VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgavi -590014, Karnataka State, India

A PHASE-2 PROJECT REPORT
ON

"AN EFFICIENT MSB PREDICTION BASED METHOD FOR HIGH CAPACITY REVERSIBLE DATA SECURITY IN ENCRYPTED IMAGES"

Submitted in partial fulfilment of the requirement for the award of the degree

BACHELOR OF ENGINEERING
IN
INFORMATION SCIENCE AND ENGINEERING

Submitted by

DEEPTHI S	**1SJ15IS018**
LIKHITHA D	**1SJ15IS042**
NETRA L	**1SJ15IS058**
NISHA N	**1SJ15IS059**

Under the guidance of
Prof. Chandrashekar J M
Assistant Professor
Dept. of ISE, SJCIT

S J C INSTITUTE OF TECHNOLOGY
Department of Information Science and Engineering
Chickballapur – 562101

2018-19

||Jai Sri Gurudev||

Sri Adichunchanagiri Shikshana Trust®

S.J.C INSTITUTE OF TECHNOLOGY

Information Science & Engineering Department

Chickballapur-562101

CERTIFICATE

Certified that the project work entitled "**An Efficient MSB Prediction Based Method For High Capacity Reversible Data Security In Encrypted Images**" carried out by **DEEPTHI S (1SJ15IS018), LIKHITHA D(1SJ15IS042), NETRA L(1SJ15IS058), NISHA N (1SJ15IS059)** in partial fulfilment for the award of **Bachelor of Engineering in Information science and Engineering in Eighth semester of the Visvesvaraya Technological University**, Belagavi during the year **2019.** It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

.....	.....	.....
Signature of the Guide 17/06/19	**Signature of HOD** 17.6.19	**Signature of Principal** 17/06/19.
Mr.Chandrashekhar J M	**Prof. Satheesh Chandra Reddy**	**Dr. RaviKumar K M**
Assistant Professor	**HOD**	**Principal** *Principal*
Dept. of ISE, SJCIT.	**Dept. of ISE, SJCIT.**	**SJCIT** S.J.C. Institute of Technol
	Prof & Head Department of Information Science & Engg SJC Institute of Technology Chickballapur-562101	Chickballapur - 562 10

External Viva:

Name of Examiners	**Signature with date**
1) Satheesh Chandra Reddy	17.6.19
2) Karitha C	17/6/19

ABSTRACT

Reversible data hiding in encrypted images (RDHEI) is an effective technique to embed data in the encrypted domain. An original image is encrypted with a secret key and during or after its transmission, it is possible to embed additional information in the encrypted image, without knowing the encryption key or the original content of the image. During the decoding process, the secret message can be extracted and the original image can be reconstructed. In the last few years, RDHEI has started to draw research interest.

Indeed, with the development of cloud computing, data privacy has become a real issue. However, none of the existing methods allows us to hide a large amount of information in a reversible manner. In this paper, we propose a new reversible method based on MSB (most significant bit) prediction with a very high capacity.

We present two approaches, these are: high capacity reversible data hiding approach with correction of prediction errors (CPEHCRDH) and high capacity reversible data hiding approach with embedded prediction errors (EPE-HCRDH). With this method, regardless of the approach used, our results are better than those obtained with current state of the art methods, both in terms of reconstructed image quality and embedding capacity.

ACKNOWLEDGEMENT

With Great pride I would like to convey my gratitude and appreciation to our **"S.J.C Institute of Technology"** for giving us the required platform for the fulfillment of the project on Credit card fraud detection using majority voting and adaboost as per the V.T.U requirements, for the 8th semester.

We express my sincere thanks to **Dr. RAVIKUMAR K M**, Principal of **SJCIT**, for providing me with excellent infrastructure to complete the project.

We express wholehearted gratitude to **Prof. SATHEESH CHANDRA REDDY**, Associate Professor and **HOD of Information Science and Engineering Department**. We wish to acknowledge his help in making my task easy by providing me with his valuable help and encouragement.

We are extremely thankful to our project coordinator **Prof. Bhanumathi S**, Assistant Professor, Department of ISE, for the gudience, support and advice during the course of the project.

It is my pleasure to thank my guide **Mr.Chandrashekhar J M,** Assistant Professor, **Department of Information Science and Engineering,** SJCIT for their guidance, encouragement and valuable suggestion from the beginning of the project work till the completion without which this project work would not have been accomplished.

We also thank all those who extended their support and co-operation while bringing out this project.

DEEPTHI S	**1SJ15IS018**
LIKHITHA D	**1SJ15IS042**
NETRA L	**1SJ15IS058**
NISHA N	**1SJ15IS059**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Digital image security plays a significant role in all fields, especially in highly confidential areas like the military and medical worlds. With the development of cloud computing, the growth in information technology has led to serious security problems where confidentiality, authentication and integrity are constantly threatened, by illegal activities like hacking, copying or malicious use of information.

The aim of encryption methods is to guarantee data privacy by fully or partially randomizing the content of original images. During the transmission or the archiving of encrypted images, it is often necessary to analyze or to process them without knowing the original content, or the secret key used during the encryption phase.

In particular, methods of reversible data hiding in theencrypted domain (RDHEI) have been designed for data enrichmentand authentication in the encrypted domain, when the encryption phase is necessarily done in the first place as, forexample, in a cloud computing scenario. Without knowing theoriginal content of the image or the secret key used to encryptthe image, it is then possible to embed a secret message inthe encrypted image.

During the decoding phase, the originalimage must be perfectly recoverable and the secret messagemust be extracted without error. Therefore, there exists atrade-off between the embedding capacity and the qualityof the reconstructed image. In recent years, many methodshave been designed. The space to embed the message may bevacated after or before the encryption phase and, during thedecoding phase, image reconstruction and data extraction canbe processed at the same time.

1.2 OBJECTIVE

- Reversible data hiding in encrypted images (RDHEI) is an effective technique to embed data in the encrypted domain. An original image is encrypted with a secret key and during or after its transmission, it is possible to embed additional information in the encrypted image, without knowing the encryption key or the original content of the image. During the decoding process, the secret message can be extracted and the original image can be reconstructed.

- The main aim of this system is to present two approaches, these are: high capacity reversible data hiding approach with correction of prediction errors (CPEHCRDH) and high capacity reversible data hiding approach with embedded prediction errors (EPE-HCRDH).

Idea of reversible data hiding in encrypted images (RDH-EI) originates from reversible data hiding (RDH) in plaintext images . It is feasible in the applications like cloud storage and medical systems. In cloud storage, a content owner can encrypt an image to preserve his/her privacy, and upload the encrypted data onto cloud. On the cloud side, when managing huge amount of encrypted images, an administrator can embed additional messages (*e.g.*, labels, time stamps, category information, etc.) into the ciphertext. This embedding not only saves the storage overhead, but also provides a convenient way of searching encrypted images. On the recipient side, when a user downloads the encrypted data containing additional messages from the server, he/she can losslessly recover the original images after decryption.

1.3 SCOPE OF THE PROJECT

In particular, methods of reversible data hiding in the encrypted domain (RDHEI) have been designed for data enrichment and authentication in the encrypted domain, when the encryption phase is necessarily done in the first place as, for example, in a cloud computing scenario. Without knowing the original content of the image or the secret key used to encryptthe image, it is then possible to embed a secret message in the encrypted image.

During the decoding phase, the original image must be perfectly recoverable and the secret message must be extracted without error. Therefore, there exists a trade-off between the embedding capacity and the quality of the reconstructed image. In recent years, many methods have been designed. The space to embed the message may be vacated after or before the encryption phase and, during the decoding phase, image reconstruction and data extraction can be processed at the same time [17], [27] or separately [12], [27], [28].

In all cases, the presented methods are not able to propose a high embedding rate together with a very good reconstructed image quality. In [12], the payload can be high (0.5 bpp), but the reconstructed image is altered when compared to the original (PSNR $\approx$ 40 dB). Moreover, other methods, such as Wu and Sun's version, propose a "high" embedding capacity, but it is only possible to embed approximately 0.1 bit per pixel at most [27]. Furthermore, in many of the existing methods, data hiding is made by LSB (least significant bit) substitution. However, in the encrypted domain, it is difficult to detect if an image contains a hidden message or not because pixels have pseudorandom values. For this reason, we propose to substitute the MSB (most significant bit) values instead of the LSB values. In fact, in the clear domain, MSB prediction is easier than LSB prediction and in the encrypted domain, confidentiality remains the same. Moreover, we do not need to preserve the high quality of the encrypted image compared to the clear domain.

1.4 ORGANIZATION OF REPORT

In this paper, we present a new high capacity reversible data hiding scheme for encrypted images based on MSB prediction. Due to the local correlation between a pixel and its neighbors in a clear image, two adjacent pixel values are very close. For this reason, it seems natural to predict a pixel value by using already decrypted previous ones, as in many methods of image coding and compression. However, in some cases, there are some errors. So, the first step of our method consists of identifying all the prediction errors in the original image and to store this information in an error location binary map (note that using overhead such an additional map is not necessary for our proposed method).

After that, we propose two different approaches: the CPEHCRDH (high-capacity reversible data hiding with correction of prediction errors) and the EPE-HCRDH (high-capacity reversible data hiding with embedded prediction errors). The CPE-HCRDH approach consists of correcting the prediction errors (CPE) before encryption. According to the error location map, the original image is pre-processed in order to avoid all the prediction errors and then, the pre-processed image is encrypted. In the EPE-HCRDH approach, the original image is directly encrypted, but after the encryption step, the location of the prediction errors is embedded (EPE). During the data hiding phase, in both approaches, the MSB of each available pixel is substituted in the encrypted image by a bit of the secret message. At the end of the process, the embedded data can be extracted without any errors and the clear image can be reconstructed losslessly by using MSB prediction.

The rest of the paper is organized as follows. Section II gives an overview of related work on reversible data hiding in encrypted images. Then, the proposed method is described in detail in Section III. Experimental results and analysis are provided in Section IV. Finally, the conclusion is drawn and future work is proposed in Section V.

CHAPTER 2
LITERATURE SURVEY

CHAPTER-2

LITERATURE SURVEY

Reversible data hiding (RDH) is particularly suitable for authentication and data enrichment. It consists of embedding a hidden message into an image. At the end of the process, it is possible to extract the secret message and to recover losslesslythe original image. Methods are based on losslessany errors and the clear image can be reconstructed losslessly by using MSB prediction.

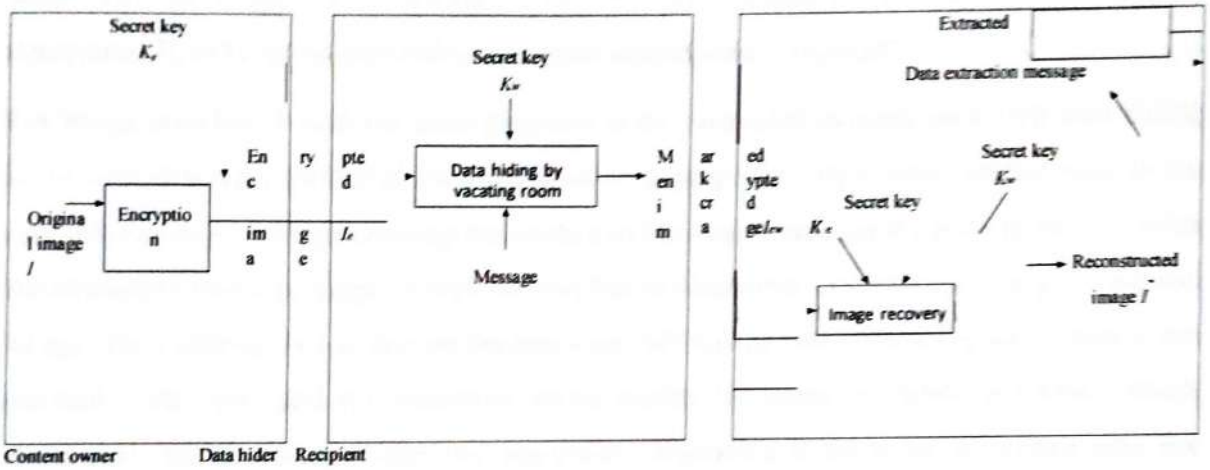


Fig 2.1 VRAE

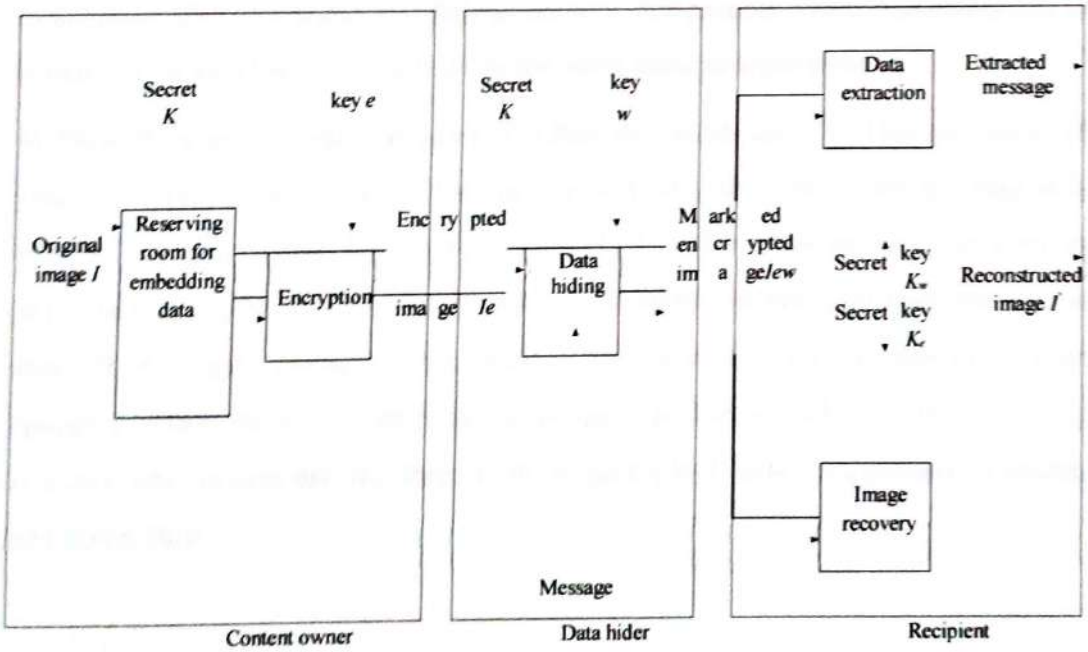


Fig 2.2 RRBE

Compression appending, difference expansion [23], [24], histogram shifting [5], [15], [22] or a combination of these schemes [16], [21]. Also, by randomizing the content of an original image, encryption provides in particular visual confidentiality. Cryptosystems can be divided into two groups according to the method used: block cipher, or stream cipher [25]. Furthermore, encryption can be selective, when only certain details are hidden in the encrypted image [9], [18], [26], or fully when the global meaning of the image is kept entirely secret [13]. Sometimes, it is necessary to be able to analyze or process encrypted images without knowing the original content, or the secret key used during the encryption phase. Many applications exist, such as visual secret sharing (VSS) [3], [14], research and indexing in encrypted databases [7], [11] or recompression of crypto-compressed images [8].

For image notation or authentication purposes in the encrypted domain, reversible data hiding in encrypted images (RDHEI) methods have been proposed. They allow embed data in the encrypted domain without knowing the content of the clear image nor the encryption key. After the extraction of the message, it must be possible to reconstruct without distorting the original image. The challenge lies in finding the best trade-off between the embedding rate – also called payload – (in *bpp*), and the recovered image quality (in terms of PSNR or SSIM). These techniques can be classified into two categories, depending if the room is vacated after the encryption phase (VRAE) or reserved before image encryption (RRBE), as presented in Fig. 1. In addition, encryption and data hiding can be a joint process, when data extraction and original image reconstruction are completed at the same time, or separately.

In 2008, Puech*et al.* proposed one of the first joint methods [17]. They encrypted the original image by using AES and, after that, they embedded a bit of the secret message at a randomly selected position in each block of 4 × 4 pixels. In order to reconstruct the cover image, they performed an analysis of the local standard deviation. In this approach, the payload is quite small (0.0625 *bpp*). Zhang, in [30], suggests encrypting the original image with a simple XOR operation. Then, the encrypted image is divided into blocks and each of them was partitioned into two sets. In one set, the three LSB of each pixel were compressed to vacate room for additional data.

During the decoding phase, the block smoothness was observed to recover the original information and to extract the message. Hong *et al.* improved this approach by using a side match technique and an advanced formula to smoothness evaluation [6]. However, the reconstructed image quality remains of poor quality (with globally a PSNR less than 30 *dB*) when the payload is high. Zhou *et al.* designed a joint method where the image encryption was partial [33]. After the encryption phase, they used a public key modulation mechanism to embed additional data, without any access to the encryption key. To reconstruct the original image, they have to know which blocks of the image have been encrypted by using a SVM classifier.

Ma *et al.* were the first to describe a RRBE technique [12]. They proposed to release a part of the original image by applying a RDH method of histogram shifting. After that, they encrypted the image and then inserted information by substituting some LSB values in the encrypted image. With this method, the payload is higher than in previous methods (0.5 *bpp*) but the reconstructed image is altered when compared with the original (PSNR close to 40 *dB*). Zhang *et al.* analyzed the prediction errors (PE) of some pixels and made space to hide data by using PE-histogram shifting before image encryption [29]. Zhang designed a separable method, where a part of the encrypted image was compressed to vacate room for the message embedding [31]. In this case, data extraction can be done before or after image decryption. In [28], Xu and Wang propose a new method based on histogram shifting and difference expansion. They used a stream cipher during the encryption phase and designed a specific encryption mode in order to encrypt the interpolation-error. In [2], Cao *et al.* propose a sparse coding technique. By exploiting the local correlation between pixels, they could vacate a large space to hide information. Qian and Zhang, in [20], described a method based on distributed source coding (DSC). They first encrypted the original image with a stream cipher and, after that, they compress some bits of the MSB planes to make room for the secret data. In [32], Zhang *et al.* encrypted the cover image by using public key cryptography with probabilistic and homomorphic properties. After the encryption phase, they embed data in the LSB planes of the encrypted pixels. During the decoding phase, as the introduced distortion was quite low, the embedded data is extracted and the original image was recovered losslessly.

In [27], Wu and Sun propose an advanced method, developed in two ways. The first approach is joint. They encrypted the original image in the same way as Zhang in [31] and, according to a data hiding key, selected some pixels to conceal data by histogram shifting. The second approach is separable: they hid bits of the secret message by MSB substitution. During the decoding phase, a median filter is applied on the marked image. Although the embedding capacity of this scheme was described as high, it is only possible to embed0.1563 *bpp*at most.

CHAPTER 3
ANALYSIS

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In all cases, the presented methods are not able to propose ahigh embedding rate together with a very good reconstructed image quality. In, the payload can be high (0:5 bpp),but the reconstructed image is altered when compared to theoriginal (PSNR _ 40 dB). Moreover, other methods, such asWu and Sun's version, propose a "high" embedding capacity,but it is only possible to embed approximately 0:1 bit per pixel at most. Furthermore, in many of the existing methods,data hiding is made by LSB (least significant bit) substitution.However, in the encrypted domain, it is difficult to detect ifan image contains a hidden message or not because pixelshave pseudorandom values.

3.1.1 DISADVANTAGE

None of the existing methods succeed in combining high embedding capacity (near 1 bpp) and high visual quality (greater than 50 dB). In most cases, the methods based on prediction error analysis (PE) or using a histogram shifting technique, the LSB values of some pixels are replaced to hide bits of the secret message. However, if an image is encrypted, it is difficult to detect if it contains a hidden message or not. In fact, the pixel values of an encrypted image are pseudo randomly generated.

3.2 PROPOSED SYSTEM

We propose two different approaches: the CPEHCRDH(high-capacity reversible data hiding with correctionof prediction errors) and the EPE-HCRDH (high-capacityreversible data hiding with embedded prediction errors). The CPE-HCRDH approach consists of correcting the predictionerrors (CPE) before encryption. According to the error locationmap, the original image is pre-processed in order to avoidall the prediction errors and then, the pre-processed image isencrypted. In the EPE-HCRDH approach, the original imageis directly encrypted, but after the encryption step, the locationof the prediction errors is embedded (EPE). During the datahiding phase, in both approaches, the MSB of each availablepixel is substituted in the encrypted image by a bit of thesecret message. At the end of the process, the embedded datacan be extracted without any errors and the clear image canbe reconstructed losslessly by using MSB prediction.

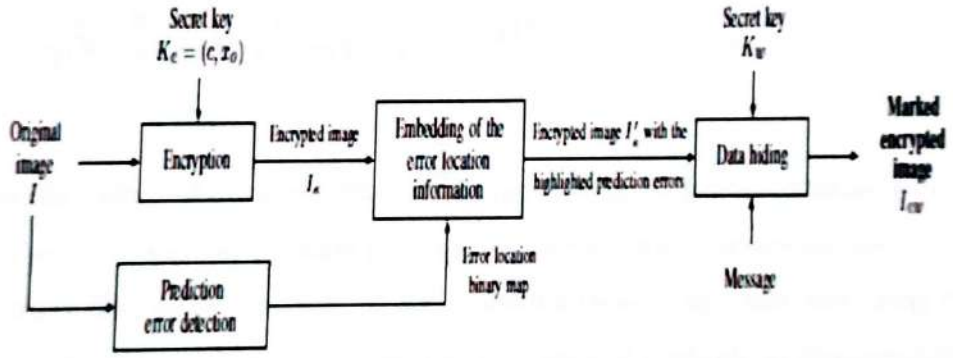


Figure 3.1 EPE-HCRDS approach encoding phase

The encoding phase includes three main steps, these are: the MSB prediction error detection, the joint MSB error consideration and encryption, and the data hiding by MSB substitution. This process is shown in Fig. 2. The goal of our proposition is that an original image I, with $m \times n$ pixels, could be encrypted by using a secret key K_e and that another person could embed a message by using a data hiding key K_w, without knowing K_e. After this process, we obtain a marked encrypted image I_{ew}, which has exactly the same size as the original image.

hat since the encryption phase is fully reversible without overflow, it is then possible er the clear image without any alteration. Moreover, we also observe that even if we otic generator in our method, it is quite possible to generate a pseudo-random sequence yptographically secure pseudo-random number generator (CSPRNG), or for example e AES algorithm in OFB mode. The only requirement is to use a stream cipher during yption phase. .

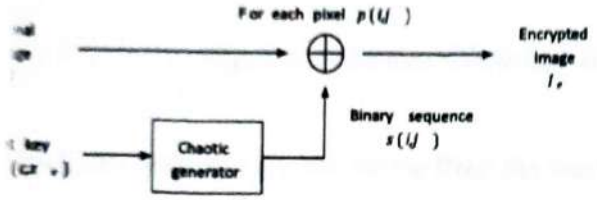


Fig 3.2 : Encryption Step

Data embedding: In the data embedding phase, it is possible to embed data in the encrypted image without knowing either the encryption key $K_e$used during the previous step or the original content of the image. By using the data hiding key K_w, the to-be-inserted message is first encrypted in order to prevent its detection after embedding in the marked encrypted image. Next, pixels of the encrypted image are scanned from left to right, then from top to bottom (scan line order) and the MSB of each available pixel is substituted by one bit b_k, with $0 \leq k < m \times n$, of the secret message:

$$p_{ew}(i,j) = b_k \times 128 + (p_e(i,j) \bmod 128). \quad (3)$$

te that only the first pixel cannot be marked because its value is not predictable, thus its e must not be changed.

Data extraction and image recovery: For the decoding phase, since our method is separable, we can extract the secret message and reconstruct the clear image I' separately. I' may be exactly like the original image I itself or a processed image I^0 very similar to the original image, depending upon which approach is used. There are three possible outcomes:

) the recipient has only the data hiding key K_w, 2) the recipient has only the encryption ey K_e, 3) the recipient has both keys.

Note that since the encryption phase is fully reversible without overflow, it is then possible to recover the clear image without any alteration. Moreover, we also observe that even if we use a chaotic generator in our method, it is quite possible to generate a pseudo-random sequence with a cryptographically secure pseudo-random number generator (CSPRNG), or for example to use the AES algorithm in OFB mode. The only requirement is to use a stream cipher during the encryption phase. .

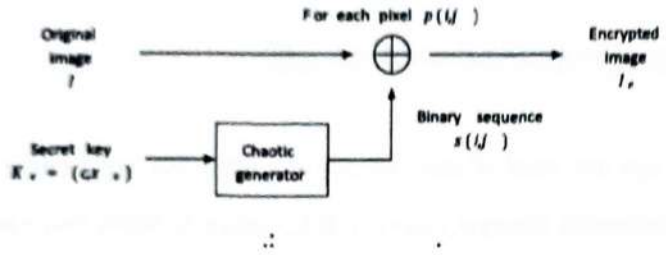


Fig 3.2 : Encryption Step

2) **Data embedding:**In the data embedding phase, it is possible to embed data in the encrypted image without knowing either the encryption key $K_e$used during the previous step or the original content of the image. By using the data hiding key K_w, the to-be-inserted message is first encrypted in order to prevent its detection after embedding in the marked encrypted image. Next, pixels of the encrypted image are scanned from left to right, then from top to bottom (scan line order) and the MSB of each available pixel is substituted by one bit b_k, with $0 \leq k < m \times n$, of the secret message:

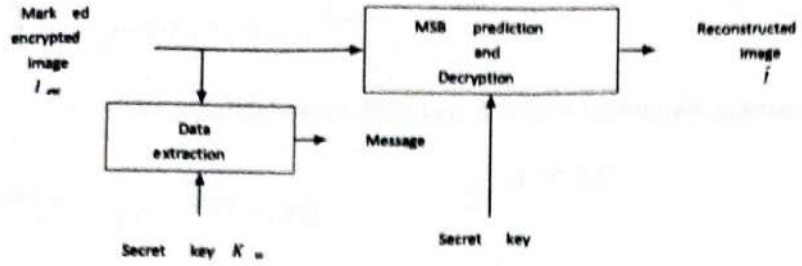
$$p_{ew}(i,j) = b_k \times 128 + (p_e(i,j) \bmod 128). \quad (3)$$

Note that only the first pixel cannot be marked because its value is not predictable, thus its value must not be changed.

3) **Data extraction and image recovery:**For the decoding phase, since our method is separable, we can extract the secret message and reconstruct the clear image I' separately. I' may be exactly like the original image I itself or a processed image I^0 very similar to the original image, depending upon which approach is used. There are three possible outcomes:

1) the recipient has only the data hiding key K_w, 2) the recipient has only the encryption key K_e, 3) the recipient has both keys.

An overview of the decoding method is presented in Fig. 4.



$K_e = (c, x_0)$

Fig. 3.3: Overview of the decoding method.

If the recipient only has K_w, the pixels from the marked encrypted image are scanned in the scan line order and the MSB of each pixel are extracted in order to retrieve the encrypted secret message:

$$b_k = p_{ew}(i,j)/128, \qquad (4)$$

where $0 \leq k < m \times n$ and refers to the index of the extracted bit in the message.

Then, by using the data hiding key K_w, the corresponding plaintext can be obtained.

In the second scenario, if the recipient only has K_e, the image ^{r} can be reconstructed, before the data hiding and the encryption steps, by proceeding as follows:

1) The encryption key K_e is used to generate the sequence $s(i,j)$, with $m \times n$ pseudo-random bytes.

2) The pixels of the marked encrypted image are scanned in the scan line order, and for each pixel, the seven LSB are retrieved by XORing the marked encrypted value $p_{ew}(i,j)$ with the associated binary sequence $s(i,j)$ in the pseudo-random stream:

$$p^{\sim}(i,j) = s(i,j) \oplus p_{ew}(i,j), \qquad (5)$$

where $\oplus$ represents the XOR operation. 3) The MSB value is predicted:

- With the values of the previously decrypted adjacent pixels, the value of the predictor $pred(i,j)$ is computed.
- The pixel value is considered with MSB = 0 and with MSB = 1 and the differences between each of these two values and $pred(i,j)$ are calculated. These values are recorded as Δ^0 and Δ^1:

$$\begin{cases} \Delta^0 = \left| pred(i,j) - \tilde{p}(i,j)^{MSB=0} \right|, \\ \Delta^1 = \left| pred(i,j) - \tilde{p}(i,j)^{MSB=1} \right|. \end{cases} \quad (6)$$

- The smallest value between Δ^0 and Δ^1 gives the searched pixel value:

$$\tilde{p}(i,j) = \begin{cases} \tilde{p}(i,j)^{MSB=0} \\ \tilde{p}(i,j)^{MSB=1}, \text{ else.} \end{cases} \quad (7)' \text{ if } \Delta^0 < \Delta^1,$$

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

In this paper, we develop more the three main steps and give more explanations and justifications. Moreover, we present more results by using a larger database for the experiments and provide a statistical analysis to evaluate the security level of the scheme. The second approach, where the original image is perfectly reconstructed, but where we have to adapt the to-be inserted message, is called high-capacity reversible data hiding approach with embedded prediction errors (EPE-HCRDH).

CHAPTER 4
REQUIREMENT SPECIFICATION

CHAPTER 4

REQUIREMENT SPECIFICATION

4.1 HARDWARE REQUIREMENTS:

1. **System** : Pentium IV 2.4 GHz.
2. **Hard Disk** : 500 GB.
3. **Ram** : 4 GB

4.2 SOFTWARE REQUIREMENTS:

1. **Operating system** : Windows XP / 7
2. **Coding Language** : Java (Jdk 1.7)
3. **Web Technology** : Servlet, JSP
4. **Web Server** : TomCAT 6.0
5. **IDE** : Eclipse Galileo
6. **Database** : My-SQL 5.0
7. **UGI for DB** : SQLyog
8. **JDBC Connection** : Type 4 Driver

4.3 SOFTWARE SPECIFICATION

A **software requirements specification** (SRS) is a description of a software system to be developed Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.[1] Used appropriately, software requirements specifications can help prevent software project failure.

4.3.1 Operating system

The type of the operating system used to implement this project is Windows XP and its features are discussed below

4.3.1.1 Windows XP

Windows XP is a personal computeroperating system produced by Microsoft as part of the Windows NT family of operating systems. It was released to manufacturing on August 24, 2001, and broadly released for retail sale on October 25, 2001.

Development of Windows XP began in the late 1990s as "Neptune", an operating system (OS) built on the Windows NT kernel which was intended specifically for mainstream consumer use. An updated version of Windows 2000 was also originally planned for the business market; however, in January 2000, both projects were scrapped in favor of a single OS codenamed "Whistler", which would serve as a single OS platform for both consumer and business markets. As such, Windows XP was the first consumer edition of Windows not to be based on MS-DOS.

Upon its release, Windows XP received critical acclaim, with critics noting increased performance and stability (especially in comparison to Windows Me, the previous version of Windows aimed at home users), a more intuitive user interface, improved hardware support, and expanded multimedia capabilities. However, some industry reviewers were concerned by the new licensing model and product activation system.

Extended support for Windows XP ended on April 8, 2014, after which the operating system ceased receiving further support or security updates (with exceptional security updates being made e.g. in 2019, to address potential ransomware threats) to most users. As of May 2019, 1.81% of Windows PCs run Windows XP.

4.4 Coding Language

The coding language used in our project is JAVA and its featues are discussed below.

4.4.1 JAVA

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then update the browser's display accordingly

Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags

<Script> ..... </Script>

<Script Language = "JavaScript">

JavaScript statements

</Script>

Here are a few things we can do with JavaScript

- Validate the contents of a form and make calculations.
- Add scrolling or changing messages to the Browser's status line.
- Animate images or rotate images that change when we move the mouse over them.
- Detect the browser in use and display different content for different browsers.

- Detect installed plug-ins and notify the user if a plug-in is required.
- We can do much more with JavaScript, including creating entire application.

JavaScript and Java are entirely different languages. A few of the most glaring differences are:

- Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
- While JavaScript is best suited to simple applications and adding interactive features to Web pages; Java can be used for incredibly complex applications.

There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact they can be used together to combine their advantages.

- JavaScript can be used for Sever-side and Client-side scripting.
- It is more flexible than VBScript.
- JavaScript is the default scripting languages at Client-side since all the browsers supports it.

Java Technology

Initially the language was called as "*oak*" but it was renamed as "*Java*" in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer's language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.
- Finally, Java is to Internet programming where C was to system programming.

IMPORTANCE OF JAVA TO THE INTERNET

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

JAVA CAN BE USED TO CREATE TWO TYPES OF PROGRAMS

Applications and Applets: An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java's ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java –compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

FEATURES OF JAVA SECURITY

Every time you that you download a "normal" program you are risking a viral infection. Prior to java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a "firewall" between a network application and your computer. When you use a java-compatible web browser, you can safely download java applets without fear of virus infection or malicious intent.

4.5 IDE

An integrated development environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of at least a source code editor, build automation tools, and a debugger.

4.5.1 Eclipse

Eclipse was inspired by the Smalltalk-based VisualAge family of integrated development environment (IDE) products. Although fairly successful, a major drawback of the VisualAge products was that developed code was not in a component-based software engineering model. Instead, all code for a project was held in a compressed lump (somewhat like a zip file but in a proprietary format called .data). Individual classes could not be easily accessed, certainly not outside the tool. A team primarily at the IBM Cary NC lab developed the new product as a Java-based replacement. In November 2001, a consortium was formed with a board of stewards to further the development of Eclipse as open-source software. It is estimated that IBM had already invested nearly $40 million by that time The original members were Borland, IBM, Merant, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft, and WebGain The number of stewards increased to over 80 by the end of 2003. In January 2004, the Eclipse Foundation was created

Eclipse 3.0 (released on 21 June 2004) selected the OSGi Service Platform specifications as the runtime architecture.

Eclipse uses plug-ins to provide all the functionality within and on top of the run-time system. Its run-time system is based on Equinox, an implementation of the OSGi core framework specification.

In addition to allowing the Eclipse Platform to be extended using other programming languages, such as C and Python, the plug-in framework allows the Eclipse Platform to work with typesetting languages like LaTeX and networking applications such as telnet and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK, with support for other version control systems provided by third-party plug-ins.

With the exception of a small run-time kernel, everything in Eclipse is a plug-in. Thus, every plug-in developed integrates with Eclipse in the same way as other plug-ins; in this respect, all features are "created equal".[50] Eclipse provides plug-ins for a wide variety of features, some of which are from third parties using both free and commercial models. Examples of plug-ins include for

Unified Modeling Language (UML), for Sequence and other UML diagrams, a plug-in for DB Explorer, and many more.

The Eclipse SDK includes the Eclipse Java development tools (JDT), offering an IDE with a built-in Java incremental compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat filespace allowing external file modifications as long as the corresponding workspace resource is refreshed afterward.

Eclipse implements the graphical control elements of the Java toolkit called Standard Widget Toolkit (SWT), whereas most Java applications use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also uses an intermediate graphical user interface layer called JFace, which simplifies the construction of applications based on SWT. Eclipse was made to run on Wayland during a Google Summer of Code (GSoC) Project in 2014.

As of 2017, language packs being developed by the Babel Project provide translations into over 40 natural languages.

4.6 Web technology

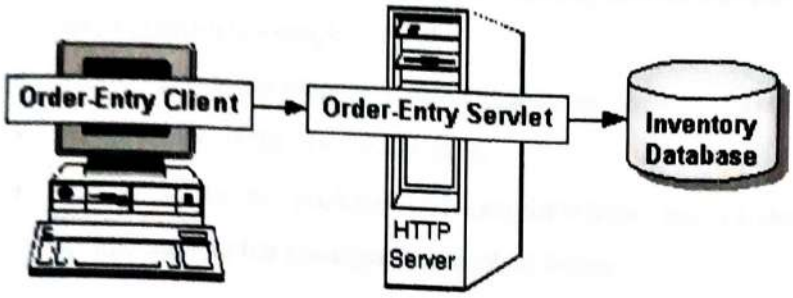
4.6.1 Servlet

Servlets provide a Java(TM)-based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions, platform-specific APIs, and incomplete interfaces.

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side -- object byte codes that can be dynamically loaded off the net. They differ from applets in that they are faceless objects (without graphics or a GUI component). They serve as platform-independent, dynamically loadable, plug gable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

What is a Servlet?

Servlets are modules that extend request/response-oriented servers, such as Java-enabled web servers. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a company's order database.



Servlets are to servers what applets are to browsers. Unlike applets, however, Servlets have no graphical user interface. Servlets can be embedded in many different servers because the servlet API, which you use to write Servlets, assumes nothing about the server's environment or protocol. Servlets have become most widely used within HTTP servers; many web servers support the Servlet API.

Use Servlets instead of CGI Scripts

- Servlets are an effective replacement for CGI scripts. They provide a way to generate dynamic documents that is both easier to write and faster to run. Servlets also address the problem of doing server-side programming with platform-specific APIs: they are developed with the Java Servlet API, a standard Java extension.
- So use Servlets to handle HTTP client requests. For example, have Servlets process data posted over HTTPS using an HTML form, including purchase order or credit card data. A servlet like this could be part of an order-entry and processing system, working with product and inventory databases, and perhaps an on-line payment system.

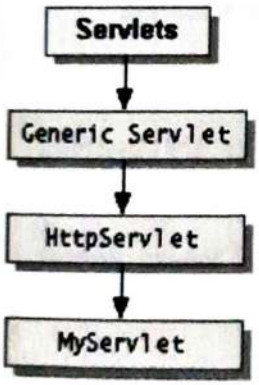
Other Uses for Servlets

Here are a few more of the many applications for Servlets

- Allowing collaboration between people. A Servlet can handle multiple requests concurrently, and can synchronize requests. This allows Servlets to support systems such as on-line conferencing.
- Forwarding requests. Servlets can forward requests to other servers and Servlets. Thus Servlets can be used to balance load among several servers that mirror the same content, and to partition a single logical
- service over several servers, according to task type or organizational boundaries.
- Architecture of the Servlet Package
- The javax.servlet package provides interfaces and classes for writing Servlets. The architecture of the package is described below.

The Servlet Interface

- The central abstraction in the Servlet API is the Servlet interface. All Servlets implement this interface, either directly or, more commonly, by extending a class that implements it such as HttpServlet.



- The Servlet interface declares, but does not implement, methods that manage the servlet and its communications with clients. Servlet writers provide some or all of these methods when developing a servlet.

Client Interaction

- When a servlet accepts a call from a client, it receives two objects:
- A ServletRequest, which encapsulates the communication from the client to the server.

- A ServletResponse, which encapsulates the communication from the servlet back to the client.
- ServletRequest and ServletResponse are interfaces defined by the javax.servlet package.

The ServletRequest Interface

- The ServletRequest interface allows the servlet access to: Information such as the names of the parameters passed in by the client, the protocol (scheme) being used by the client, and the names of the remote host that made the request and the server that receive the input stream, ServletInputStream. Servlets use the input stream to get data from clients that use application protocols such as the HTTP POST and PUT methods.

Interfaces that extend ServletRequest interface allow the servlet to retrieve more protocol-specific data. For example, the HttpServletRequest interface contains methods for accessing HTTP-specific header information.

THE SERVLETRESPONSE INTERFACE

The ServletResponse interface gives the servlet methods for replying to the client. It

- Allows the servlet to set the content length and MIME type of the reply.

4.7.1 Features of My SQL

MySQL is offered under two different editions: the open source MySQL Community Server and the proprietary Enterprise Server.[72] MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and is built from the same code base.

Major features as available in MySQL 5.6:

- A broad subset of ANSI SQL 99, as well as extensions
- Cross-platform support
- Stored procedures, using a procedural language that closely adheres to SQL/PSM[73]
- Triggers
- Cursors
- Updatable views
- Online Data Definition Language (DDL) when using the InnoDB Storage Engine.
- Information schema
- Performance Schema that collects and aggregates statistics about server execution and query performance for monitoring purposes

4.8 Functional and Non Functional Requirements

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in use cases. Functional requirements are supported by non-functional requirements (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>," while non-functional requirements take the form "system shall be <requirement>."[3] The plan for implementing functional requirements is detailed in the system design, whereas *non-functional* requirements are detailed in the system architecture.

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements, which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.

A typical functional requirement will contain a unique name and number, a brief summary, and a rationale. This information is used to help the reader understand why the requirement is needed, and to track the requirement through the development of the system.The crux of the requirement is the description of the required behavior, which must be clear and readable. The described behavior may come from organizational or business rules, or it may be discovered through elicitation sessions with users, stakeholders, and other experts within the organization.Many requirements may be uncovered during the use case development.

In systems engineering and requirements engineering, a **non-functional requirement** (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing *functional* requirements is detailed in the system *design*. The plan for implementing *non-functional* requirements is detailed in the system *architecture*, because they are usually architecturally significant requirements.[1]

Broadly, functional requirements define what a system is supposed to *do* and non-functional requirements define how a system is supposed to *be*. Functional requirements are usually in the form of "system shall do <requirement>", an individual action or part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of "system shall be <requirement>", an overall property of the system as a whole or of a particular aspect and not a specific function. The system's overall properties commonly mark the difference between whether the development project has succeeded or failed.

Chapter-5

SYSTEM DESIGN

5.1. Data Flow Diagrams

Encryption Process

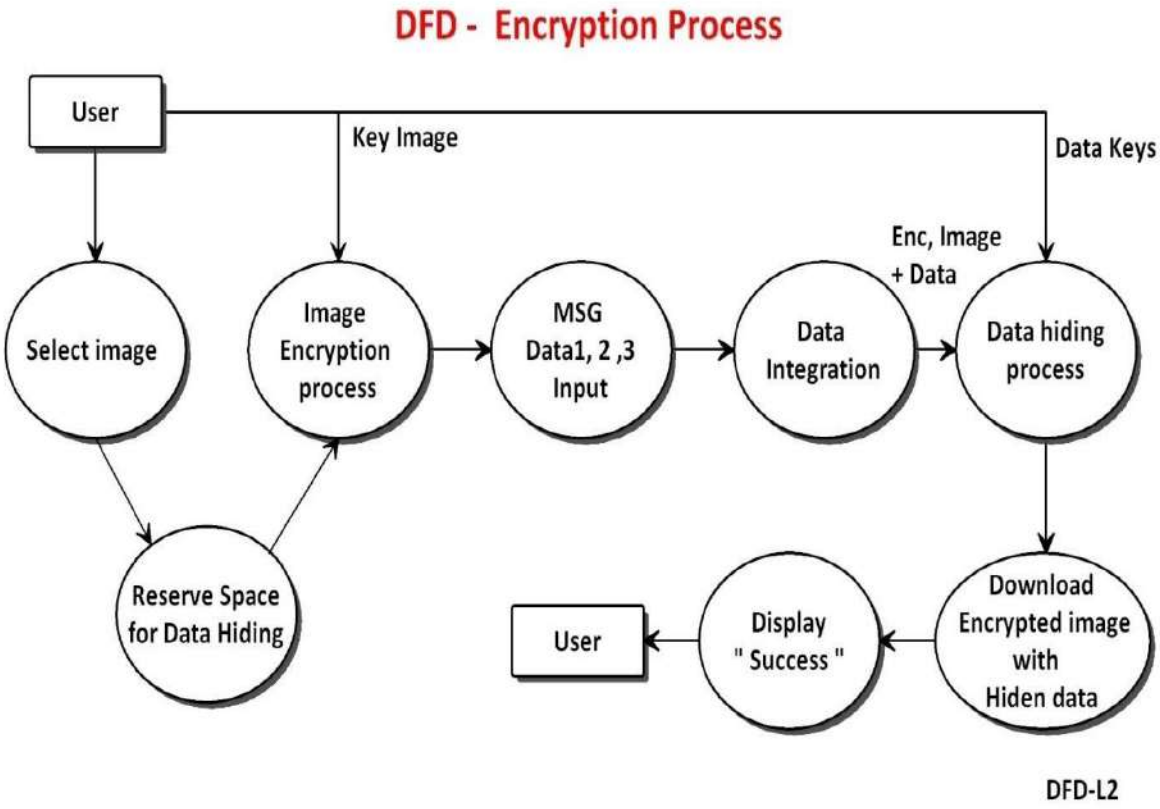


Figure 1.1 Dataflow Diagram for Encryption Process

To perform the encryption process user first needs to select an image from the local system then reserve space for data hiding. After reserved the space the image is sent to encryption process, user provide a key to process. We can also hide the message, to hide message user need to provide the message that need to be hide. To hide the data user need to provide different key. We can download the encrypted image with hidden data to the local system. Then system display success to the user.

Decryption Process

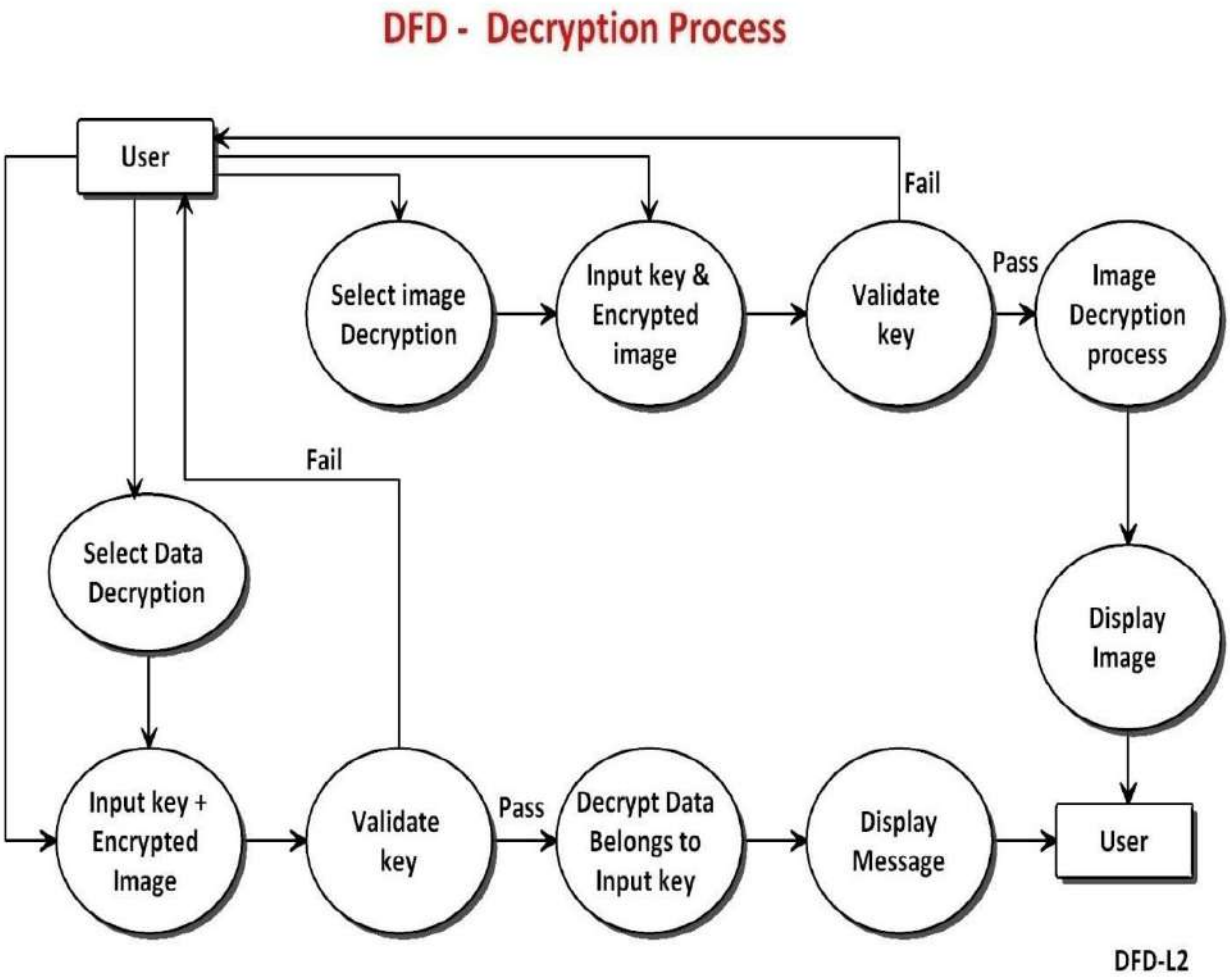


Figure 1.2 Dataflow Diagram for Decryption Process

- To perform decryption process user need to select an image, provide the key with an encrypted image to the system. The key will be validate if the key is valid it moves to image decryption process. If the key is invalid it again ask for user to enter valid key. If the decryption process completed successfully it display the image to the user.
- If user need to decrypt the data user selects the data that need to be decrypt. User provides the key and encrypted image to system. Key is again validated, if key is valid it moves to decryption process. If the key is invalid it again ask for user to enter valid key. If the decryption process completed successfully it display the message to the user.

5.2. Sequence Diagram

Data Hiding

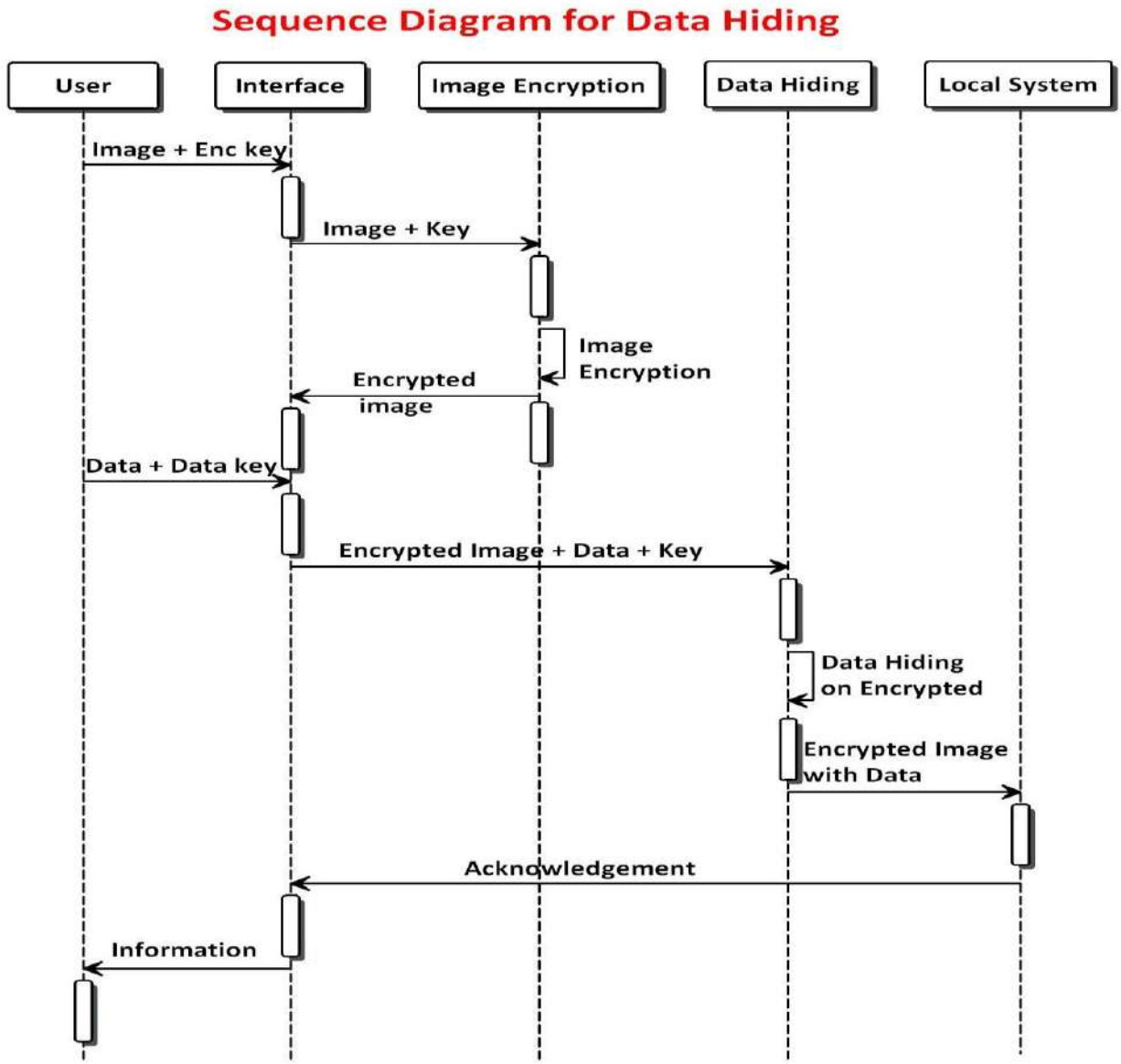


Figure 1.3 Sequence Diagram for Data Hiding

To perform the encryption process user first needs to select an image from the local system then reserve space for data hiding. After reserved the space the image is sent to encryption process, user provide a key to process. We can also hide the message, to hide message user need to provide the message that need to be hide. To hide the data user need to provide different key. We can download the encrypted image with hidden data to the local system. Then system display success to the user.

Image + Data Retrieval

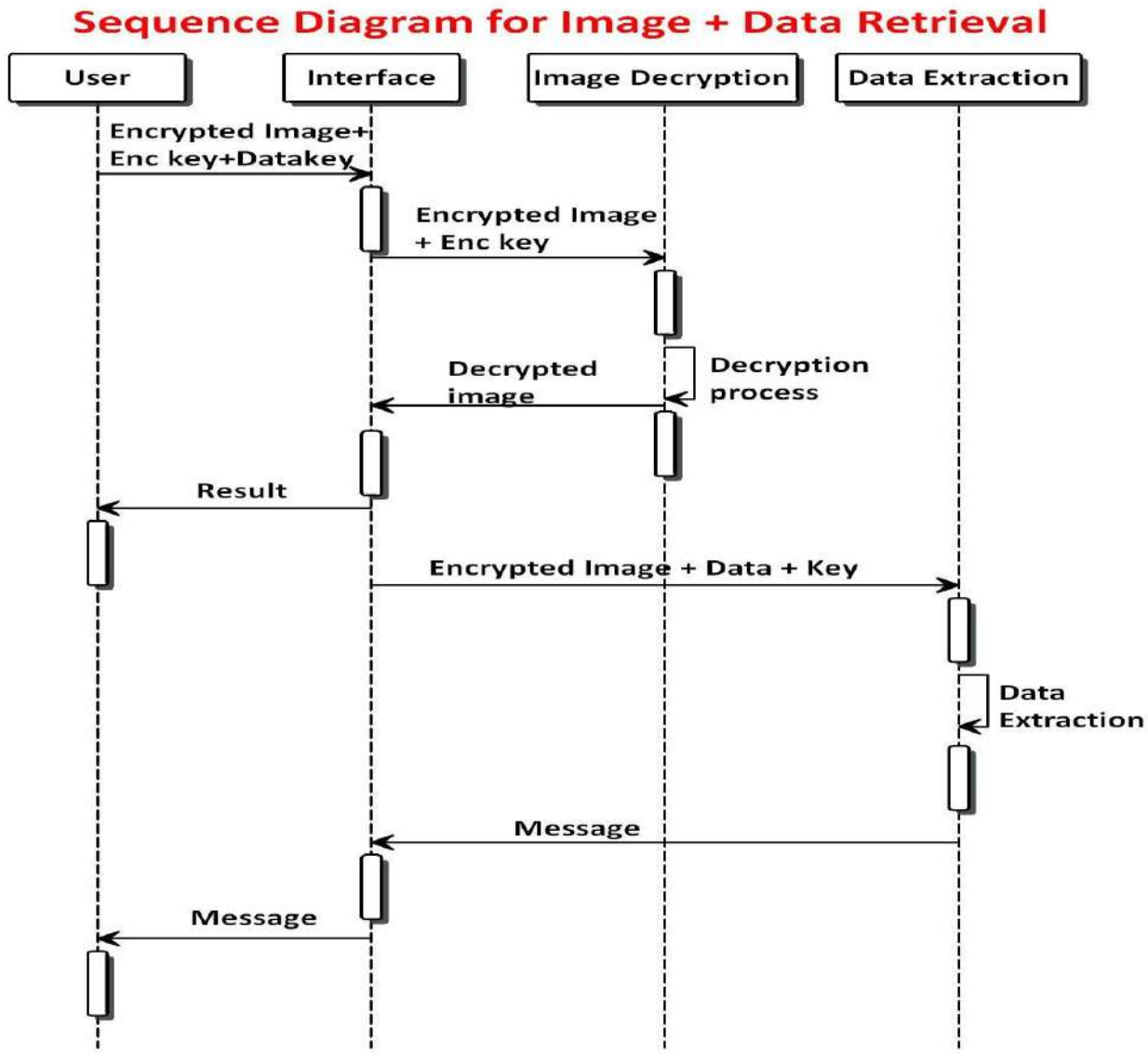


Figure 1.4 Sequence Diagram fpr Image+Data Retrieval

- To perform decryption process user need to select an image, provide the key with an encrypted image to the system. The key will be validate if the key is valid it moves to image decryption process. If the key is invalid it again ask for user to enter valid key. If the decryption process completed successfully it display the image to the user.
- If user need to decrypt the data user selects the data that need to be decrypt. User provides the key and encrypted image to system. Key is again validated, if key is valid it moves to decryption process. If the key is invalid it again ask for user to enter

valid key. If the decryption process completed successfully it display the message to the user.

5.3 Modules

Admin Module

- Login Session
- Show Profile
- User Details (View, Edit)

  Change Password

Description

Admin will login by using login session. Admin user also has the privilege to delete & edit the Member User. Admin also can change their password and details.

Member Module

- Registration
- Login Session
- Member Profile
- Image Encryption & Data Hiding

(a) Image Select from local system

(b) Image Uploaded to Web Server using FTP

(c) Encryption Key Input

(d) Image Encryption process (produce Encrypted Image)

(e) Hidden Data Input(Data1,Data2,Data3)

(f) Data Hiding Key Input (key1,key2,key3)

(g) Data Hiding process (Produce Encrypted Image where Data is hided)

(h) Downloading the Encrypted Image to the local system.

- Image Decryption & Retrieval of Data

(a) Selecting the Encrypted Image with Hidden Data from Local system

(b) Image Encryption Key Input (Optional)

(c) Data Hiding Key Input (Option)

(d) Based on the key provided do the following process

(e) If Image Encryption Key only given

(f) Do Image Decryption Process

(g) Show the Encrypted Image

(h) Download the Image to local system

- If Data Hiding Key only given

(a) Do Data Retrieval Process

(b) Display the Hidden Data

Description

Member will create account by them self by using Register session, after that member will login by using login session. Member can encrypted image by using key and hiding the data into the image using different key. Decryption Process: He has two options to view encrypted data. Those are viewing only image, or only data. After selecting option based on option we have to give key value and see the result.

Chapter-6

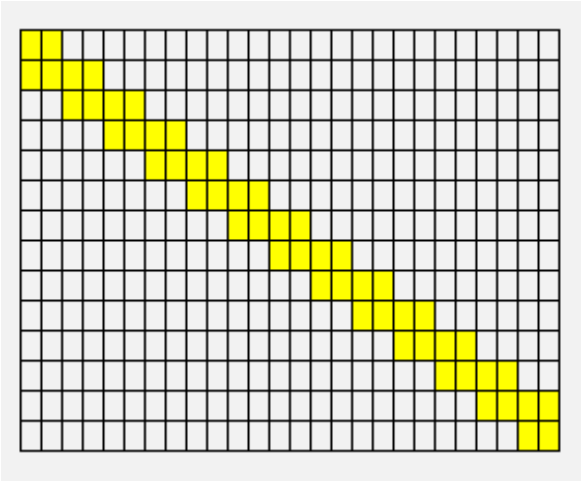
IMPLEMENTATION

A crucial phase in the system development life cycle is successful implementation of new system design. Implementations simply mean converting new system design into operation. This is the moment of truth, the first question that strikes in every one's mind is whether the system will be able to give all the desires results as expected from system. The term implementation has different meanings, ranging from the conversion of a basic application to a complete replacement of computer system. Implementation is used here to mean the process of converting a new or revised system design into an operational one. This chapter presents the implementation details and reasons for choosing language, platforms and also the coding guidelines used.

6.1 ALGORITHM

Encryption Process

Step 1: Reserving Room for Embedding Data.



Consider the above matrix is a pixel representation of a image. This system reserves the pixels which are highlighted for embedding the data.

Step 2: Image Encryption.

- For Image Encryption this system uses the technique called XOR Operation.

- For this operation system need two input.

  1.Image

  2.Image Encryption Key

- And the output will be Encrypted Image.
- For Example Input Encryption Key is INDIA then from the input key, this system will generate a 8-bit key value (KV)by following method.
- Bit(ASCII(I)) XOR Bit(ASCII(N)) XOR Bit(ASCII(D)) XOR Bit(ASCII(I)) XOR Bit(ASCII(A))

Step 3: Data Hiding on Encrypted Image

- For this operation system need Three input.

1. Encrypted Image
2. Data Hiding Key
3. Data to Hide

- In this process we are using following two algorithms

1. Key Based Pixel Selection algorithm
2. MSB replacement Algorithm

- Consider our Data Hiding Key is ABCDEF
- Separate each characters A,B,C,D,E,F
- For A find ascii value that is 65 convert it into 8 bit binary “0100 0001”
- For B find ascii value that is 66 convert it into 8 bit binary “0100 0010”…..

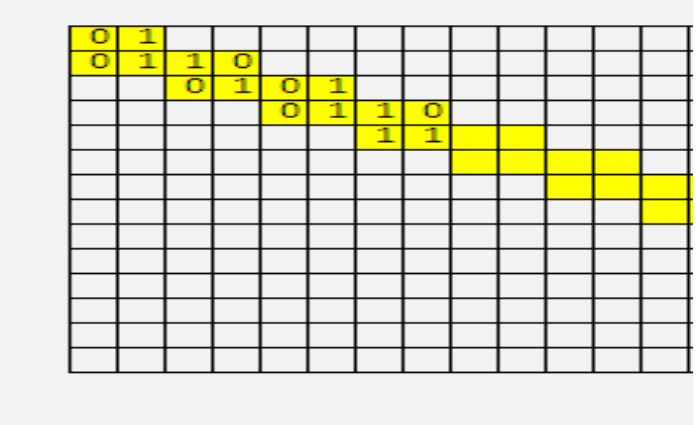


Figure 2.1 Converting Binary values in reserved pixel space

- Map the converted binary values in the reserved pixel space, which is shown in above image.
- This system going to hide the data in the pixels which are having value 1.
- In Selected pixel the colour channel R is going to hold the hidden data.
- We are changing the last 4 MSB for only one colour channel, there will not be any damage to the real image.

Example: if we are planning to hide a text X

X ASCII Value 88 then it Binary value is 0101 1000

Split Binary part into A and B like that A = 0101 and B = 1000

In Pixel selection matrix-

Let R value in 1,2 is 120 (R1) and R Value in 2,2 is 91(R2)

R1 = Binary(120) = 0111 1000 R2 = Binary(91) = 0101 1011

Replace last 4 bit in R1 by A and Replace last 4 bit in R2 by B

After Conversion R1 = 0111 0101 and R2 = 0101 1000 (R1 = 117 , R2= 88)

Decryption Process

- In decryption process this system has following two options
- Data Retrieval

Input : Encrypted Image with Data + Data Hiding Key

Output : Hidden Data

- Image Recovery

Input : Encrypted Image with Data + Image Encryption Key

Output : Recovered Image without any damage.

Chapter-7

CONCLUSION

- We are going to propose an efficient method of reversible data hiding in a encrypted image based on MSB prediction with high embedded capacity.
- From our knowledge this is one of the first methods, which proposes to use MSB instead of LSB for RDHEI . Due to the fact that MSB prediction is easier than LSB prediction in the original domain.
- The original image is slightly modified in order to avoid all the prediction errors . After substituting all MSB in image, it is possible to hide one bit per pixel.
- We have seen that the proposed scheme provides a good security level and can be used to preserve the original image content confidentiality.

REFERENCES

[1] P. Bas and T. Furon, "Image database of BOWS-2," http://bows2.eclille. fr/.

[2] X. Cao, L. Du, X. Wei, D. Meng, and X. Guo, "High capacity reversible data hiding in encrypted images by patch-level sparse representation, IEEE Transactions on Cybernetics, vol. 46, no. 5, pp. 1132–1143, 2016.

[3] T.-H. Chen and K.-H. Tsao, "User-friendly random-grid-based visual secret sharing," IEEE Transactions on Circuits and Systems for Video Technology, vol. 21, no. 11, pp. 1693–1703, 2011.

[4] Z. Erkin, A. Piva, S. Katzenbeisser, R. L. Lagendijk, J. Shokrollahi, G. Neven, and M. Barni, "Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing," EURASIP Journal on Information Security, vol. 2007, p. 17, 2007.

[5] X. Gao, L. An, Y. Yuan, D. Tao, and X. Li, "Lossless data embedding using generalized statistical quantity histogram," IEEE Transactions on Circuits and Systems for Video Technology, vol. 21, no. 8, pp. 1061–1070, 2011.

[6] W. Hong, T.-S. Chen, and H.-Y.Wu, "An improved reversible data hiding in encrypted images using side match," IEEE Signal Processing Letters, vol. 19, no. 4, pp. 199–202, 2012.