Enhanced approach to Information Security using Steganography and Cryptographic Hash

A Dissertation Report

Submitted by

Sahil Mittu
(Registration No. 11302221)
(Roll No. RK2305A12)

to

Department of CSE/IT

In partial fulfilment of the requirements for the award of the Degree of

Master of Technology in Computer Science Engineering

Under the guidance of

Mr. Sami Anand
Assistant Professor,
School of Computer Science Engineering, LPU

May, 2015

Research Approval Form (PAC)

LOVELY PROFESSIONAL UNIVERSITY
Transforming Education, Transforming India

School of: Computer Science and Engineering

DISSERTATION TOPIC APPROVAL PERFORMA

Name of the student	: Sahil Mittu	Registration No	: 11302221
Batch	: 2013-2015	Roll No	: RK2305A12
Session	: 2014-2015	Parent Section	: K2305

Details of Supervisor:

Name	: Sami Anand	Designation	: Assistant Professor
UID	: 16840	Qualification	: M.Tech
		Research Exp.	: 2.5 years

Specialization Area: Databases (pick from list of provided specialization areas by DAA)

Proposed Topics:-

1. A technique to improve information security combining steganography and cryptography
2. Mobile Ad-hoc networks protocols.
3. Networks and Security.

Sami Anand
Signature of supervisor

PAC Remarks: Topic "1" is approved. Paper expected

M
13742
25/9/14

APPROVAL OF PAC CHAIRMAN Signature: 1/10/14 Date:

*Supervision should finally encircle one topic out of three proposed topics and put up for an approval before Project Approval Committee (PAC).
*Original copy of this format after PAC approval will be retained by the student and must be attached in the Project/Dissertation final report.
*One copy to be submitted to supervisor.

Abstract

Security of the information has been a prime concern since long. It has taken a major leap in present day scenario due to diversity of applications and users over the internet. Organisations and individual users/researchers are striving hard to figure out new techniques to ensure security of the data being transmitted over the networks.

Cryptography is one such class of techniques generally used in which the secret messages are modified (encrypted) before transmitting so that even if they are intercepted by intruders during transmission, they cannot be read in their original form.

Cryptographic Hashing is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string. It is a one way process, original data cannot be recovered from converted value or key.

Similarly *Steganography,* another class of technique adopted by people using which they can transmit secret messages securely by hiding them into other messages or some another media (termed as cover file). The intruders on the network may not know about the presence of some secret data. Hence it can be safely transmitted.

This study is aimed to achieve enhanced security by the use of steganography as well as cryptographic hashing together so that even if intruder may guess the presence of data and modified it, then on the destination receiver will know about the variation in data by verifying its hash value.

Certificate

This is to certify that Sahil Mittu has completed M.Tech dissertation titled "Enhanced approach to Information Security using Steganography and Cryptographic Hash" under my guidance and supervision. To the best of my knowledge, the present work is the result of his original investigation and study. No part of the dissertation proposal has ever been submitted for any other degree or diploma.

The dissertation is fit for the submission and the partial fulfilment of the conditions for the award of M.Tech Computer Science & Engineering.

Date: _____ Signature of Advisor

Name: Sami Anand

UID: 16840

Acknowledgement

Foremost, I would like to express my sincere gratitude to my advisor *Mr. Sami Anand (Assistant Professor, LPU)* for the continuous support in this study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in the time of research and writing of this dissertation.

Last but not least I would like to thank my friends and my loving family: my parents and my brother for their support while I worked on the study.

Declaration

I hereby declare that the dissertation entitled, "Enhanced approach to Information Security using Steganography and Cryptographic Hash" submitted for the M.Tech degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma.

Date: _____ Investigator

Sahil Mittu

Registration No.:11302221

Table of Contents

List of Figures

Chapter 1
INTRODUCTION

The research study title "Enhanced approach to information security using Steganography and Cryptographic Hash" focuses on the security aspect of the data communication between two parties via encapsulating the secret message with its corresponding cryptographic hash within another media like image file or audio file.

1.1 Theoretical Foundation

1.1.1 Introduction to Information Security

Now a days, large amount of information is transmitted on the network through some medium either on the wired or wireless network. The information travelled on the network may be crucial depending on the purpose of its use. So the security of this type of information from the intruders on the network is required. Security of information transmitted over the network is the major concern in these days. We cannot protect the entire network from the intruders, but we can secure the information that we want to transmit. The security of information can be achieved by the use of two technologies:

- Steganography
- Cryptography

1.1.2 Introduction to Steganography

The foundation of this research is a class of techniques for transmitting secret data units by hiding them in other data units. This technique is known as *steganography*. The file in which the secret data is to be hidden or embedded is known as *cover media* whereas the file generated as the result of integration of *secret data* unit into the cover media is known as *stego media*. The steganography terminology was coined by Pfitzmann. The media (cover/stego) can be text files, audio files, images, or any binary files. This implies that any one type of file can be hidden into any other type of file. For instance, a text file can be embedded into an image file or an audio file. An image file can be embedded into any binary file, etc. [20]

When such fusions happen, most of the default application software will discard the data portion that cannot be understood by them. E.g. Microsoft Windows Picture Viewer will discard any extra text bytes padded at the end of the stego-image binary data. When the stego-image will be opened using some non-default application (like notepad) the secret

message would be readable somewhere. In the latter case, the image would not be displayed. [16]

Figure 1: ***The concept of steganography [16]***

Use of Steganography ***Error! Reference source not found.***

- Transmit information securely
- Protection of Property
- Watermarking of Digital Media (Proof of Ownership)

Misuse of steganography ***Error! Reference source not found.***

- Criminal communication
- Hacking
- Electronic Payment
- Viruses
- Fraud
- Gambling
- Hacking

1.1.3 Introduction to Cryptography

This research is not only based on the steganography, other technique known as cryptography is also a part of research. Cryptography provide more feature to provide security to the information transmitted over the network.

Cryptography is the concept of converting data from one form to another so that even if the intruder detects the data, its meaning could not be understood. Encryption, decryption, hashing, digital signature etc. are the mechanisms classified under cryptography techniques.

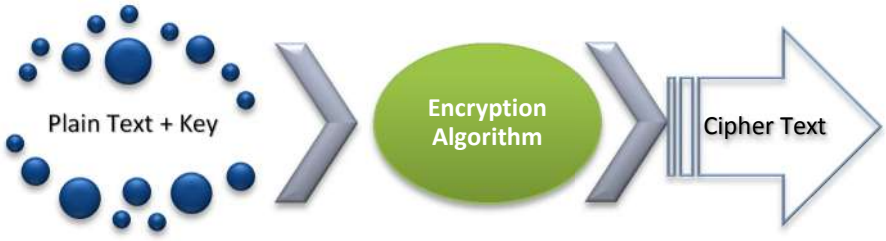


Figure 2: ***The concepts of Cryptography (Encryption)***

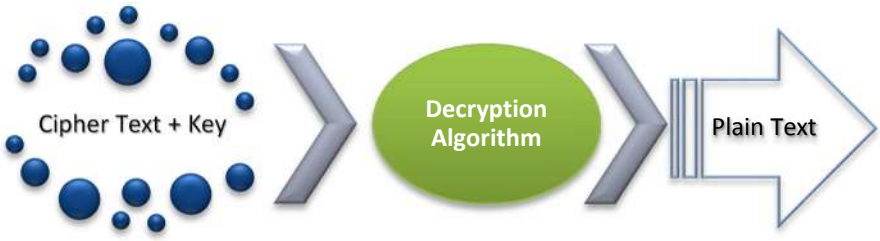


Figure 3: ***The concepts of Cryptography (Decryption)***

Cryptography focuses on keeping the contents of message secret. Cryptography provides the features *Confidentiality, Integrity and Availability (CIA)* triad. [26]

Confidentiality

Confidentiality ensures that only authorized person can access information/data or information system. Confidentiality can be achieved through User ID, Password, Access Control List (ACL) and Policy Based security implementation. Failed to achieve confidentiality leads to unauthorized access to data/information or information system. Confidentiality can be more illustrated by the following two concepts:

- Data Confidentiality assures that private or confidential information will not be disclosed to any user who is not authorized to access it.
- Privacy assures that individual control or influence what information related to them may be collected and stored and to whom that information may be disclosed. Only those people are allowed to access the private data of other users to whom it is disclosed.

Integrity

Integrity helps to guard the information from the unauthorized modification while transmission from source to destination or at rest. If any changes would be made to information then it ensures those would be done only by the authorized people. The loss of integrity means unauthorized changes or modification in information. Encryption and

hashing algorithms are the key techniques to ensure integrity. Integrity can be more illustrated by the following two concepts:

- Data Integrity assures that changes made to information would be in specified and authorized manner
- System Integrity assures that system performs its intended function in an unimpaired manner, free from deliberate and inadvertent unauthorized manipulation of the system.

Availability

Availability ensures that data and information system are available whenever it required to user who is authorized. The services or access only denied to unauthorized users.

The CIA can be achieved by the use of keys. Key in a cryptography is a combination of string of bits used to transfer plain text into cipher text using some encryption algorithm and to convert cipher text into plain text using corresponding decryption algorithm. Key can also be used to generate hash value for the any string input to hash algorithm. Use of key in cryptography provide more security to the information .There are two types of keys:

- *Public Key* is known to every one
- *Private Key* is known to only the sender/receiver who is the owner of private key

Cryptographic techniques like encryption, decryption, digital signature, etc. all are dependable on the keys and uses these two keys in *symmetric and asymmetric* fashion.

In *symmetric* key based approach the same key is used at both sender and receiver side whereas in *asymmetric* key based approach the one key (public/private) is used at sender side and other key is used at receiver side (private/public).

1.1.4 Hashing

Hashing is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string. A hash value *h* is the processed form of message *M* processed by hash function H (i.e. *h=H(M)*). The principle objective of hashing is data integrity. A small change in any bit of message M results in different value of hash code *h. [26]*

Hashing is a one way mechanism, in which obtaining original data from the hash value is not possible because of

- Its fixed length size (which may be smaller as compared to original data size) and

- The number of operations performed on it with the concept of compression.

Even if hash value may be processed for obtaining actual data it may not be true that the data obtained is same as of whose hash value is.

Some variations are always there in each hash value for different bits of message. Same hash value will not be generated for any two different messages but hash value is always same for same message (same output for same input) every time. [19]

These are the characteristics of the hash algorithm.

Figure 4: ***Block Diagram of Hashing***

Hashing not only provides the mechanism to achieve integrity, it also provide mechanism to ensure message authentication, generate pseudo random number, digital signature, etc.

Applications of Hashing

Hashing can be used in wide range of applications. Some of the leading applications are discussed below. [26]

Message Authentication is the mechanism or service used to verify the integrity of a message. It assures that data received are exactly same as sent (i.e. no any kind of modification in the existing content, insertion, deletion or replay). Message authentication is achieved when the hash function is used with some key, and that kind of hash function is known as *keyed hash function*. The hash value is known as *message digest* when it is generated by hash function for the purpose of message authentication.

Digital Signature is similar to message authentication, only difference is that the hash value is encrypted with user's private key. Anyone who knows the user's public key will able to verify the integrity of the message that is associated with the digital signature.

One-Way Password, hash function can also be used for generating one way passwords. Using this approach the password of user are not directly stored on the database, instead of actual password only the hash value of that password is stored. By this the actual password

users is also secured from the intruders because actual password is not retrievable by the intruders. So when user provides password on the interface then its hash value is generated and compared with the corresponding hash value stored on the server for that user.

Intrusion detection or ***virus detection*** can also be performed by the use of hash functions. For this user needs to generate hash value for each file on the system and also secure the hash value on some external source (once writable medium like CD, etc). Later determine the hash values and match with each other. If any variation between the values found then it means some wrong on the system either due to intruders or virus.

Cryptographic hash function can also be used to construct *pseudorandom function* ***(PRF)*** or *pseudorandom number generator* ***(PRNG)***.

Requirements and Security

The basic requirements of hash algorithm that each hash algorithm must fulfil for its success are *variable length input, fixed length output, collision resistant and preimage.* The following table defines the requirement and gives corresponding description for the requirement.

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that H(y) = h.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find y ! x with H(y) = H(x).
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).
Pseudorandomness	Output of H meets standard tests for pseudorandomness

Figure 5: ***Requirements for a Cryptographic Hash Function H*** **[26]**

This research implements a new hash algorithm in which input of 256 bits is provided to hash function and it will generate 128 bits of hash value for that input.

1.2 Evolution

1.2.1 Steganography

In ancient time the steganography technique (only on cover media) was relied on physical resources like human skin, game, etc. whereas in current time it is relies on the digital media like image file, text file, audio file, video files, etc. The general working principle of steganography remained same. **Error! Reference source not found.**

John and Jajodia (1996), stated that the word steganography is derived from the Greek words *stegos* meaning *covered* and *graphia* meaning *writing*, thus means *covered writing*. Steganography has been used in various forms for thousands of years. [13]

- Looking back to 500 BC, Histaiacus shaved a slave's head and tattooed a secret message on his scalp. After some months when hairs on scalp of slave grew sufficiently to conceal tattooed message then salve was sent to Miletus to convey secret message to Histaiacus's son-in-law Aristagoras.
- In Greece, another method used for conveying secret message was to peel the wax off a wax-covered tablet, then write a secret message and re-wax tablet. The recipient would simply need to rid of the wax from tablet to get secret message.
- The modern digital era of steganography is credited to Kurak and McHugh who proposed the method similar to embedding secret message to LSBs of the data bytes of cover media (Image file). They examined the image downgrading and contamination techniques now known as image based steganography. [15]

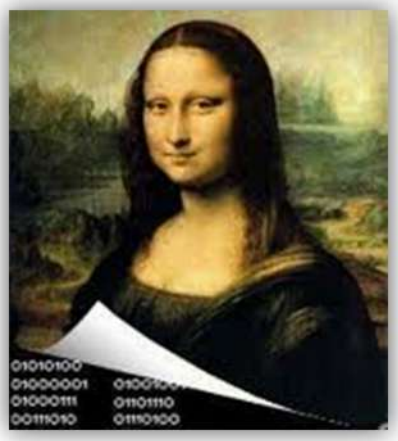


Figure 6: ***Famous Painting of Mona Lisa used for Steganography in History***

1.2.2 Cryptography

The word cryptography was derived from the greek work *kryptos* which means secret and *graphos* which means writing, resulting this means secret writing. Primarily cryptography use the codes and ciphers to protect the secret data. *Cipher* is coded text used on the behalf of some other text. Cryptography is the process of converting original text into its equivalent cipher text using some algorithms. Major mechanism used to perform cryptography is encryption and decryption. Encryption is the process of converting original text into cipher text whereas decryption is the process of reforming original text from the cipher text.

The Roman ruler Julius Caesar (100 BC – 44 BC) evolved and used a very simple method for secret communication. What actually happen he needs to send a highly confidential military instructions to his generals through messenger but he was not able to trust on his messenger. He was afraid from the fact that his message may be leaked to his opponents and can be used against him. So he decided to do some fusion so that messenger if handover the secret information to opponents then it would be useless for them. He did shifting of the character of alphabets and forms a mechanism of encryption known as *Caesar cipher*. This method was very efficient and fast at the time when it was derived because of non-availability of machine processing. The Caesar cipher is the mechanism of shifting alphabets of the word by 3 forward at sending side to generate cipher and 3 backward at receiving side to forms the original text from the cipher. This means alphabet 'A' would be translated as 'D', 'B would be 'E' and so on. So all the alphabets of secret message are shifted forward and backward on sender and receiver side. All the alphabets and their corresponding equivalent Caesar has been illustrated in below figure. [27]

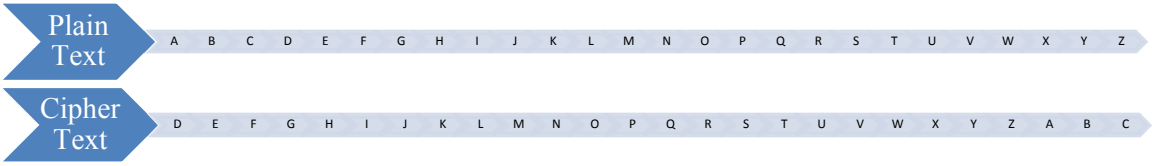


Figure 7: *Caesar Cipher* equivalent to Plain Text

1.2.3 Cryptography to Cryptographic Hash

The first cryptographic hash function designed date back to late 1970s. While presenting the paper on Public key cryptography by Diffie and Hellman, they identified the need of one way hash function as building block of digital signature scheme. The first work on hash

proposed by Rabin based on block cipher DES as a result of 64 bit Hash value. To improve the result of hash value proposed by Rabin (1970). Yuval showed an enhancement in the hashing mechanism to show how to find collisions for *n*-bit hash function in time $2^{(n/2)}$. Later Merkle's work introduced requirements of collision resistance and second preimage resistance and preimage resistance. In 1987, the definition of collision resistance is formalized by the Damgard and after two year Naor and Yung defined the variant of second preimage resistant functions called universal one-way hash function (UOWHF) also reffered as function offering eSEC. [21]

In 2004 Rogaway and Shrimpton formally studied the relationship between collision resistance and various variants of preimage resistance and second preimage resistance.

Hash function implementation mechanism goes through from various phases of design model. In the beginning the Block Cipher technique used for Hash functions. Later hash functions built using Arithmetic Primitives then a dedicated hash function are built like N-Hash, MD2, MD4, MD5, SHA1, etc. [21]

1.3 Image Steganography Techniques

Several techniques exist in modern steganography for embedding secret message into image files. Some of common names complied from several source of literature are as following: [2]

- LSB substitution and Palette based methods.
- Optimum Pixel Adjustment Procedure (OPAP)
- Inverted Pattern (IP) approach
- Pixel Value Differencing (PVC) method
- Discrete Cosine Transform (DCT) and other JPEG Steganography methods
- Patchwork and Spread Spectrum method
- Vector Quantization Method using codebook
- Fixed Range Equalization method

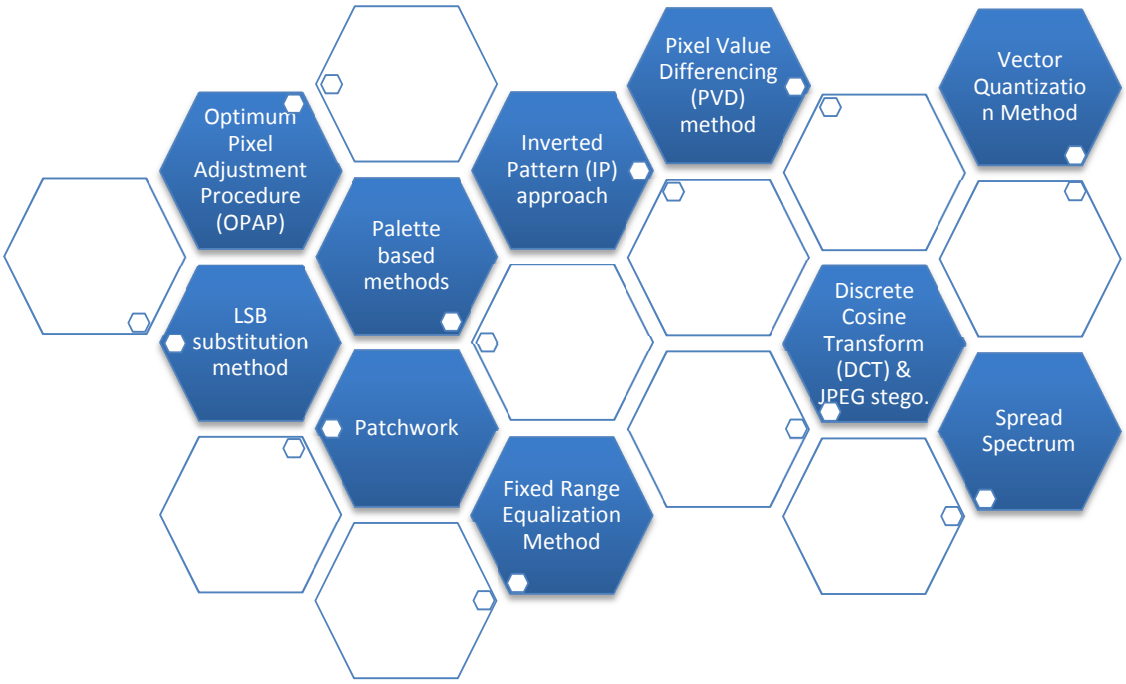


Figure 8: ***Image Steganography Method [16]***

Sara et.al. (2011) and Hamid et.al. (2012) classify these Image based steganography methods in following board categories:

- Direct Method
- Spatial or Image Domain Method
- Frequency or Transform Domain Method
- Adaptive Steganography Method

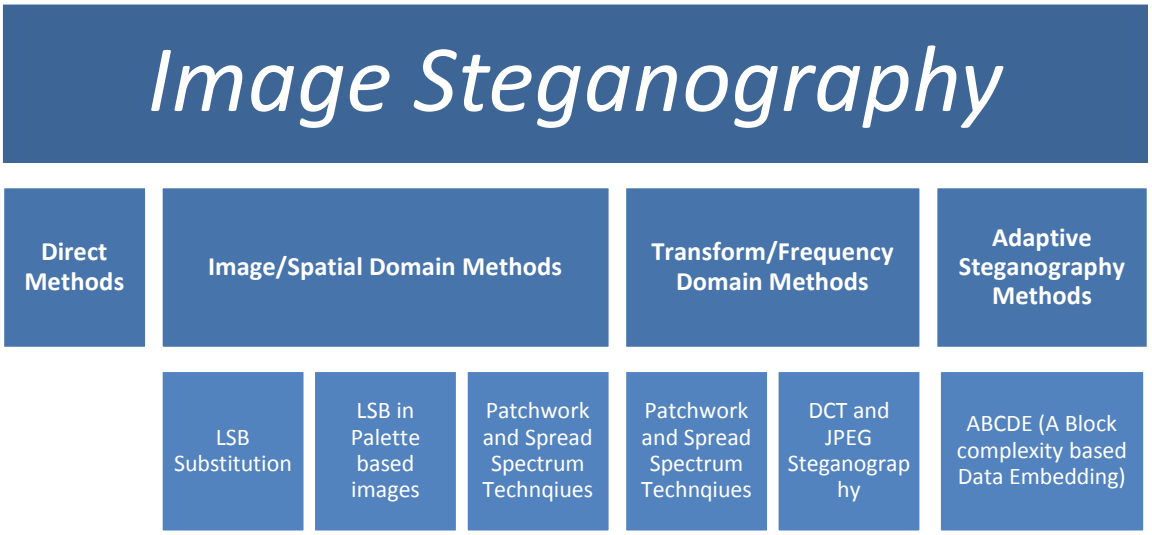


Figure 9: ***Categorization of Image Steganography Methods [16]***

1.3.1 Direct Methods

These methods include direct embedding of secret message into cover media without any manipulation over existing bytes of the cover media. The secret message is appended to the cover media as extra information. Cheddad et al. (2010) evaluates that neither the image histogram nor the visual preceptions can detect any difference between the cover image and stego image due to secret message being hidden after the end of file tag for these approaches. [16]

- Using simple *Text Editor*
    - Open the cover (image) file with any simple text editor like notepad.
    - Type the secret message into the file (at end of file).
    - Save the file (now this image would be termed as *stego* image)
    - When the image file would be open using standard image viewer application, the secret message would not be visible.
    - To read the secret message embedded into stego image, open it in some text editor like notepad, etc.
- Using *copy* command of DOS

  The same task can be performed using *copy* command of DOS
    - Syntax: X:\> COPY cover_image /B + message_file /B stego_image
    - Example: C:\> COPY cartoon.jpg /B + secret.txt /B cartoons.jpg
    - The above command will append the content of *secret.txt* file to cover image file *cartoon.jpg* and generate a stego image *cartoons.jpg*
    - The switch */B* indicates that the files are to be treated as binary files.
    - The operator + is used to concatenate the file *cartoon.jpg* and *secret.txt*

1.3.2 Spatial or Image Domain Method

In these methods, the modification of cover media for embedding secret message involves encoding at LSBs level. These methods uses bitwise techniques that apply bit insertion and noise manipulation. Message is directly embedded in the intensity of pixels and are most suitable for lossless image format like GIF and BMP. [16]

1.3.3 Frequency or Transform Domain Method

This method involves manipulation of algorithms and images transforms. The images are first transformed and then secret message is embedded into them. Most of these methods are independent of image format. Algorithms attempt to hide message in more significant

areas of the cover image and the embedded may survive between lossy and lossless compression. Lossy image format like JPG supports these methods. [16]

1.3.4 Adaptive Steganography Method

Adaptive steganography method is a special case of the spatial/image domain and frequency/transform domain methods. Adaptive steganography methods are "statistically-aware embedding" methods. Such methods consider the statistical global feature of the image prior working on LSB coefficients. The statistics define where to make changes in the image bits. One of these methods is characterized by a random adaptive section of pixels depending on cover image and the selection of pixels in a block with large standard deviation in order to avoid areas of uniform colour (or, smooth areas). This method is proven to be robust in terms of image compression, cropping and manipulation. [16]

1.4 Steganography and Cryptography

Steganography and cryptography are two different concepts. Both of them lead to security of information to be transmitted over the network. The secrecy aspect of the message addressed by steganography and cryptography is stated as under: [17]

- Using *Cryptography* the contents of the message are kept secret.
- Using *Steganography* the existence of the secret message is kept secret.

Both steganography and cryptography are the ways to protect information from unwanted parties but neither of the two technologies is perfect alone and can be compromised. Once the presence of hidden information is revealed or even suspected, the purpose of steganography is partly defeated. The strength of steganography can thus be amplified by combining it with cryptography.

The level of security can be enhanced further if the secret message to be embedded in some cover media is encrypted using algorithms like RSA, 3DES, etc. prior embedding. In such a scenario, even after the message presence is identified, the message cannot be directly read by the hacker unless it is decrypted.

Thus, it can be concluded that *steganography is NOT a substitute to cryptography and vice versa.* Rather, both the concepts have different objectives and the integration of both could lead to stronger and secure message communication system.

1.5 Audio Steganography Techniques

The set of techniques in which the cover media is an audio file (say, .wav, .mp3, etc.) are known as audio steganography techniques. Most of these techniques are also based on LSB substitution method as in case of image steganography techniques. Below we introduce the most widely used MP3 steganography algorithm named MP3Stego. [16]

1.5.1 The MP3Stego algorithm

This algorithm was developed *Fabien A. Petitcolas* in 2002 at Cambridge University for the purpose of hiding information in MP3 files. It was the first tool to hide information in MP3 files. MP3Stego uses the parity based embedding scheme to hide secret data in MP3 files during the compression process. The data is first compressed, encrypted and then hidden in the MP3 bit stream. The hiding process is executed during the Layer III encoding carried out using two nested loops. The inner loop is a quantization and encoding loop. The quantization step-size is increased to get a smaller quantization value if the bits produced by quantization and encoding exceed the available maximum length bits. This operation is repeated until Huffman coding bits are short enough. The bits are encoded as its parity by changing inner loop termination condition. The outer loop is noise control loop that controls the quantization noise according to threshold value and adjusts the scale factor.

The operation of MP3Stego can be simply understood by the following:

- For each block of music information to be encoded, MP3Stego decides whether or not or encode a hidden bit of information. Two factors decide whether to hide a bit: if there are more bits in the hidden message yet to be encoded and whether this particular block has been randomly chosen to be embedded with a bit.
- If it has been determined that no bit is to be embedded, no change is made from the standard encoding process, and the block is created as normal.
- If a bit is to be embedded, the block is first encoded as normal. After the initial encoding, the parity of the block is examined. Once the parity of the block is determined, one of two scenarios will occur:
    - The parity bit matches the bit to be encoded. In this case, since the parity bit is what we want it to be, the initial encoding is suitable, since it produces the

correct bit for the hidden message. No change is made and the initial encoded block is placed in theMP3 file.

- The parity bit doesn't match the bit to be hidden. In this case the block is unsuitable for carrying the hidden message bit because the parity is incorrect. The quantization factor that is used by the encoding application is changed, and the block is re-encoded in an attempt to change the parity bit of the block. This may happen several times before the parity of the block is correct. This also changes the length of the block.

- If the file cannot embed the file for some reason, MP3Stego fails. If there are no errors, once the hidden file is completely embedded, the remainder of the MP3 file is encoded normally.

The MP3Stego is implemented using two command line programs named *encode* and *decode* to hide secret message into cover file and extract secret message from stego file.

1.6 The ID3 Tag

ID3 is a metadata container prominently used with MP3 files. It allows storing information (like title, album, artist, etc.) about the file within the MP3 file itself. Although used globally in prominence, ID3 is just a *de facto* standard rather than *de jure* standard. No standardization body was involved in its creation nor has it been approved formally by any such organization. The ID3 tag is credited to Eric Kemp (1996) who had the idea of adding a small chunk of data to the audio file which soon became the *de facto* standard for storing metadata in MP3 files. The format was released by Damaged Cybernetics, an underground group that specialized in cracking console gaming systems.

ID3 has two unrelated versions: **ID3v1** and **ID3v2** with sub versions *ID3v1.1, ID3v2.2, ID3v2.3,* and *ID3v2.4. [16]*

1.6.1 The ID3v1

The ID3v1 tag was placed at the end of the MP3 file to maintain compatibility with older media players. It occupies *128 bytes* beginning with the string **TAG**. It allows 30 bytes for each of the fields: title, artist, album and a comment. Four bytes are used to store year value and one byte to represent one of the 80 predefined genres values. The list of genre values was later extended to 148 by Winamp. A revision to ID3v1 made in 1997 allowed the comment field to be trimmed by two bytes to store tack number. This revision was known

as ID3v1.1 as there was no major change in format of tag. Strings are either padded with spaces or zeroes to cover full length (say, 30 bytes for *album* name). [9]

TAG	*title*	*artist*	*album*	*year*	*comment*	*zero-byte*	*track*	*genre*
3 B	30 B	30 B	30 B	4 B	28 B	1 B	1 B	1 B

Figure 10: ***Format of ID3v1 Tag***

Extended ID3v1 tag is an extra adds an extra data block *before* ID3v1 tag that extends title, artist and album filed by additional 60 bytes each. It also offers a free text genre, one byte playing speed (values 0 through 5), the start time and stop time of music in MP3 file e.g. for fading in. If none of the fields are used, it is automatically omitted. If present the extended ID3v1 tag consumes another *227 bytes*. This tag begins with a string **TAG+** and any information in its field is continuation of information in the native tag fields. [16]

TAG+	*title*	*artist*	*album*	*speed*	*Genre*	*start time*	*stop time*
4 B	60 B	60 B	60 B	1 B	30 B	6 B	6 B

Figure 11: ***Format of Extended ID3v1 tag***

Following are the characteristics of several fields of ID3v1 tag:

Field	**Length**	**Description**
Header	3 bytes	***"TAG"***
title	30 bytes	30 characters for song title
artist	30 bytes	30 characters for artist name
album	30 bytes	30 characters for album name
year	4 bytes	4 characters for year number e.g. 2012
comment	28 bytes	The comment (30 bytes, if zero-byte and track not allowed)
zero-byte	1 byte	If a track number is stored, this byte contains value 0
track	1 byte	Track number on album (invalid if zero-byte is non-zero)
genre	1 byte	Index in list of genres or 255

Following are the characteristics of several fields of extended ID3v1 tag:

Field	Length	Description
Header	4 bytes	***"TAG+"***
title	60 bytes	Next 60 characters for song title
artist	30 bytes	Next 60 characters for artist name
album	30 bytes	Next 60 characters for album name
speed	1 bytes	play speed (**0**:unset, **1**:slow, **2**:medium, **3**:fast, **4**:hard-core)
genre	30 bytes	A free text field of 30 characters for the genre name
start time	6 byte	The start time of music as *mmm:ss*
stop time	6 byte	The end time of music as *mmm:ss*

1.6.2 The ID3v2

A totally new specification of ID3 tag known as ID3v2 was created by several contributors in 1998 that had no relationship with ID3v1. These tags are variable sized and usually occur at start of file. Metadata is contained in several frames inside ID3v2 tag. The length of each frame can be up to *16 MB* and the entire tag can be up to *256 MB*.

There are 84 types of frames and applications can also define their own new frame types. The standard frames are those that contain:

- Cover art image (album art), *which is the very target of this research*
- Beats per minute (BPM)
- Copyright
- License
- Lyrics
- Arbitrary text
- URL data, etc.

There are 3 sub versions of ID3v2

- **ID3v2.2** was the first public version of ID3v2. It used three character frame identifiers (like TT2 for frame containing song title) rather than 4 character identifiers like TIT2. This standard is now considered out-of-date.
- **ID3v2.3** expanded frame identifier to 4 characters (e.g. TIT2 for frame containing song title) and added several new frames. A frame could also contain multiple values separated with a / character. It is the most widely used version of ID3v2 tags.

- **ID3v2.4** was published in November 2000. It allows textual data to be encoded using UTF-8. Multiple values of a frame are separated by null character instead of a / character, so / can appear again in text data as value rather than a separator. It also allows adding tags at the end of file rather than in the beginning.

Surprising but true, Windows Explorer and Windows Media Player cannot handle ID3v2.4 tags in any version up to and including Windows 8 and Windows Media Player 12. Windows can understand ID3v2 up to and including ID3v2.3

The metadata frames can also contain images representing the following:

- 32x32 pixels 'file icon' (PNG only)
- Cover image (Front)
- Cover Image (Back)
- Media Label (e.g. label side of CD)
- Lead artist/performer
- Artist/performer
- Band/orchestra
- Composer
- Lyricist/text writer
- Movie/video screen capture
- Band/artist logo
- Publisher/studio logo

Several image file formats like JPEG, PNG and BMP etc. are supported.

Apart from the above image based frames, there are several text based frames like:

- COMM Comments
- EQUA Equalization
- MCID Music CD identifier
- PCNT Play Counter
- POPM Popularity Meter (Rating)
- RVAD Relative Volume Adjustment
- RVRB Reverb
- TALB Album name
- TCOM Composer name

- TEXT Lyricist or text writer
- TLEN Length (duration)
- TIT1 Content group description
- TIT2 Title or song name description
- TIT3 Subtitle or description refinement
- TRCK Track number or position in a set
- TDRC Recording/release Date and Time
- TYER Recording/release Year
- WOAF Official audio file webpage URL
- WOAR Official artist/performer webpage URL

The tags can be read and edited using a variety of software available commercially or free of cost over the internet. Such types of software are known as *taggers.*

1.7 The BMP File format

The BMP file format is one of the most common formats for image representation. It is a raster graphics format in which each pixel is represented by a triad of bytes with each byte representing the value for one of primary colours – Blue, Green and Red respectively. Thus, it requires 24-bits per pixel in memory (i.e. the BMP images have 24-bit colour depth). With 8-bits per colour component, the intensity levels of each primary colour can range from 0 through 255. With 24-bits per pixel, the format supports 2^{24} (i.e. 16777216) distinct colours per pixel in the bitmap image. [5]

These images are independent of the display device and a variant of bitmap images is also known as DIB (Device Independent Bitmap) image. Although, BMP images can be compressed, it is not a common practice. Thus, this study deals with the uncompressed BMP formats only.

1.7.1 The BMP Header

The official specification of the BMP file format requires the binary representation of BMP image be preceded by a 54 byte header immediately before the pixel data:

Field	**Offset**	**Length**	**Description/Value**
Magic Identifier	0	2 bytes	***"BM"***
Size	2	4 bytes	size of the bitmap file (in *bytes*)

Reserved1	6	2 bytes	reserved (must be **0**)
Reserverd2	8	2 bytes	reserved (must be **0**)
Offset	10	4 bytes	offset to start of image data (in *bytes*)
Header size	14	4 bytes	size of bitmap info header (must be **40**)
Width	18	4 bytes	width of the image (in *pixels*)
Height	22	4 bytes	height of the image (in *pixels*)
Planes	26	2 bytes	number of planes (must be **1**)
Bits	28	2 bytes	colour depth (*1* or *4* or *8* or **24**)
Compression	30	4 bytes	**0** (**no compression**), 1 (*8-bit RLE*), 2 (*4-bit RLE*), 3 (*RGB with mask*)
Image size	34	4 bytes	size of image data (in *bytes*)
Resolution-X	38	4 bytes	horizontal resolution per meter (in *no. of pixels*)
Resolution-Y	42	4 bytes	vertical resolution per meter (in *no. of pixels*)
Colours	46	4 bytes	no. of colours in image (generally **0**)
Important Colours	50	4 bytes	no. of important colours in image (generally **0**)

Another important aspect about the BMP files is that the pixel information in binary starts from the pixel at bottom left corner and ends at the pixel at top right corner. [4]

1.7.2 Examples of BMP headers of sample images

- *[For 28x24 image]*
  66 77 22 8 0 0 0 0 0 0 54 0 0 0 40 0 0 0 28 0 0 0 24 0 0 0 1 0 24 0 0 0 0 0 224 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- *[For 32x20 image]*
  66 77 182 7 0 0 0 0 0 0 54 0 0 0 40 0 0 0 32 0 0 0 20 0 0 0 1 0 24 0 0 0 0 0 128 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- *[For 64x48 Image]*
  66 77 54 36 0 0 0 0 0 0 54 0 0 0 40 0 0 0 64 0 0 0 48 0 0 0 1 0 24 0 0 0 0 0 0 36 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Chapter 2
REVIEW OF LITERATURE

Owen Harrison et. al. (2007). "Practical Symmetric Key Cryptography on Modern Graphics Hardware"

This paper presents an application oriented approach to block cipher processing on GPUs. A new block based conventional implementation of AES with 2-4x speed increase over the previous fastest AES GPUs implementation. The general purpose data structure is used for representing cryptographic client request, which is also suitable for execution on a GPU.

In this research paper the optimized AES algorithm is implemented in CTR (Counter Feedback Mode) on an Nvidia 8800GTX graphic processor. This is an ideal non-generalized implementation which can be used as source of comparison with generalized approaches. G80 architecture supports integer bitwise operation and 32 bit integer data types.

The research use the GPU as the dedicated processor for performing cryptographic functions. The different cryptographic function to be compiled into the single payload. One of the main criteria for data model in this context is to allow the buffering of as many messages as possible that require processing into single stream, permitting the GPU to reach its full performance potential. Exposing the payload structure to user rather than per message API allows the grouping of multiple messages. The payload structure contain separate pointers for data and keys as these are generally maintained separately. A level mode for which cryptographic service is required, is described within the payload descriptor or within the individual messages. The message descriptor is used to provide pointers to the arrays of messages, Initialisation Vector, Associated data, tags, etc. All of these elements uses the generic element descriptor which allow the description of any data unit within the data and key stream.

So the model is based on AES algorithm that must be used to store keys and reuse of keys across messages the *hashtable* cache is used. This is not for just to aid efficiency at the key schedule generation stage but also to generate smallest key schedule stream. This is important for on-chip GPU caching of the key schedule. Whenever the client application generates the key stream for payload inclusion, it's important for the same key to use the

same position within the stream. This allows fast optimisation of key schedule caching based on key offset rather than key compression.

The CTR mode of operation is used for symmetric key method. This determine how specified block cipher is used to implement a cryptographic system which support message greater than one block in length. Over GPU is suitable for bulk data encryption and can also be employed in a general manner while still maintain its performance in many circumstances for both parallel and serial modes of operation messages. [7]

Research Gap: So it can be concluded that further authenticated encryption modes of operation should be researched.

Joel Alwen et. al. (2009). "Public-Key Encryption in the Bounded-Retrieval Model"
They research on how to minimize the "key leakage" attacks for the security of keys used for encryption and decryption purpose. The model is constructed named Bounded-Retrieval Model (BRM) to provide security against these attacks.

The first public-key encryption scheme in the Bounded-Retrieval Model (BRM) is to provide security against various forms of adversarial "key leakage" attacks. In this model, the adversary is allowed to learn arbitrary information about the decryption key, subject only to the constraint that the overall amount of "leakage" is bounded by at most ℓ bits. The goal of the BRM is to design cryptographic schemes that can tolerate leakage bounds ℓ bits or more by increasing secret key size but keeping all the other parameters including the size of the public key, cipher text, encryption/decryption time and the number of secret-key bits accessed during decryption with small and independent of ℓ (no. of bits). [12]

As for main technical tool, they introduce the concept of an Identity-Based Hash Proof System (IB-HPS), which generalized the notion of hash proof systems of Cramer and Shoup to the identity-based setting. They had three different constructions for this based on:

- Bilinear Groups
- Lattices
- Quadratic Residuosity

As a result of this IB-HPS almost immediately produce an Identity-Based Encryption (IBE) scheme which is secure against partial leakage of target identity's decryption key. Similarly

IB-HPS also used to construct public-key encryption (and IBE) schemes in the Bounded-Retrieval Model.

Amritpal Singh et. al. (2013). "An Enhanced Run Length Coding for JPEG Image Compression"

Run Length coding is used for compressing the images, especially when images are compressed by block transformation. An Enhanced Run length coding method counts the number of repeated zeros which is represented as ***Run*** and appends the non-zero coefficient represented as ***level*** following the sequence of zeros.

In this paper, only the data item of zero from the string is encoded either it is single or repeating. The single zero between two non-zero coefficients is coded as (1, 0) where 1 is number of occurrence means length and 0 denoted to zero data item. This is an optimized approach of run length encoding. This technique will increase size of compressed data as compared to original when there is minimum repetition of data items in the data. [24]

To improve more in the term of compression ratio, if the occurrence of zero is single then instead of coded it in (1,0) it is simple coded as default as 0 and when it is repeating then it is coded in (n,0) where n is the number of continuous repetitions. This proposed method of Run Length is only used when continuous occurrence of zero is more than one.

Let us consider the example of both schemes (Optimized and Proposed Run Length Coding)

Data Items: {-33 21 -3 -2 -3 -4 -3 0 2 1 0 1 0 0 0 0 0)}

Optimized RLE: {-33 21 -3 -2 -3 -4 -3 (1, 0) 2 1 (1, 0) 1 (5, 0)}

Proposed RLE: {-33 21 -3 -2 -3 -4 -3 0 2 1 0 1 (5, 0)}

Research Gap: This scheme of RLE is that, this scheme is able to improve the compression ratio only if there are multiple continuous data items in the data set and reliability of RLE is still maintained.

Sumit Mittu, et. al. (2013). "Conveying Secret Messages through Album Art in MP3 Files"

This research paper is based on the steganography technique for hiding the data behind image (known as Album Art) that is also embedded in audio file. The purpose of this paper is to make information secure while transmission from source to destination. By this

method the intruder on the network will not identify that the audio file contain some secret or sensitive information in it.

In this Research, for the implementation of steganography on Album Art the ***LSB (Least Significant Bit)*** method was used. In this method the LSB of pixel intensity values of the album art or cover image is replaced by the secret data bits. [16]

As war as the quality of image is concerned, there was not any major difference in the quality of both images original album art and stego album art (contain secret information). This minor difference that may be found will not be observed by the human eye even by keeping both images in front. Taking the difference to level of pixel the maximum change would be 1/256*1/256*1/256 = 1/16777216 or nearly 0.00000006% which is impossible to differentiate by human eye. [1]

For Instance, when using an image with 24-bit pixel depth, one bit (the LSB) of each color component – Red, green and blue of the pixel can be replaced with one bit of secret data bit. This would result in strong 3 bits of secret data per pixel.

The Smart LSB pattern substitution algorithm proposed by Sumit Mittu et.al customizes/extend the traditional LSB algorithms to provide relatively stronger security maintaining all the positive of LSB substitution algorithms. These algorithms define three different patterns using which the bits of secret data replaced the LSB's of specific bytes of cover image.

Three different Smart LSB patterns proposed (and used with the Smart LBS pattern substitution algorithms are as under.

i. Diamond pattern
ii. SM pattern
iii. S-less pattern

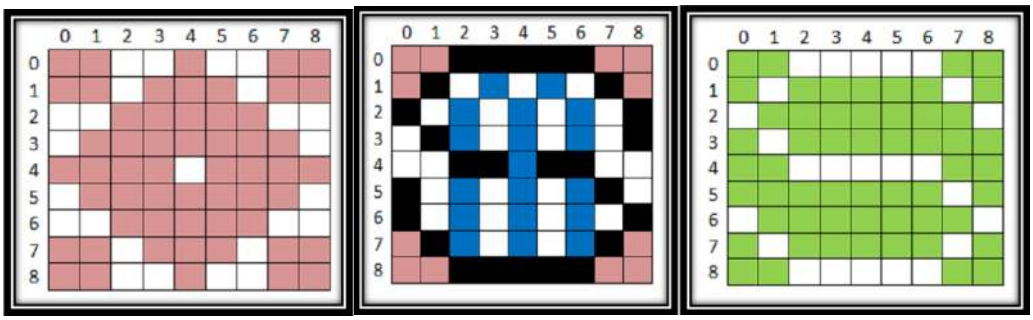


Figure 12: ***SMart LSB Patterns [16]***

The Smart LSB pattern substitution algorithm takes the LSBs of first 81 bytes of the image, arranges them into a grid of 9 rows x 9 columns and replaces the 56 of these 81 LSBs with

the first 56 bits of secret data. The same procedure is repeated for the next 81 LSB s of the cover image byte and next 56 bits of secret data. This is done until either of the two (cover image LSBs or secret data bits) exhausts.

The Algorithms are strong but more dynamism can be introduced in the algorithms to strengthen information security by introducing randomized bit distribution in its patterns, by performing cryptographic hash on secret data, in order to get more hiding capacity compression on secret data can be performed.

S.Joseph et. al. (2013). "A Novel Approach of Modified Run Length Encoding Scheme for High Speed Data Communication Application"

This paper presents a modified run length encoding scheme for high speed data compression to reduce the memory occupancy and to improve the performance of system.

This paper is based to improve compression rate. The basic idea behind this modified RLE scheme is to improve the compression rate so that the more system memory can be saved by the mean of compression. [14]

The modifications that are ade on RLE scheme in method is on the run length factor by replacing the few nearer bytes as well. In this. If the input data contain nearest value with its adjacent data (like 23 with 22 or 21 with 22) then both values are considered as same data item i.e. 22. Let us consider a compared example of RLE with Modified RLE

Input data {5 5 4 64 64 64 12 12 12 13 13}

RLE {5 2 4 1 64 3 12 3 13 2}

Modified RLE {5 3 64 3 12 5}

Research Gap: By this technique the data compression ratio can be improved but the integrity and reliability of data is compromised.

Samir Kumar B et. al. (2013). "Image Compression using Approximate Matching and Run Length"

This paper concentrate on lossless compression of image using approximate matching technique and run length encoding. The performance of this method is compared with the available JPEG compression technique over a wide number of images.

This paper is the combination of Run Length encoding scheme and replacement of matching pattern with the functional representation module called as edit operation. This operation is expressed as (p;char) where p describes replacing character at position p by character char. [3]

Let C denote “copy", and R denote “replace” then the following are two ways to convert the string “11010001011101010" to “11010001111001010" (0,1 are stored in ASCII)via different edit operation sequences:

C C C C C C C C R C C R C C C C C

1 1 0 1 0 0 0 1 0 1 1 1 0 1 0 1 0

1 1 0 1 0 0 0 1 1 1 1 0 0 1 0 1 0

A list of edit operations that transform a string u to another string v is called an Edit Transcription of the two strings. This will be represented by an edit operation sequence (u; v) that orderly lists the edit operations.

For example, the edit operation sequence of the edit transcription in the above example is

(\11010001011101010",\11010001111001010") = (9;1),(12,0);

Approximate matching method. In this case, the string \11010001011101010" can be encoded as f(17; 2) = (9; 1),(12,0), storing the ASCII characters require 136 bit or 17 byte whereas storing 4 characters will require 4 byte. Thus a compression of approximate 76.4% is achieved. This technique is very useful in image compression because of the inherent property of an image; two consecutive rows of an image has almost same string of pixel values. Only a few pixel varies. Experimental results prove this hypothesis.

The concept of edit operation is similar to run length encoding scheme where Edit operation is the replacement of pattern with some distinct bit as well.

Research Gap: This paper is concluded that this technique is only useful where the chances of occurrence of same pattern is maximum, so that the size of data can be reduced to maximum possibility.

Sukhdeep Singh et. al. (2014), “Public Cloud Storage using NAS”

This thesis work have some part of hashing, which is used for the username and password security. Password in the system is stored in the form of hash value rather than its original string so every time whenever a user logged in to his/her account using username and

password then his/her password would always be matched by its hash value corresponding to the value entered by the user in login screen and the value stored in the systems database. Anytime if user (may be an intruder) tries to retrieve password using "forget password" option, system would generate new password rather than displaying existing password. As it is already known that formulating the original string is not possible from the hash value. [25]

Research gap: This study work gives an idea to use hashing concept in proposed research for maintain integrity, accountability and confidentiality of secret data.

Chapter 3
PRESENT WORK

3.1 Rationale and Scope of Study

Since long ago, messages have been securely exchanged over networks via several of the techniques like encoding and cryptography.

A hacker is tempted to intercept and decipher these messages due to the fact that he/she can notice the presence of the message (although not in a form/format that can be directly understood). Alternatively, if the presence of the message is obscured and the hacker is not at all aware that some message is actually being transmitted over the network the probability of the hacker to intercept and decipher that message can be reduced down almost by 50 per cent.

SMart LSB tool for sending secret data over the network provide the feature to hide data from the intruders but still if some other person who aware (knowingly or unknowingly) about the existence of secret data then he/she may tries to intercept the data and make modification to the content of secret message embedded in stego media. Then receiver has no option to know about the changes had been made to the content of secret data. Keeping this thing in mind enhancement can be performed in the SMart LSB tool to achieve Integrity of secret data.

By performing Hash on secret data before embedding it to cover media and by send that hash value along with the secret message, the receiver would get to know about the alteration of secret data when the hash value of secret data would regenerated at recipient side with same hash function (if different hash value formed means data compromised)

3.2 Problem Formulation

This research study will be based on the Smart LSB substitution algorithms proposed by Sumit Mittu et.al. This research work would focus on providing enhanced security by integrating Steganography and Cryptographic hash function to ensure *confidentiality, accountability and integrity.*

A hash code would be generated for secret message to be transmitted over the network. The secret message along with this hash code would then be embedded into an image file using customized LSB based substitution steganography algorithms. The image file thus

generated would be further embedded in album art of an mp3 file (thereby providing yet another level of security by multilevel hiding of secret data) at the sender side. Similar process would be performed in reverse at the recipient side.

The above process ensures the following aspects of security:

- Confidentiality is achieved when the message is concealed within another file.
- Accountability is ensured through the organisation key used to generate hash values.
- Integrity is maintained when at recipient side the hash value is regenerated from the secret message to match with that generated at sender side.

3.3 Objectives of Problem

The main objectives of the study include:

1. To integrating cryptography and steganography techniques together to generate yet stronger information security technique.
2. To ensure integrity of and accountability on secret data transmission by use of hashing techniques.
3. Without increase in size of image the security parameter will be adjusted within the image.

3.4 Research Methodology

During the proposed study the following possible improvements as suggested by [Mittu 2013] in "Conveying Secret Message through Album Art in MP3 file" are shall be considered to further enhance Smart LBS substitution algorithms.

1. Key based approach for steganography to achieve integrity.
2. The sender cannot deny sending the message (accountability or non-repudiation).

To achieve integrity for the secret message, the concept of hashing is being introduced in the Smart LSB substitution algorithms. Prior to embedding secret data in cover image, it goes through the hash function and corresponding hash value of secret data would also be embedded into the cover media with secret data. A hash code will be sent along with the message in the same file to ensure the integrity of message which is regenerated on the recipient side from the received message with same hash function. If the hash value matched with received hash value, it means data is same as it is sent by the receiver. If any variation found in the hash value then it means data has been compromised while

transmission. In this research the new hash algorithm is defined which makes it more secure because no one else the maker of hash function will aware about the process or method of generating hash value.

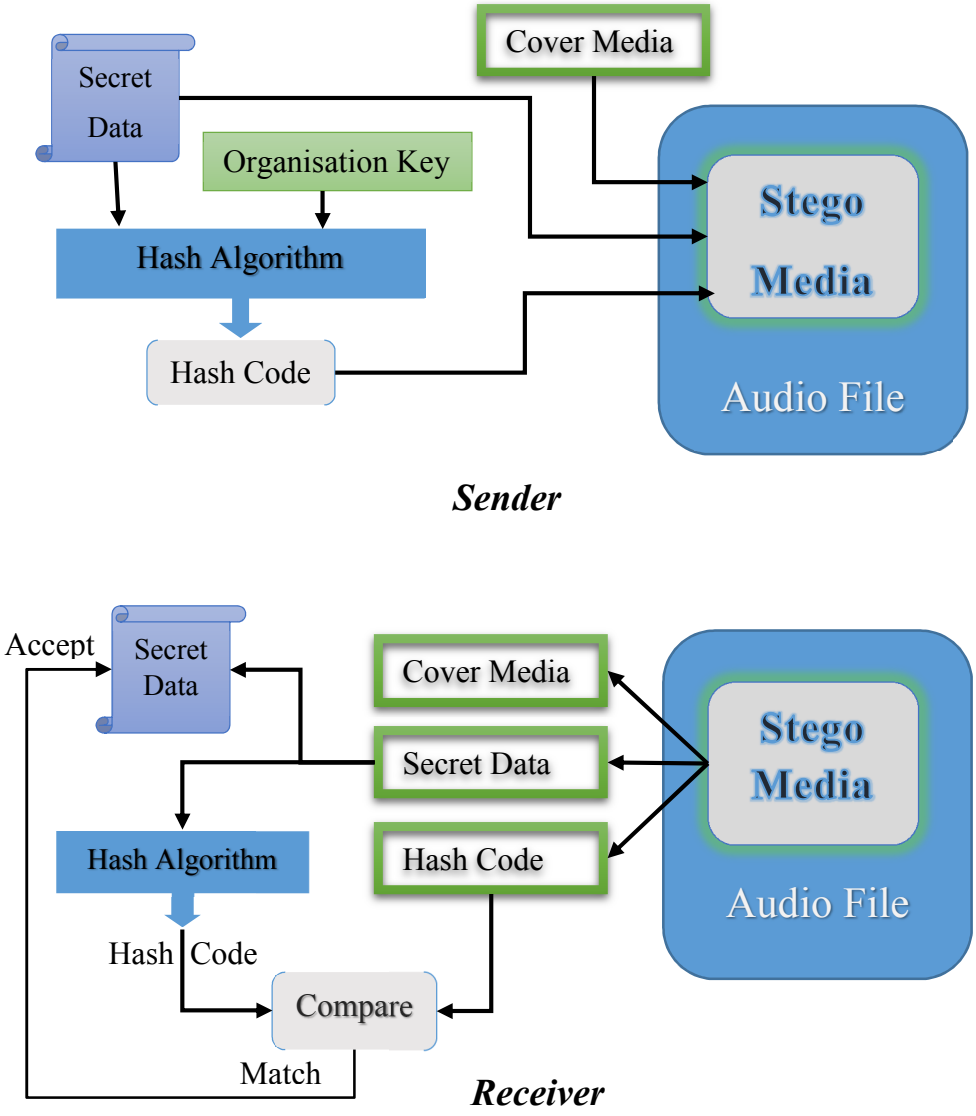


Figure 13: ***Block Diagram of Methodology***

3.3.1 Why new hash algorithm?

The purpose of creating new hash algorithm is to make hash value more secure from the intruders on the network. New hash algorithm will generate different values from other existing algorithms and the mechanism for generating hash value will also be different from those in existing hash algorithms. This will restrict intruder to change data and create new hash value for that changed data. The variation in hash value generated by existing algorithm and proposed algorithm will restrict them to do changes. New hash algorithm has been designed as per the specifications and requirements proposed in current study.

To achieve accountability in this research the hash algorithm being proposed in current study uses an organisation specific key. The organisation key is different for different organisations. It would be unique among all the keys for the hash algorithm and provided by the hash algorithm itself. Thus, when the key of particular sender is used for generating the hash value for any message, sender will not be able to deny sending the message (i.e. non-repudiation).

3.3.2 Proposed Hash Algorithm

The proposed hash algorithm accepts input of any length and it would always generate output (hash value) of fixed length (i.e. of 128 bits) of binary.

First of all the secret data is passed to the hash algorithm then it will convert all the secret data into its equivalent binary form. After conversion to binary the length of binary is verified that whether it is divisible by 256 or not. If the length is not divisible by 256 then additional bits would be padded at the end of binary so as to make it divisible by 256. When length is multiple of 256 then all binary is converted into its equivalent hexadecimal form with nibble (4bits of binary) represents one hexadecimal value and so on.

At this step you have data in the form of hexadecimal, now all the hexadecimal values are converted into its equivalent ASCII code and stores binary of that ASCII value. At this step again length of binary is verified (i.e. divisible by 256) if not then padding would be performed again.

Now the real process to generate Hash value will be initialized at this step. The another Hash function named HASH_OP will be called by passing 256 bits from the binary generated at previous step and this process will be repeated until all the binaries will not be processed by HASH_OP function. The value returned by HASH_OP function is stored in the variable HCODE and all the values that are returned by this function are XORed with each other.

The HASH_OP function will return the binary value of length 128 buts w.r.t 256 bits passed to it. HCODE variable also have length of 128 bits, XORed value of multiple 128 bits (if function called multiple times). Then this HCODE (final Hash Value) will be returned to the user which is embedded with the secret data in the cover media.

The function *HASH_OP()* receive the binary of length 256 bits from the function *HASH()* (i.e. input of 256 bits binary data) and split that 256 bits into two parts of L and R of length

128 bits. After this XOR operation is performed on L and R and the result produced after XOR operation is stored into the variable X.

Now IMPLIES operation will be performed between (L→X) and (R→X) and the result of both will be stored L' and R' respectively. After this Right circular shift of bits on combined data of L'R' will be performed

At this step the major input is given to the algorithm i.e. Developer Key (DK) and Organisation Key (OK) which are used in the algorithm to achieve authentication and accountability.

Here, a loop will be initialized to perform same operation multiple times to generate unique value for the input given to algorithm so that final hash value would result in different or unique every times when different input would be given to it. The number of rounds for the loop are set to 11. It was observed that 11 iterations lead to a relatively distinct hash value with optimal combination of computational overhead and security.

In loop body the M represent message will be the concatenate result of L' and R' variable. Now the developer key will be applied on L' and R' and the resultant value will be stored in the variable called L" and R". The DK developer key will be re generated by XORing its existing value with Organisation key. After processing DK XOR operation on L" and R" with L' and R' will be performed respectively and stored into M.

Now AND operation will be performed on the set of first 8 bits with the set of next 8 bits and so on up to set of last 16 (8 and 8) bits. Again L' and R' will be recalculated with the update value of L", R" and M' by XORing L" with M' and R" with M'.After this value of M will be updated with the values of L'and R' and next round of loop will be started (if current round not a last round).

After executing all the rounds of loop the final value of M' after performing IMPLIES operation on it with Organisation Key is returned to the calling agent

The entire process is represented with the help of flow chart in Figure 10 and Figure 11.

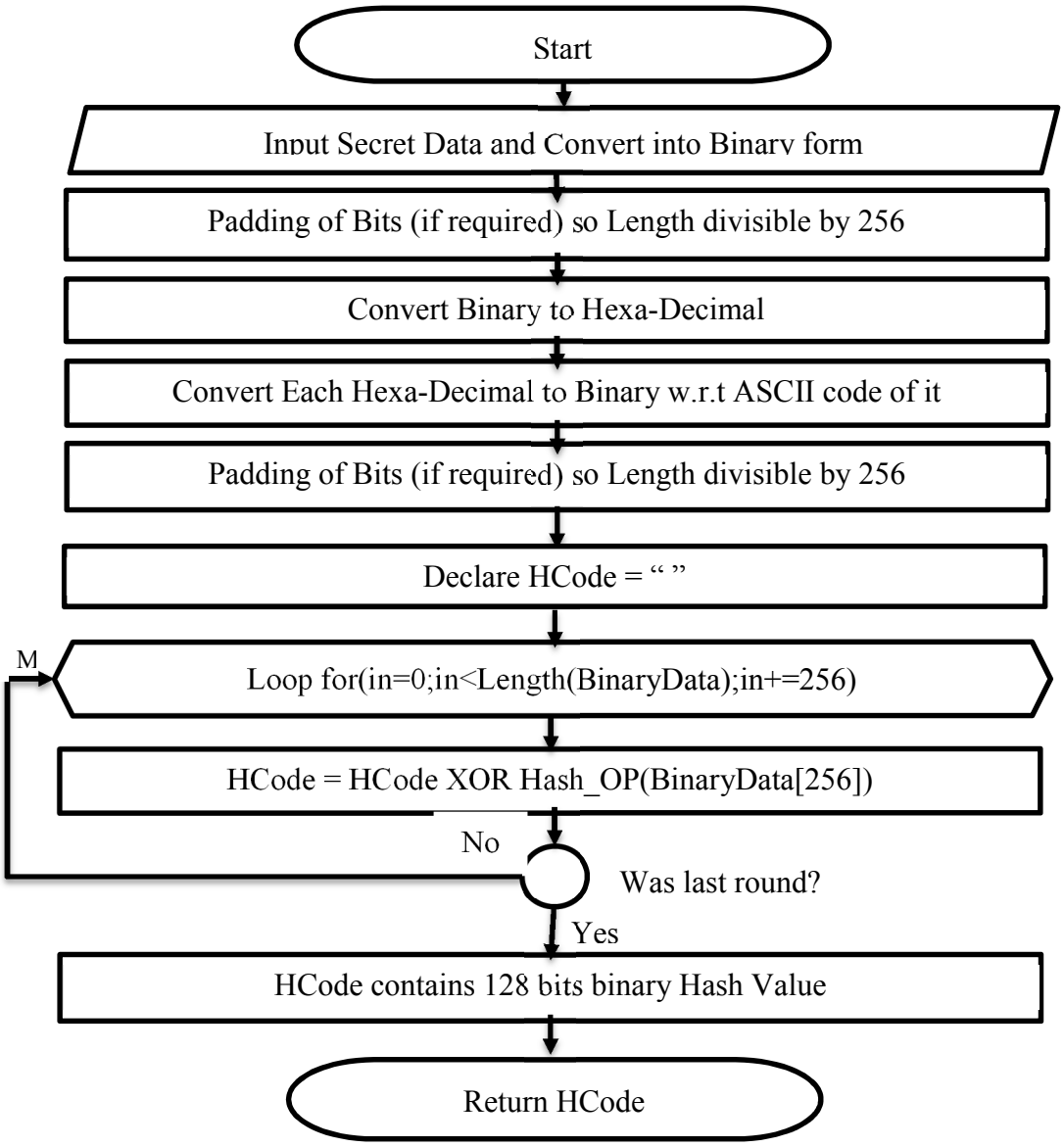


Figure 14: ***Hash Algorithm***

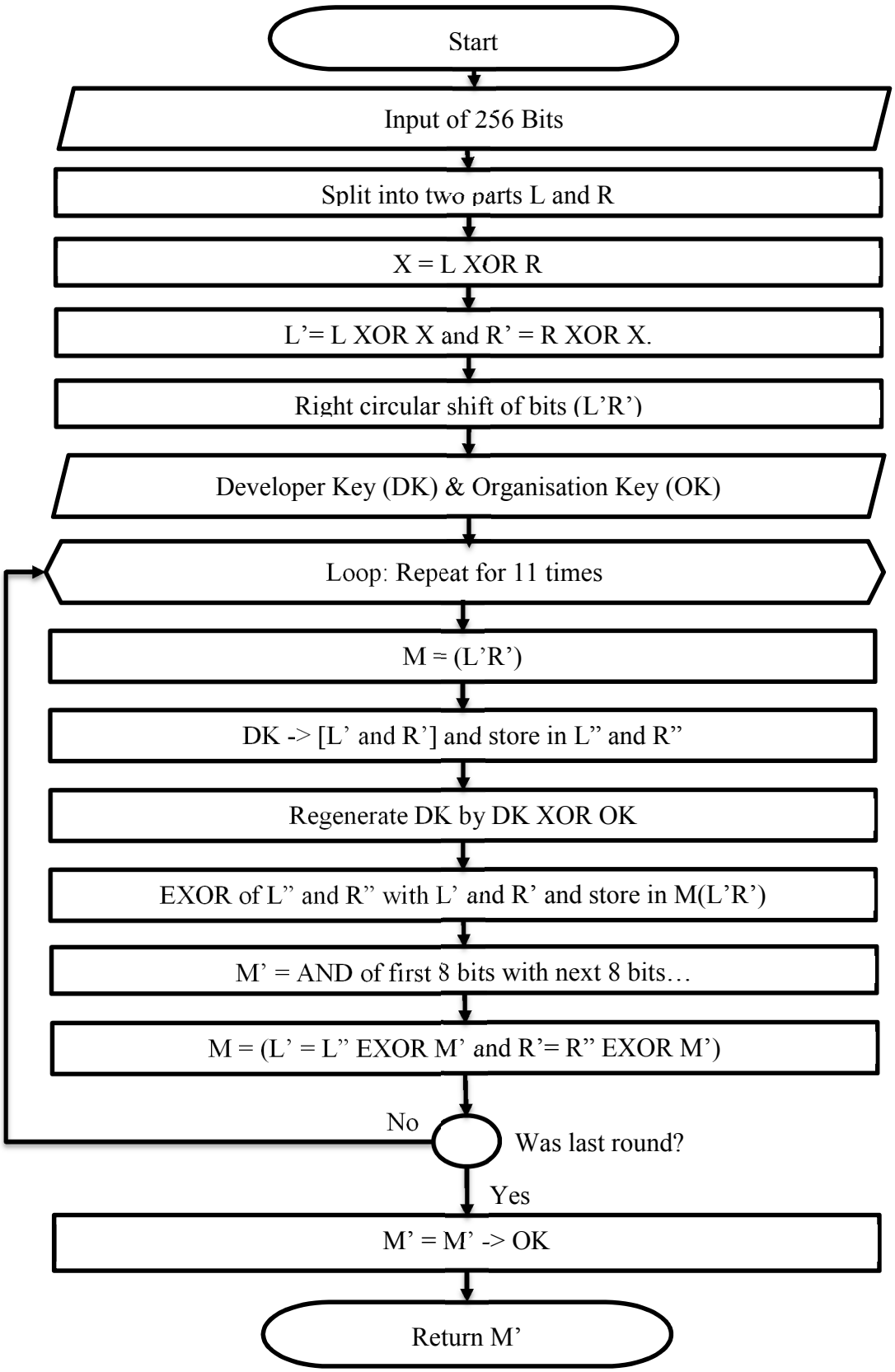


Figure 15: ***Hash_OP Algorithm***

Chapter 4
RESULTS and DISCUSSIONS

With the use of cryptography and steganography techniques together, a stronger security mechanism is built for the current research work. Using this mechanism the integrity of secret message sent within cover file is achieved. Formation of new hash algorithm makes it more secure because intruders on the network would not aware about the technique used for generating hash value of the secret message and the operations that are performed on the input to generate its hash value.

The particular *organisation key* may vary from organisation to organisation which helps to accomplish accountability for the message in this research. The organisation specific key can be generated or provided by the developer of algorithm and it will be unique among all the keys generated for this purpose.

The outcome of this research established its way starting with the development of new hash algorithm for generating hash value of secret message. Secret message is conveyed using existing SMart LSB substitution method. The tool iSMart LSB created to embed secret message with its hash value. This tool gives an option to send secret message with or without its hash value (depends upon sender choice) and hash value will also be embedded into the cover media (which is BMP format picture of any size) with secret data.

4.1 The iSMart LSB tool

The iSMart LSB tool is developed so as to implements the proposed iSMart LSB substitution algorithms. The tool derives certain characteristic features from SMart LSB tool [16]. The iSMart LSB tool provides following features:

- Generate stego image that contains secret data along with its corresponding hash.
- Extract the secret data from the stego image and also report about its integrity.

The iSMart LSB tool provisions the sender to embed secret data into the cover image without the hash if required. The extraction module will accordingly not verify the integrity of the secret data being received through (and extracted from) stego image. In such case, the iSMart LSB tool imitates the Smart LSB tool.

The discussion below details the steps used to embed the secret data (with or without hash) in the cover image and extract it back (with or without integrity verification).

Step 1: Start iSMart LSB Tool. First screen of iSMart LSB tool gives four option:

- **Hide Secret Data**
  - Provide interface to hide secret data with/without hash
- **Extract Secret Data**
  - Provide interface to extract secret data perform integrity check
- **About iSMart LSB**
  - Display information about iSMart LSB tool
- **Exit**
  - Quit from iSMart LSB tool.

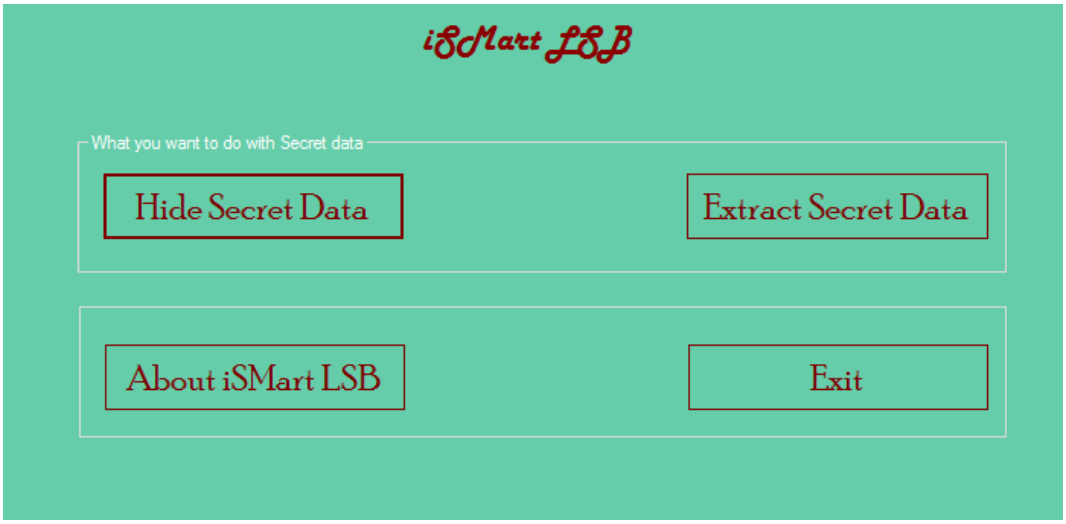


Figure 16: ***Main Screen of iSMart LSB Tool***

Step 2: Click on "Hide Secret Message" button to display the stego image generation interface.

Step 3: Browse cover image file in which you want to embed secret data. When an image will be loaded, iSMart LSB tool will display the capacity of image to embed secret data.

Step 4: Choose the text file in which secret data is kept. If file size is greater than hiding capacity of image then iSMart tool will generate alert message and would not embed secret data.

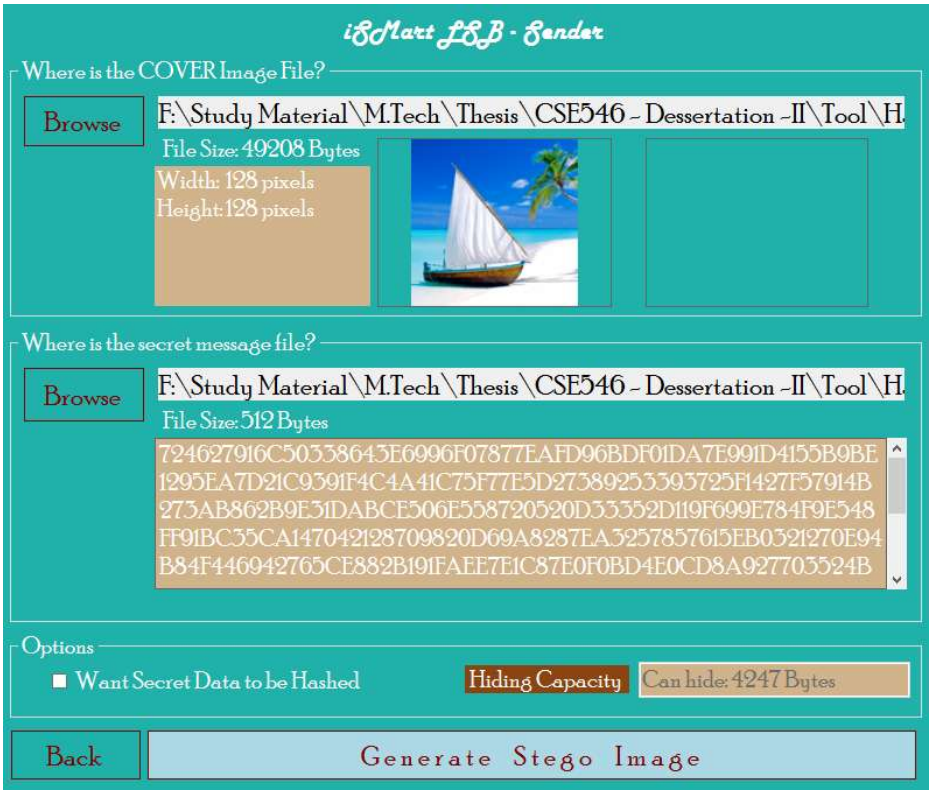


Figure 17: ***iMSart LSB - Sender window***

Step 5: Uncheck Hash option if not interested to check integrity of secret data on recipient side (by default it is enabled).

Step 6: Click on "Generate Stego Image" button. It will again verify the capacity of image to carry secret data and size of secret data (with hash value if chosen). If everything satisfied then it will embed secret data (with hash value if chosen) to cover media and thus stego media is generated.

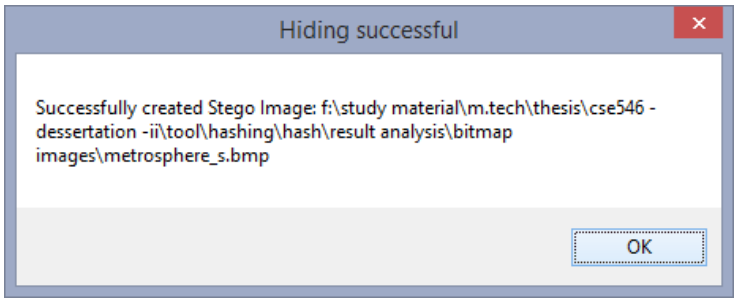


Figure 18: ***Alert generated after Stego Image created successfully***

Step 7: Now you can embed this stego media to any Audio file that supports ID3 tag feature. Most of media players provides option to add or delete cover media from audio files. Below is the example given for itunes.

- Select the song and right click on it. Choose option “Get Info” from popup menu and select Artwork tab as shown in below figure.
- Click on Add button and choose the stego image as cover media to embed into audio file and click OK button to apply settings.

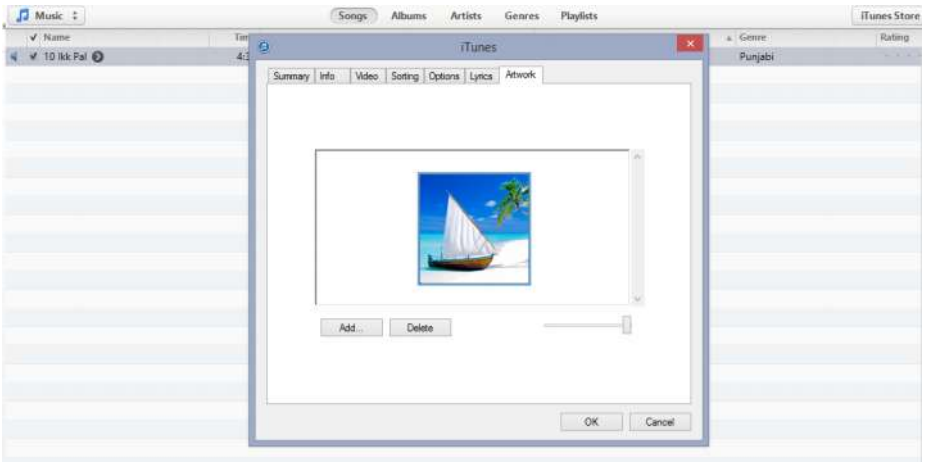


Figure 19: *Embedded Stego Image via iTunes*

Now test existence of cover media in other media players also.

Figure 20: *Preview of Stego Image in iTunes*

Figure 21: *Preview of Stego Image in Windows Media Player*

Figure 22: ***Preview of Stego Image on Android Device***

Step 8: After embedding of stego media in audio file you can transmit that audio file to its destination.

Step 9: On receiver side

- Extract stego media from audio file received using same tool or any other tool that support or provide ID3 tag editing feature
- Keep extracted stego image separate to provide it to iSMart LSB tool later for extracting secret data from it.
- Repeat the same steps that were performed while embedding stego image in audio file (if using same tool on both sender and receiver side) except the last step of click on Add button.
- Now instead of click on Add button you need to right click on cover media shown on interface. This will popup a menu with options cut, copy, paste and delete.
- Click on copy option and save this image using any image editor software like MS Paint, etc.

Figure 23: ***Option to copy Cover Media i iTunes***

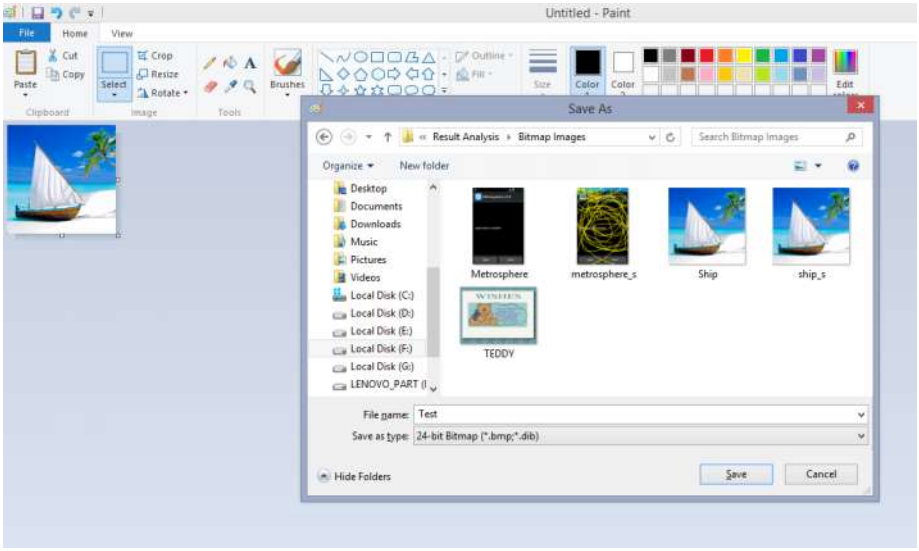


Figure 24: ***Saving Stego Media using MS Paint***

Step 10: Now open iSMart LSB tool on recipient side and choose "Extract secret data" option from its main window.

- Main screen of iSMart LSB tool remains same as in figure 16.

Step 11: New window "iSMart LSB – Receiver" will give you the interface to extract secret data from image (load stego image saved previously by clicking on Browse button)

Figure 25: *iSMart LSB - Receiver*

Step 12: Stego file will be loaded and few important information related to stego image will be displayed on interface.

Step 13: Click on "Extract Secret Message" button

- This will extract the secret message from stego image and display it on the interface.

Step 14: If integrity check was performed on secret data while embedding process then automatically iSMart LSB tool at this step performs the integrity check.

- The same hash function will be used for extraction and embedding process of secret data.
- Organization key to calculate hash value for the secret data would also remain same at both side. If different key would be used it leads to different hash value for same secret data.
- If hash value generated at recipient side matched with hash value received in stego media, means data is same as sent by the sender otherwise data is compromised.
- User would be intimated by message alert about match or mismatch of hash value.

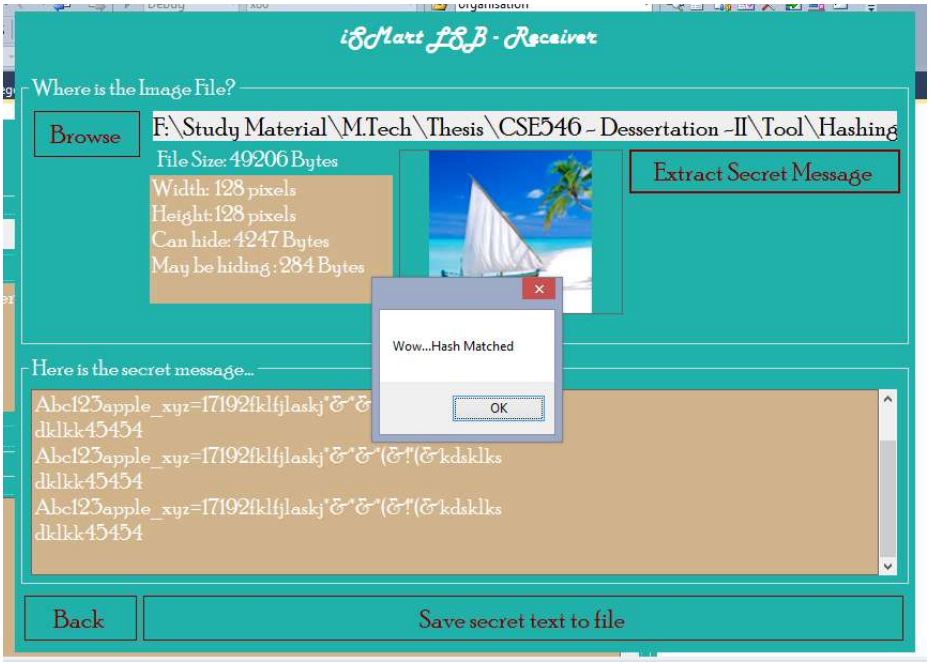


Figure 26: ***Hash Value matched***

Step 15: Interface gives the option to save extracted secret data in both scenario even if integrity check fails.

- To save secret data in text file format, user just need to click on "Save Secret text to file" button.
- The dialog box appears on screen and choose the destination folder and give name to secret data text file.

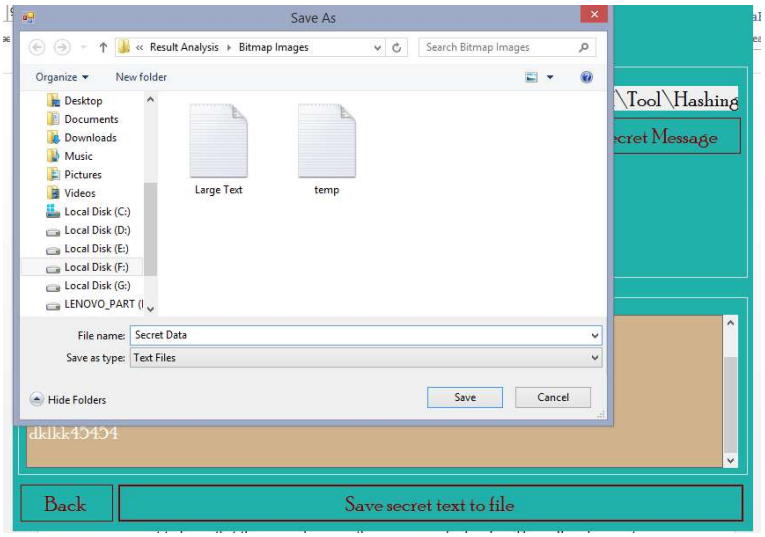


Figure 27: ***Saving Secret Data to new text file***

4.2 Interpretation of Results

Result analysis has been performed on various images and secret data items. Few of them has been discussed here which illustrate the quality and efficiency of the hash algorithm proposed in this research and how integrity of secret data helps users to know about the authenticity of the secret data.

Various test has been performed on the basis of following data set:

Images used:

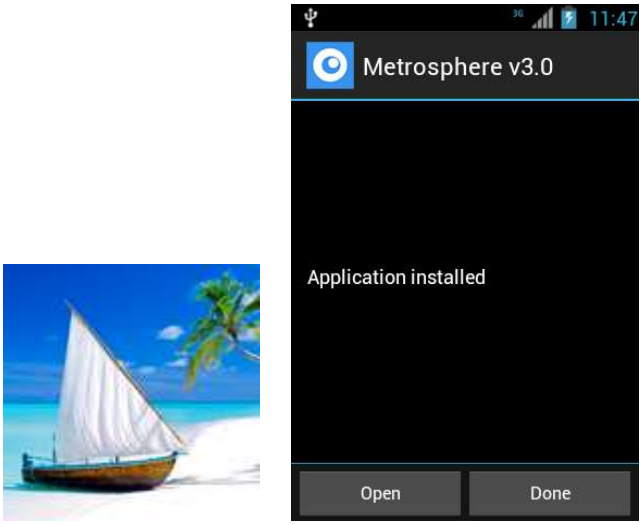


Figure 28: ***Test Images (Ship and Metrosphere)***

Secret Data:

Secret Data 1	Data 2
724627916C50338643E6996F07877EAFD96B DF01DA7E991D4155B9BE1295EA7D21C9391F 4C4A41C75F77E5D27389253393725F1427F57 914B273AB862B9E31DABCE506E558720520D 33352D119F699E784F9E548FF91BC35CA1470 42128709820D69A8287EA3257857615EB032 1270E94B84F446942765CE882B191FAEE7E1C 87E0F0BD4E0CD8A927703524B559B769CA4E CE1F6DBF313FDCF67C572EC4185C1A88E86E C11B6454B371980020F19633B6B95BD280E4F BCB0161E1A82470320CEC6ECFA25AC73D09F 1536F286D3F9DACAFB2CD1D0CE72D64D197F 5C7520B3CCB2FD74EB72664BA93853EF41EA BF52F015DD591500D018DD162815CC993595 B195	Abc123apple_xyz=17192fklfjlaskj*&*&*(&!* (&kdsklks dklkk45454 Abc123apple_xyz=17192fklfjlaskj*&*&*(&!* (&kdsklks dklkk45454 Abc123apple_xyz=17192fklfjlaskj*&*&*(&!* (&kdsklks dklkk45454 Abc123apple_xyz=17192fklfjlaskj*&*&*(&!* (&kdsklks dklkk45454

4.2.1 Without Integrity

Test Plan 1

Image used: #1 Secret Data used: #1 Integrity Check = Uncheck

Figure 29: *Result Analysis of Test Plan 1*

Secret data extracted without any integrity check. No authentication of secret data even if it is modified by an intruder, receiver will not be confident about secret data.

Test Plan 2

Image used: #1 Secret Data used: #2 Integrity Check = Uncheck

Figure 30: *Result Analysis of Test Plan 1*

Again no identification even if data modified

Test Plan 3

Image used: #2 Secret Data used: #1 Integrity Check = uncheck

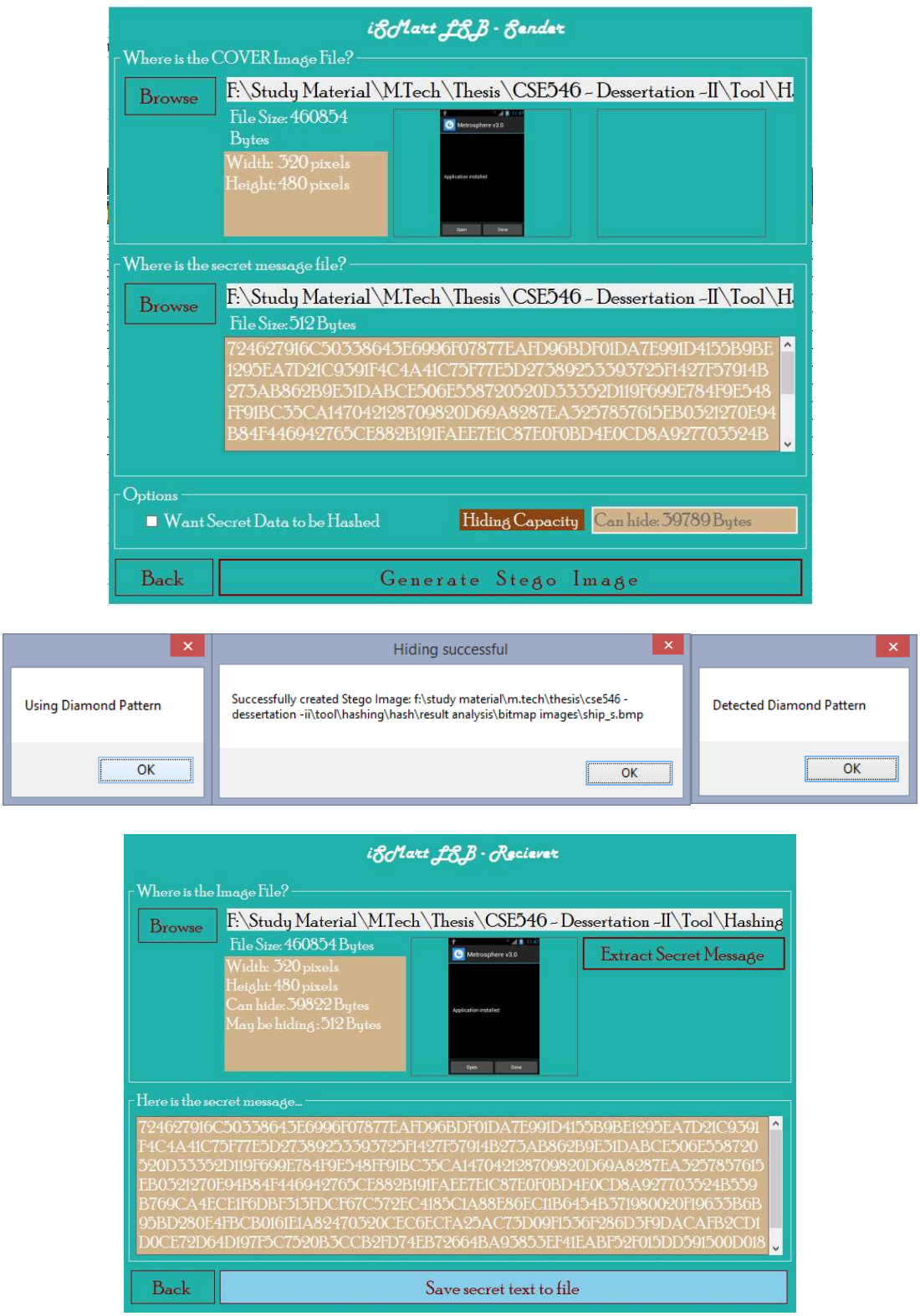


Figure 31: ***Result Analysis of Test Plan 3***

Same case as of test plan 1. Different image used with old secret data

4.2.2 With Integrity

Test Plan 4 - Hash Matched

Image used: #1 Secret Data used: #1 Integrity Check = Checked

In this test plan the integrity check (i.e. Secret data to be hashed) is checked. In this case secret data alone is not embedded into the cover media. Its hsh value is also calculated by the hash function and that 128 bit hash value will be embedded with the secret data into the cover media.

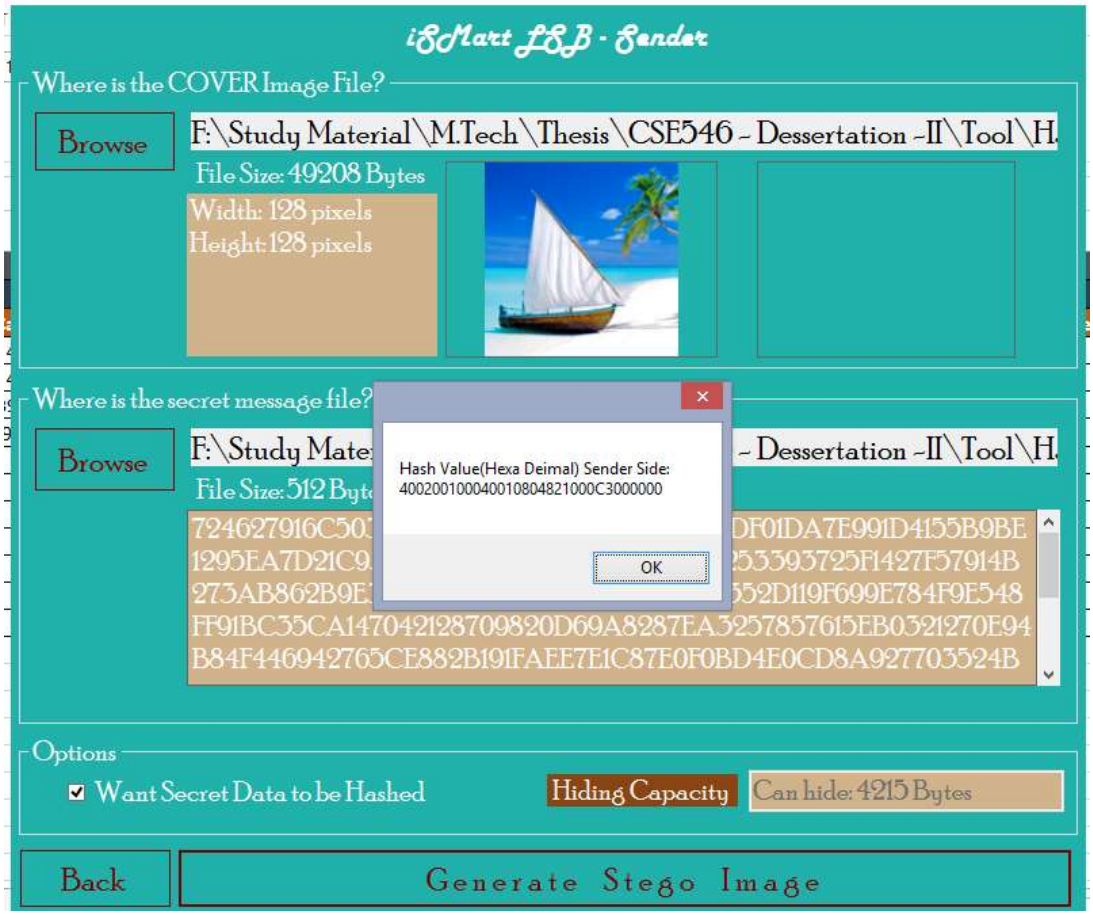
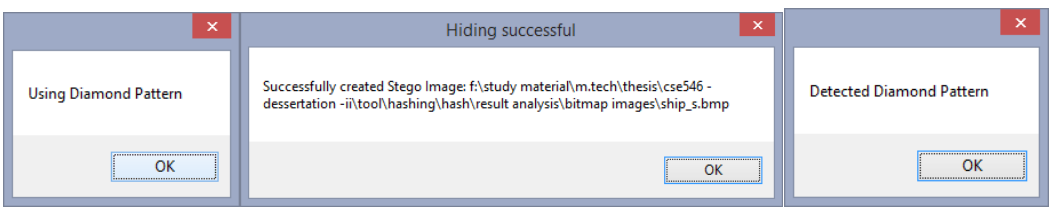


Figure 32: ***Hash Value (in Hexa-decimal Form) of secret data is shown***



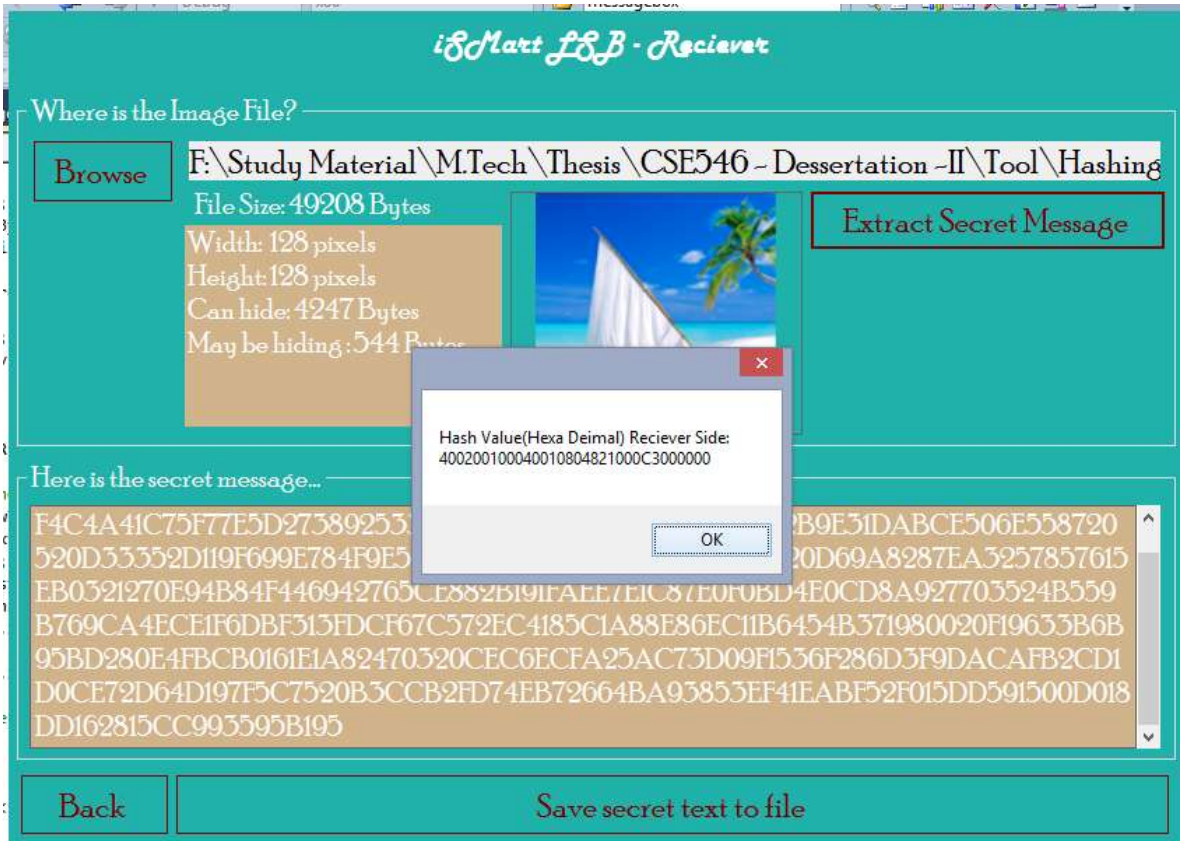


Figure 33: ***Same hash value generated at recipient side from secret data***

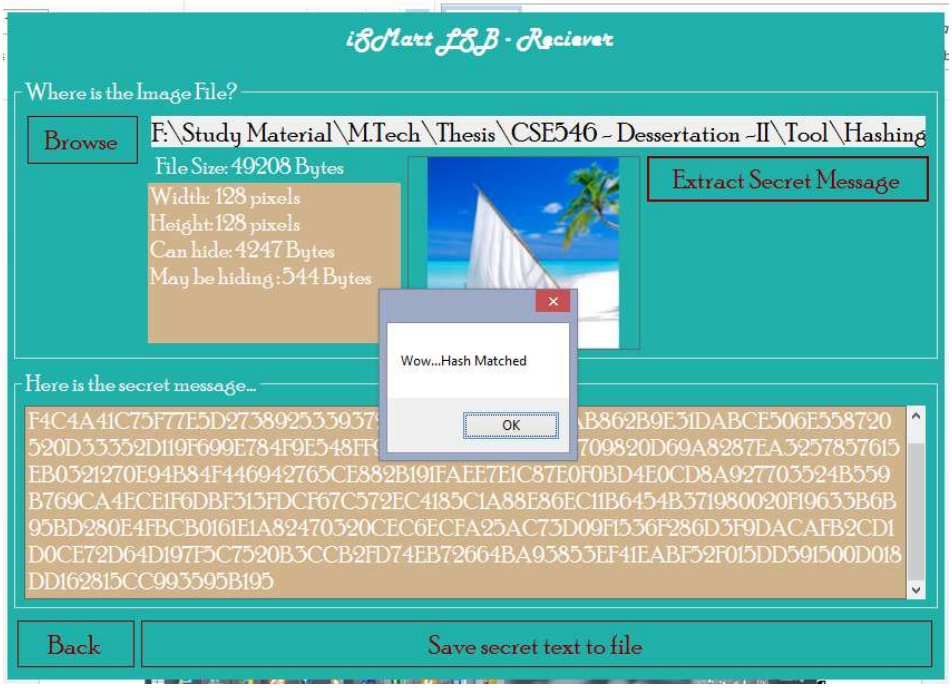


Figure 34: ***Regenerated Hash value from secret data received is matched with, stored in stego media***

Test Plan 5 – Hash Matched

Image used: #1 Secret Data used: #2 Integrity Check = Checked

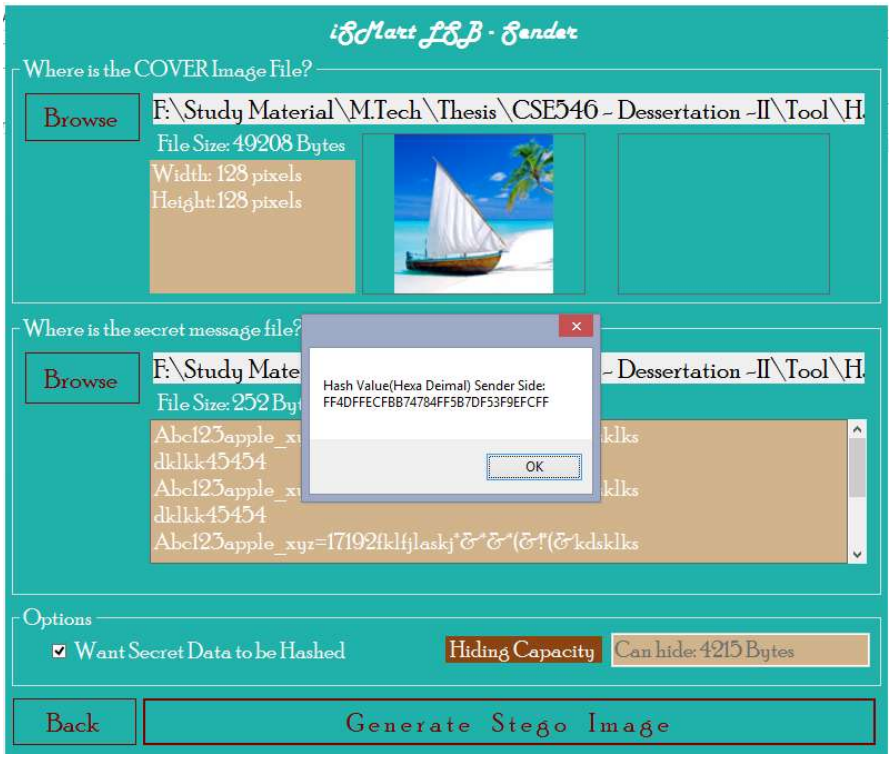


Figure 35: ***Hash Value (in Hexa-decimal Form) of secret data is shown***

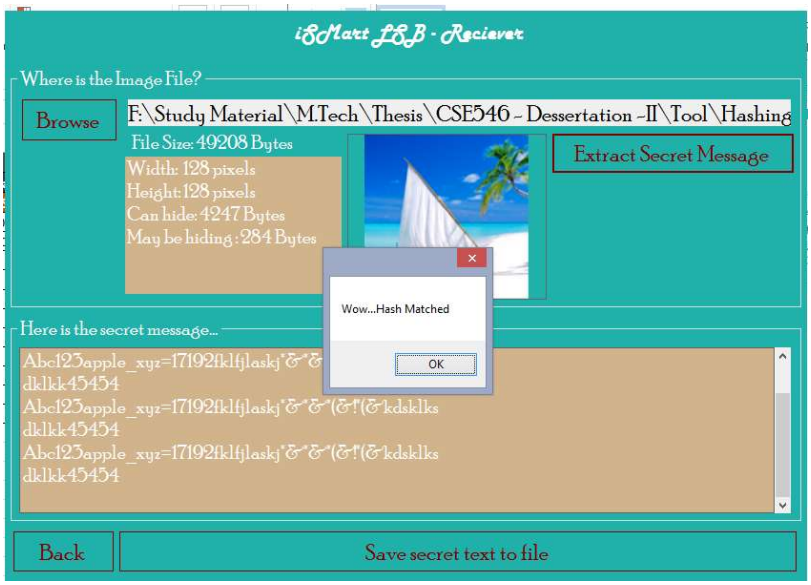


Figure 36: ***Regenerated Hash value from secret data received is matched with, stored in stego media***

Test Plan 6 – Hash Matched

Image used: #2 Secret Data used: #2 Integrity Check = Checked

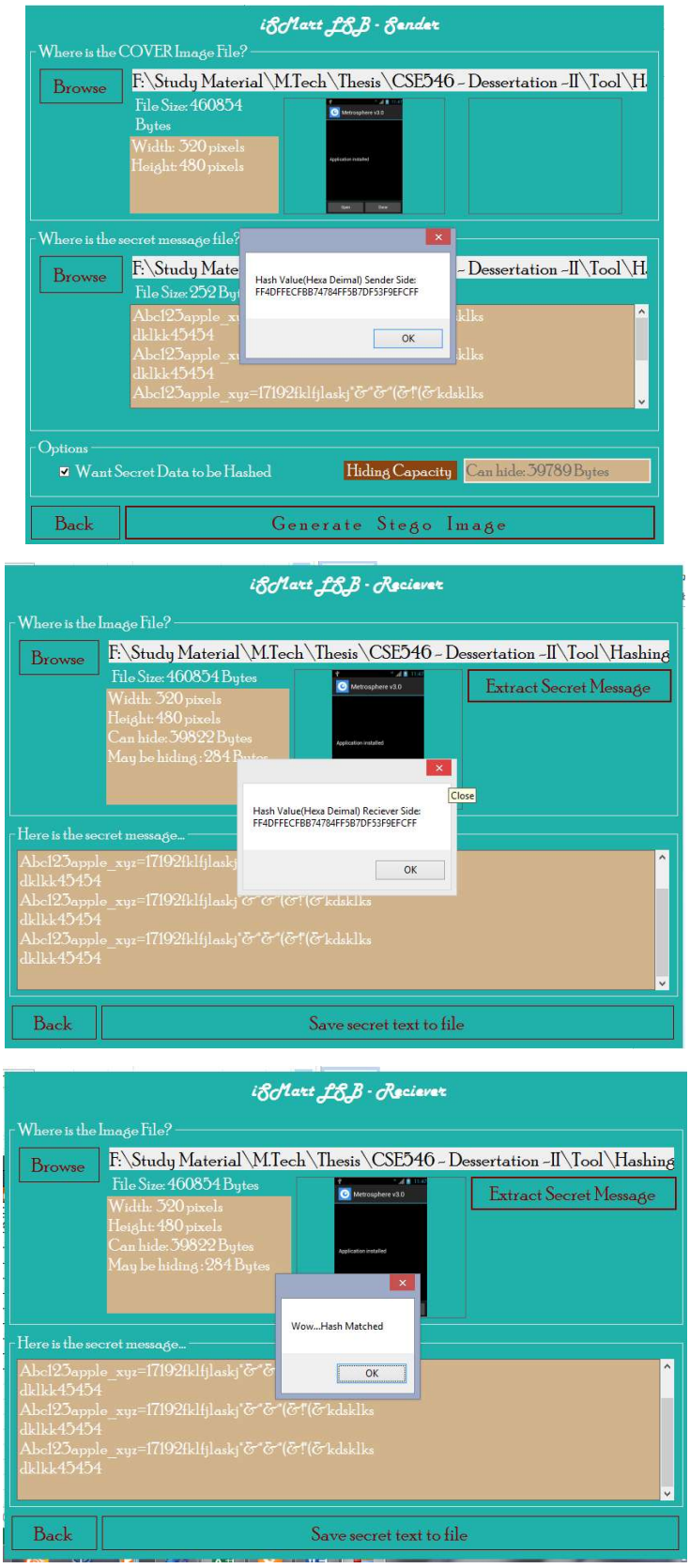


Figure 37: ***Result Analysis of Test Plan 6***

Test Plan 7 – Hash not Matched

Image used: #2 Secret Data used: #2 Integrity Check = Checked

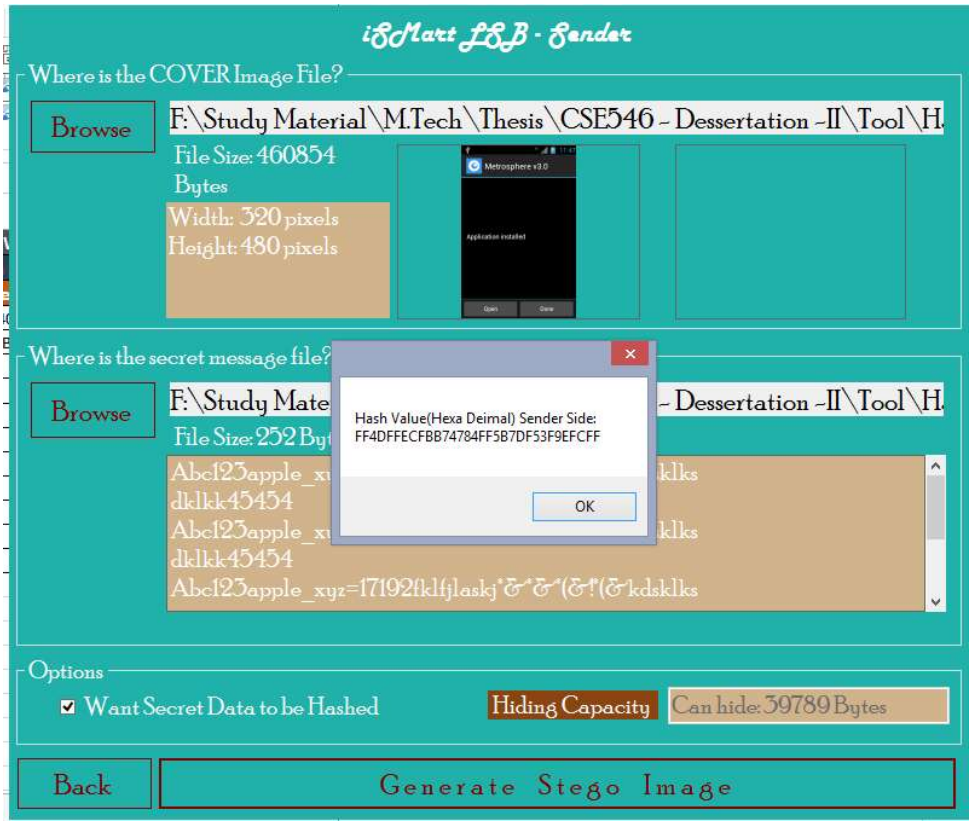


Figure 38: ***Hash Value (in Hexa-decimal Form) of secret data is shown***

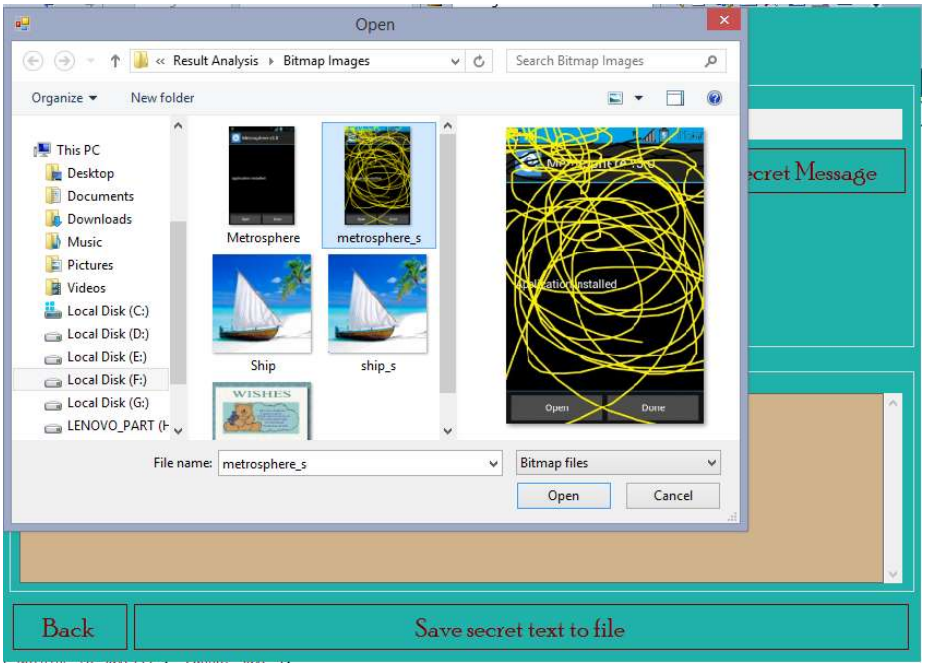


Figure 39: ***Image Altered***

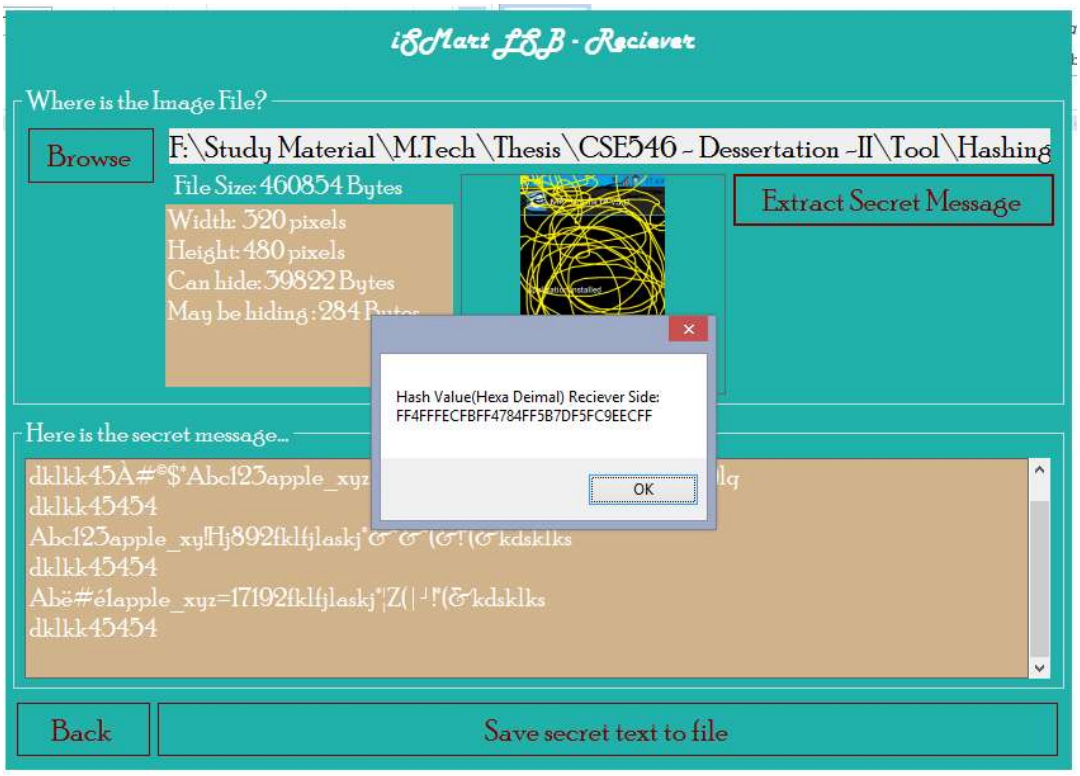


Figure 40: ***Hash value regenerated from the secret data received***

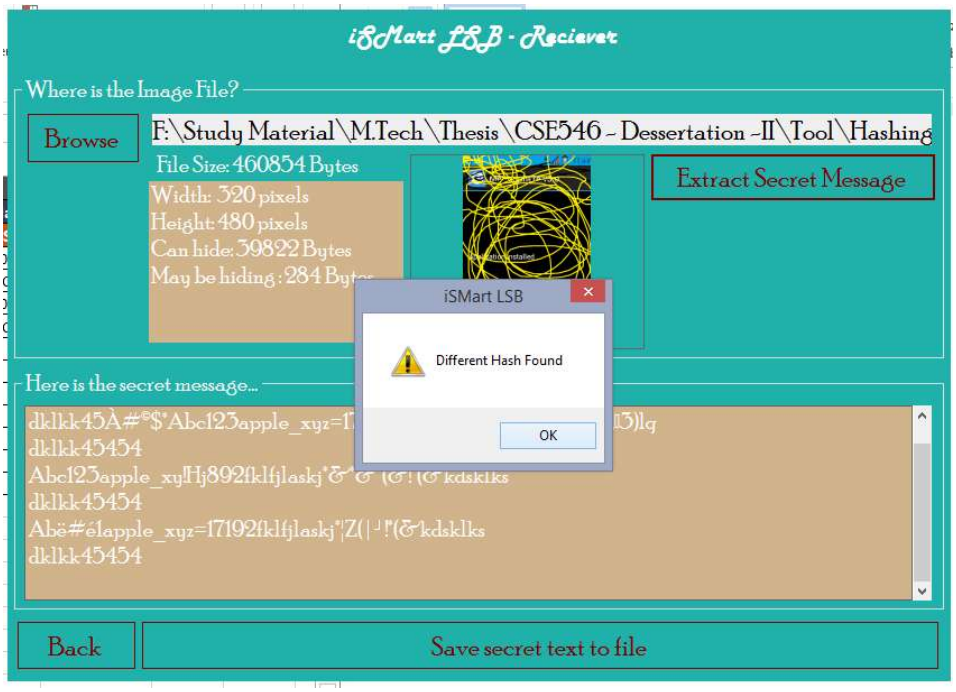


Figure 41: ***Regenerated hash value when compared with, stored in stego media found altered image***

Test Plan 8 – Hash not Matched because of different Organisation Key used

Image used: #2 Secret Data used: #2 Integrity Check = Checked

Organisation key used at Sender's site "TestPurpose"

```
//Input Developer Key and Organisation Key
string o = OrganisationKey("TestPurpose");      //Call to ---Organisation Key--- function
ok.Append(o);

int cr = 0;
int j = 0;
//----------------Hash Rounds----------------------
```

Figure 42: ***Organisation Key Used at Sender Side***

Organisation key used at Receivers's site "TestPurposeHacked"

```
//Input Developer Key and Organisation Key
string o = OrganisationKey("TestPurposeHacked");      //Call to ---Organisation Key--- function
ok.Append(o);

int cr = 0;
int j = 0;
//----------------Hash Rounds----------------------
```

Figure 43: ***Organisation Key Used at Receiver Side***

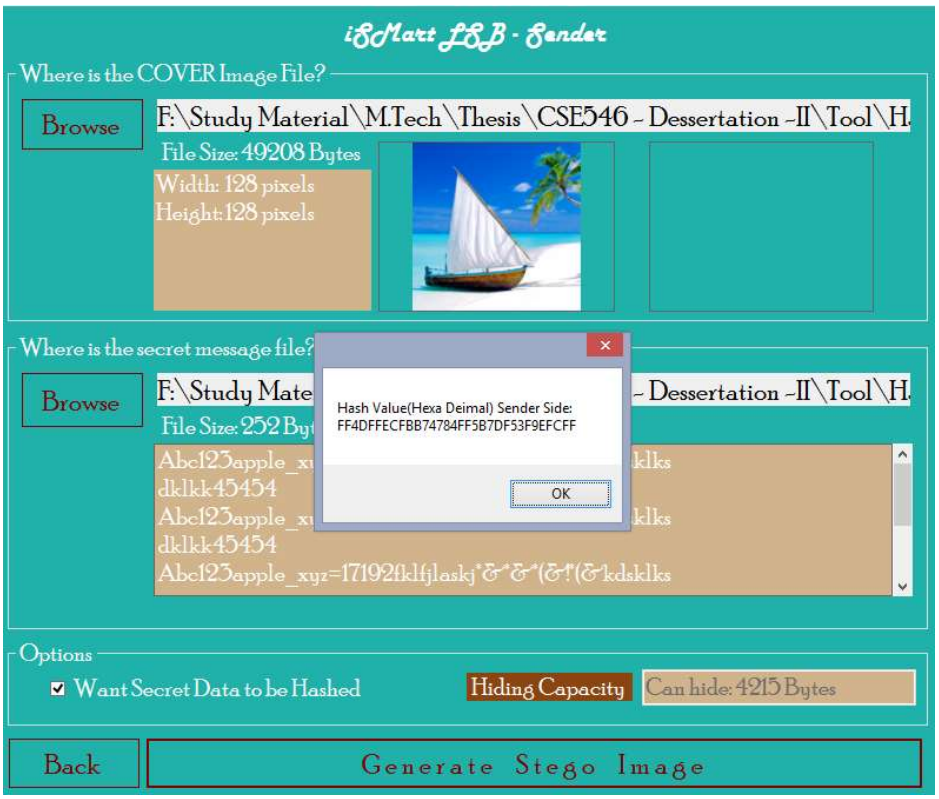


Figure 44: ***Hash Value (in Hexa-decimal Form) of secret data is shown***

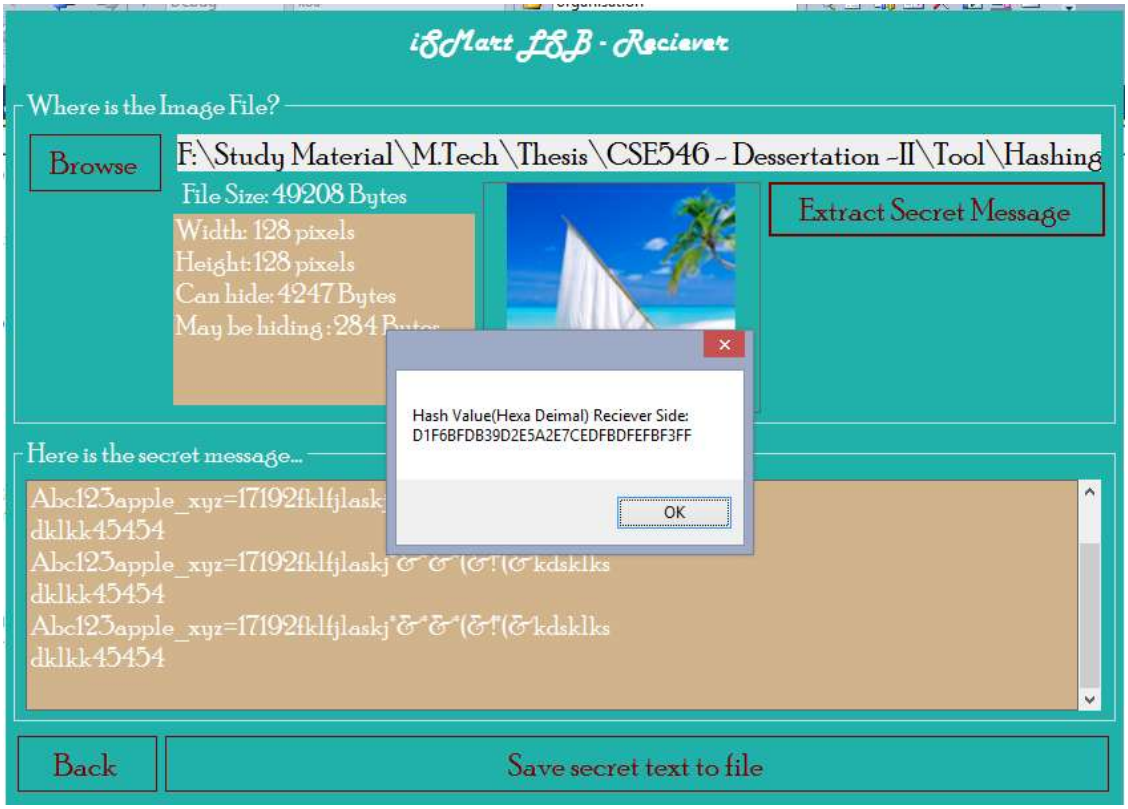


Figure 45: ***Hash value regenerated from the secret data received***

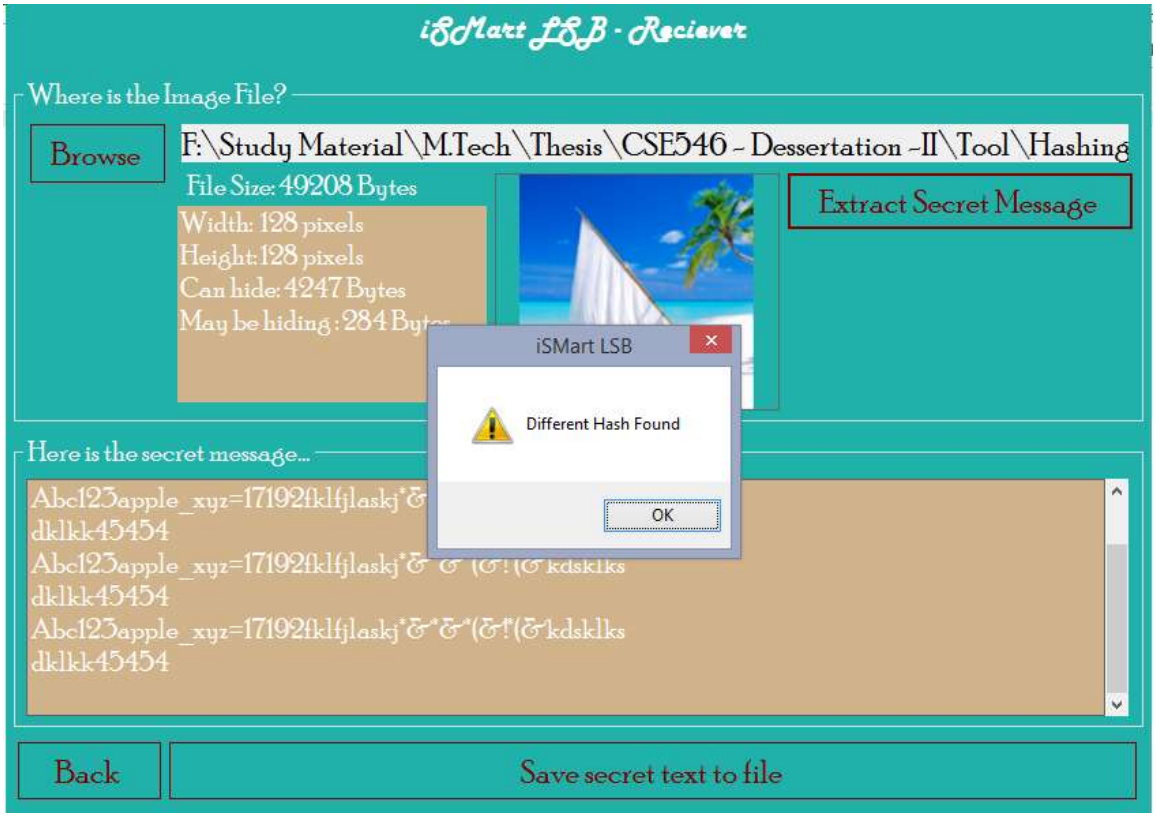


Figure 46: ***Regenerated hash value when compared with, stored in stego media found variation***

So these test case illustrates that Hash algorithm applied on secret data results in more secure information transmission. Even if intruders on the network detects the secret data and tried to alter it, receiver would get know about the variation in secret data when hsh value is regenerated and compared with the received hash value (stored in stego media).

4.3 Summary

The motive of this research work is to achieve confidentiality, accountability and integrity of the secret message transmitted on the network through cover file. Even if the intruder on the network is able to read and modify the content of secret message still the receiver would get to know that there is a change in the message or hash value. Thus, all such securities would be achieved by the use of cryptographic hash function, which would generate a fixed size number of bit pattern corresponding to bits input to it. Research would be stronger when the algorithm generating hash value use some different technique as compared to existing and working if unknown to public users. This would make hash value unique in itself and intruder on the network if change the content of secret message; unable to generate hash value same as by proposed hash algorithm.

Chapter 5
Conclusion and Future Scope

5.1 Conclusion

The results of this study proves that this technique provides more security to the secret data in the following manners:

- This study achieved integrity with the use of hash algorithm
- The hash algorithm used in this study also provide authentication of message

This study helps the communicating parties (i.e. Sender and receiver to whom the secret data belongs) to detect the alteration made to secret data on the network. .Even if the intruder on the network is able to read and modify the content of secret message embedded into cover media, the receiver would still get to know that there is a change in the message or hash value. Thus this method would provide integrity, the way to detect the change in the content of secret message through the hash code which was generated on sender side. Research would be stronger when the algorithm generating hash value uses some different techniques as compared to existing and working.

Personally, this study has offered immense knowledge, a glimpse into the ethos of the steganography, cryptographic hash and a hands-on experience in developing a security tool that could serve the society as well. To the society this system could serve as a cost-effective, time-saving yet secure mechanism to enable sharing secret messages between the two parties with a minimal risk of information stealth.

5.2 Future Scope

The results of the current study can opens up the scope for possible improvements in following respects:

- The iSMart LSB Tool to support for displaying/saving multiple formats of secret data (beyond text file formats).
- Development of specific tool that performs the entire process of embedding stego media into audio file. This will eliminate the need for a third party tool like windows media player, winamp, itunes, id3tag editor, etc.
- In addition to image format of BMP, additional image or file types like JPEG, PNG, etc. can be supported.

- Any other or improved application of cryptography over the secret data prior embedding it in album art image file.
- Support for dealing with multiple album art images as capable in ID3v2 tags.
- Concept of public/private key for more secure hashing.
- Key based approach for steganography.
- Addition of noise during steganography process.
- Providing support for handling transmission impairments to the level possible.

REFERENCES

[1] Amirtharajan R., Ramkrishnan K., Krishna M. V., Balaguru R. J. B. (2012) "Who decides hiding capacity? I, the Pixel Intensity", IEEE.

[2] Amritharajan R., Akila R., Deepikachowdavarapu P., (2010) "A Comparative Analysis of Image Steganography", International Journal of Computer Applications (0975 – 8887) Volume 2 – No.3.

[3] Bandyopadhyay S. K., Paul T. U., Raychoudhury A. (2013) "Image Compression using Approximate Matching and Run Length", IJACSA.

[4] BMP (Windows) Header Format; Available online at http://www.fastgraph.com/help/bmp_header_format.html

[5] BMP Image File Format; Available online at: http://www.fastgraph.com/help/bmp_header_format.html

[6] Cheddad A., Condell J., Curran K., McKevitt P. (2010) "Digital Image Steganography: Survey and Analysis of current methods" Signal Processing 90 Elsevier, pp. 727–752.

[7] Harrison O., Waldron J. (2007) "Practical Symmetric Key Cryptography on Modern Graphics Hardware".

[8] History of Steganography, Available online at: stegano.net/tutorial/steg-history.html

[9] ID3 Tag Description; Available online at: http://id3.org

[10] ID3 Tag Editor - Software to embed and extract the ID3 Tag information into and from an MP3 file, Available for download at: http://file.softsea.com/MP3_Tag_Editor/id3tageditor_setup.exe

[11] James C., Judge, (2011) "Steganography: Past, Present, Future" http://www.sans.org/reading-room/whitepapers/stenganography/steganography-past-present-future-552

[12] Joel A., Yevgeniy D., Moni N., Gil S., Shabsi W., Daniel W. (2009) "Public-Key Encryption in the Bounded-Retrieval Model".

[13] Johnson N. F., Jajodia S. (1998) "Exploring Steganography: Seeing the Unseen" IEEE Computer 31 (2), pp. 26-34.

[14] Joseph S., Srikanth N., Abhilash E. N. (2013) "A Novel Approach of Modified Run Length Encoding Scheme for High Speed Data Communication Application", IJSR.

[15] Kurak C., McHugh J., (1992) “A cautionary note on image downgrading”, Proc. of the IEEE 8th Annual Computer Security Applications Conference, pp. 153–159.

[16] Mittu S., Sobti R. (2013) “Conveying Secret Messages through Album Art in MP3 Files” Proc. Of Wilkies 100 International Conference on Computing Sciences, Elsevier.

[17] Morkel T., Eloff J. H. P., Olivier M. S., (2005) “An Overview of Image Steganography. Proc. Fifth Annual Information Security South Africa Conference (ISSA2005), Sandton, South Africa.

[18] Natanj S, Taghizadeh S. R. (2011) “Current Steganography Approaches: A survey” International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) Volume 1, Issue 1, Dec. 2011.

[19] NIST, Computer Security Division, Computer Security Resource Center. http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html. Accessed on 02 Mar 2015.

[20] Pfitzmann B. (1996) “Information Hiding Terminology,” Proc. First Int’l Workshop Information Hiding, Lecture Notes in Computer Science No. 1,174, Springer-Verlag, Berlin, pp. 347-356.

[21] Preneel B., “First 30 Years of Cryptographic Hash Functions and the NIST SHA-3 Competition” https://www.cosic.esat.kuleuven.be/publications/article-1532.pdf

[22] Provos N., Honeyman P. (2003) “Hide and Seek: An Introduction to Steganography” IEEE Security & Privacy Magazine 1, pp. 32-44.

[23] SHA 3 Competition on NIST http://csrc.nist.gov/groups/ST/hash/sha-3/

[24] Singh A., Singh V. P. (2013) “An Enhanced Run Length Coding for JPEG Image Compression”, IJCA.

[25] Singh S. (2014) *Public Cloud Storage using NAS*, Thesis M.Tech., Lovely Professional University, Phagwara.

[26] Stallngs W. (2011) *Cryptography and Network Security Principles and Practice*, Pearson Education Publications.

[27] Zaid M., Mohd W., Zulkifli (2007) “Evolution of Cryptograpghy” https://idazuwaika.files.wordpress.com/2008/06/evolution_of_cryptorgarphy.pdf