STEGANOGRAPHY

Secure Way to Hide Text within Image

Atalay Aşa & Halis Çağrı Akdeniz & Büşra Gedeli - 14 June 2017

Abstract

The art of information hiding has received much attention in the recent years as security of information has become a big concern in this internet area. As sharing of sensitive information via a common communication channel has become inevitable, Steganography – the art and science of hiding information has gained much attention. We are also surrounded by a world of secret communication, where people of all types are transmitting information as innocent as an encrypted credit card number to an online-store and as insidious as a terrorist plot to hijackers[1].

Steganography is a technology where modern data compression, information theory, spread spectrum, and cryptography technologies are brought together to satisfy the need for privacy on the Internet.

Steganography is technically process of hiding a secret message within a larger one in such a way that someone can not know the presence or contents of the hidden message. The cover media may be text, image, voice or video. Many different carrier file formats may be used , but the digital images are the most popular one because of their availability on the Internet. In order to hiding secret information in images, there are several ways of steganographic techniques exist, some of are more complex than others and all of them have strong and weak points according to their specific features. This project report intends to give an overview of image steganography, its usability and techniques. In this paper, LSB (Least Significant Bit) technique will be proposed for Image

Steganography to hide secret message and and AES encryption will be used to encrypt that secret message in order to make the message more difficult for unauthorised people to extract the original message. We explore the performance of Java image processing applications designed with multithreading approach. In order to test how the multithreading influences on the performance of the program, we tested several image processing algorithms implemented with Java language using the sequential one thread and multithreading approach on single and multi-core CPU.

1. Introduction

In today's world, the communication is the basic necessity of every growing area. Everyone wants the secrecy and safety of their communicating data. In our daily life, we use many secure pathways like internet or telephone for transferring and sharing information, but it's not safe at a certain level.One of the reasons that makes intruders successful is the most of the information they acquire from a system is in a form that they can read and comprehend. Intruders may reveal the information to others, modify it to misrepresent an individual or in organization, or use it to launch an attack. The main goal of steganography is to communicate securely in a completely undetectable manner such that no one can suspect that there is some secret information.

Unlike cryptography, which secures data by transforming it into another unreadable format, steganography makes data invisible by hiding (or embedding) them in another

piece of data [2]. Thus, cryptography is science of overt secret writing while steganography as covert secret writing. The cover, host or the carrier is the target media in which information is hidden so that other person will not notice the presence of the information. The modified cover, including the hidden data, is referred to as a stego object which can be stored or transmitted as a message. The secret information can be embedded in various types of covers. If information is embedded in a cover text (text file), the result is a stego-text object. Similarly, it is possible to have cover audio, video and image for embedding which result in Stego Audio, Stego Video and Stego Image, respectively.Nowadays, the combinations of steganography and cryptography methods are used to ensure data confidentiality and to improve information security. Following work presented here revolves around steganography in digital images and does not discuss other types of steganography(such as linguistic or audio) and in the Table-1 comparison of steganography with watermarking and encryption has been made.

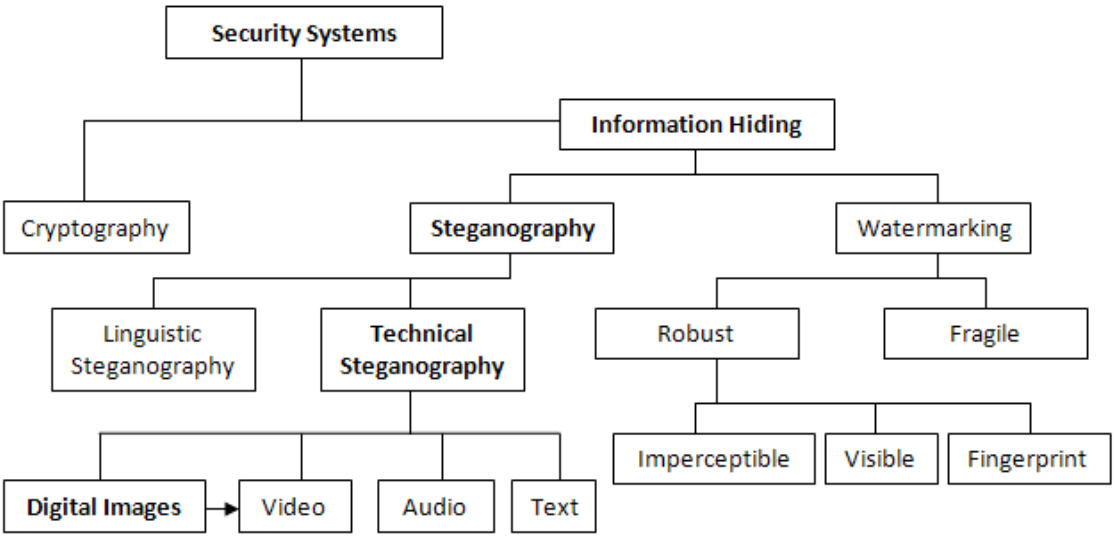


Fig. 1. The different embodiment disciplines of information hiding. The arrow indicates an extension and bold face indicates the focus of this project.

Table 1. Comparison of steganography, watermarking and encryption.

Criterion/Method	Steganography	Watermarking	Encryption
Carrier	any digital media	mostly image/audio files	usually text based, with some extensions to image files
Secret data	payload	watermark	plain text
Key	optional		necessary
Input files	at least two unless in self-embedding		one
Detection	blind	usually informative (i.e., original cover or watermark is needed for recovery)	blind
Authentication	full retrieval of data	usually achieved by cross correlation	full retrieval of data
Objective	secrete communication	copyright preserving	data protection
Result	stego-file	watermarked-file	cipher-text
Concern	delectability/ capacity	robustness	robustness
Type of attacks	steganalysis	image processing	cryptanalysis
Visibility	never	sometimes (see Fig. 2)	always
Fails when	it is detected	it is removed/replaced	de-ciphered
Relation to cover	not necessarily related to the cover. The message is more important than the cover.	usually becomes an attribute of the cover image. The cover is more important than the message.	N/A
Flexibility	free to choose any suitable cover	cover choice is restricted	N/A
History	very ancient except its digital version	modern era	modern era

1.2 Steganography Applications

Steganography is employed in various useful applications, e.g., copyright control of materials, enhancing robustness of image search engines and smart IDs (identity cards) where individuals' details are embedded in their photographs. Other applications are video-audio synchronization, companies' safe circulation of secret data, TV broadcasting, TCP/IP packets (for instance a unique ID can be embedded into an image to analyze the network traffic of particular users), and also checksum embedding [3]. Petitcolas [4] demonstrated some contemporary applications, one of which was in Medical Imaging Systems where a separation is considered necessary for confidentiality between patients' image data or DNA sequences and their captions, e.g., physician, patient's name, address and other particulars. A link however, must be maintained between the two. Thus, embedding the patient's information in the image could be a useful safety measure and helps in solving such problems.[5]. Steganography would provide an ultimate guarantee of authentication that no other security tool may ensure.

Inspired by the notion that steganography can be embedded as part of the normal printing process, the Japanese firm Fujitsu is developing technology to encode data into a printed picture that is invisible to the human eye (data), but can be decoded by a mobile phone with a camera as exemplified in Fig. 2-a and shown in action in Fig. 2-b. The process takes less than one second as the embedded data is merely 12 bytes.

Hence, users will be able to use their cellular phones to capture encoded data. They charge a small fee for the use of their decoding software which sits on the firm's own servers. The basic idea is to transform the image colour scheme prior to printing to its Hue, Saturation and Value components (HSV), then embed into the Hue domain to which human eyes are not sensitive. Mobile cameras can see the coded data and retrieve it.

This application can be used for "doctor's prescriptions, food wrappers, billboards, business cards and printed media such as magazines and pamphlets" [6], or to replace barcodes.

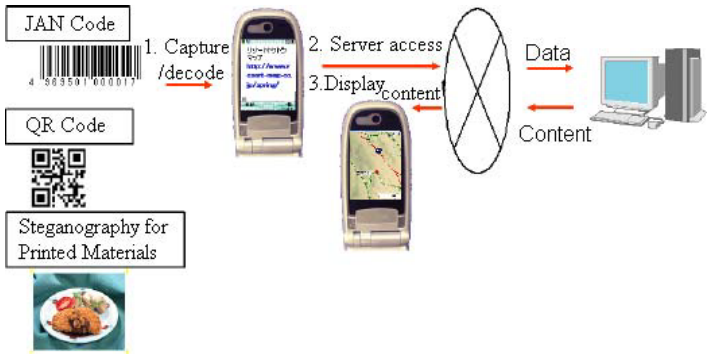


Figure 2-a

Figure 2-b

Fig. 2. Fujitsu exploitation of steganography: (a) a sketch representing the concept and (b) the idea deployed into a mobile

2. Image Steganography

Steganography can be classified into various types, depending upon necessity. Hence, it can be set to occur in four types which names are text, image, audio and video. In this paper we will focus on the use of steganography within digital images (PNG, JPEG, BMP). This steganography technique is more popular in the recent years than other steganography techniques, possibly because of the flood of electronic image information available with the advent of digital cameras and high-speed internet distribution. It can involve hiding information in the naturally occurred noise within the image. Most kinds of information contain some kind of noise. Noise refers to the imperfections inherent in the process of rendering an analog picture as a digital image. In image steganography we can hide message in pixels of an image. An image steganographic scheme is one kind of steganographic systems, where the secret message is hidden in a digital image with some hiding method [7]. Someone can then use a proper decoding procedure to recover the hidden message from the image. The original image is called a cover image in steganography, and the message-embedded image is called a stego image [8] [9]. Various methods of image steganography are:

i) **Data Hiding Method:** Hiding the data, a username and password are required prior to use the system. Once the user has been login into the system, the user can use the information (data) together with the secret key to hide the data inside the chosen image. This method is used to hiding the existence of a message by hiding information into various carriers. This prevents the detection of hidden information [10].

ii) **Data Embedding Method:** For retrieving the data, a secret key is required to retrieving back the data that have been embedded inside the image. Without the secret key, the data cannot be retrieved from the image. This is to ensure the integrity and confidentiality of the data. The process of embedding the message inside the image, a secret key is needed for retrieving the message back from the image, the secret message that is extracted from the system is transfer into text file and then the text file is compressed into the zip file and zip text file is converting it into the binary codes [11].

iii) **Data Extracting Method:** It is used to retrieve an original message from the image; a secret key is needed for the verification. And for extracting method, a secret key is needed to check the key is correct with the decodes from the series of binary code. If key is matched, the process continues by forming the binary code to a zipped text file, the unzip the text file and transfer the secret message from the text file to retrieve the original secret message [11].

2.1 Features of Image Steganography

- **Transparency :** The original image should not be affected steganography process. It should remain same.
- **Data payload or capacity :** This feature is about how much data should be embedded as a steganography to extract data successfully.
- **Robustness :** Robustness feature is the ability of the hidden message to remain undamaged even if the steganographic image undergoes transformation, sharpening, linear and non-linear filtering and other various techniques.

3. Image Steganography Techniques

There are several techniques to conceal information inside cover-image. Image steganography techniques can be divided into two groups: those in the Image Domain and those in the Transform Domain [12]. Image – also known as spatial – domain techniques embed messages in the intensity of the pixels directly, while for transform – also known as frequency – domain, images are first transformed and then the message is embedded in the image [13].

Image domain techniques encompass bit-wise methods that apply bit insertion and noise manipulation and are sometimes characterised as “simple systems” [14]. The image formats that are most suitable for image domain steganography are lossless and the techniques are typically dependent on the image format [15].

Steganography in the transform domain involves the manipulation of algorithms and image transforms [14]. These methods hide messages in more significant areas of the cover image, making it more robust [16]. Many transform domain methods are independent of the image format and the embedded message may survive
conversion between lossy and lossless compression [15].

In the next sections steganographic algorithms will be explained in categories according to image file formats and the domain in which they are performed.

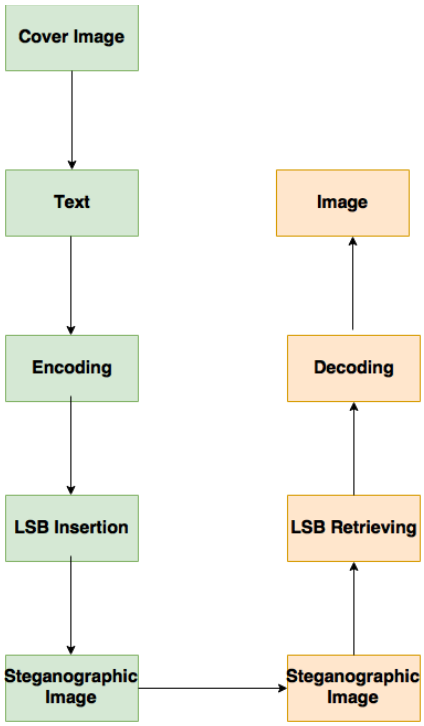


Image Steganography Process with LSB Technique

3.1 Image(Spatial) Domain

In this technique images are represented by pixels. Simple watermarks could be embedded by modifying the pixel values or the least significant bit (LSB) values . It directly loads the raw data into the image pixels. Some of its algorithms are LSB, SSM Modulation based technique.

- **Least Significant Bit**

The Least Significant Bit (in other words the 8th bit) is one of the main techniques in spatial domain image steganography. LSB Substitution works by iterating through the pixels of an image and extracting the ARGB values. It then separates the colour channels and gets the least significant bit. Meanwhile, it also iterates through the characters of the message setting the bit to its corresponding binary value. Thus, if we use 24-bit image, a bit of each of the red, green and blue colour components can be used, since they are represented by bits. In other words, one pixel can store 3 bits in each pixel. For example if we have 800 x 600 pixel image, we can store 1,440,000 bits or 180,000 bytes of

R	G	B
0101110	1011010**0**	1110010**0**
1110011	0011010**1**	1011011**0**
0101001	1101101**0**	1101101**0**

embedded data. The data that hides may be increased through big size pictures. Following demonstration is for 3 pixels of 24-bit coloured image :

Bold binary numbers is represented as least significant bit of each pixel. Now, if we have ‘A’ letter which binary representation is ‘01000001’ and if we embed that ‘A’ letter into our sample pixel representation, the resulting grid as follows :

R	G	B
0101110	1011010**1**	1110010**0**
1110010	0011010**0**	1011011**0**
0101000	1101101**1**	1101101**0**

Although the number was embedded into the first 8 bytes of the grid, only 5 underlined bits needed to be changed according to ‘A’ letter binary representation. Since there are 256 possible intensities of each primary color, changing the LSB of a pixel results in small changes in the intensity of the colors. These changes cannot be visible by the human eye due to the message hidden. In these consecutive bytes of the image data – from the first byte to the end of the message – are used to embed the information. And easy to detect, more secure system for the sender and receiver to share a secret key that specifies only some pixels to be changed In its simplest form, LSB makes use of BMP images, since they use lossless compression. It hide a secret message inside a BMP file, one would require a very large cover image. In BMP images of 800x600 Pixels are not often used on the Internet and might arouse suspicion. For this reason, LSB steganography has also been developed for use with other image file formats [17]. It is a simple method for embedding data in a cover image. This is the simplest algorithm in

which information can be inserted into every bit of image information. Given an image with pixels, and each pixel being represented by an 8-bit sequence, the watermarks are embedded in the last (least significant bit) of selected pixels of the image proposed a simple data hiding technique by simple LSB substitution. In this technique last bit of host data is randomly changed and produce the watermarked data at output. The cover LSB media data are used to hide the message [18].

- **Pixel Value Differencing:**

This method provides both high embedding capacity and outstanding imperceptibility for the stego-image; this segments the cover image into non overlapping blocks containing two connecting pixels and it modifies the pixel difference in each pair for data embedding.

- **Pixel Indicator:**

This method gives the stego images of better quality than the traditional method while maintaining a high embedding capacity and it also uses concept of hiding the data using the difference between the pixel values. It's more complex way of hiding information in an image. Transformations are used on the image to hide information in. Transform domain embedding can be termed as a domain of embedding techniques in frequency domain; image is represented in terms of its frequencies.

3.2 Frequency(Transform) Domain

Transform domain techniques embed secret information in a transform space of the signal. With the transform domain steganography algorithms data can be embedding into a couple type of file formats such as JPEG, BMP etc. and among them JPEG file format is the most popular image file format on the Internet, because of the small size of the images.

- **Discrete Cosine Transformation:**

These methods convert the uncompressed image into JPEG compressed type[19].It is based on data hiding used in the JPEG compression algorithm to transform successive 8x8- pixel blocks of the image from spatial domain to 64 DCT coefficients each in frequency domain. [20] The main advantage of this method is its ability to minimize the block like appearance resulting when boundaries between the 8x8 sub-images become visible (known as blocking artefact).

- **Discrete Wavelet Transformation:**

It gives the best result of image transformation. It splits the signal into set of basic functions and there are two types of wavelet transformation one is continuous and other is discrete. This is the new idea in the application of wavelets, in this the information is stored in the wavelet coefficients of an image, instead of changing bits of the actual pixels.

It also performs local analysis and multi-resolution analysis. DWT transforms the object in wavelet domain and then processes the coefficients and performs inverse wavelet transform to show the original format of the stego object.

4. Combining Steganography Technique with Cryptography

Cryptography and steganography are well known and widely used techniques that manipulate information (messages) in order to cipher or hide their existence respectively. As it is mentioned earlier, steganography is the art and science of communicating in a way which hides the existence of the communication and cryptography scrambles a message so it cannot be understood.

In this application we will focus on a system that uses both cryptography and Steganography for better confidentiality and security. In the steganography side LSB technique will be used in order to embed the text data and cryptography side Advanced Encryption Standard(AES) algorithm will be used for encrypting the data. The reason behind using AES algorithm with LSB technique is that, some adversaries can predict that LSB technique is used in the given picture and they tried to extract the data and when they do that resulting data must be meaningless for them. So, the data given to the our application will be first encrypted with AES algorithm which is the most secure public key encryption algorithm then will be embedded into the picture.

4.1 Advanced Encryption Standart(AES) Features

The Advanced Encryption Standard (AES) original name was Rijndael, as it was named after the developers of the algorithm; Vincent Rijmen and Joan Daemen. The Algorithm was submitted as a proposal to the NIST (National Institute of Standards and Technology) in 1996 during a competition done to select an encryption standard to replace the existing one at that time DES (Data Encryption Standard) which was published in 1977. The NIST committee announced the AES as the US standard in November 26, 2001. AES uses symmetric key encryption where the same key is used for encrypting and decrypting the data, and the main challenge is to exchange that key with complete privacy as if this key is found then all the encryption process is compromised and useless but of course there are couple ways to prevent this happened.

AES belongs to a family of ciphers known as block ciphers. A block cipher is an algorithm that encrypts data on a per-block basis. The size of each block is usually

measured in bits. AES, for example, is 128 bits long. Meaning, AES will operate on 128 bits of plaintext to produce 128 bits of ciphertext. Like almost all modern encryption algorithms, AES requires the use of keys during the encryption and decryption processes. AES supports three keys with different lengths: 128-bit, 192-bit, and 256-bit keys.

The longer the key mean that the stronger encryption. So, AES 128 encryption is the least strong, while AES 256 bit encryption is the strongest.

In terms of performance though, shorter keys result in faster encryption times compared to longer keys so in our project since we are unaware of the amount of total data AES 128 bit key will be used in order to provide fast processing.

4.2 AES with Block Cipher Modes

Earlier we mentioned that AES belongs to a family of ciphers known as block ciphers but block ciphers encrypt only fixed-size blocks. A block cipher, as it shown, encrypts data one block at a time. Many common block ciphers have block sizes of 64 and 128 bits, and for most practical applications, we need to encrypt data of sizes much greater than a single block. A mode of operation is a scheme in which a block cipher is adapted to handle data encryption and decryption in bulk. So, using AES encryption without proper mode would not be secure and robust, as a result we have to select appropriate block cipher mode for our AES encryption.

National Institute of Standards and Technology(NIST) has defined 5 modes of operation for AES and other FIPS-approved ciphers[MODES]:

- CBC(Cipher Block Chaining)
- ECB(Electronic Codebook)
- CFB(Cipher Feedback)
- OFB(Output Feedback)
- CTR(Counter)

The CBC mode is well-defined and well-understood for symmetric ciphers and they are accepted as the most secure way to implement AES algorithm. The encryption modes that we consider is for preventing an eavesdropper from reading the traffic. They do not provide any authentication, so an attacker can still change the message-sometimes in any way she/he wants.

4.2.1 Cipher Block Chaining(CBC)

The cipher block chaining (CBC) mode is one of the most widely used block cipher modes for bulk encryption and the problems of other methods are avoided by "randomizing" the plaintext with XORing each plaintext block. In CBC mode encryption, a plaintext is divided into same sized blocks. Then each block Pi is XORed with the previous

ciphertext block Ci−1 before being input to the block cipher to produce the corresponding ciphertext block. Thus encryption and decryption are given by;

$$Ci=eK(Pi\oplus Ci-1) \text{ and } Pi=dK(Ci)\oplus Ci-1$$

respectively, for $1 \leq i \leq m$. Equal plaintext blocks will typically encrypt to different ciphertext blocks, significantly reducing the information available to an attacker.

One question mark still left with the question of which value to use for C0(Initial Cipher). This value is called the Initialization Vector, or IV also in CBC mode the length of input into CBC mode has to be a multiple of the block cipher size. Padding bits have to be applied to input data that does not satisfy this requirement before the data can be divided into blocks and encrypted.

The major advantage of CBC mode over other modes lies in its ability to hide statistical properties of the plaintext blocks. As a consequence of CBC mode's chaining structure, each ciphertext block Ci depends on all plaintext bits from P1 to Pi and the IV (or C0). Encryption of two identical plaintexts using the same key and IV produces identical ciphertexts. This is undesirable especially in applications where the data to be protected is chosen from a small set of predefined messages. Fortunately this risk can be easily mitigated by choosing unique IVs for each encryption, which ensures different ciphertexts are always produced even when encrypting the same plaintext with the same key.

CBC mode encryption is shown in the following graph.

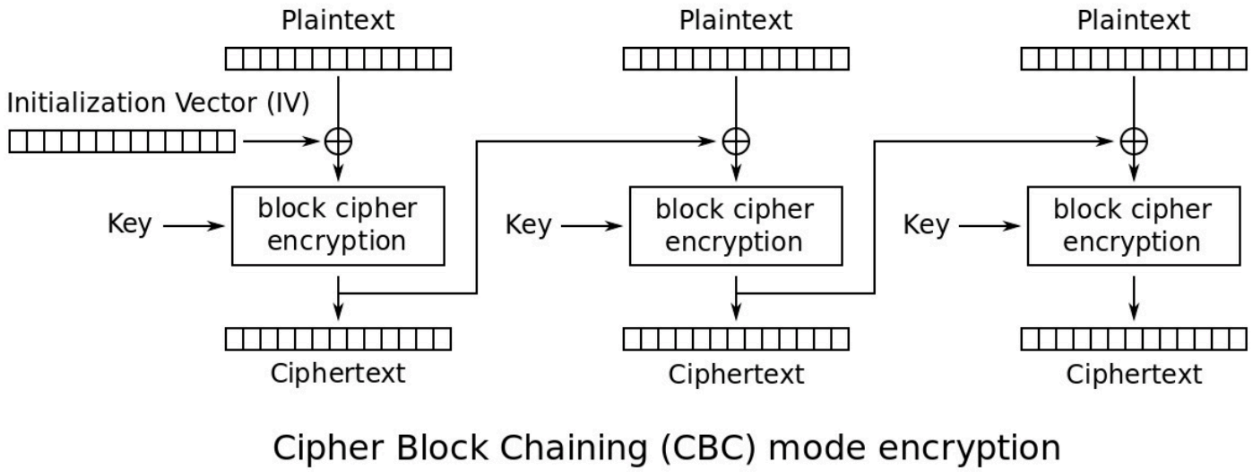


Figure 3.

4.2.2 Initialization Vector

The IV is used at the beginning of an encryption process and is needed by the decrypting party to recover the first plaintext block. Using two different IVs, IV and IV ′, say, encrypting a plaintext P produces two completely unrelated, random looking ciphertexts C and C′. This is due to the difference in input (IV ⊕ P1 versus IV ′ ⊕ P1) into the encryption function that produces two unpredictably different ciphertext blocks C1 and

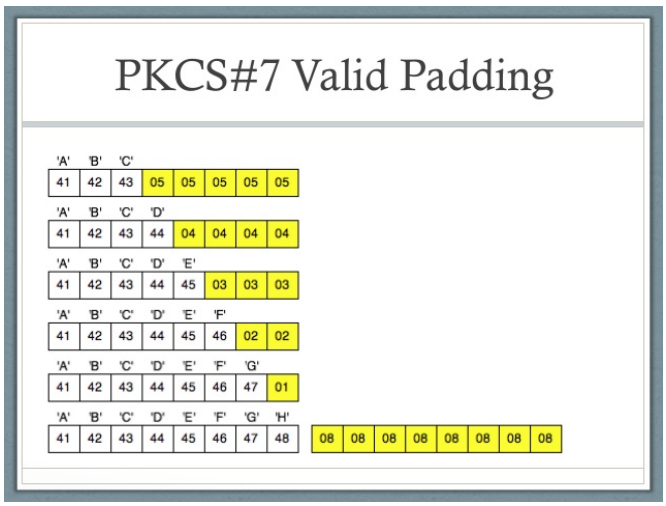
C1′ . This causes the input into the next encryption, i.e. C1 ⊕ P2 versus C1′ ⊕ P2, to differ randomly, and produces two randomised blocks C2 and C2′ . A similar argument applies to the next block and so on, and the random differences propagate along the chaining structure all the way to blocks Cm and Cm′ . This is a desirable property as an attacker cannot discern identical plaintexts with a ciphertext-only attack, as long as IVs are unique per encryption.Using a pseudo-random per- mutation to model the encryption algorithm, the authors concluded that it is sufficient for the IV to be random to provide adequate security (in a formally defined way), provided the underlying block cipher was strong enough.Further (though not part of [42]), the IV can be manipulated by an adversary to make predictable modifications to the first recovered plaintext block without detection (see Section 2.3.2.3 below). Using a secret IV (by encryption, indexing to a list, or synchronous- generation) is one way to prevent this attack.

4.2.3 Padding

In general, a block cipher mode is a way to encrypt a plaintext(P) to a ciphertext(C), where the plaintext and cipher text are of an arbitrary length. Most modes require that the length of the plaintext P be an exact multiple of the block size. This requires some padding. There are many ways to pad the plaintext, but the most important rule is that the padding must be reversible. It must be possible to uniquely determine the original message from a padded message.In this project Standard Padding algorithm which is PKCS#7 Padding is used. PKCS#7 Padding schema is actually very simple. It follows the following rules:

- The number of bytes to be padded equals to "8 - numberOfBytes(clearText) mod 8". So 1 to 8 bytes will be padded to the clear text data depending on the length of the clear text data.
- All padded bytes have the same value - the number of bytes padded.

PKCS7 Padding schema can also be explained with the diagram below;



So, when program apply encryption with padding it will fill the block with the number that shows how many bits deficient to fill up the block. On the other hand, after the program decrypting the ciphertext, the padding also has to be removed. The code that removes the padding should also check that the padding was correctly applied. Each of the padding bytes has to be verified to ensure it has the correct value.

So far it is shown that this cryptographic algorithm must use AES/CBC mode with using completely random IV value in order to achieve data confidentiality, also PKCS#7 padding must be applied to CBC mode, since this algorithm only works with 16 bits of blocks. A strong encryption algorithm protects confidentiality of data, i.e. encrypted traffic can not feasibly be deciphered by an eavesdropper without the decryption key. However, AES encryption on it's own does not provide any integrity of the data (unless using GCM mode to provide Authenticated Encryption with Associated Data- AEAD) so it is recommended to use something like HMAC_SHA256. The HMAC should be applied after encryption (Encrypt-then-MAC) to provide protection to Padding Oracle Attacks. When applying the HMAC it should be across the Salt, IV and Ciphertext – a common mistake is only performing it on the Ciphertext, which means any changes to the IV which alter the Ciphertext won't be noticed. So, let's explain what is the real mean of message integrity and authenticity before proceeding any further.

4.2.4 Message Integrity and Authenticity

The Oxford English Dictionary defines the act of authenticating as to "certify the authorship of something". In the context of information security, the meaning of data authentication follows the dictionary definition above in a more precise way. The concepts of genuineness and authorship of data translates to message integrity and data origin authenticity, defined as:

Message integrity: The assurance that the message has not been modified in transit.

Data authentication: The guarantee that the message is sent by a particular person or entity.

In practice, data origin authentication implies message integrity. If a message is modified by an adversary in transit, then it can be equivalently considered to have originated from the adversary. Conversely, common integrity protection mechanisms involve generating a bit pattern that is dependent on the message and a secret key known to the sender and recipient only. Verification of the correctness of the computed value proves knowledge of the key on the part of the entity who generated that value, and hence the origin of the message. Note that encryption alone does not generally provide authentication guarantees.

Generally it is accepted that encryption should be accompanied with integrity protection.One of the themes in this project is to provide users have a communication that can not be changed by the adversaries with integrity protection system alongside the

encryption, so message authentication code also will be added into the encryption mechanism.

4.2.5 Message Authentication Codes

A message authentication code, or MAC, is a construction that detects tampering with messages. Encryption prevents Eve from reading the messages but does not prevent her from manipulating the messages. This is where the MAC comes in. Like encryption, MACs use a secret key, K, known to both Alice and Bob but not to Eve. Alice sends not just the message m, but also a MAC value computed by a MAC function. Bob checks that the MAC value of the message received equals the MAC value received. If they do not match, he discards the message as unauthenticated. Eve cannot manipulate the message because without K she cannot find the correct MAC value to send with the manipulated message.

A MAC is a function that takes two arguments, a fixed-size key K and an arbitrarily sized message m, and produces a fixed-size MAC value. We'll write the MAC function as MAC(K, m). To authenticate a message, Alice sends not only the message m but also the MAC code MAC(K, m), also called the tag. Suppose Bob, also with key K, receives a message m' and a tag T. Bob uses the MAC verification algorithm to verify that T is a valid MAC under key K for message m'.

Given that the ideal MAC is a random mapping with keys and messages as input and that we already have hash functions that try to behave like random mappings with messages as input, it is an obvious idea to use a hash function to build a MAC. This is called as HMAC. The designers of HMAC were of course aware of the problems with hash functions, for this reason, they did not define HMAC to be something simple like MAC(K, m) as h(K II m), h(m II K), or even h(K II m il K), which can create problems if you use one of the standard iterative hash functions. Instead, HMAC computes h(K EB a II h(K EB b II m», where a and b are specified constants. The message itself is only hashed once, and the output is hashed again with the key.There are several hash functions which are MD5,SHA-1, SHA-2, SHA-256 etc. and since SHA-256 provides enough amount of security it will be used in HMAC algorithm for providing secure message authentication mechanism.

5. GENERAL STRUCTURE OF ENCRYPTION AND DECRYPTION

An iteration count has traditionally served the purpose of increasing the cost of producing keys from a password, thereby also increasing the difficulty of attack.

A salt in password-based cryptography has traditionally served the purpose of producing a large set of keys corresponding to a given password, among which one is selected at random according to the salt. So, salt value will be concatenate to password and they will be hashed together.This will prevent the attackers operate brute force attacks

against program since every time same password used resulting hash value always will be different unless same salt value used.
In this project our purpose is to provide a program that user can embed a picture with using his/her own password with using AES/CBC algorithm for encrypting the message and then LSB technique to embed the encrypted text. In the Encrypting phase user must choose an image to enter the message and then password should be entered in order to derive the encryption and authentication keys.
Keys will be generated according to PKCS #5 also known as Password-Based Cryptography Specification's key derivation standard.
A key derivation function produces a derived key from a base key and other parameters. In a password-based key derivation function, the base key is a password and the other parameters are a salt value and an iteration count. Two functions are specified in as a key derivation function but PBKDF2 is recommended for new applications. A typical application of the key derivation functions defined here might include the following steps;

1. Select a salt S and an iteration count c,
2. Select a length in octets for the derived key
3. Apply the key derivation function to the password, the salt, the iteration count and the key length to produce a derived key
4. Output the derived key.

An iteration count has traditionally served the purpose of increasing the cost of producing keys from a password, thereby also increasing the difficulty of attack.

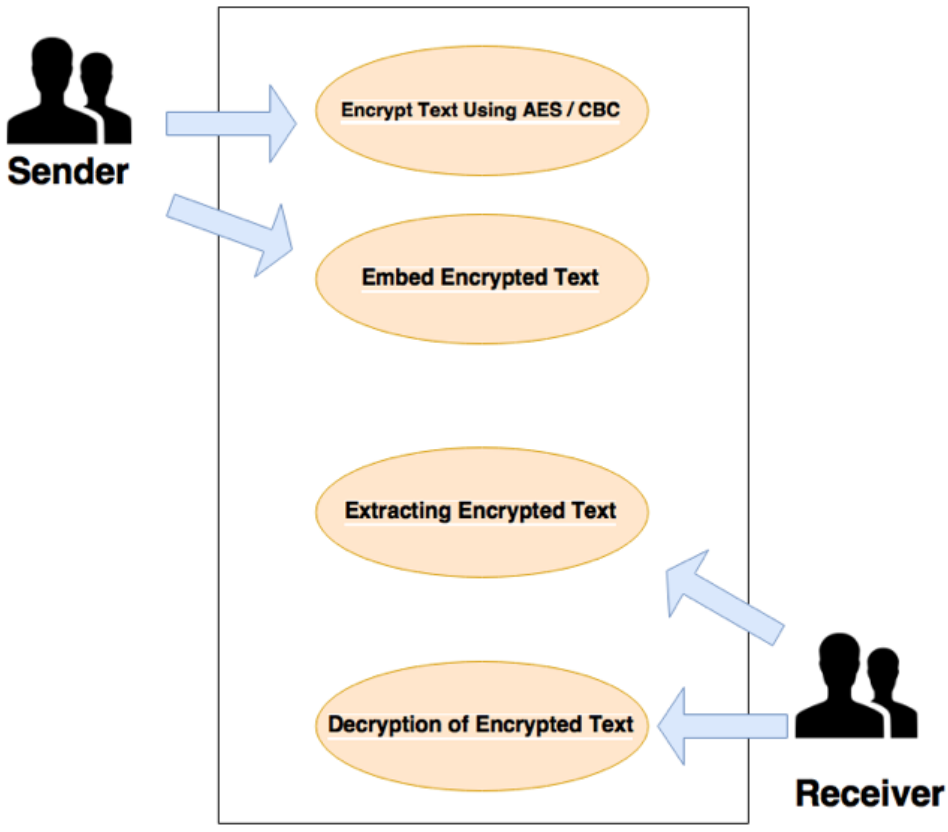
A salt in password-based cryptography has traditionally served the purpose of producing a large set of keys corresponding to a given password, among which one is selected at random according to the salt. So, salt value will be concatenate to password and they will be hashed together.This will prevent the attackers operate brute force attacks against program since every time same password used resulting hash value always will be different unless same salt value used.

As a result of what has been told till this time, following elements will be used throughout the program encryption.

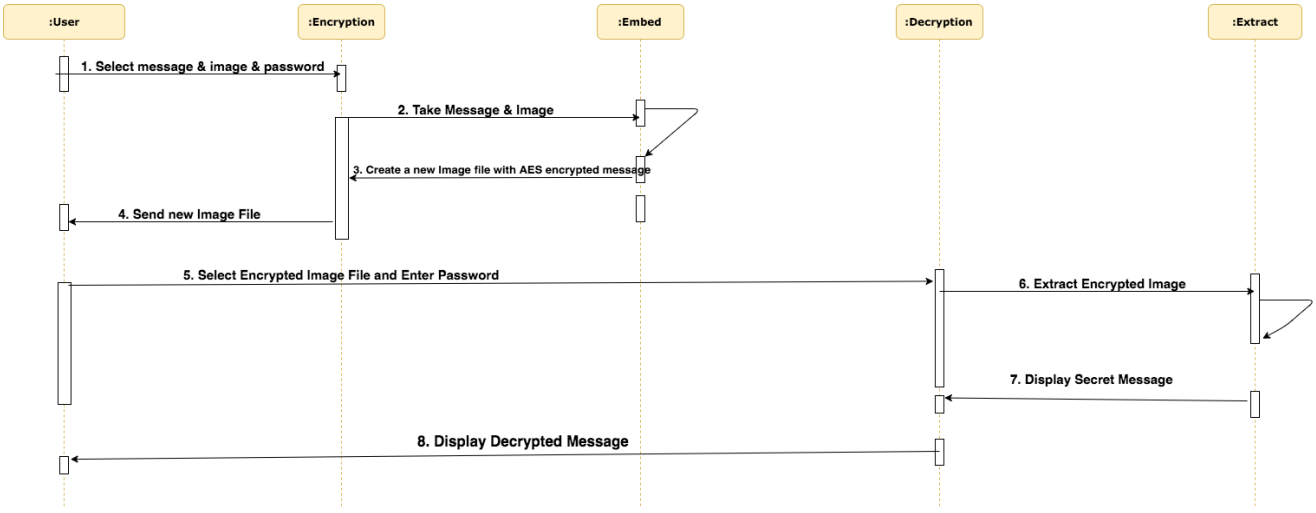
Generated CipherText = AES/CBC/PKCS5Padding
Our Keygen generator = PBKDF2WithHmacSHA256
Iterations = 65352
Salt Length= 128 bits
AuthenticationKey_Length = 128 bits
Initialization Vector(IV)=128 bits

6. ANALYSIS & DESIGN

6.1 Use Case Diagram



Steganography Use Case Diagram



Sequence Diagram

7. IMPLEMENTATION

- **Using java.awt.image**

That package provides classes for creating and modifying images. Images are processed using a streaming framework that involves an image producer, optional image filters, and an image consumer. This framework makes it possible to progressively render an image while it is being fetched and generated. [21]

- **Using ImageIO Class**

That class containing static convenience methods for locating ImageReader and ImageWriter, and performing encoding and decoding. [22]

- **Using java.awt.Image.BufferedImage Class**

That BufferedImage Class extends the Image class to allow the application to operate directly with image data. [21] In that project **getRGB** is used for getting an integer value of a pixel before embedding process and **setRGB** method for setting a pixel value while embedding.

- **Using java.util.concurrent.atomic Class**

The java.util.concurrent.atomic package defines classes that support atomic operations on single variables. All classes have get and set methods that work like reads and writes on volatile variables. That is, a set has a happens-before relationship with any subsequent get on the same variable. The atomic compareAndSet method also has these memory consistency features, as do the simple atomic arithmetic methods that apply to integer atomic variables. In that project, Atomic Integer travels through the image.

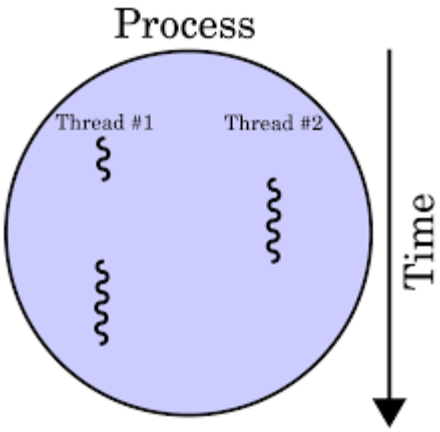
- **Using javax.swing Class**

Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms. In this project, we use javax.swing class for graphical user interface design.

7.1 Embedding Secret Text into an Image

7.1.1 Serial Embedding

- In the object of BufferedImage, using ImageIO.read method in order to take original image from user and using Scanner class to take text which user would like to hide.
- Text file is taken as input and it separated in stream of bytes.(i.e Ankara -> A-n-k-a-r-a)
- We used flag for reducing memory and that flag keeps last bit of every pixel. In our project we used LSB technique for embedding last digit of every pixel. Using getRGB and setRGB methods in order to embed encrypted text into user specified image. Now each bit of these bytes is encoded in LSB of each next pixel using OR (|) & AND (&) operators. If the pixel's last digit is 1, it is used OR operation with 0x00000001 otherwise, if it is 0, using AND operation with 0xFFFFFFFE to store and after set last bit of original image. It provides us using less memory and better speed.



- After every bit embedding operation, bitwise right shift operation is used for getting the next digit from a byte.
- We simply insert an exclamation point immediately after the embedding of the hidden text to let us know until where to read in extraction process.
- After embedding process using ImageIO.write method for saving the image which contains the secret information to another image file at specified path given by user in PNG or JPEG format.

7.1.2 Parallel Embedding

- It is possible to perform more than one operation in a single program stream with the Thread class. In parallel embedding process, we used threads to be able to use the system more effectively and high performance.

- It will be possible to write applications that will work with multiple threads at the same time by developing a multithread application. However, these threads do not work synchronously. To run synchronized; we defined an array that holds all the bits of the text to prevent confusion. We created two indexes from this array index. These two indexes helped me to keep bits which are embedded in the text. Thus, we have prevented the confusion of the bits which are embedded.
- Implementation algorithm is similar to serial implementation except minor differences. We use run() method for multithreading algorithm and start() the Thread.

- After embedding process using Thread, write method for saving the image which contains the secret information to another image file at specified path given by user in PNG or JPEG format.

7.2 Extracting Secret Text from Image

7.2.1 Serial Extracting

- Using again ImageIO.read method to get stego image from user.
- It is used again getRGB method for taking every pixel's last digit.
- The data of secret message is taken into char array.
  That char array is gathered together using for loop and after that process completed, user can acquire secret message.

7.2.2 Parallel Extracting

- Implementation algorithm is similar to serial extracting implementation except minor differences. Same as the parallel embedding process we use run() method for multithreading algorithm and start() the Thread.
- To run synchronized with thread; we defined an array that holds all the bits of the text to prevent confusion. We created two indexes from this array index. These two indexes helped me to keep bits which are embedded in the text. Thus, we have prevented the confusion of the bits which are embedded.
- Data extraction lasts until see the '!' symbol. This symbol indicates where to stop.

8. EXPERIMENTAL RESULTS

In order to test the scalability of the application we used 3 different image sizes. Every algorithm was repeated 10 times for all 3 images and the average of all the results was recorded. From the implementation of the tests of the single thread approach algorithms on the single-core and multiple-core platforms we recorded the results shown below.

Test Case 1:

This is the smallest test image used in our experiments, with 1600 pixels in width and 1200 in height (517KB). When using steganography algorithm, this image can hide up to 240,000 bits of data.

IMAGE	PERFORMANCE TIME (ms)			
	EMBEDDING		EXTRACTING	
	Serial Embedding	Parallel Embedding	Serial Extracting	Parallel Extracting
test1.jpg	1700 (average)	961 (average)	243 (average)	234 (average)

Table 2

Figure 3-a is the image before data hiding and Figure 3-b is the image after data hiding.

Figure 3-a

Figure 3-b

Test Case 2:

This is the median test image used in our experiments, with 3200 pixels in width and 1800 in height(1323KB).

When using our algorithm, this image can hide up to 720,000 bits of data.

IMAGE	PERFORMANCE TIME (ms)			
	EMBEDDING		EXTRACTING	
	Serial Embedding	Parallel Embedding	Serial Extracting	Parallel Extracting
test2.jpg	3502 (average)	2925 (average)	424 (average)	419 (average)

Table 3

Figure 4-a is the image before data hiding and Figure 4-b is the image after data hiding.

Figure 4-a

Figure 4-b

Test Case 3:
This is the largest test image used in our experiments, with 4252 pixels in width and 2835 in height(3317KB). When using our algorithm, this image can hide up to 1,506,801 bits of data.

IMAGE	PERFORMANCE TIME (ms)			
	EMBEDDING		EXTRACTING	
	Serial Embedding	Parallel Embedding	Serial Extracting	Parallel Extracting
test3.jpg	9979 (average)	9129 (average)	596 (average)	571 (average)

Table 4

Figure 5-a is the image before data hiding and Figure 5-b is the image after data hiding.

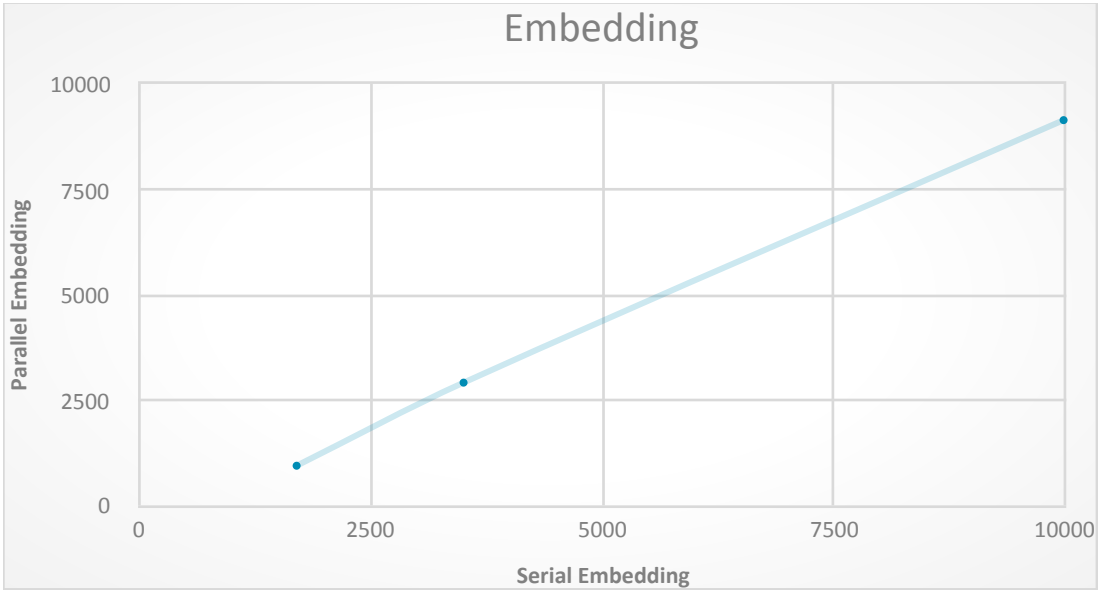
Figure 5-a

Figure 5-b

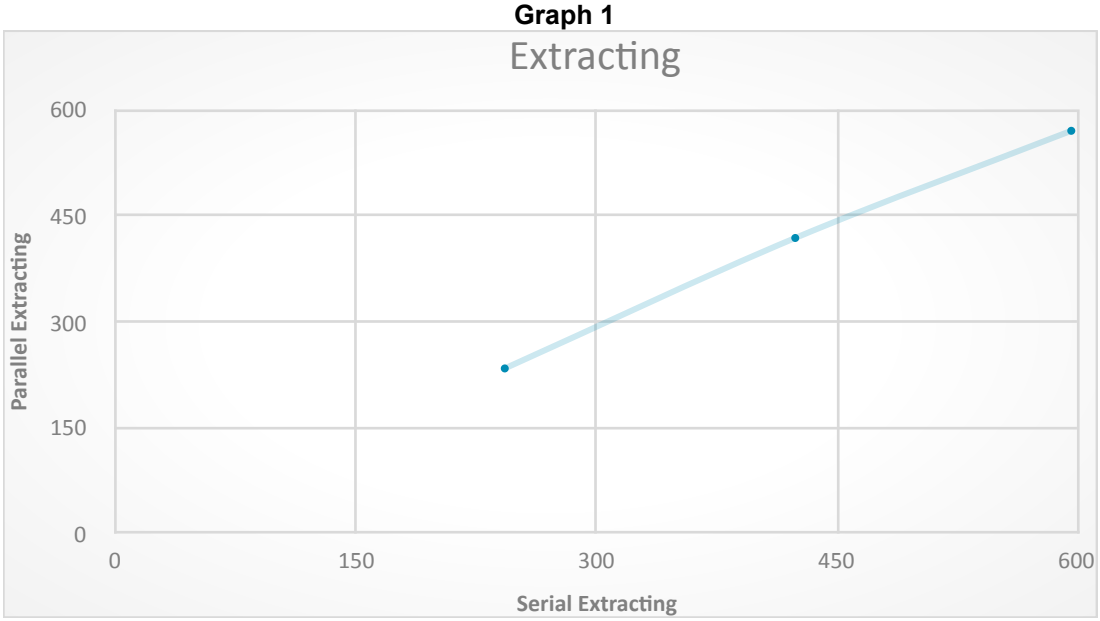
9. PERFORMANCE ANALYSIS

The run-time performance values on the 3 different test image of the program running in serial version and parallel version with 4 threads are depicted in the following graphs.

The serial and parallel operation of the embedding process is shown in Graphic 1. The x-axis represents the serial runtime and the y-axis represents the parallel runtime. There are 3 points which are represent the run-time values of the test pictures we mentioned above. The graph line is drawn according to these 3 values.The serial and parallel operation of the extracting process is shown in Graphic 2. Same as the embedding process the x-axis represents the serial runtime and the y-axis represents the parallel runtime. Also there are 3 points which are represenst the run-time values of the test pictures we mentioned above. The graph line is drawn according to these 3 values.



Graph 1



Graph 2

10. USER INTERFACE

We have created a user-friendly, easy-to-use user interface for this project with a minimalist and simple design. With these features, it is an interface that allows users to interact with users in the most effective way, offering a pleasant experience.

In user interface, After the login screen, the user has 2 options to choose between 'Encryption' and 'Decryption'. Following the selected process, there are steps such as choosing an image, entering a mask to hide, and so on. Our interface provides the user with the ability to see the original image and the data embedded image together. However, the user who performs the data extraction can easily see the extracted text. As a result of all these operations, the user can also switch between pages.

In Figure 6, a section of our interface has been enhanced.

Figure 6

11.CONCLUSION

Although only some of the main image steganographic techniques were discussed in this paper, one can see that there exists a large selection of approaches for hiding information in images. All the major image file formats have different methods of hiding messages, with different strong and weak points respectively. Where one technique lacks in payload capacity, the other lacks in robustness.

Least significant bit (LSB) in both BMP and PNG makes up for this, but both approaches result in suspicious files that increase the probability of detection when in the

presence of a warden. LSB provides biggest storage, speed and least distortion in the image among other techniques.

Even though, steganography provides us to carry messages in a secret way, we enhanced data's security, confidentiality and integrity combining with cryptography that is AES encryption method. Also, we used PBKDF2 standard in order to create our keys which is durable against brute force attacks, rainbow table attacks and also least dangerous attacks.

As a future work, we may use frequency domain for embedding text into an image. Also, for the cryptography side we may use longer key length. It can be used also SCIP protocol in order to enhance security.

12. REFERENCES

1) C. P. Sumathi, T. Santanam and G. Umamaheswari, "A Study of Various Steganographic Techniques Used for Information Hiding". International Journal of Computer Science & Engineering Survey (IJCSES) Vol.4, No.6, December 2013.

2) Navneet Kaur, Sunny Behal, "A Survey on various types of Steganography and Analysis of Hiding Techniques ".International Journal of Engineering Trends and Technology (IJETT) – Volume 11 Number 8 - May 2014.

3) W. Bender, W. Butera, D. Gruhl, R. Hwang, F.J. Paiz and S. Pogreb, Applications for data hiding, IBM Systems Journal, 39 (3&4)(2000) 547-568.

4) F.A.P. Petitcolas, "Introduction to information hiding", in: S. Katzenbeisser and F.A.P. Petitcolas, (ed.) (2000) Information hiding techniques for steganography and digital watermarking, Norwood: Artech House, INC.

5) A. Cheddad, J. Condell,K. Curran and P. Mc Kevitt, "Digital Image Steganography: Survey and Analyses of Current Methods". Signal Processing, Volume 90, Issue 3 ,March 2010, Pages 727-752.

6) D. Frith, "Steganography approaches, options, and implications", Network Security, 2007(8)(2007) 4-7.

7) Hitesh Singh, Pradeep Kumar Singh, Kriti Saroha "A Survey on Text Based Steganography",Proceedings of the 3rd National Conference; INDIACom-2009 Computing For Nation Development, February 26 – 27, 2009 Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi

8) Jain, Nitin, Sachin Mesh ram, and Shikha Dubey. "Image Steganography Using LSB and Edge–Detection Technique." , International Journal of Soft Computing and Engineering (IJSCE) ISSN (2012): 2231-2307

9) M. M. Amin, M. Salleh, S. Ibrahim, M.R. Katmin, M.Z.I. Shamsuddin, "Information Hiding using Steganography" Proceedings of 4th National Conference on Telecommunication Technology, Shah Alam , Malaysia, 2003.

10) Amin, Mohamed "Muhalim and Ibrahim, Subariah and Salleh, Mazleena and Katmin, Mohd rozi (2003) , Information hiding using steganography.

11) Ibrahim, Rosziati, and Teoh suk Kuan. "Steganography Algorithm to hide secret message inside an Image." arXiv preprint arXiv: 1112.2809 (2011).

12) Silman, J., “Steganography and Steganalysis: An Overview”, SANS Institute , 2001

13) Lee, Y.K. & Chen, L.H., “High capacity image steganographic model”, Visual Image Signal Processing , 147:03, June 2000

14) “Reference guide: Graphics Technical Options and Decisions”, http://www.devx.com/projectcool/Article/19997

15) Venkatraman, S., Abraham, A. & Paprzycki, M., “Significance of Steganography on Data Security”, Proceedings of the International Conference on Information Technology: Coding and Computing , 2004

16) Wang, H & Wang, S, “Cyber warfare: Steganography vs. Steganalysis”, Communications of the ACM , 47:10, October 2004

17)Morkel, Tayana, Jan HP Eloff, and Martin S. Olivier. "An overview of image steganography." ISSA. 2005.

18) Goel, Arun Rana, and Stuti Manpreet Kaur. "A Review of Comparison Techniques of Image Steganography. " Global Journal of Computer Science and Technology 13.4 (2013)

19) Sohag, Saeed Ahmed, Md Kabirul Islam, and Md Baharul Islam. "A Novel Approach for Image Steganography Using Dynamic Substitution and Secret key. "

20) Bhattacharyya, Souvik, and Gautam Sanyal. "A Robust Image Steganography using DWT Difference Modulation (DWTDM)." International Journal of Computer Network & Information Security 4.7 (2012) .

21) https://docs.oracle.com/javase/7/docs/api/java/awt/image/package-summary.html#package_description

22) https://docs.oracle.com/javase/7/docs/api/javax/imageio/ImageIO.html