x512)!* or 1.39635576863E+1306593 possible permutations. Therefore, this approach is not effective and will require too much computational power.

The solution was found in this paper. The ideal permutation can be obtained by sorting *z* and *x* in ascending order, taking sorting indices of *z* i.e. the permutation, and reapplying them to *x*. The result of this is a legitimate decoy sorted as the forgery which guarantees hiding of artefacts.

Take-home message
This strategy is one of the best seen so far, therefore it will be discussed further in the implementation section. The idea of finding the best permutation based of sorting, will be used in 'Future work' section, where we will be attempting the attack against the Benford's law forensics.

3.5 The optimal attack to histogram-based forensics detectors is simple(x)

In this paper, Comesana-Alfaro P and Perez-Gonzalez F. attack Benford's law forensics to hide the artefacts produced by a jpeg compression. Benford's law based forensic detectors are based on the relative frequency of most significant digits (MSD).

Benford's law: theory of leading digits. It holds true for a dataset that grows exponentially, but also appears to hold true for many cases in which an exponential growth pattern is not obvious. Benford law can recognize the probabilities of highly likely or unlikely frequencies of numbers in a dataset. The probabilities are based on mathematical logarithms of the occurrence of digits in a large dataset. In other words, Benfords law distribution is a probability histogram, representing nine digits. The visual Benford's law representation of some number sets can be found on the website [7].

In this paper, a number set represent the AC components of 8x8 DCT blocks. JPEG compression affects AC components such that their distribution does not comply with Benford's law. The proposed solution is as follows:

- Computation of AC components of each 8x8 DCT block.
- For each of these frequencies the non-null coefficients are modified in order to comply with Benford's law.

The modification step was implemented using simplex algorithm, since the problem is classified as a linear optimization problem. The two equality constraints are required.

- Each variable must only have one value.
- The distribution of variables must comply with Benford's law.

Simplex constraints will be discussed further in the implementation part.

Take-home message
After reading this paper, an idea has occurred, to try using simplex on the spatial domain i.e. instead of looking at AC components to hide jpeg compression, simplex will be tested on actual pixels in order to make a detectable histogram of a forged image to look like a decoy's histogram.

3.6 The ubiquitous Kronecker product

$\otimes$ The Kronecker product has a rich and very pleasing algebra that supports a wide range of fast and practical algorithms (Van Loan C.F. (2000)). The Kronecker product is used in thriving application areas such as signal processing, image processing and quantum computing. Also, sparse factorizations and Kronecker product are proving to be very effective way to look at fast linear transforms. As computers get more and more powerful, researchers are dealing with problems of high dimension and this leads to Kronecker products whenever low-dimension techniques are tensored together.

Basic properties
Let's say we have two matrices B and C, then their Kronecker product would be $B \otimes C$.

$$\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} = \left[\begin{array}{ccc|ccc} b_{11}c_{11} & b_{11}c_{12} & b_{11}c_{13} & b_{12}c_{11} & b_{12}c_{12} & b_{12}c_{13} \\ b_{11}c_{21} & b_{11}c_{22} & b_{11}c_{23} & b_{12}c_{21} & b_{12}c_{22} & b_{12}c_{23} \\ b_{11}c_{31} & b_{11}c_{32} & b_{11}c_{33} & b_{12}c_{31} & b_{12}c_{32} & b_{12}c_{33} \\ \hline b_{21}c_{11} & b_{21}c_{12} & b_{21}c_{13} & b_{22}c_{11} & b_{22}c_{12} & b_{22}c_{13} \\ b_{21}c_{21} & b_{21}c_{22} & b_{21}c_{23} & b_{22}c_{21} & b_{22}c_{22} & b_{22}c_{23} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \end{array}\right]$$

Kronecker product basics Fig(2)

Basically, we take each element of B and multiply it by matrix C.

The basic properties of the Kronecker product are straight forward:

- $(B \otimes C\ )^T = B^T \otimes C^T$
- $(B \otimes C\ )^-1 = B^{-1} \otimes C^{-1}$
- $(B \otimes C)(D \otimes F) = DB \otimes CF$
- $B \otimes (C \otimes D) = (B \otimes C) \otimes D$

In Kronecker product work, matrices are sometimes reshaped as vectors and vectors are sometimes made into matrices. Matrix reshaping into a vector is represented by *vec(.)*. Vectorization happens by stacking columns of the matrix.

For example, for the 2×2 matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, the vectorization is $\mathbf{vec}(A) = \begin{bmatrix} a \\ c \\ b \\ d \end{bmatrix}$.

Matrix to vector[12]Fig(3)

The Kronecker product can be used to represent linear equations in which the unknowns are matrices such as *AXB=C*(Schacker K. (2013)), where A,B,C are given matrices and the matrix X is the unknown, then it is possible to rewrite the equation as:
($B^T \otimes$ A) vec(X) = vec(AXB) = vec(C)
Note that such equations are used to represent a linear optimization problem (simplex) and equality constraints

Take-home message
Kronecker product is very useful when optimizing a function where the unknown is a matrix. This is because of the relationship between the Kronecker product and the vectorization operator vec(.) that takes a matrix and unwinds it into a long vector. Kronecker product will be used in the calculation of simplex equality constraints in section 4.2.

3.7 Image Quality metrics: PSNR vs SSIM

In this paper, Alain Hore et al. analyze two well-knows objective image quality metrics: the peak-signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM)(Hore A. et al(2010)). The PSNR value approaches infinity, therefore a small value of the PSNR implies high numerical differences between images. The SSIM is a well-known metric developed by Z.Wang et al(2004), and is considered to be correlated with the quality perception of the human visual system. In contrast to PSNR, SSIM represents image distortion based on three factors: loss of correlation, luminance distortion, and contrast distortion.
There are no precise rules for selecting the SSIM or the PSNR when the evaluation of image quality is required. Studies have shown that the PSNR, has the best performance in assessing the quality of noisy image (e.g. DCT block image manipulation).
Both measures were tested against Gaussian noise, blur, jpeg and jpeg2000 compression. F-score measure was used to test the sensitivity of both measures and results have shown that PSNR is very sensitive to Gaussian noise and SSIM is more sensitive to jpeg compression.

Take-home message
SSIM was designed to improve the traditional methods like PSNR and MSE, which have proven to be inconsistent with human eye perception. Therefore, SSIM will be used as a distortion measure section 4. The main reason is that both permutation decoding and 4.2. guarantee a statistically legitimate image (histograms), due to legitimacy of a decoy, the only thing we need to worry about is the human visual system.

Chapter 4: **Implementation**

This paper will now describe the two counter-forensic strategies implemented in this project. First strategy, is fully based on work proposed by Balado F.(2014). Second, is an attempt to reproduce results from permutation decoding, by using simplex linear optimization, rather than permutation decoding. In this chapter we are only going to focus on the actual counter-forensic algorithms. Forgeries will be discussed in the Results chapter. The decision was made to use MATLAB 2015a for permutation decoding and Octave 4.0 for linear optimization. The main reason for using MATLAB and Octave, is that they come with a lot of image processing, statistics and visualization functions. Having such functionality will make the programmer's job easier. The experiments were conducted by using images from the MIRFLICKR dataset. The dataset contains 25000 legitimate images.

4.1 Permutation decoding

The main idea of this strategy is to find a permutation of a decoy's pixels such that the output image looks like the forged image. This is done in three steps,

- A decoy selection based on distance function.
- Find a rearrangement matrix to make a decoy look like the forgery.
- Use a SSIM image distortion metric to obtain the best result.

4.1.1 Decoy selector

Recalling Balado. F(2014), we can go ahead and try to implement this strategy. We start by taking the forged image and retrieving its histogram. Then a decoy selector was implemented based on chi-squared distance function. The chi-squared distance is a nonlinear metric and is widely used to compare histograms (Yang W. et al (2015)).

The main idea is to use K-NN algorithm to obtain *N* decoys, whose histogram is most similar to forgery. The whole decoy selection process is as follows:

- Compute a histogram of the forged image *h(x)*.
- Calculate the distance between *h(x)* and all images in the database, using chi-squared distance.
- Create a HashTree<distance, filename>.
- Select top N entry values from the HashTree, denoted as *D*.

The reason of using a HashTree is that MATLAB's 'map' function automatically sorts keys, therefore we can save some time. Each potential decoy will be denoted as d_i and its histogram as

h(d_i). Notice that, in this phase of the strategy, we are more interested in retrieving a histogram that is near to *h(x)* from the 'shape' point of view rather than the statistical one. Therefore, it may happen that the best matching histogram is statistically too distant from *h(x)*, which will cause a serious image distortion. Consequently, we select the top *N* candidates from the database, and run the permutation decoding on all of them. Among these *N* candidates, the one resulting in the least distortion will be used to produce the output.

4.1.2 Permutation decoding

The optimal permutation decoding solution is as follows:

- Put both a forged image and d_i into a vector form.
- Sort pixels of the forged image in ascending order.
- Record the original positions of pixels (denoted as Π).
- Sort pixels of d_i.
- Unsort d_i using Π.
- Reshape the output to the original size

MATLAB provides a sort() function, which outputs the sorted data set and Π. Therefore, the whole permutation decoding process can be done like this:

```
vecI = I(:);
vecDecoy = decoy_res(:);
[~, ind] = sort(vecI, 'ascend');
sort_vecDec = sort(vecDecoy, 'ascend');
out_vec(ind) = sort_vecDec;
out = reshape(out_vec, size(I,2), size(I,1));
```

Permutation decoding Fig(4)

,where *ind* is Π, *vecI* is the forged image in vector form, *vecDecoy* is d_i in vector form, *out_vec* is the output. Note that the decoy and the forged image must be the same size, otherwise a permutation would be impossible.

4.1.3 Selecting the best output

Since we know that there is a chance that the best decoy (based of distance function) may not produce the best output, instead of just using d_1 all *N* decoys tribute as candidates for the best solution. The SSIM value is computed for each output. The decoy with the best SSIM, is selected for the final result. In the Result chapter, the entire process is demonstrated.

Note that with this approach, by the end, we have completely eliminated the forged image, instead the pixels of a decoy were rearranged such that it looks like the forgery. Since the intensity values of pixels were never actually touched, the decoy histogram never changes. Therefore, because a decoy is a legitimate image, so is the output.

4.2 The role of simplex in histogram counterforensics.

The motivation for trying out simplex on pixel histogram counter-forensics came from work proposed by Comesana-Alfaro et al(2014).The idea is to get the same or similar result as in permutation decoding by using simplex. The decoy selection process stays as normal, but pixel remapping will be done using linear optimization.

4.2.1 Problem definition

First, let's start with recalling permutation decoding. The output can be represented as $y{=}\Pi x$, where y is the output, Π is the permutation matrix and x is a decoy. Another way to represent an image would be *Av*, where A is a large matrix of ones and zeros with *(mxn)x256* elements where *m.n* are dimensions of the image and v is the column vector of pixel intensities ranging from 1..256. Each row of A represents an individual pixel, and contains only a single 1 and all other elements of the row are zeros. Each column must add up to the number of times a pixel occurs in the image i.e. its frequency. After multiplying Av, the result is an image in a vector form.
glpk Octave function was used for this strategy. The function accepts 5 compulsory parameters:

$$A = glpk(f,C,b,lb,ub)$$

f a column vector containing the objective function coefficients. f is obtained by using a Kronecker product.

$$f = -v \otimes double(z)$$

,where v is the vector of intensities, z is the forged image. Our objective is to obtain matrix A that minimizes the following function:

$$min\ f'A, \text{ equivalent to } max\ -f'A$$

Since the linear optimization solver performs its operations on vectors, therefore f=f(:).

4.2.2 Equality constraints

C is the matrix of constraints that must be satisfied in the solution. In this subsection, we will show the way of representing constraints using Kronecker product and their right hand side. First, we want to make sure that each row of A, adds up to 1 i.e. each pixel can only have one single intensity value in its row.

$$A*1{=}1$$

The constraint can be represented as:

$$vec(I_m A1) = vec(1)$$

equivalent to,

$$(1^T \otimes I_m) * vec(A) = vec(1)$$

,where I_m is the identity matrix of size mxm, *m* is the size of a forged image (r x q), 1 is the vector of pixel intensities. The right hand side of the equation for this constraint is a column vector of ones of size *m x 1*
This is where problems started to occur. Assume we have a 500 x 500 image and a vector of pixel intensities $v = 1 : 256$, then m = 500 x 500. The result matrix of this constraint would be (1*m)*(256*m) = 262144 x 67168864, too big for the normal computer to process. At this stage we can surely say, that this approach is not efficient and will have no use in the real world. But we still want to prove the concept, that this works, therefore in the results part of this report, we will stick to small 64 x 64 bit image to prove the concept.

Second constraint, ensure that each column of matrix A will add up to the same value as in the decoy. This will guarantee that the output image will have identical to the decoy histogram.

$$1^T A = h^T$$

The constraint can be represented as:

$$vec(1^T A I_q) = vec(h^T)$$

equivalent to,

$$(I_q^T \otimes 1^T) * vec(A) = vec(h^T)$$

, where I_q is the identity matrix of size 256x256, 1 is a column vector on ones of size (m*m)x1 and h^T is the histogram of a decoy. The right hand side is the histogram of a decoy. Now, we can stack two constraints to obtain *C*, and stack right hand side vectors to obtain *b*. *ub* and *lb* represent upper and lower bounds. Since output matrix A must only contain ones and zeros, upper bound is a column vector of ones and lower bound is a column vector of zeros, both are size (m*256)*1 (number of columns in *C*). Now, once we figured out all required inputs, we can finally plug them into *glpk* function and wait for the result.

4.2.3 Output computation

The result comes in a form of a column vector *s* of size (m*256) x 1. To obtain the image we shall follow the steps:

- Reshape *s* so it looks like *A*: *A = reshape(s,m,256)*.
- Multiply A by a column vector on pixel intensities: *A = A*v'*, the result is a column vector of pixels.
- Reshape A to the original image size: *A = reshape(A, r, q)*.
- Finally, since pixels values are doubles at this stage, cast them to uint8 to obtain the final result: *A = uint8(A)*.

Chapter 5: **Results**

In this chapter, the results produced by two strategies will be discussed. First,a few different types of forgeries will be demonstrated and their effect on histograms, and later the results and efficiency of two strategies described in previous chapter will be compared.

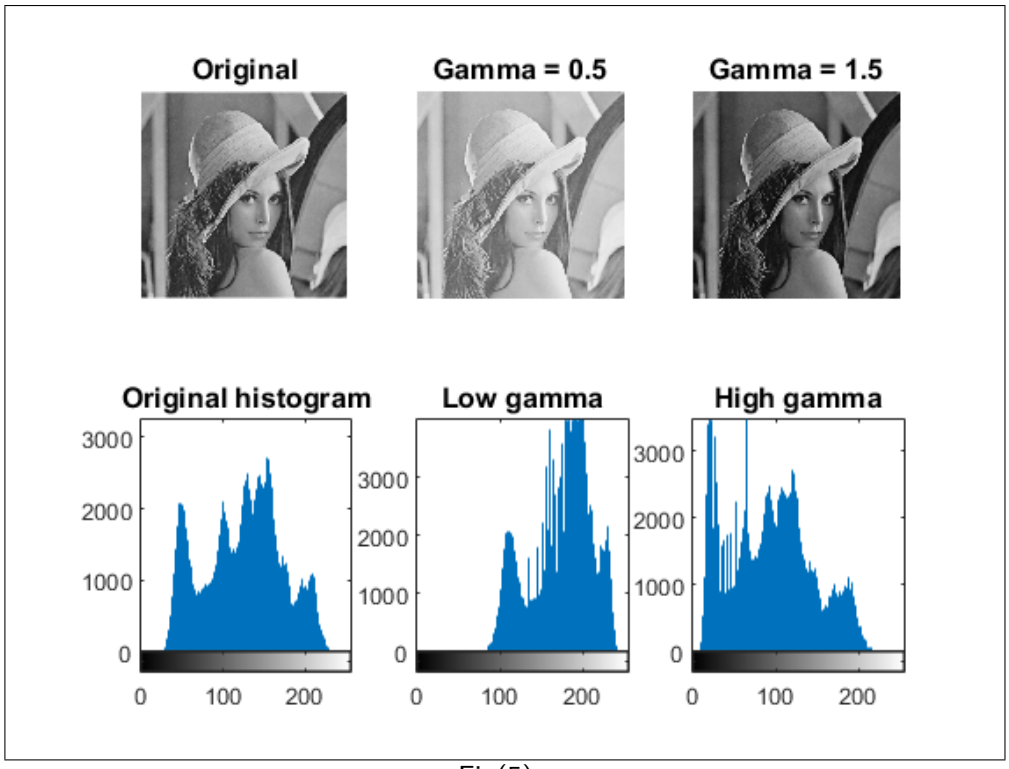
5.1 Forgery

Before testing counter-forensic strategies, it is desired to know what how a legitimate histogram looks like and what kind of artefacts does the forgery produce. Let's look at two main types of image tampering based on contrast enhancement.

Gamma-Correction
Gamma correction controls the overall brightness of an image. The formula is:

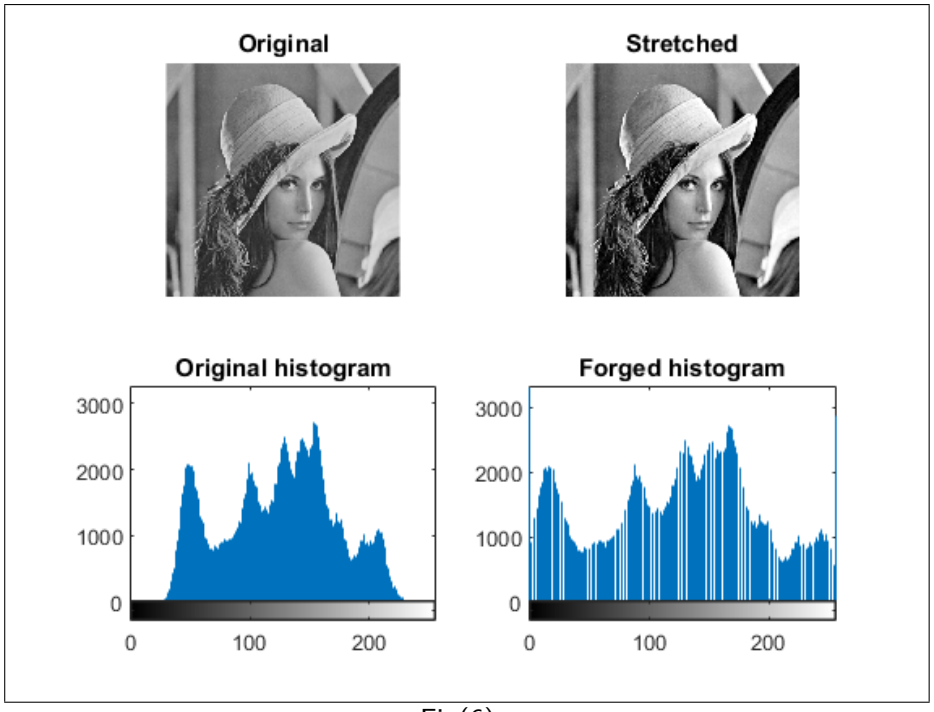
$$z_i = \text{round}(255(w_i/255)^{\gamma})$$

,where z is the output pixel, w is the input pixel. As can be observed from the formula, gamma-correction performs rounding. This causes peaks appear on the histogram. Also, depending on γ value, the histogram will shift to the left if γ is high, to the right is it is low.



Fig(5)

Histogram stretching
Histogram stretching is an attempt to improve the contrast on the image by stretching its histogram across all possible intensity values. The artefacts produced are gaps in between histogram bins.



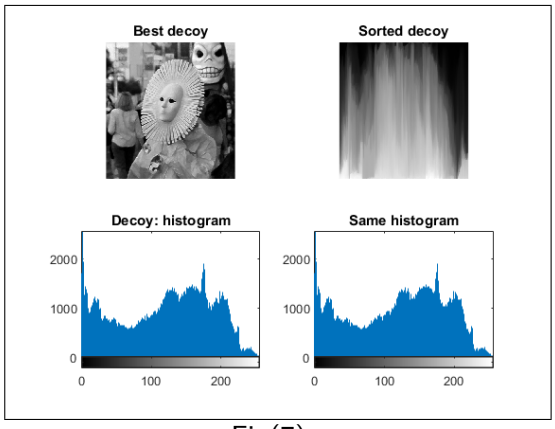
Fig(6)

As can be observed from these figures, the original, legitimate image's histogram should look nice and smooth, there should not see any large gaps or peaks in the histogram. This is what we are trying to achieve by applying counter-forensics strategies.

5.2 Output

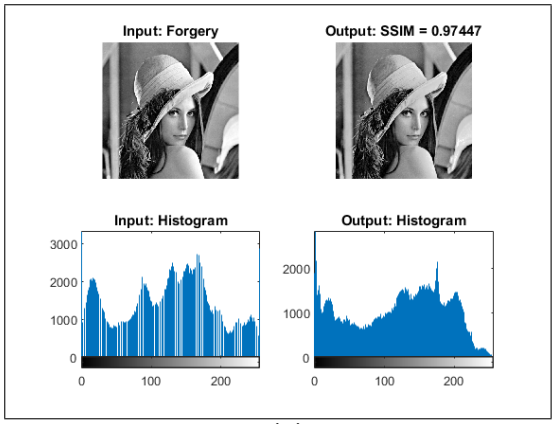
5.2.1 Permutation decoding

To demonstrate the result or permutation decoding strategy, the original Lena image was taken and its histogram was stretched(Figure(6)). Now the decoy selector comes into action. We will look ahead and pick the best decoy for demonstration.



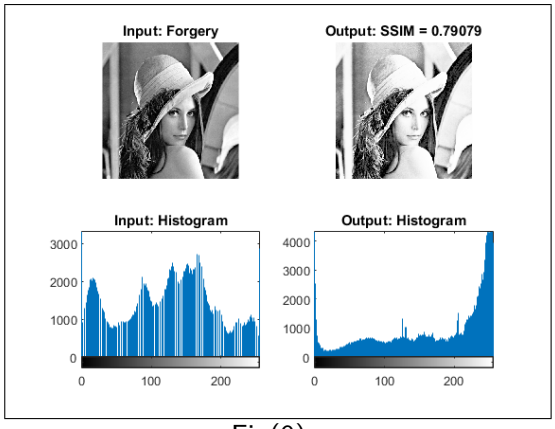
Fig(7)

Fig(7) is the best possible decoy to Fig(6). Note that the histogram never changes, since no changes were performed to actual pixels, only their positions were swapped. Then, the same thing was done to Fig(6), and a permutation matrix was obtained. After applying Π onto the sorted decoy, this is the achieved result:



Fig(8)

The SSIM is pretty impressive. The output image has a very little distortion compared to the forged image. This means, the properties of histogram stretching were saved, while a histogram was turned from detectable to undetectable.
But what happens if a decoy is too distant from the forged image? To demonstrate this, a random decoy was selected from the database.



Fig(9)

As can be observed, the distortion is too big, since a decoy selection was not based on the distance

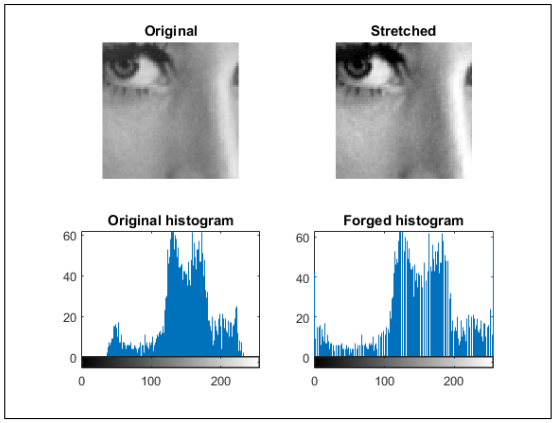
function. Note that, we cannot use the original Lena image as a decoy, if we do, all properties of histogram stretching will be lost, resulting the forgery step being useless.

5.2.2 Simplex

Reminder: since simplex constraints are too big for the normal computer to process, the tests will be performed on smaller images of size 64 x 64. Since, such images can automatically be classified as not-legitimate, the simplex approach will be used as a proof of concept, i.e. the strategy will work perfectly fine on a stronger computer, with no limitations on the size of constraint matrices, but still will take a lot of time to produce the result. Therefore, the image distortion will be ignored and we will focus on the histogram only.

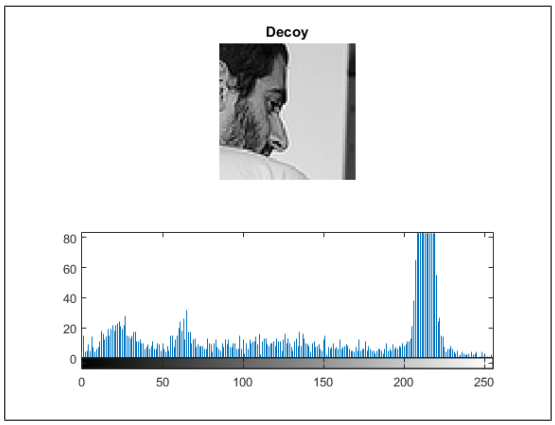
In this part, it is demonstrated, that the result of simplex is the same or very near to permutation decoding. Simplex guarantees mathematically the best possible result, and if it is the same as in permutation decoding, then we are proving mathematically rather than analytically, that permutation decoding also produces the best possible result.

Start by taking Lena image, cutting out a 64 x 64 piece and applying histogram stretching on it.



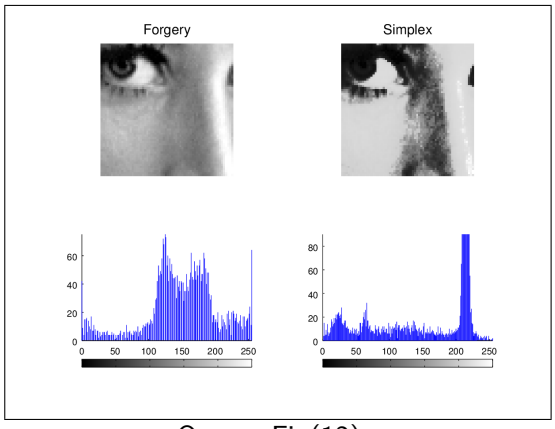
Fig(10)

Then a decoy was picked, by cutting out a 64 x 64 piece from a random image from the database.



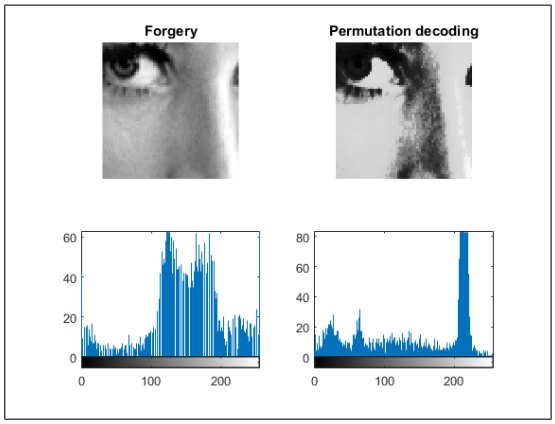
Fig(11)

Now, simplex is applied, the task is to change Fig(10) such that its histogram looks like in Fig(11).



Octave Fig(12)

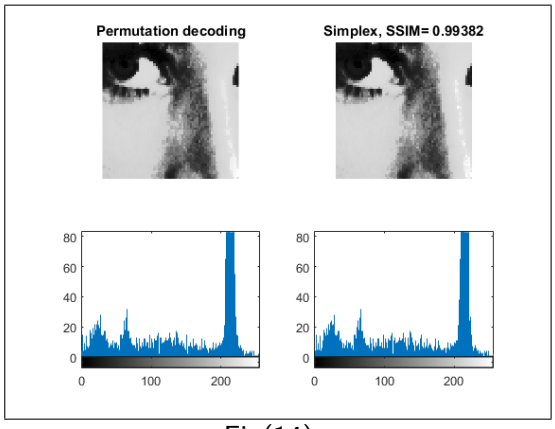
As can be seen, the image's histogram has been successfully forged to match that of the decoy. Permutation decoding then produces:



MATLAB Fig(13)

Comparing the forgery histograms, a slight difference can be observed because of differences in how Octave and MATLAB display histograms. However, forgeries produced by Octave and MATLAB are the same, hence SSIM $= 1$.

Now let's compare the results produced by simplex and permutation decoding.



Fig(14)

The SSIM $= 0.99382$, which is a great result. We have reached near ideal score.

5.2.3 Conclusion

We have compared two different counter-forensic strategies. Permutation decoding produces the analytically best result and simplex produces mathematically best result. We have seen that the output of both strategies is nearly the same.

Simplex runs at the complexity *O(2^n)* and permutation decoding at *O(nlog(n))*, where n is the number of pixels. Such high complexity, makes simplex very inefficient for the analysis of large matrices, therefore there is no point in using it in counter-forensics in the spatial domain. Since, permutation decoding, produces almost the same result, that makes it the most optimal and efficient strategy for histogram counter-forensics.

Chapter 6: **Future work**

In this section the another attempted counter-forensic strategy will be discussed and demonstrated, that unfortunately turned out to be ineffective. This can be considered as an idea for future research.
In [5], we have seen a mathematical approach of attacking Benford's law forensics using simplex to hide traces of jpeg compression. AC components of every non-compressed image in the DCT domain, must follow the Benford's law pmf. This was an inspiration to try an analytical approach and see if it is possible to achieve positive results using solely permutation decoding.

Plot Benford's Law histogram
To see if the image follows Benford's Law, we shall compute the histogram of leading digits of an image's AC components. This is done by dividing an image into 8 x 8 DCT blocks and removing the DC components. Then for each AC component, its most significant digit is obtained using the formula from the simplex paper by Comesana-Alfaro et al(2014).

$$d(x) = \left\lfloor \frac{|x|}{10^{\lfloor \log_{10}(|x|) \rfloor}} \right\rfloor .$$

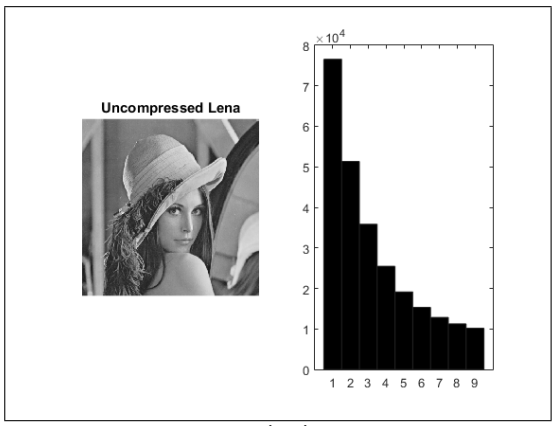
Fig(15)

A set of numbers verify Benford's law if their MSD is distributed according to the following probability mass function:

$$P(d) = \log_{10}\left(1 + \frac{1}{d}\right) .$$

Fig(16)

After obtaining most significant digits, we can plot a histogram.



Fig(17)

As we can observe, the histogram of untampered image, follows the Benford's law pmf. On the other hand, jpeg compressed images does not.

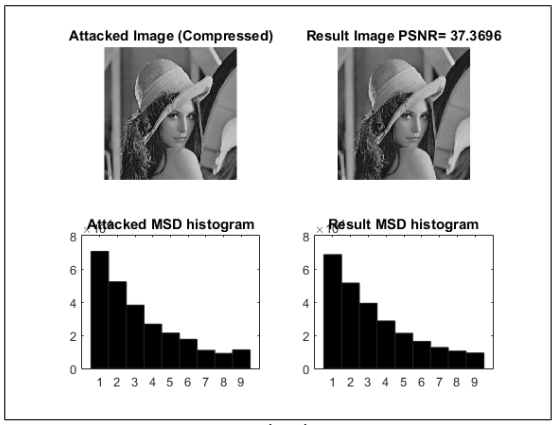
DCT decoy
Now let's move onto actual counter-forensic process. First, a new decoy selector is required since we are working in different domain. To obtain a decoy, we attempted following steps:

- Divide an image into 8 x 8 DCT blocks.
- Create a mask m = ones(8,8).
- Set $m(1,1) = 0$.
- Apply the mask onto each block to remove the DC components.

We do the same thing for the forged image and for each potential decoy in the database. We then select the top N decoys based on the distance function.

Attacking Benford's law forensics, with permutation decoding
Once a list of decoys is obtained, permutation decoding is applied and PSNR is used for the quality measure.

- Start by dividing both forgery and a decoy into 8 x 8 DCT blocks.
- Remove DC components, but save DC components in a separate matrix (to restore image later)
- Put resulting matrices into a vector form.
- Apply permutation decoding as in [4].
- Reshape the result back to image matrix.
- Restore DC components by adding them to the result matrix.
- Perform inverse DCT, and cast the result to uint8 to obtain final output image.



Fig(18)

The histogram of most significant digits was fixed such that they comply with Benford's Law, but unfortunately the result images PSNR is below acceptable i.e. there are noticeable artefacts in the output image.
Conclusion
This approach is not useful in counter-forensics, since it causes image distortion. Anyway, this can be used as an idea for future research for how to defeat Benford's law forensics analytically rather than mathematically[5].

Chapter 7: **Conclusion and Learning outcome**

I would like to conclude this report by giving my opinion about the final year project process and learning outcomes.

Everybody likes positive results. This is why many scientific journals tend to accept only papers with positive results. But producing negative result is also very important. As scientists continue publishing negative results, the newer generation of researchers will not waste their time and money repeating the same studies and finding the same results. This is the case for 5.2.2 and 6. We tried two counter-forensic strategies that have never been done before, and obtained negative result so nobody has to do it anymore. Scientists have the responsibility to study the topic and report everything, including the negative findings.

I picked this project because in my opinion it is a lot more challenging than the majority of proposed projects. I really wanted to get a taste of the research process, and am overall happy about accomplishing it. The most challenging part was reading and understanding scientific papers related to image processing which even professional scientists find challenging. As a hard working student, who has never really done any image processing or research before, I eventually learned how to correctly read papers, how to make sense of mathematical notations and how to extract useful information from poorly/well written papers. My mathematical skills has improved a lot in the past 7 months and also I learned how to program scripts.

I am delighted to be a part of the hectic academic environment. I have learned more than I expected during the period of the final year project. I would like to thank UCD School of Computer Science for the given opportunity. This ends my final year project report.

Bibliography

[1] Balado F. (2014). *The role of permutation decoding in minimum-distortion perfect counter-forensics.* 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[2] Barni M., Fontani M., Tondi B. (2012). *A universal technique to hide traces of histogram-based image manipulations.* Coventry: Proceedings of the on Multimedia and security.

[3] Comesana-Alfaro P. , Perez-Gonzalez F. (2013). *Optimal counterforensics for histogram-based forensics.* Vancouver: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing.

[4] Comesana-Alfaro P., Perez-Gonzalez F. (2014). *The optimal attack to histogram-based forensic detectors is simple(x).* Atlanta: 2014 IEEE International Workshop on Information Forensics and Security (WIFS).

[5] Gloe T., Kirchner M., Winkler A., Bohme R. (2007). *Can We Trust Digital Image Forensics?* Augsburg: ACM MM 2007.

[6] Hore A., Ziou D. (2010). *Image Quality Metrics: PSNR vs. SSIM.* Istanbul: 20th International Conference on Pattern Recognition.

[7] Huiskes M., Thomee B., Lew M. (2008). *The MIRFLICKR Retrieval Evaluation.* http://press.liacs.nl/mirflickr/.

[8] Kirchner M., Bohme R. (2007). *Tamper Hiding: Defeating Image Forensics.* Saint Malo: Information Hiding, 9th International Workshop, IH 2007.

[9] Long J. Thornton B. *Testing Benford's Law.* http://testingbenfordslaw.com/

[10] Popescu A.C., Farid H. (2005). *Exposing Digital Forgeries by Detecting Traces of Resampling.* IEEE Transactions on Signal Processing, vol. 53, no. 2.

[11] Schacke K. (2013). *On the Kronecker Product.* Pages 2-11.

[12] Van Loan C.F. (2000). *The ubiquitous Kronecker product.* Journal of Computational and Applied Mathematics. Pages 85-100.

[13] Wang Z., Bovik A.C., Sheikh H.R., Simoncelli E.P. (2004). *Image quality assessment: from error visibility to structural similarity.* IEEE Transactions on Image Processing, vol. 13, no. 4.

[14] Wikipedia - https://en.wikipedia.org/wiki/Vectorization_(mathematics)

[15] Yang W., Xu L., Chen X. , Zheng F., Liu Y. (2015). *Chi-Squared Distance Metric Learning for Histogram Data.* Mathematical Problems in Engineering, Volume 2015 (2015), Article ID 352849, 12 pages.