HIDING DATA INSIDE A CLOUD

Sorin Radoveneanu[1], Cătălin Leordeanu[2], Valentin Cristea[3]

Maintaining data confidentiality is one of the most important problems of protection against attacks or against accidental misuse. The rapid development of the Internet, along with storage and data transfer technologies, increased the complexity of such problems. Therefore, robust and secure data storage is an important issue for which a single appropriate solution is difficult to find. Among the most common techniques we find steganography and cryptography, which address target different objectives.

This paper proposes a solution which combines the two techniques in order to improve confidentiality. Due to the proposed solution an attacker is unable to understand the secret message and through the inclusion of steganography techniques the existence of the message is hidden inside an image. We prove that the proposed solution is efficient and that it provides data confidentiality, making it useful to transfer data over unsecured channels, or for data storage on an environment for which the user has no absolute control, such as a Cloud System.

Keywords: confidentiality, steganography, privacy, plausible deniability

1. Introduction

As the Internet is an open communication medium, any transferred files are exposed to attacks. They can be intercepted or modified in transit, without the knowledge of the sender or the intended destination. Data stored on remote servers, such as Cloud-based ones, is even more exposed to attacks[1]. Cloud data storage services offer very few security guarantees, and the users have no direct control over their data[2].

Encryption helps protect the data by modifying the data and making it very difficult to understand by an attacker. However, only encryption algorithms are not enough to protect very sensitive data. Given enough time or computing power an attacker could attempt to break the encryption and understand the secret data, or the owner of the data could be constrained to reveal the secret key through other means. Steganography[3] is a different

[1] Faculty of Automatic Control and Computers, University "Politehnica" of Bucharest, Romania, e-mail: sorin.radoveneanu@cti.pub.ro

[2]Faculty of Automatic Control and Computers, University "Politehnica" of Bucharest, Romania, e-mail: catalin.leordeanu@cs.pub.ro

[3]Faculty of Automatic Control and Computers, University "Politehnica" of Bucharest, Romania, e-mail: valentin.cristea@cs.pub.ro

approach to the issue of data confidentiality, which attempts to hide the existence of the sensitive data. This method also offers plausible deniability for the owner of the secret data.

For example, one of the most common methods in steganography is called "LSB insertion"and it is used to hide secret messages into images. The principle of this method is based on the modification of the least significant bit from the 24 bit representation of each pixel, to form a secret message. Since only the least significant bit is modified the difference in the image will be minimal and the secret message will only be read by someone who knows of its existence. This paper aims to exceed these limitations and propose a robust data hiding solution which offers a sufficient level of confidentiality, as well as plausible deniability for the owner.

The rest of the paper is structured as follows. Section 2 describes similar research efforts, with an emphasis on deniable encryption solutions. Section 3 describes the proposed solution and the flow of operations to store and retrieve a secret message. Section 5 contains experimental results using the proposed solution and Section 6 concludes this paper and outlines directions for future research.

2. **Related work**

There are many research efforts which handle data confidentiality. One such solution is covered by the Deniable File Systems (DFS)[5]. This refers to file systems which can hide a small part of itself. It differs from classic Encrypted File Systems through the fact that the folders and files are not visible.However, a disadvantage of the Deniable File Systems is the fact that it covers the entire partition or file system. It is not very useful if the user only wishes to hide a small secret message inside another file. StegFS[7] is also a file system which is able to hide the existence of different sets of data. It is actually an extension of the Ext2fs file system for linux and the data can be encrypted as part of different security levels. An attacker would not be able to know if he has discovered the secret keys to all the security levels.

Another solution to ensure data confidentiality is TrueCrypt[6]. It falls into the category of Deniable File Systems, which we mentioned earlier. It is able to use different cryptographic algorithms, such as AES, Serpent, Twofish and others to encrypt in real-time an entire disk or a partition. It can also offer plausible deniability through the creation of a hidden disk partition, which is kept inside a visible partition. It has however a number of security vulnerabilities[6] and it is also difficult to configure.

A basic solution for deniable encryption is presented in [8]. It is however difficult to implement and use. Most other solutions have used the term "plausible deniability", which brings it closer to steganography through the insertion of secret data inside encrypted files.

3. Data hiding

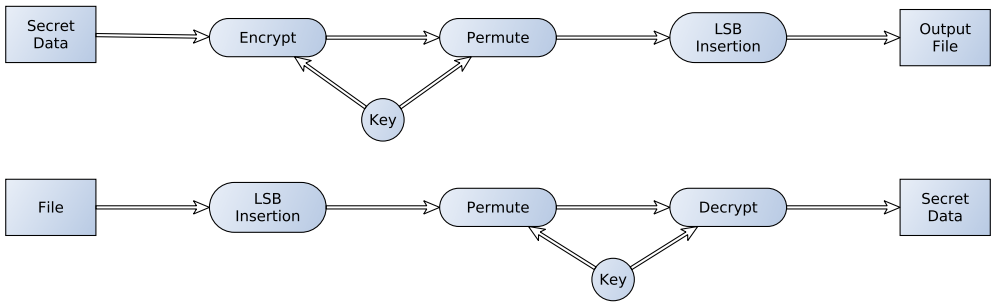

FIG. 1. Operation flow for storage and retrieval of secret data

Figure 1 presents the necessary operation flow to obtain a file with an embedded secret message. As an additional security measure, the encrypted information is permuted, meaning that the initial data is divided in sub-blocks of fixed length, which are then shuffled according to a key-based pattern. The same permutation key is also used to encrypt the data using the AES algorithm. The last step in the storage operation is the insertion of the resulted encrypted secret data into a target file using the LSB method. To encrypt the data we obtained we used the Advanced Encryption Standard (AES) algorithm[13] using keys of 256bits. Currently, the use of 256bit keys for AES is considered secure[14].

To obtain the secret information from a file, the operation flow is similar. First, the embedded message is extracted through the same LSB method. After this, the permutation is reversed and then the data is decrypted using the same key. Of course, we will need the same key as the one used in the storage operation due to the use of symmetric encryption algorithms.

3.1. LSB insertion

Least Significant Bit(LSB)[4] insertion is one of the most common steganographical methods today, which is used to hide small amounts of data in audio or image files. In the case of image steganography, this approach is based on the modification of the least significant bit and exploits the inability of the human eye to distinguish very small differences for each pixel. In a similar way, the human ear cannot detect very small differences for the frequencies in audio files.

This approach is easier to use in the case of PNG and BMP image files, which are lossless formats. For a 24 bit encoded image, the RGB values are encoded using 8 bits each. When adding hidden data we can modify the least significant bit of each group of 8 bits. Therefore, we could encode 3 bits of hidden date for each pixel in the image. Based on the existing values for these bits, we have a 50% chance of changing the existing value of the Red, Green

or Blue LSB bit from 0 to 1 or from 1 to 0. It is possible to apply this method for lossy image formats, such as JPG. In this case the approach involves the modification of DCT(Discrete Cosine Transformation) coefficients [9].

Such a simple steganography solution has a number of disadvantages. For example, if an attacker is aware of the existence of the secret data, it is very simple for him to extract it since it is not encoded in any way. Another important aspect is the choice of the audio or image file in which to encode the secret data. If the user chooses a very common or widespread file to encode a hidden message, then it could be detected by comparing the original hash of the file.

3.2. Key-based permutation

As an additional security measure, the entire data block is divided into blocks of the same size, which are then shuffled. The actual permutation is based on a key derived from the passwords used for the encryption and the number of blocks that the data has been divided into. The permutation we used is based on the Key Based Random Permutation (KBRP) algorithm presented in [10].

This permutation can be divided into three steps:

(1) Initialization of the permutation key. Considering that the data we need to permute has been divided into N blocks we need to create a vector of the same size which will be the permutation key. Each element of the permutation key will be filled according to the ASCII value of a character from the password. Considering that the password which we use has a number of characters n¡N we are left with a number of N-n positions in the key which we need to fill. This is done by adding two adjacent values and inserting the result on the first position. Afterwards, the values of the permutation key are brought into the interval $[0, N)$. Considering that the character of the password has an ASCII value of X, the value of the element of the permutation key will be X mod N. This step ends when all the N positions of the permutation key have been filled.

(2) Elimination of duplicates. If there are any duplicates inside the permutation key then the first value will be kept unchanged and the rest will be replaced by the value 0.

(3) Filling the blanks. All the values of the permutation key must be brought into the interval $[1, N]$. All the values of 0 will be replaced with values which are not already present in the permutation key, starting from both ends of the key. This step ends when all the elements of the permutation key belong to the interval $[1, N]$.

The permutation is useful because it adds another level of protection for the data. An attacker would not be able to analyze the cyphertext until he has reversed the permutation. Since the permutation scheme is based on the passwords for the files it also does not require any other secret key from the

user. After the end of the permutation stage, the result is the final form of the file containing the secret information. The process of retrieving the stored information is identical with the storage operations which we described in this section, but the necessary stages are executed in reverse order.

4. The Chi-Square Attack

Since the LSB method may change the frequency of the displayed colours of the image, we used the Chi Square Attack[15] to attempt to detect hidden data. The Chi Square attack uses statistical methods to try to detect the amount of hidden data.

The algorithm compares the Pairs of Values (PoVs) which are formed by the pixels during the steganographic process due to the change of the least significant bit. If the embedded data is equally distributed then the frequencies of both the odd and even components of the PoVs should be equal. The Chi-Square attack [16] compares the theoretical expected frequency of the least significant bit of the image using an equally distributed message, as opposed to the measured frequency in the sample image. The result is a probability of existence of embedded data. Since we usually do not have access to the original image the theoretically expected frequency distribution is calculated as the arithmetic mean of the odd and even frequencies of the PoVs. We consider that each pixel can belong to one of k colour categories. In the case of 8-bit greyscale images there are a total of 128 categories or PoVs, since we consider a category to contain both an odd and an even value.

Therefore, the theoretically expected frequency in category i after embedding an equally distributed message is:

$$n_i^* = \frac{|\{colour|sortedIndexOf(colour)\epsilon\{2i, 2i+1\}\}|}{2}$$

The measured frequency of occurence for category i is:

$$n_i = |\{colour|sortedIndexOf(colour) = 2i\}|$$

The x^2 statistic, with $k-1$ degrees of freedom, is defined as:

$$x_{k-1}^2 = \sum_{i=1}^{k} \frac{(n_i - n_i^*)^2}{n_i^2}$$

Finally, the probability p that embedded data exists in the image is defined as the integration of the density function, considering that the distribution of n_i and n_i^* are equal. The probability p is defined as:

$$p = 1 - \frac{1}{2^{\frac{k-1}{2}}\Gamma\left(\frac{k-1}{2}\right)} \int_0^{x_{k-1}^2} e^{-\frac{x}{2}} x^{\frac{k-1}{2}-1} dx$$

When working with images where pixels are represented using 24 bits (8 bits for each colour channel), the number of categories rises to 2^{23}.

5. Test Cases

For the implementation of the image manipulation part of the solution we used OpenCV [11]. It is an open source computer vision library, first released in 1999 under the BSD license, free for both commercial and academic use. For the AES encryption algorithm we used the BouncyCastle cryptographic library [12]. Due to the characteristics of LSB steganography, since we can only add data in the least significant bit, the theoretical limit for the secret data which we can insert is 12.5% of the total size of the image file.

5.1. Results

We used an image of size 302x446 pixels, to which we added an increasing amount of hidden data, through the aforementioned methods. The size of the hidden data was 3,95KB, 9.07KB, 14,6KB and 25KB.

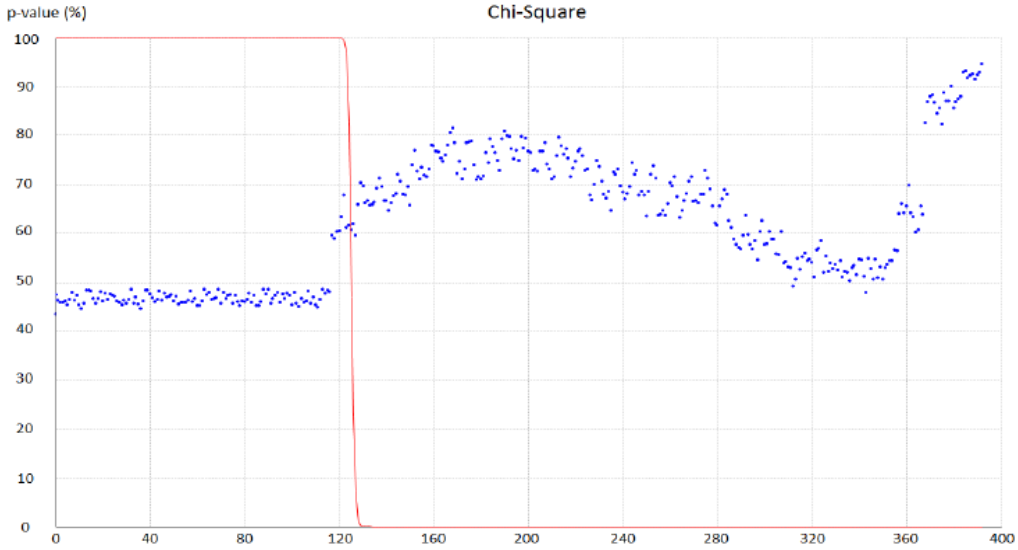

FIG. 2. Chi-Square attack for an image with 14,6KB of hidden data using only the LSB method.

In Figure 2, Figure 3 and Figure 4 we tested the probability $p(\%)$ for 14,6KB of hidden data using the elements of the proposed solution. In Figure 5 we showed all the experiments for the Chi-Square attack side by side to show the use rate of the hidden data.

As expected, the experiments where we used all the stages of the proposed solution (AES+KBRP+LSB) added the most additional data, which is visible in Figure 5. The Key Based Random Permutation (KBRP) step also adds some additional data due to the presence of the normalization and padding steps of the algorithm. However, the most additional data is added by the padding for the AES encryption algorithm. As a tradeoff for the lack of efficiency this

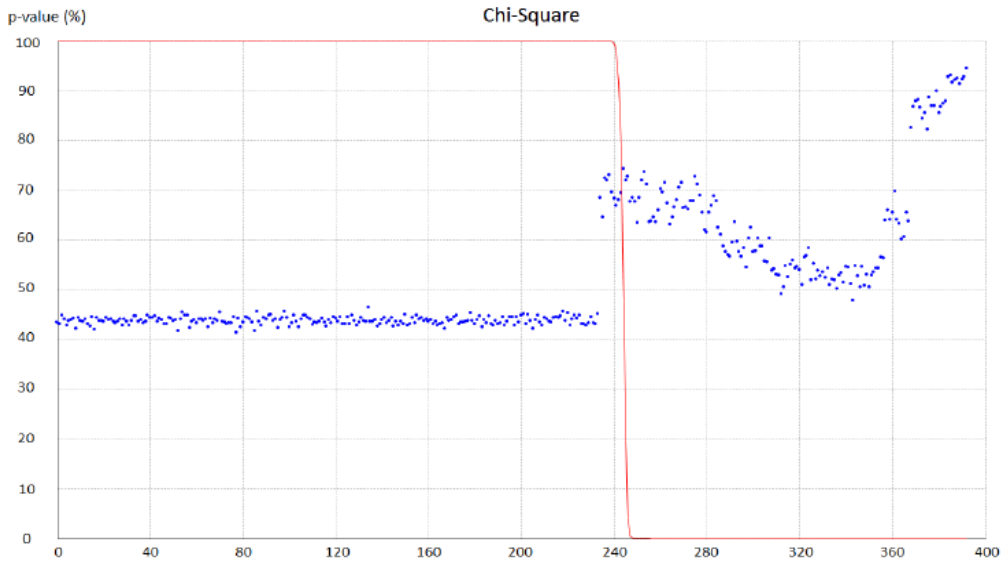

FIG. 3. Chi-Square attack for an image with 14,6KB of hidden data using AES encryption and LSB.

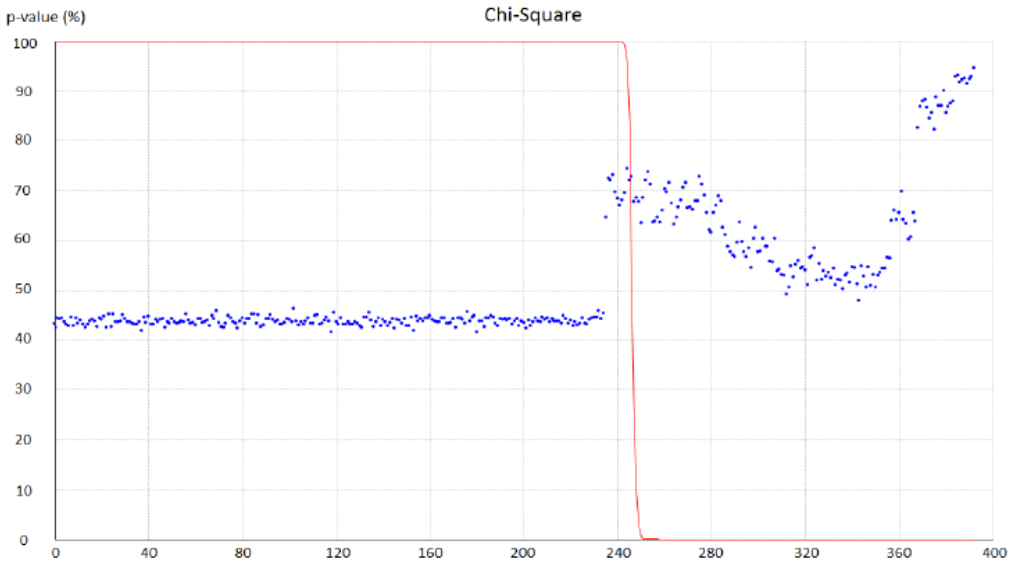

FIG. 4. Chi-Square attack for an image with 14,6KB of hidden data using the entire confidentiality solution (AES + KBRP + LSB).

method offers the highest level of security for the hidden data. From the figures above we can observe a relatively uniform distribution of the PoVs.

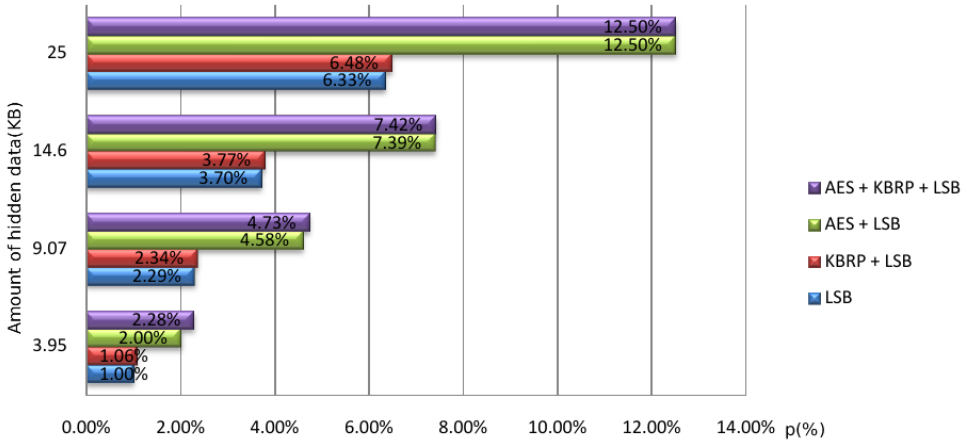

FIG. 5. The probability p for different amounts of hidden data.

We also performed a set of experiments to determine the performance of the proposed solution, shown in Figure 6. These experiments were performed on a machine with an Intel i5 processor and 4GB of RAM.

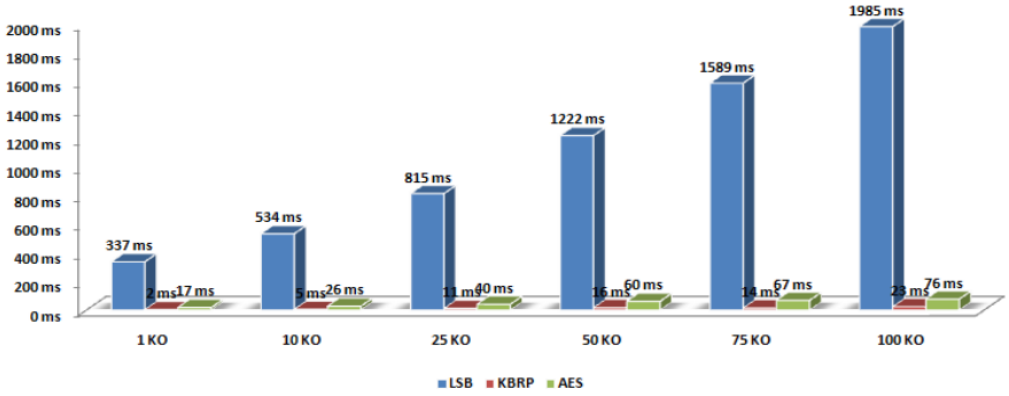

FIG. 6. Performance in milliseconds of each step of the solution.

Since the overall performance depends only on the amount of hidden data, we performed experiments where we inserted 1KB, 10KB, 25KB, 50KB, 75KB and 100KB respectively on larger images. The results obtained for the data insertion and data extraction operations were identical. The actual values can be seen in Figure 6.

From these results we can draw the conclusion that the operation with the largest running time is the LSB insertion. This is understandable because the algorithm needs to sequentially alter the value of each pixel. The KBRP and AES encryption require the least amount of computational resources, especially

for very small amounts of data. Also, the execution time is proportional to the amount of hidden data.

6. Conclusions and future work

The solution for data confidentiality proposed in this paper can be used when sending data over an unsecure communication channel, as well as storage in a remote environment where the user has no security guarantees, such as a Cloud System. There are many online services offering image storage and this solution would enable users to hide confidential information from potential attackers.

As opposed to traditional confidentiality methods the user is therefore granted plausible deniability regarding the existence of the embedded data, as well as another layer of protection due to the permutation and encryption steps. The efficiency of the proposed solution was tested thoroughly using the Chi-Square attack and we can see the effect of embedding different amounts of hidden data and for different steps of the workflow. The security of the solution is directly dependent on the security of the encryption algorithm. For this solution we used the AES cypher which is currently considered secure. We also added an additional level of security through a key-based permutation of the resulting data. The proposed solution combines steganography and cryptography methods in order to ensure data confidentiality. Usually such a method is used to hide only small amounts of confidential data and we have tested for up to 100kB.

As future work we intend to improve the performance of the solution. Since the Least Significant Bit stage uses the most computational resources, its performance could be improved by using a GPU implementation in CUDA or OpenCL. Since the operations performed on each pixel can be easily separated the algorithm is suitable for a parallel implementation.

Acknowledgement

The work has been funded by the "*Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds* " through the Financial Agreement POSDRU/159/1.5/S/ 134398.

REFERENCES

[1] *Kandukuri, Balachandra Reddy, V. Ramakrishna Paturi, and Atanu Rakshit.* Cloud security issues. Services Computing, 2009. SCC'09. IEEE International Conference on, pp. 517-520, IEEE, 2009.

[2] *Wang, Cong, Qian Wang, Kui Ren, and Wenjing Lou.* Privacy-preserving public auditing for data storage security in cloud computing. In INFOCOM, 2010 Proceedings IEEE, pp. 1-9. IEEE, 2010.

[3] *Johnson, Neil F., and Sushil Jajodia.* Exploring steganography:Seeing the unseen. Computer 31, no. 2 (1998): 26-34.

[4] *Fridrich, Jessica, Miroslav Goljan, and Rui Du.* Reliable detection of LSB steganography in color and grayscale images. In Proceedings of the 2001 workshop on Multimedia and security: new challenges, pp. 27-30. ACM, 2001.

[5] *Anderson, Ross, Roger Needham, and Adi Shamir.* The steganographic file system. In Information Hiding, pp. 73-82. Springer Berlin Heidelberg, 1998.

[6] *Czeskis Alexei, David J. St Hilaire, Karl Koscher, Steven D. Gribble, Tadayoshi Kohno, and Bruce Schneier.* Defeating Encrypted and Deniable File Systems: TrueCrypt v5. 1a and the Case of the Tattling OS and Applications. In HotSec.2008.

[7] *McDonald, Andrew D., and Markus G. Kuhn.* Stegfs: A steganographic file system for linux. In Information Hiding, pp. 463-477. Springer Berlin Heidelberg, 2000.

[8] *Canetti, Rein, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky.* Deniable encryption. In Advances in CryptologyCRYPTO'97, pp. 90-104. Springer Berlin Heidelberg, 1997.

[9] *Cox, Ingemar J., Joe Kilian, F. Thomson Leighton, and Talal Shamoon.* "Secure spread spectrum watermarking for multimedia." Image Processing, IEEE Transactions on 6, no. 12 (1997): 1673-1687.

[10] *S.M. Hussain, N.M. Ajlouni*, Key Based Random Permutation, Journal of Computer Science 2 (5): 419-421, 2006

[11] *Culjak, Ivan, David Abram, Tomislav Pribanic, Hrvoje Dzapo, and Mario Cifrek.* "A brief introduction to OpenCV." In MIPRO, 2012 Proceedings of the 35th International Convention, pp. 1725-1730. IEEE, 2012.

[12] *Castle, Bouncy.* "Bouncy castle crypto apis." URL http://www.bouncycastle.org/.(Cited on page 82.) (2007).

[13] *Miller, Frederic P., Agnes F. Vandome, and John McBrewster.*"Advanced Encryption Standard." (2009).

[14] *Rogawski, Marcin, Kris Gaj, and Ekawat Homsirikamol.* "A high-speed unified hardware architecture for 128 and 256-bit security levels of aes and grstl." Embedded Hardware Design: Microprocessors and Microsystems (2013).

[15] *Westfeld, Andreas, and Andreas Pfitzmann.* "Attacks on steganographic systems." In Information Hiding, pp. 61-76. Springer Berlin Heidelberg, 2000.

[16] *Zanganeh, Omid, and Subariah Ibrahim.* "Adaptive image steganography based on optimal embedding and robust against chi-square attack." Information Technology Journal 10, no. 7 (2011): 1285-1294.