COMBINATION OF CRYPTOGRAPHY AND STEGANOGRAPHY FOR SECURE COMMUNICATION IN VIDEO FILE

Sharone Gorla
B.Tech, Jawaharlal Nehru Technological University, 2006

PROJECT

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

FALL
2009
COMBINATION OF CRYPTOGRAPHY AND STEGANOGRAPHY FOR SECURE COMMUNICATION IN VIDEO FILE

A Project

by

Sharone Gorla

Approved by:

__________________________________, Committee Chair
Dr. Isaac Ghansah

__________________________________, Second Reader
Prof. Dick Smith

Date
Student: Sharone Gorla

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

__________________________, Graduate Coordinator       ________________
Dr. Cui Zhang          Date

Department of Computer Science
“Combination of cryptography and Steganography for secure communication” is an application, which combines both Cryptography methods (i.e. Encryption, decryption) and Steganography techniques to make the communication more secure. The outcome of this project is to create a cross-platform tool that can effectively hide a message (i.e. Word document) inside a digital video file. It is concerned with embedding information in a secure and robust manner.

The application first compresses the word document with secret message, and then encrypts the compressed file and uses the resulted file as the secret message to hide in the digital video file generating a Stego-object. The intended receiver de-embeds decrypts and decompresses the Stego-object respectively to get the hidden message. This paper also attempts to identify the requirements of a good Steganographic algorithm and briefly reflects on different types of steganalysis techniques.

The application is developed in Java and uses Tiny encryption algorithm and Discrete Cosine Transformation-Least significant bit algorithm for implementing Cryptography and Steganography respectively.

Dr. Isaac Ghansah
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. CRYPTOGRAPHY</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Secret Key Cryptography</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Public-Key Cryptography</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Hash Functions</td>
<td>8</td>
</tr>
<tr>
<td>3. TEA (Tiny Encryption Algorithm)</td>
<td>10</td>
</tr>
<tr>
<td>3.1 Notations for Bitwise Shifts and Rotations</td>
<td>11</td>
</tr>
<tr>
<td>3.2 XOR</td>
<td>11</td>
</tr>
<tr>
<td>3.3 Integer Addition and Subtraction</td>
<td>11</td>
</tr>
<tr>
<td>3.4 Encryption Routine</td>
<td>12</td>
</tr>
<tr>
<td>3.5 Decryption Routine</td>
<td>16</td>
</tr>
<tr>
<td>4. STEGANOGRAPHY</td>
<td>19</td>
</tr>
<tr>
<td>4.1 DCT-LSB</td>
<td>21</td>
</tr>
<tr>
<td>5. DEFLATE COMPRESSION ALGORITHM</td>
<td>24</td>
</tr>
<tr>
<td>6. PROPOSED SYSTEM</td>
<td>26</td>
</tr>
<tr>
<td>6.1 Proposed System</td>
<td>26</td>
</tr>
<tr>
<td>6.2 Modules of the Application</td>
<td>26</td>
</tr>
<tr>
<td>7. STEGANALYSIS</td>
<td>28</td>
</tr>
<tr>
<td>7.1 Steganalysis</td>
<td>28</td>
</tr>
<tr>
<td>7.2 Attacks on Steganography</td>
<td>28</td>
</tr>
<tr>
<td>7.2.1 Detection</td>
<td>29</td>
</tr>
<tr>
<td>7.2.2 Destruction</td>
<td>29</td>
</tr>
<tr>
<td>8. UNIFIED MODELING LANGUAGE</td>
<td>31</td>
</tr>
<tr>
<td>8.1 Use-Case Diagrams</td>
<td>31</td>
</tr>
<tr>
<td>8.2 Sequence Diagrams</td>
<td>35</td>
</tr>
<tr>
<td>8.3 Other Unified Modeling Diagrams</td>
<td>36</td>
</tr>
</tbody>
</table>
9. USER GUIDE FOR THE SYSTEM................................................................................ 38
10. CONCLUSION............................................................................................................... 49
References.................................................................................................................... 50
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Figure 1.1: Types of Steganography</td>
<td>2</td>
</tr>
<tr>
<td>2.</td>
<td>Figure 2.1: Overview of Cryptology</td>
<td>5</td>
</tr>
<tr>
<td>3.</td>
<td>Figure 2.2: Secret Key Cryptography</td>
<td>6</td>
</tr>
<tr>
<td>4.</td>
<td>Figure 2.3: Public Key Cryptography</td>
<td>7</td>
</tr>
<tr>
<td>5.</td>
<td>Figure 2.4: Hash Functions</td>
<td>8</td>
</tr>
<tr>
<td>6.</td>
<td>Figure 3.1: Encryption Routine for TEA</td>
<td>12</td>
</tr>
<tr>
<td>7.</td>
<td>Figure 3.2: Two Feistel Rounds (one cycle) of TEA</td>
<td>13</td>
</tr>
<tr>
<td>8.</td>
<td>Figure 3.3: Encryption Process for TEA</td>
<td>15</td>
</tr>
<tr>
<td>9.</td>
<td>Figure 3.4: Decryption Routine for TEA</td>
<td>16</td>
</tr>
<tr>
<td>10.</td>
<td>Figure 3.5: Decryption Process for TEA</td>
<td>17</td>
</tr>
<tr>
<td>11.</td>
<td>Figure 4.1: Steganographic System</td>
<td>19</td>
</tr>
<tr>
<td>12.</td>
<td>Figure 4.2: LSB Process</td>
<td>22</td>
</tr>
<tr>
<td>13.</td>
<td>Figure 4.3: Embedding Process of DCT-LSB</td>
<td>22</td>
</tr>
<tr>
<td>14.</td>
<td>Figure 4.4: Extracting Process of DCT-LSB</td>
<td>23</td>
</tr>
<tr>
<td>15.</td>
<td>Figure 8.1.1: Use Case Diagram for Sender Module of the Application</td>
<td>32</td>
</tr>
<tr>
<td>16.</td>
<td>Figure 8.1.2: Use Case Diagram for Receiver Module of the Application</td>
<td>34</td>
</tr>
<tr>
<td>17.</td>
<td>Figure 9.1: How to Use the System</td>
<td>39</td>
</tr>
<tr>
<td>18.</td>
<td>Figure 9.2: Compress the File Step-1</td>
<td>40</td>
</tr>
<tr>
<td>19.</td>
<td>Figure 9.3: Compress the File Step-2</td>
<td>40</td>
</tr>
<tr>
<td>20.</td>
<td>Figure 9.4: Encrypt the File Step-1</td>
<td>41</td>
</tr>
<tr>
<td>21.</td>
<td>Figure 9.5: Encrypt the File Step-2</td>
<td>42</td>
</tr>
<tr>
<td>22.</td>
<td>Figure 9.6: Embed the File Step-1</td>
<td>42</td>
</tr>
<tr>
<td>23.</td>
<td>Figure 9.7: Embed the File Step-2</td>
<td>43</td>
</tr>
<tr>
<td>24.</td>
<td>Figure 9.8: De-embed the File Step-1</td>
<td>44</td>
</tr>
<tr>
<td>25.</td>
<td>Figure 9.9: De-embed the File Step-2</td>
<td>44</td>
</tr>
<tr>
<td>26.</td>
<td>Figure 9.10: Decrypt the File Step-1</td>
<td>45</td>
</tr>
<tr>
<td>27.</td>
<td>Figure 9.11: Decrypt the File Step-2</td>
<td>46</td>
</tr>
</tbody>
</table>
28. Figure 9.12: Decrypt the File Step-3 46
29. Figure 9.13: Decompress the File Step-1 47
30. Figure 9:14: Decompress the File Step-2 47
Chapter 1
INTRODUCTION

In the field of Data Communication, security-issues have the top priority. Classical cryptography is one of the ways to secure plain text messages. Cryptography addresses the necessary elements for secure communication namely privacy, confidentiality, key exchange, authentication, and non-repudiation but reveals the fact that communication is happening. Steganography takes cryptography a step farther by hiding the existence of the information. Steganography comes from the Greek words Steganós (Covered) and Graptos (Writing). Markus Kahn defines Steganography as an art and science of communicating in a way that hides the existence of the communication. Steganographic technology plays a vital role in the future of computer security, primarily privacy on open systems such as the Internet.

There are a large number of Steganographic methods, which most of us are familiar with ranging from invisible ink and microdots to secreting a hidden message in the second letter of each word of a large body of text etc. With computers and networks there are many other ways of hiding information, such as hiding text within Web pages, Null ciphers etc. Steganography however is significantly more sophisticated than the examples above.

Figure1.1 taken from [1], gives different applications of Steganography. Protection against detection (Data hiding) and protection against removal (Document Marking) are two major areas Steganographic methods are used. Steganographic Data hiding
algorithms allows user to hide large amounts of information within digital files like Image, audio and video files. These forms of Steganography often used in conjunction with cryptography adding layers of security.

The Other major area of Steganography is document marking where the message to be inserted is used to assert copyright over a document. This can be further divided into watermarking and fingerprinting. Copyright abuse is the motivating factor in developing new document marking technologies like digital watermarking and digital fingerprinting. “Digital Watermarking is a way to hide a secret or personal message to protect a product’s copyright or to demonstrate data integrity”. “Digital Fingerprinting is an emerging technology to protect multimedia from unauthorized redistribution. It embeds a

![Figure 1.1 Types of Steganography](image-url)
unique ID into each user's copy, which can be extracted to help identify culprits when an unauthorized leak is found” [2].

Neither Cryptography nor Steganography is a turnkey solution to privacy of open systems. To add multiple layers of security it is always a good practice to use both Cryptography and Steganography together. The aim of this paper is to describe a method for integrating together cryptography and Steganography for secure communication using a Video file. The proposed system first compresses the secret message (i.e. word document) and then implements cryptographic algorithms to the compressed message. The resulted file is used as the secret message to be hidden in the digital video file. Once the video file is embedded with the secret message, it is sent to the intended receiver. The video file should be de-embedded, decrypted and decompressed to get the original secret message hence, adding three layers of security to the communication.

In chapter two, we will define Cryptography and explain various types of Cryptography. Chapter 3 will review Tiny Encryption Algorithm (TEA). In chapter four, will discuss various Steganographic methods and will review Discrete Cosine Transformation-Least Significant Bit Steganography algorithm. In chapter five, we will look in detail at Deflate compression algorithm and in chapter six, we will provide with a method for integrating Cryptography and Steganography adding multiple layers of security. In chapter seven, we will look at various types of attacks possible on Steganographic methods. In Chapter 8, we will look at various Unified modeling language diagrams and Chapter 9 will provide
with a user guide for the system. Chapter 10 will conclude with a brief discussion of the implications of Steganographic technology. At the end, we will list the resources used in researching and developing the application.
Chapter 2

CRYPTOGRAPHY

One of the classic techniques used for ensuring privacy of files and communication is Cryptography. Lorenzo Cappelletti refers cryptography to “the science of keeping secrecy of messages exchanged between a sender and a receiver over an insecure channel. The objective is achieved by encoding data so that it can only be decoded by specific individuals.”

![Figure 2.1: Overview of Cryptology](image)

Figure 2.1 taken from [3], gives an overview of the cryptology. Cryptanalysis is a study of how to compromise (defeat) cryptographic mechanism. Cryptology is the study of Cryptography and Cryptanalysis. The goal of cryptography is to make it possible for two communication entities to exchange a message in such a way that no third party can understand the message. Cryptography methods generally alter the original message in such a way that the recipient can undo the alteration to get the original message. The
original message is termed “plaintext” and the encoded or altered message “ciphertext”.
The process of conversion from plaintext to ciphertext called “Encryption”, and the opposite operation known as “decryption” [3].

In general, three types of cryptographic schemes are in practice to achieve the Cryptography goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions (or Protocols). The type and length of the keys utilized depend upon the encryption algorithm.

2.1 Secret Key Cryptography:

Secret key Cryptography, also known as symmetric encryption uses a single key for both encryption and decryption. The sender uses the key to encrypt the plaintext and sends the ciphertext to the receiver. The recipient applies the same key to decrypt the message and recover the plaintext.

\[
\begin{align*}
\text{Plaintext} & \quad \xrightarrow{K} \quad \text{Ciphertext} & \quad \text{Plaintext} \\
E(\cdot) & \quad \xrightarrow{K} \quad D(\cdot)
\end{align*}
\]

K-key, E-Encryption, D-Decryption

Figure 2.2: Secret key Cryptography

Figure 2.2 shows the process of secret key cryptography. The biggest difficulty with this approach is the distribution of the key. Secret key cryptography schemes fall into either
stream ciphers or block ciphers. Stream ciphers operate on a single bit (byte or computer word) at a time and implement some form of feedback mechanism so that the key is constantly changed.

A block cipher gets its name from the fact that the scheme encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same ciphertext when using the same key in a block cipher whereas the same plaintext will encrypt to different ciphertext in a stream cipher [3]. Block ciphers can operate in one of the several modes. Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB) are the most important modes. Data Encryption Standard (DES), Advanced Encryption Standard (AES), CAST-128/256, Rivest Ciphers (aka Ron's Code), Blowfish are some of the Secret key cryptography algorithms [3].

2.2 Public-Key Cryptography:

\[
\begin{array}{c}
\text{Plaintext} \\
\rightarrow \\
E( ) \\
\rightarrow \\
\text{Ciphertext} \\
\rightarrow \\
D( ) \\
\rightarrow \\
\text{Plaintext}
\end{array}
\]

K-key, E-Encryption, D-Decryption

Figure 2.3: Public key Cryptography

Public key cryptography is a two-key crypto system in which two parties can engage in a secure communication without having to share a secret key. One key is used to encrypt
the plaintext, designated the public key which can be advertised. The other key is used to decrypt the ciphertext to plaintext and is designated the private key which is never revealed to another party. This approach also called as asymmetric cryptography, because we use a pair of keys. Figure 2.3 shows the process of the public cryptographic algorithms. Public key cryptography depends upon the one-way functions, which are easy to compute whereas their inverse function is relatively difficult to compute. RSA, Diffie-Hellman, Digital signature Algorithm (DSA), ElGamal, and Elliptic Curve Cryptography (ECC, are the examples of Public-key cryptography algorithms [3].

2.3 Hash Functions:

Hash functions, are also called message digests and one-way encryption. Hash function algorithms do not use a key to carry out the encryption and decryption process. Instead, the algorithm computes a fixed length hash value based upon the plaintext that keeps both the contents and the length of the message secure.

![Hash Function Diagram]

Figure 2.4: Hash Functions

Figure 2.4 shows the process of Hash function cryptographic algorithms. Hash functions algorithms are typically used to provide a digital fingerprint of file contents, often used to ensure that the file has not been altered by an intruder or virus. Message Digest (MD)
algorithms, Secure Hash Algorithm (SHA), RIPEMD, Hash of Variable Length (HAVAL), Tiger are some of the examples of Hash function algorithms.
Chapter 3

TEA (Tiny Encryption Algorithm)

Tiny Encryption Algorithm is a Feistel cipher encryption algorithm that uses operations from mixed orthogonal algebraic groups like XOR, ADD and SHIFT. David Wheeler and Roger Needham of the Cambridge University Computer Laboratory designed TEA in the year 1994.

A **Feistel cipher** is a block cipher with a particular structure known as a **Feistel network**. In a Feistel cipher, the data been encrypted is split into two halves. The round function \( F(\cdot) \) is applied to one half using a sub key and the output of \( F \) is XORed with the other half and the two halves are swapped. Each round follows the same pattern except for the last round where there is no swap. A nice feature of a Feistel cipher is that encryption and decryption are structurally identical i.e. the sub keys used during encryption at each round are taken in reverse order during decryption [4].

The main goal of TEA is to minimize memory footprint and maximize speed. TEA is simple to implement, has less execution time, and takes minimal storage space. TEA is” highly resistant to differential cryptanalysis, and achieves complete diffusion (where a one bit difference in the plaintext will cause approximately 32 bit differences in the cipher text) after only six rounds.”[4]. It uses a large number of iterations rather than a complicated program.

**Notation:** Any number subscripted with “h” represents a Hexadecimal number
e.g: \(10_h\) represents 16 in decimal values.

### 3.1 Notations for Bitwise Shifts and Rotations:

- \(x << y\): denotes logical left shift of \(x\) by \(y\) bits.
- \(x >> y\): denotes logical right shift of \(x\) by \(y\) bits.
- \(x <<< y\): denotes left rotation of \(x\) by \(y\) bits.
- \(x >>> y\): denotes right rotation of \(x\) by \(y\) bits.

### 3.2 XOR:

In computer science, an XOR is a mathematical operation that combines two bits. It returns value is TRUE if either of the two bits is TRUE, but false if both are equal. For our cryptography algorithm, we do an XOR combining two strings of bits. Say \(x\) and \(y\) are two string patterns then XOR for \(x\) and \(y\) is denoted by \(x \oplus y\) [4].

### 3.3 Integer Addition and Subtraction:

The operation of integer addition modulo \(2^n\) is denoted by \(x \oplus y\). and subtraction modulo \(2^n\) is denoted by \(x \ominus y\). Where \(x, y \in Z_2^n\) (The value of \(n\) should be clear from the context)

The key is set at 128 bits and the key schedule algorithm splits the 128-bit key \(K\) into four 32-bit blocks \(K = (K[0], K[1], K[2], K[3])\). The 128-bit key is enough to prevent simple search techniques being effective [4].
### 3.4 Encryption Routine:

The Encrypt Routine given in figure 3.1 taken from [4], is written in the C language and assumes a 32-bit word size. The 128 bit key is split into four parts and is stored in K[0] - k[3] and the Data is stored in v[0] and v[1].

```c
void code(long* v, long* k) {
    unsigned long y=v[0],z=v[1], sum=0, /* set up */
    delta=0x9e3779b9,       /* a key schedule constant */
    n=32 ;
    while (n-->0) { /* basic cycle start */
        sum += delta ;
        y += ((z<<4)+k[0]) ^ (z+sum) ^ ((z>>5)+k[1]) ;
        z += ((y<<4)+k[2]) ^ (y+sum) ^ ((y>>5)+k[3]) ;
    }             /* end cycle */
    v[0]=y ; v[1]=z ;
}
```

Figure 3.1: Encryption Routine for TEA

The constant delta is given as $\delta = \sqrt{5} - 1) \cdot 2^{31}$ i.e. 9E3779B9h and is derived from the golden number ratio to ensure that the sub keys are distinct and its precise value has no cryptographic significance.

TEA uses addition and subtraction as the reversible operators instead of XOR. The TEA encryption routine relies on the alternate use of XOR and ADD to provide nonlinearity. The algorithm has 32 cycles (64 rounds). TEA is short enough to write into almost any program on any computer. TEA on one implementation is three times as fast as a good software implementation of DES, which has 16 rounds. Figure 3.2 taken from [4], gives an overview of two rounds i.e. one cycle of TEA.
Key size: 128 bit key is split into four subkeys $K = \{ K[0], K[1], K[2], K[3] \}$

Block size: 64 bits

Structure: Feistel Network

Rounds: Variable (64 Feistel rounds (32 cycles) is recommended).

![Figure 3.2: Two Feistel Rounds (one cycle) of TEA](image-url)

- Represents Integer addition modulo
- Represents XOR
- Represents logical left shift by 4 bits
- Represents logical right shift by 5 bits
**Inputs for the Encryption routine**: Plaintext P, Key K

The plaintext is split into two halves as $P = (\text{Left}[0], \text{Right}[0])$

**Output for the Encryption routine**: The cipher text is $C$

Where $C = (\text{Left}[64], \text{Right}[64])$.

The plaintext block is split into two halves, $\text{Left}[0]$ and $\text{Right}[0]$ and each half is used to encrypt the other half over 64 rounds of processing then combined to produce the cipher text block. Each round $i$ has inputs $\text{Left}^[i-1]$ and $\text{Right}^[i-1]$, derived from the previous round, as well as a sub key $K[i]$ derived from the 128 bit overall $K$.

The Output and the delta constant of the $i^{th}$ cycle of TEA are given as

\[
\begin{align*}
\text{Left}^[i+1] &= \text{Left}^i \oplus F(\text{Right}^i, K[0,1], \delta^i), \\
\text{Right}^[i+1] &= \text{Right}^i \oplus F(\text{Right}^i+1, K[2,3], \delta^i), \\
\delta^i &= (i+1)/2 * \delta,
\end{align*}
\]

The sub keys $K[i]$ are different from $K$ and from each other.

The Round function $F$ contains the key addition, bitwise XOR and both left and right shift operations, and given as

\[
F(M, K[j,k], \delta[i]) = ((M \ll 4) \oplus K[j]) \oplus (M \oplus \delta[i]) \oplus ((M \gg 5) \oplus K[k])
\]
F - Round function and K[i] – key for the \text{i}^{\text{th}} round

Figure 3.3: Encryption Process for TEA
The keys $K[0]$ and $K[1]$ are used in the odd rounds and the keys $K[2]$ and $K[3]$ are used in even rounds. The round function of TEA encryption algorithm differs slightly from a classical Feistel cipher structure where integer addition modulo-$2^{32}$ is used instead of XOR as the combining operator. Figure 3.3 taken from [4], gives an overview of the encryption process for TEA.

### 3.5 Decryption Routine:

```c
void decode(long* v, long* k) {
    unsigned long n = 32, sum, y = v[0], z = v[1],
    delta = 0x9e3779b9 ;
    sum = delta<<5 ;
    /* start cycle */
    while (n-->0) {
        z -= (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;
        y -= (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
        sum -= delta ; }
    /* end cycle */
    v[0] = y ; v[1] = z ; }
```

Figure 3.4: Decryption Routine for TEA

The decryption routine given in figure 3.4 taken from [4], is same as the encryption routine with the cipher text as input and the sub keys $K[i]$ are used in the reverse order.

**Inputs for the Decryption routine:** Cipher text C, Key K

The cipher text is split into two halves as $C= (DLeft[0],DRight[0])$

Where $Dleft[0]=ERight[64]$ and $DRight[0]=Eleft[64]$
Output for the Decryption routine: The plain text is $P$, where $C = (D\text{Left}[64], D\text{Right}[64])$.

F - Round function and $K[i]$ – key for the $i^{th}$ round.

Figure 3.5: Decryption Process for TEA
The figure 3.5 taken from [4], gives the structure of the decryption algorithm for TEA. The intermediate value for the decryption process equals the corresponding value of the encryption process with the two halves of the value swapped. For example say the output of the \(n^{th}\) round of the encryption process is \(E_{\text{Left}}[i]\) concatenated with \(E_{\text{Right}}[i]\) then the input to the \((64-i)^{th}\) decryption round is \(D_{\text{Right}}[i]\) concatenated with \(D_{\text{Left}}[i]\). It is important to note that while cryptography is \textit{necessary} for secure communication, it is not by itself \textit{sufficient}.
Chapter 4

STEGANOGRAPHY

Steganography is the art and science of writing hidden messages inside innocent looking containers such as digital files, in such a way that no one apart from the sender and intended recipient realizes the existence of a hidden message [5]. Steganography uses redundant portions of the container file such as Video files to embed the secret message.

![Steganographic System Diagram](image)

Figure 4.1: Steganographic System

Figure 4.1 taken from [6], gives an overview of the Steganographic system. There are three different types of Steganographic algorithms namely Injection, Substitution and Generation.
**Injection (or insertion):** This technique adds bits to unused sections of digital files to hide the secret message. By doing this we avoid modifying those file bits that are relevant to an end-user—leaving the cover file perfectly usable. For example, we can add additional harmless bytes in an executable or binary file. Because those bytes do not affect the process, the end-user may not even realize that the file contains additional hidden information. Using an insertion technique changes file size.

**Substitution:** This technique is used to replace the least significant bits of information that determine the meaningful content of the original file with new data in a way that causes the least amount of distortion. The main advantage of this technique is that the cover file size does not change after the execution of the algorithm. On the other hand, this approach has few drawbacks. The resulting stego-file, may be adversely affected by quality degradation and that may raise suspicion. Another drawback is substitution method limits the amount of data that can be hide to the number of insignificant bits.

**Generation:** Unlike injection and substitution, this technique does not require an existing cover file. This technique generates a cover file for the sole purpose of hiding the message. The main flaw of the insertion and substitution techniques is that people can compare the stego file with any pre-existing copy of the cover file (which is supposed to be the same file) and discover differences between the two. We will not have that problem when using a generation approach, because the result is an original file, and is therefore immune to comparison tests. Among the substitution techniques, a very popular
methodology is the LSB (Least Significant Bit) algorithm, which replaces the least significant bit in some bytes of the cover file to hide a sequence of bytes containing the hidden data. That is usually, an effective technique in cases where the LSB substitution does not cause significant quality degradation (such as in 24-bit bitmaps).

4.1 DCT-LSB (Discrete Cosine Transformation-List Significant Bit Encoding): DCT-LSB is a Steganographic method is a substitution algorithm used for hiding information behind Video files. Each frame in the video holds a part of the secret message. Discrete Cosine Transform (DCT) transforms successive $8 \times 8$ pixel blocks of the frame into 64 DCT coefficients each. The DCT coefficients $D(i, j)$ of an $8 \times 8$ block of image pixels $p(x, y)$ are given by the formula below

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i) C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos \left( \frac{(2x+1)i\pi}{2N} \right) \cos \left( \frac{(2y+1)j\pi}{2N} \right)$$

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{if } u > 0 \end{cases}$$

Least Significant Bit (LSB) is a simple Steganographic method that takes the individual pixels of the frame and replaces the least significant bits with the secret message bits. It is by far the most popular of the coding techniques used. Figure 4.2 shows the process of LSB algorithm.
We can commandeer the least significant bit of 8-bit true color image to hold each bit of our secret message by simply overwriting the data that was already there. The impact of changing the least significant bit is almost imperceptible.

**Input**: message, cover image  
**Output**: steganographic object containing message

```
while data left to embed do
    get next DCT coefficient from cover file
    if DCT ≠ 0 and DCT ≠ 1 then
        get next bit from the Secret message
        replace DCT LSB with message bit
    end if
    insert DCT into steganographic object
end while
```

**Figure 4.2: LSB Process**

**Figure 4.3: Embedding Process of DCT-LSB**
Figures 4.3 and 4.4 taken from [7], gives algorithms for embedding and extracting secret information in video files using DCT-LSB algorithm respectively.

**Steganography vs. Cryptography**

Steganography and Cryptography are parallel data security techniques, both can be implemented side by side but, they differ in certain qualities like

- Steganography can use cryptography but not vice versa.
- Steganography has a very expensive payload as compared to cryptography.
- Cryptography makes the message “unreadable” where as Steganography makes it “unseen”.

Steganography implemented to cryptographic data will increase the security of the data communication.
Chapter 5

DEFLATE COMPRESSION ALGORITHM

DEFLATE is a lossless compressed data format that compresses data using a combination of the LZ77 algorithm and Huffman coding.

- Is independent of CPU type, operating system, file system, and character set
- Compatible with widely used gzip utility
- Worst case 5 bytes per 32K byte block

Each block consists of two parts:

A pair of Huffman code trees that describe the representation of the compressed data part and a compressed data part (The Huffman trees themselves are compressed using Huffman encoding.) [8].

The compressed data consists of a series of elements of two types:

Literal bytes (of strings that have not been detected as duplicated within the previous 32K input bytes),

Pointers to duplicated strings, where a pointer is given as a pair <length, backward distance>[8].

Distance: max of 32K bytes

Length: 258 bytes

Literals, distances, and lengths in the compressed data are represented using a Huffman code (one code tree for literals and lengths and a separate code tree for distances)

The code trees for each block appear in a compact form just before the compressed data for that block
Compressed block format:

Header: BFINAL (1 bit) | BTYPE (2 bits)

BFINAL = 1 (last block of the data set)

BTYPE: 00 - no compression

  01 - Compressed with fixed Huffman codes

  10 - Compressed with dynamic Huffman codes

  11 - Reserved

Non-compressed blocks (BTYPE=00)

Any bits of input up to the next byte boundary are ignored and the rest of the block consists of the following information:

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
\hline
\text{LEN} & \text{NLEN} & \ldots \text{LEN bytes of literal data...}
\end{array}
\]

LEN is the number of data bytes in the block. NLEN is the one's complement of LEN [8].
Chapter 6

PROPOSED SYSTEM

To add multiple layers of security it is a good practice to use both Cryptography and Steganography together. Steganographic algorithms implemented to cryptographic data makes communication more secure.

6.1 Proposed System: The application first compresses the document with secret message, and then encrypts the compressed file and uses the resulted file as the secret message to hide in the harmless message generating a Stego-object. The intended receiver de-embeds decrypts and decompresses the Stego-object respectively to get the hidden message.

6.2 Modules of the Application: The application has two modes of operation i.e. Sender and Receiver.

The three major modules for Sender mode of application are

Compression: The application first compresses the document to be transferred

Encryption: An Encryption algorithm encrypts the compressed file and the resulted file is used as secret message.

Embedding: The encrypted file is hidden in the Harmless Message (video file) using corresponding Steganographic algorithm, which generates a Stego Object, which is sent to the intended recipient.
The three major modules for the Receiver mode of application are

**De-Embedding:** The Stego Object is de-embedded generating an encrypted file.

**Decryption:** The encrypted file is decrypted using an the Encryption algorithm, and the resulted file is given to the compression module

**De-Compression:** The application then de-compresses the document and we have the Secret message.

Steganography and cryptography are closely related. “Cryptography scrambles messages so they cannot be understood” Whereas, “Steganography will hide the message so there is no knowledge of the existence of the message” [9]. Sending an encrypted message will arouse suspicion while an invisible message will not do so. The application developed in this project combines both sciences to produce better protection of the message. Even if the Steganography fails since the message is in encrypted form it is of no use for the third party, hence the information is secure.
Chapter 7

STEGANALYSIS

Steganographic techniques have succeeded for centuries. However, “since secret information usually has a value to the ones who are not allowed to know it, there are people or organizations who try to decode encrypted information or find information that is hidden from them” [9]. Even though the hiding algorithms are ahead advanced, the techniques to find the hidden information also grow.

7.1 Steganalysis:

Most Steganographic techniques involve altering properties of the cover source like video files and there are many ways of detecting these alterations. The process of detecting steganographically embedded hidden messages in digital data like Audio and a video file is known as Steganalysis. “Steganalysis the science utilized to disrupt the transmission of Steganographic encrypted messages, through detection, extraction, disabling or destruction of such hidden information” [9]. Steganalysis takes advantage of statistical or perceptual distinction of Stego object from the original harmless message like Audio, video files etc.

7.2 Attacks on Steganography

Two aspects of attacks on Steganography are detection and destruction of the embedded message.
7.2.1 Detection: Most Steganographic techniques involve changing properties of the original harmless messages like Image and Video files and the detection algorithms concentrate on detecting these changes [10]. Detecting the existence of a hidden message will save time in the message elimination phase by processing only those digital files that contain hidden information. Detecting an embedded message defeats the primary goal of Steganography techniques that is concealing the very existence of a message [10]. The algorithms vary in their approaches for hiding information. Without knowing which algorithm is used and which Stego-key is used, detecting the hidden information is quite complex.

7.2.2 Destruction or Defeating algorithms concentrate on removing the hidden messages from the Stego object [10].

Steganalysis techniques are similar to the cryptanalysis for the cryptography methods. As we have discussed in the previous chapters

\[ \text{Harmless Message} + \text{secret message} + \text{stego-key} = \text{stego-object} \]

Some of the known attacks for the Steganography are stego-only, known cover, known message, chosen stego, and chosen message.

A stego-only attack is similar to the ciphertext only attack where only the stego-object is available for analysis. If the "original" Harmless message and stego-object are both available, then a known cover attack is available [11].
The steganalysis may use a **known message** attack i.e attacker may attempt to analyze the stega-object for future attacks. Even with the message, this may be very difficult and is equivalent to the stego-only attack [11]. The **chosen stego** attack is one where the Steganography tool or algorithm, stego-object are known.

A **chosen message** attack is one where the steganalyst generates stego-object from some Steganographic tool or algorithm from a known message. The goal in this attack is to determine corresponding patterns in the Stego object that may point to use specific Steganographic algorithms [11]. The compression a technique works either when a file is resized or the color palette is altered. We can also change the image format using a different compression technique to remove the hidden message. “Steganos and Stools use LSB embedding in the spatial domain, while Jsteg embeds in the frequency domain. Other more sophisticated techniques include the use of quantization and dithering”.

Chapter 8

UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. This object-oriented system notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation. UML model abstracts the essential details of the underlying problem from the usually complicated software system. UML provides with nine modeling diagrams i.e. Use case diagrams, Class diagrams, Sequence diagrams, Collaboration diagrams, State chart diagrams, Activity diagrams, Component diagrams, Deployment diagrams. In order to make it easy for the viewer to understand the blueprint of the project, we have made use of use-case and sequence diagrams [13].

8.1 Use-Case Diagrams:

Use case diagrams describe what a system does from the point of an external observer. The emphasis is on what a system does rather than how. A scenario is an example of what happens when someone interacts with the system. Use-case diagrams use a scenario to model a system. Use-case diagrams have actors, use-cases and the relations among the actors and the use-cases.

Actor: Specifies the persons involved in the scenario.
Use-case: Specifies the function of a particular task.

Use-case diagrams have the following relationships between the actors and use-cases.

**Generalization:** Specifies parent-child relationship.

**Association:** Specifies cardinality.

**Use Case Diagrams for application:**

Figure 8.1.1: Use Case Diagram for Sender Module of the Application
Figure 8.1.1 is the use case diagram for the sender module of the developed application.

Compress, encrypt and Embed are the three use cases for the sender module and their functionality is given below.

**Use Case Name: Compress**

**Actors:** Sender

**Entry Condition:** User must select the file (with .doc extension).

**Exit Condition:** Successful or Un Successful Compression of file with appropriate error messages

**Events:** The user selected file will be compressed by deflate compression algorithm

**Use Case Name: Encrypt**

**Actors:** Sender

**Entry Condition:** User must select the file and should provide a key.

**Exit Condition:** Successful or Un Successful Encryption of file with appropriate error messages

**Events:** The user selected file will be encrypted by Tiny Encryption algorithm

**Use Case Name: Embed**

**Actors:** Sender

**Entry Condition:** User must select the Video file and the secret message file

**Exit Condition:** Successful or Un Successful Embedding of file with appropriate error messages
**Events:** The user secret message file is embedded into the Video file by DCT-LSB Steganographic algorithm

---

**Figure 8.1.2: Use Case Diagram for Receiver Module of the Application**

Figure 8.1.2 is the use case diagram for the sender module of the developed application. De-Compress, Decrypt and De-Embed are the three use cases for the sender module and their functionality is given below.

**Use Case Name: De-Embed**

**Actors:** Receiver

**Entry Condition:** User must select the received Video file

**Exit Condition:** Successful or Un Successful De-Embedding of file with appropriate error messages
Events: The user selected Video file will be De-embedded by DCT-LSB Steganographic algorithm.

Use Case Name: Decrypt

Actors: Receiver

Entry Condition: User must select the file and should provide a key.

Exit Condition: Successful or Un Successful Decryption of file with appropriate error messages.

Events: The user selected file will be decrypted by Tiny Encryption algorithm.

Use Case Name: De-Compress

Actors: Receiver

Entry Condition: User must select the compressed file.

Exit Condition: Successful or Un Successful De-Compression of file with appropriate error messages.

Events: The user selected file will be De-compressed by deflate compression algorithm.

8.2 Sequence Diagrams:

Sequence diagrams describe interactions among classes (Class roles describe the way an object will behave in context) in terms of an exchange of messages over time.

A Sequence diagram consists of two major behavioral elements:
**Object:** The primary element involved in a sequence diagram is an Object. Object is an instance of a class. An object is represented by a named rectangle. The name before “:” is the Object name and the name after “:” is the Class name [13].

<object name>: <class name>

**Message:** The interaction between different objects in a sequence diagram is represented by messages. A “directed arrow” denotes a message.

<------------------------> Represents a “message”

<--------------------------> Represents “return “

### 8.3 Other Unified modeling diagrams:

**Class Diagrams:** A class diagram gives an overview of a system by showing its classes and the relationships among them. Class diagrams are static – they display what interacts but not what happens when they do interact.

**Collaboration Diagrams:** Collaboration diagrams are also interaction diagrams. They convey the same information as sequence diagrams, but they focus on object roles instead of the time of message exchange.

**State Chart Diagrams:** Objects have behaviors and state. The state of an object depends on its current activity or condition. A state chart diagram shows the possible states of the object and transitions that cause a change in state.
**Activity Diagrams:** An activity diagram is essentially a fancy flowchart. Activity diagrams and state chart diagrams are related. While a state chart diagram focuses attention on an object undergoing a process (or on a process of object), an activity diagram focuses on the flow of activities involved in a single process. The activity diagram shows how those activities depend on one another [13].

**Component Diagrams:** A component is a code module. Component diagrams are physical analogs of class diagram.

**Deployment Diagrams:** Deployment diagrams show the physical configurations of software and hardware.
Chapter 9

USER GUIDE FOR THE SYSTEM

The main window of the Combining Cryptography and Steganography for Secure Communication has three main menus as given below

1. Step 1: Compression Options
   - Compress
   - De-Compress
   - Exit

2. Step 2: Cryptography Options
   - Encrypt
   - Decrypt

3. Step 3: Steganography Options
   - Embed
   - De-Embed

4. Help: How to use the tool
The Application provides with a Help menu shown in figure 9.2, which gives instructions to the user on how to use the tool.

![Help menu](image)

**Figure 9.1 : How to Use the System**
Steps for sender to hide the secret message in a Video file

Step1:- Compress the file

**Input:** File with .txt or .doc extension

**Output:** Generates a file with “cmp” extension

![Image of Compressing File](image1.png)

Figure 9.2: Compress the File Step-1

![Image of Compressed File](image2.png)

Figure 9.3: Compress the File Step-2
Figure 9.2 and Figure 9.3 gives the steps for compressing the file. Select the file to be compressed and hit Compress button. “File Compressed Successfully” message box will pop up. If the compression is not successful appropriate errors messages pop up.

**Step 2: Encrypt the Compressed file**

**Input:** File with “cmp” extension

**Output:** Generates a file with “enc” extension

![Encrypt the File Step-1](image)

Select the compressed file and hit the Encrypt button in the Encryption window as shown in figure 9.4. User will be asked to enter the Key (i.e, password) for encryption as shown in figure 9.5.
Appropriate errors messages will displayed if the encryption is not successful.

**Step3:- Embed**

**Input:** File with “enc” extension and the Videos file

**Output:** Generates a video file with data embedded
Select the encrypted file, the video file and hit the Embed button in the window as shown in figure 9.6. “Embed Process Completed” message box will pop up if the Embed process goes successfully shown in figure 9.7. If the Embed process is not successful appropriate errors messages pop up. Now the resulted video file is transferred to the intended receiver.

**Steps for the intended receiver to get the hidden data**

**Step1:- De-Embed**

**Input:** A Video file

**Output:** Generates a file with “enc” extension
Select the video file and hit the De-Embed button in the window. “De-Embed Process Completed” message box will pop up if the De-Embed process goes successfully. Appropriate errors messages are displayed if the De-Embed process is not successful. Figure 9.8 and 9.9 gives the steps to de-embed the file.
Step2:- Decrypt the encrypted file

**Input:** File with “enc” extension

**Output:** Generates a compressed file with “cmp” extension

Select the file with “enc” extension and hit the Decrypt button in the window as shown in figure 9.10. User will be asked to enter the Key (i.e. password) as shown in figure 9.11. If there is a password mismatch, the application will display unauthorized user error message.
If the decryption is not successful appropriate errors messages will be displayed.

Figures 9.10, 9.11 and 9.12 shows the step-by-step implementation of the system to decrypt a file.
Step3: De-Compress the file

**Input:** File with “cmp” extension

**Output:** Generates a file with .txt or .doc extension

Select the file with “cmp” extension and hit the De-Compress button in the window as shown in figure 9.13.

![De-compress the File Step-1](image)

Figure 9.13: De-compress the File Step-1

![De-compress the File Step-2](image)

Figure 9.14: De-compress the File Step-2
“Decompression Successful” message box will pop up if the De-compression process is successful as shown in figure 9.14 and the secret message i.e. word document is generated. If the De-compression process is not successful, appropriate errors messages are displayed.
Chapter 10

CONCLUSION

Steganography is an effective way to obscure data and hide sensitive information. The effectiveness of Steganography is amplified by combining it with cryptography. By using the properties of the DCT-LSB Steganography algorithm for video file and combining it with the TEA cryptography standards, we developed a method, which adds layers of security to the communication. Steganographic methods do not intended to replace cryptography but supplement it.

The strength of our system resides in adding multiple layers of security. First the secret message i.e. word document to be transferred is compressed, encrypted and then embedded in a video file using Steganographic algorithm hence, adding three layers of security. The weakness of the system developed is the size of the secret file i.e. word document after compression should be less than the size of the Cover object i.e. Video file. Since we are using compression algorithm this happens only for huge documents.

As future work, we intend to study more steganalytic techniques i.e. detecting whether a particular file contains any form of embedding or not. We also plan to extend our system so that it can hide digital files in other digital files, for example hiding Audio files in Videos files etc.
REFERENCES


