

CHAPTER-1

1. Introduction

1.1. Purpose

The development of the Space Science and Technology has recently attracted a growing interest from researchers and industrial communities, mainly because of large number of possible applications capable to exploit remotely sensed data and satellite images. Advances in space science, data analysis, and communication technologies present new opportunities for users to increase productivity, reduce costs, facilitate innovation and create virtual collaborative environments for addressing the new challenges. GIS and Remote sensing technologies, along with related geospatial technologies, contribute powerful tools for preserving and protecting the nation's critical infrastructure.

In such systems, a space borne platform collects scientific data and transmits them to a ground station and at the ground segment, a series of image products are created that can be made available to research or commercial organizations for exploitation. The data delivery and sharing process, usually based on CD/DVD-ROM or on shared network environment (Internet, LAN, WAN etc), provides the user with a digital version of the remote sensing data and images. In the same way as for multimedia contents, the digital format implies an inherent risk of unauthorized copy or use of the product.

Similarly many digital services, such as Medical, Military, and Space imaging systems require reliable security in storage and transmission of digital images. The rapid progress of Internet in the digital world today, the security of digital images has become more and more important. The prevalence of multimedia technology in our society has promoted digital images to play a more significant role, which demands a serious protection of users' privacy. To fulfill such security and privacy needs in various applications, encryption of images is very important to minimize malicious attacks from unauthorized parties.

1.2. Intended Audience and Reading Suggestions

This project is intended to the people who need privacy for their confidential images. It was most suitable in the networking Systems, so it was more eligible in Space Science research centers.

1.3. Scope

Although there are many image encryption techniques none of them are suitable for the networking systems. So the main scope of our project was to provide security for the images in the networking systems. Our project provides safe ways of means to transfer images between the networking systems confidentially.

1.4 ARCHITECTURE

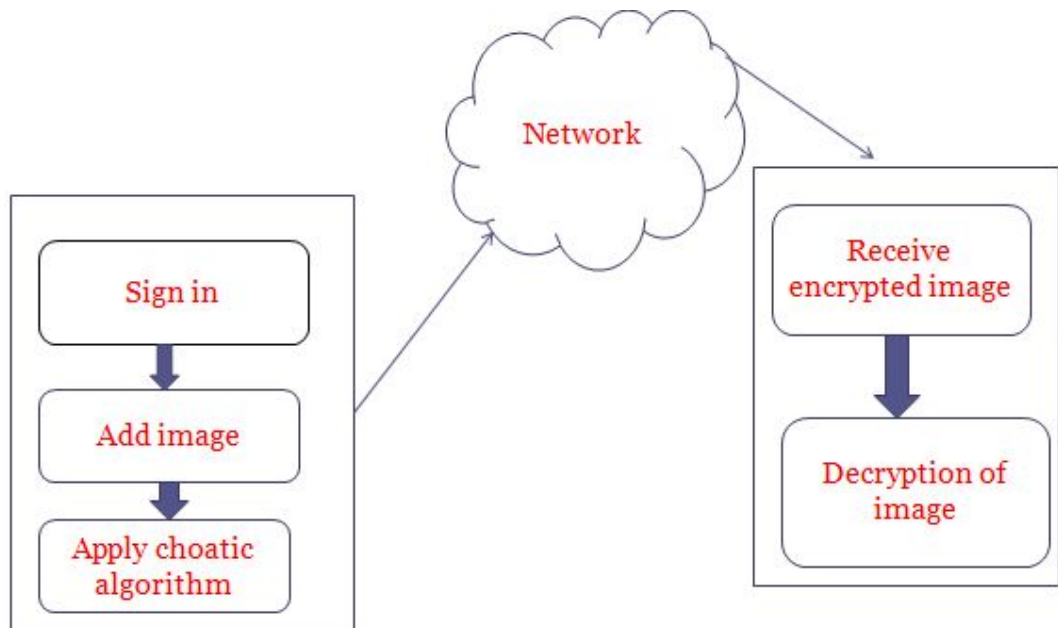


Fig.1.1 Architecture Diagram

CHAPTER-2

2. Software and Hardware Requirements

2.1. Software Requirements

The minimum software requirement specifications for developing this project are as follows:

Operating System	:	Windows XP
Presentation Layer	:	Java, Servlets, JSP
Web Server	:	Apache Tomcat 6.0
Database	:	Oracle 10g XE
IDE	:	Eclipse 3.3
Database Layer	:	JDBC
Documentation Tool	:	MS Office
UML Tools	:	Rational Rose

2.2. Hardware Requirements

The minimum hardware requirement specifications for developing this project are as follows:

Processor	:	Standard processor with a speed of 1.6GHz
RAM	:	256MB RAM or higher
Hard Disk	:	20GB or more
Monitor	:	Standard color monitor
Keyboard	:	Standard keyboard
Mouse	:	Standard mouse

CHAPTER-3

3. Literature Survey

There are many scientists who worked on the chaotic encryption techniques, here are some of them, S. Bennie et al. proposed a novel algorithm for image encryption based on mixture of chaotic maps. In this symmetric key cryptography technique, a typical coupled map was mixed with a one-dimensional chaotic map and used for high degree security image encryption while its speed is acceptable. This mixture application of chaotic maps shows advantages of large key space and high-level security. The cipher generated by this method is the same size as the plaintext. Similarly in, Jiri Fridrich showed that how to adapt certain invertible chaotic two-dimensional maps on a torus or on a square to create symmetric block encryption schemes.

The schemes are especially useful for encryption of large amount of data, such as digital images or electronic databases. In this a chaotic map is first generalized by introducing parameters and the discretized to finite square lattice of points which represent pixels or some other data items. The discretized map is further extended to three dimensions and composed with simple diffusion mechanism. As a result, a block product encryption scheme is obtained. To encrypt an $N \times N$ image, the ciphering map is iteratively applied to the image. The main features of the encryption scheme studied in are: a variable key length, a relatively large block size (several KB or more), and a high encryption rate (1Mb un-optimized C code on a 60MHz Pentium). The cipher is based on two dimensional chaotic maps, which are used for creating complex, key-dependent permutations.

Unlike most today's symmetric encryption schemes, which rely on complex substitution rules while somewhat neglecting the role of permutations, the cipher is based on complex permutations composed with a relatively simple diffusion mechanism. A somewhat different chaotic maps based image encryption method was proposed in to improve the properties of confusion and diffusion in terms of discrete exponential chaotic maps, and design a key scheme for the resistance to statistic attack, differential attack and grey code attack be-cause of floating-point analytical computation outcome of chaotic system, unauthorized users or attacker can attack cryptosystems efficiently

and effectively via certain basin structures. To increase security level of cryptosystem, they combined approaches of chaotic and conventional crypto-graphic methods in some ways as in. To achieve diffusion and resistance from differential attack, they analyze discrete exponential chaotic map performance and de-sign the permutation of the pixels of image after that performs “XOR plus mod” operation on designed permutation. Similarly, to achieve and increasing resistance level from statistic attack, grey code attack and entropy attack, Linhua et al. chose chaotic map elaborately and construct chaotic sequences which satisfy uniform distribution .

To achieve high encryption speed, in the early stages some elementary cryptographic methods using algebra-based transformation scrambling techniques were suggested. Since the operations are simple, the encryption does not require high computational cost but recent research suggest that Chaos-based transformations naturally have affine linearity while algebra-based transformations have fixed scrambling matrixes for data encryption. The cryptanalysis of show that the fixed or linear scrambling matrices are insecure for image encryption.

Similar thoughts also appeared in by Wang et al., where a Poker shuffle is used for scrambling image, which is controlled dynamically by chaotic system. This scrambling technique belongs to the position permutation and it has several features like nonlinearity, non-analytic formula and large key space. In addition, it can deal with non-square image while many scrambling methods with analytic formula only process square image and can be integrated into the existing image encryptions and digital image water-marking.

To overcome these limitations in Logistic maps the nonlinear chaotic algorithm (NCA) map uses power function and tangent function instead of linear function. Many chaotic circuits and their application schemes have been proposed to secure communications since Pecora and Carrol proposed a method to synchronize two identical chaotic systems. The chaotic secure system can be used in applications that do not require a high level of information security such as remote keyless entry system, video phone, and wireless telephone. Some existing Hard-ware based chaotic secure communication schemes have been briefly reviewed in this Section.

CHAPTER-4

4. Software Requirements Analysis

4.1. Existing System

There are some existing systems like traditional image encryption techniques DES, Triple-DES and IDEA.

1. Here in the existing systems the both the encryption and decryption will be done in a single system.
2. The user can not change his initial conditions, will not have repeated processing.
3. Have very less confusion and diffusion properties that are desirable for cryptography.

4.1.1 Problems in the Existing System

- a. They provide high security level only under CBC mode.
- b. They require large data size.
- c. The existing system will take long computational time.
- d. They need high computing power.
- e. Not efficient for networking Systems.

4.2. Proposed System

Proposed system provides a solution to existing system by extending its facilities. The proposed study aims to explore the possibility of using chaotic or chaos-based encryption techniques to protect remote sensing satellite images and provides high level of security in efficient and reliable way.

Chaos based cryptographic scheme provides high security level, less computational time and power in reliable and efficient way to deal with balky, difficult and intractable data that why many researchers recommends that it is more suitable for multimedia data, especially for images. Chaos-based system have many properties to achieve high security level, such as sensitivity to change initial conditions and parameters, periodicity (a system that tends in probability to a limiting form that is independent of the initial conditions), random behavior and unstable periodic orbits with long periods. It has very high diffusion and confusion properties that are desirable for cryptosystem.

1. Maintenance is easy.
2. Provides high security level.
3. This project takes less computational time and power in reliable and efficient way to deal with balky, difficult and intractable data.
4. It is more suitable for multimedia data, especially for images.
5. This project has many properties to achieve high security level, such as sensitivity to change initial conditions and parameters, periodicity, random behavior and unstable periodic orbits with long periods.
6. It has very high diffusion and confusion properties that are desirable for cryptosystem.
7. Encrypting and Decrypting of the image is very easy.
8. Sending or transferring the image via the network is easier, just to start server class at receiver side.
9. By entering the key with which the key was encrypted we can get the original image in the other system.
10. Viewing the status of encryption and decryption processes will also be there.

4.3. System Features and Functionalities

The user can have membership by concerning the administrator. The system dynamically generates member id on presenting the user details. The user can have transactions by having their member id. The user adds the image for encryption and encrypts it for confidential images.

4.3.2 Users of the System

1. Administrator.
2. User.
3. Other (Guests).

1. Administrator:

Administrator is having all the rights to access this portal. Admin can view all the images, and the status of the encrypting and decrypting process. He will have control over the network whether to accept the server's request for receiving image. He will provide all the benefits for image encryption and decryption.

2. User

The people who use this system for providing security for confidential images are users. There may be different set of users for this system. The users just load an image to encrypt and encrypt it and enter the destination systems IP address to transfer the image and he can view the status of the network for the sent image.

3. Guests

Guest or the new user can only view the functionalities in the portal and they have no access permissions unless they are registered.

Process Modules

Software engineering is an engineering discipline whose goal is the cost-effective development of software systems. Software engineering is concerned with development and maintenance of technological products, problem solving techniques common to all engineering disciplines.

A development strategy that encompasses the process, methods and tools layers is referred to software engineering paradigm or process model. A process model for software engineering is chosen based on the nature of the project and application, the methods and tools to be used and the controls and derivable those are required.

LINEAR SEQUENTIAL MODEL / WATERFALL MODEL:

This model suggests a systematic, sequential approach to software development that begins at the system level and progress through analysis, design, coding, testing and support. The figure below shows the waterfall model for software engineering.

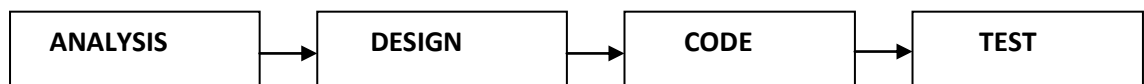


Fig: Linear Sequential Model

1. System/information engineering and modeling:

As software is always a part of a larger system, work establishing requirements for all system elements and then allocating some subset of these requirements to software. System engineering and analysis encompass requirements gathering at the system level with a small amount of top-level design and analysis. Information engineering encompasses requirements gathering at the strategic business level and at the business area level.

2. Software requirements analysis:

The requirements gathering process is intensified and focused specifically on software. To understand the nature of the program to be built, the software engineer must

understand the information domain for the software as well as required function behavior performance and interface. Requirements for both the software are documented and reviewed with the customer.

3. Design:

Software design is actually a multi-step process that focuses on four distinct attributes of a program: data structure, software architecture, interfaces representation a procedural detail. The design process translates requirements into a representation of the software that can be accessed for quality before coding begins. Like requirements, the design is documented and becomes a part of the software configuration.

4. Code generation:

The design must be translated into a machine-readable form. The code generation step performs this task. If the design is performed in a detailed manner, code generation can be performed mechanically.

5. Testing:

Once the code has been generated, program testing begins. The process focuses on the logical internals of software, ensuring that all statements have been tested and on the functional externals, that is conducting tests to uncover errors and ensure that defined input will produce actual results that agree with required results.

6. Support:

Software will undergo changes after it is delivered to the customer. Software support/maintenance reapplies each of the preceding phases to an existing program rather than a new one.

As the linear sequential model is the oldest and the most widely used paradigm for software engineering, and can be used efficiently for small projects that require less customer communication, it has been in development of data security project.

CHAPTER-5

5. Software Design

The software design used to develop this project is object-oriented design is based on function and procedures. It can be said as the development of software by building self-contained modules as objects that can be easily replaced, modified and reused. In this environment software is a collection of discrete objects that encapsulate their data as well as the functionality to model real world “objects”. Each object has attribute and methods. Objects are grouped into classes. Here each object is responsible for itself.

5.1. DFD

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system. It can also be used for the visualization of data processing (structured design).

There are two types of DFDs. They are:

- Context Level DFD
- Top Level DFD

5.1.1. Context Level DFD

In the Context Level the whole system is shown as a single process.

- No data stores are shown.
- Inputs to the overall system are shown together with data sources (as External entities).
- Outputs from the overall system are shown together with their destinations (as External entities).

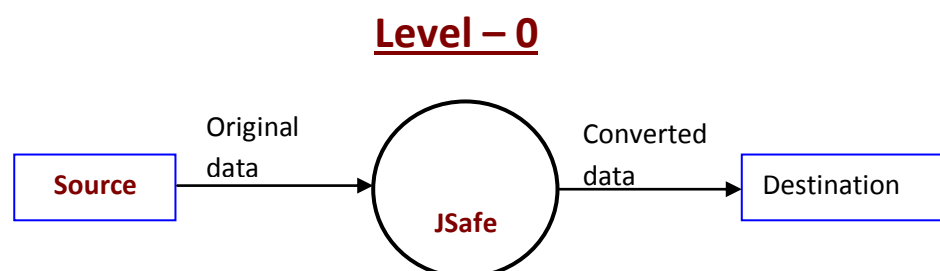


Fig. 5.1 Context Level-0 DFD

5.1.2. Level-1 DFD

The Top Level DFD gives the overview of the whole system identifying the major system processes and data flow. This level focuses on the single process that is drawn in the context diagram by ‘Zooming in’ on its contents and illustrates what it does in more detail.

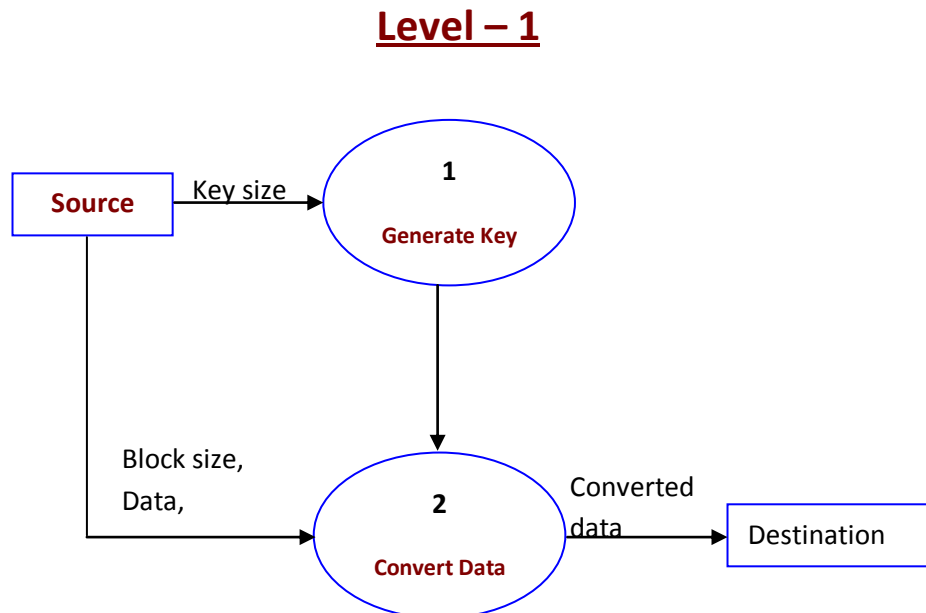


Fig. 5.2 Context Level-1 DFD

5.1.3. Level-2 DFD

The Top Level DFD gives the overview of the whole system identifying the major system processes and data flow. This level focuses on the single process that is drawn in the context diagram by ‘Zooming in’ on its contents and illustrates what it does in more detail.

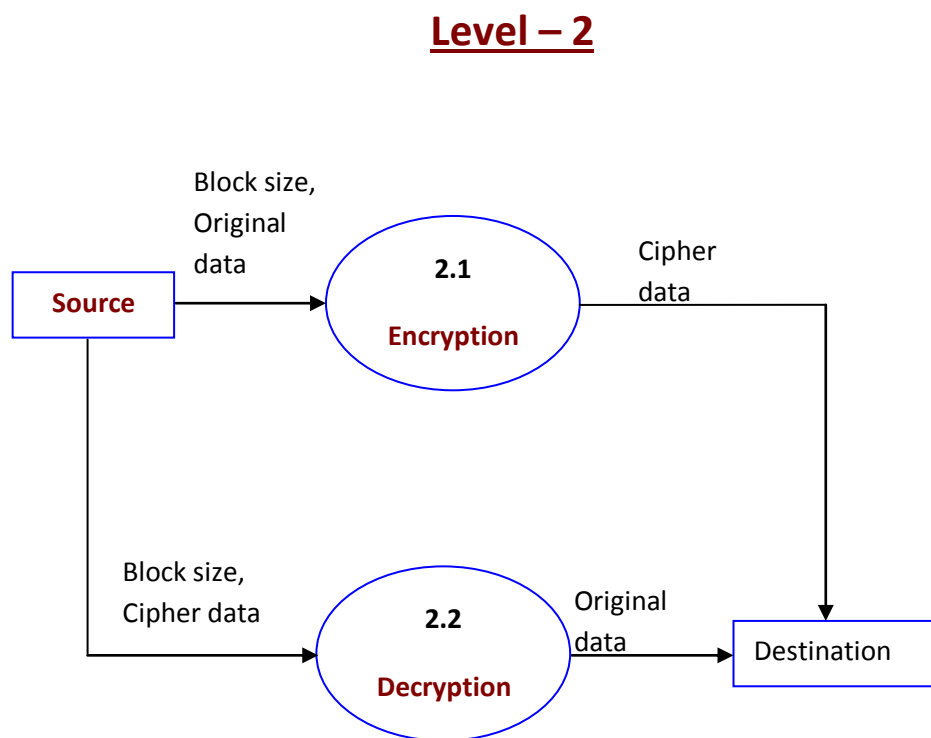


Fig. 5.3 Context Level-2 DFD

5.1.4. Level-3 DFD

The Top Level DFD gives the overview of the whole system identifying the major system processes and data flow. This level focuses on the single process that is drawn in the context diagram by 'Zooming in' on its contents and illustrates what it does in more detail.

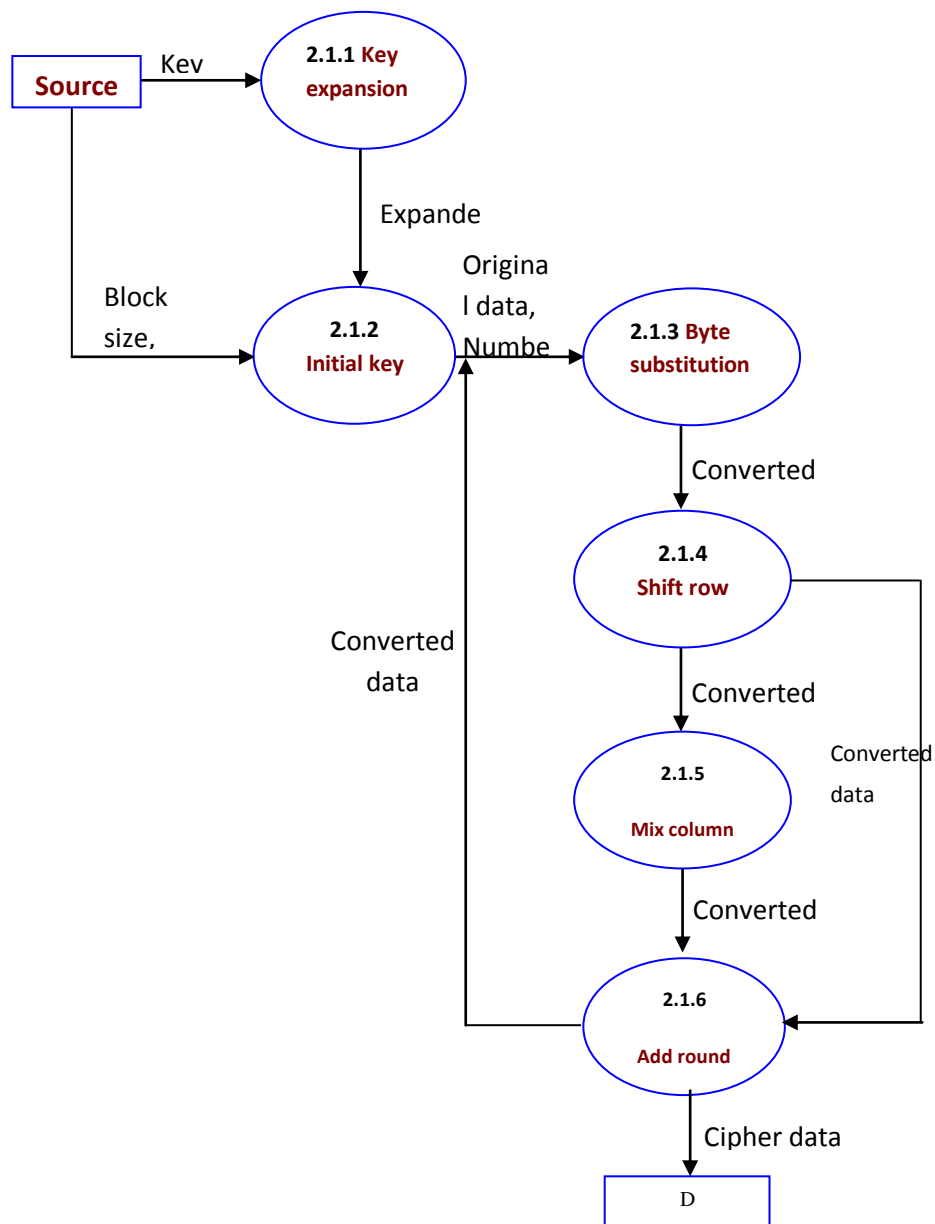


Fig. 5.4 Context Level-3 DFD

5.1.5. Level-4 DFD

The Top Level DFD gives the overview of the whole system identifying the major system processes and data flow. This level focuses on the single process that is drawn in the context diagram by 'Zooming in' on its contents and illustrates what it does in more detail.

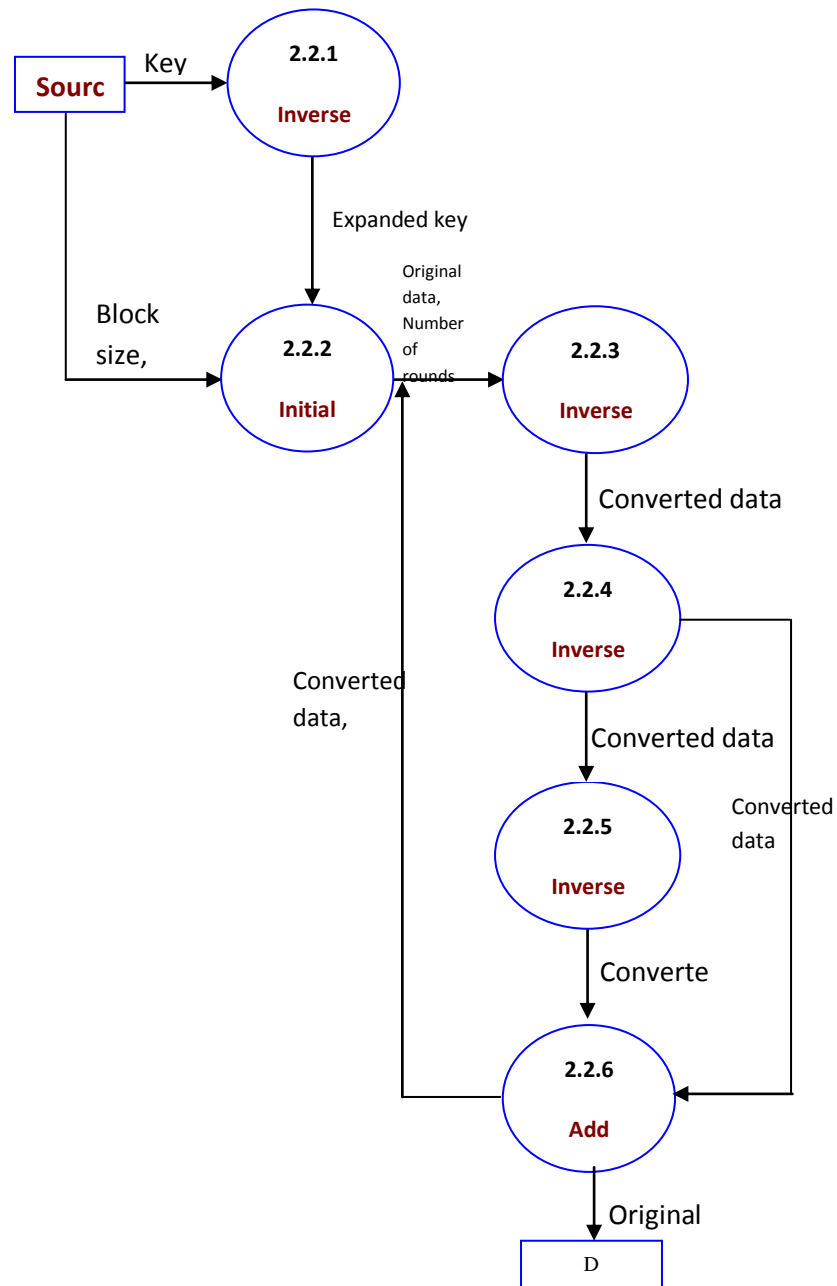


Fig. 5.5 Context Level-4 DFD

5.2. UML Diagrams

The Unified Modeling Language (UML) includes a set of graphical notation techniques to create visual models of software-intensive systems. It has structural diagrams, behavioral diagrams, and interaction diagrams.

5.2.1. Class Diagram

The class diagram is a static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

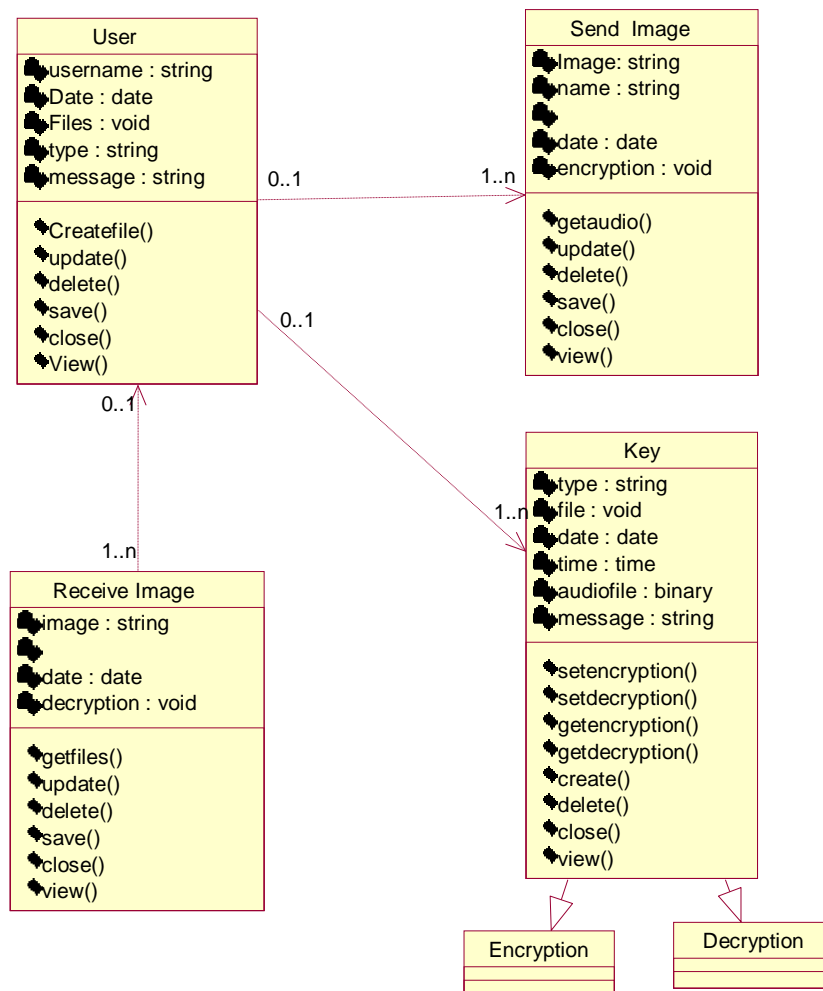


Fig. 5.6 Class Diagram

5.2.2. Interaction Diagrams

Interaction diagrams emphasize the flow of control and data among the things in the system being modeled. This includes sequence, and collaboration diagrams.

5.2.2.1. Sequence Diagram

Sequence diagram shows how processes operate with one another and in what order.

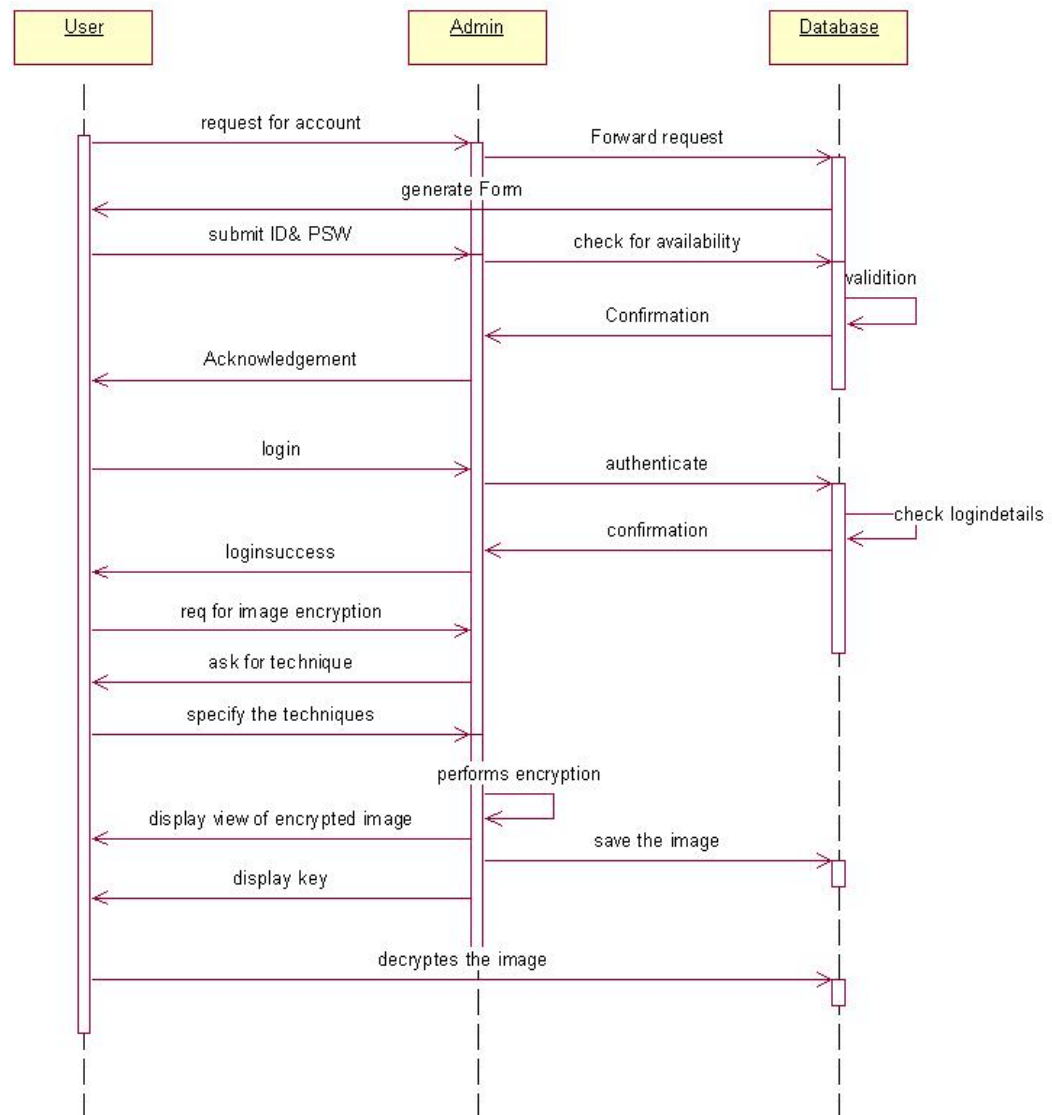


Fig. 5.7 Overall Sequence Diagram

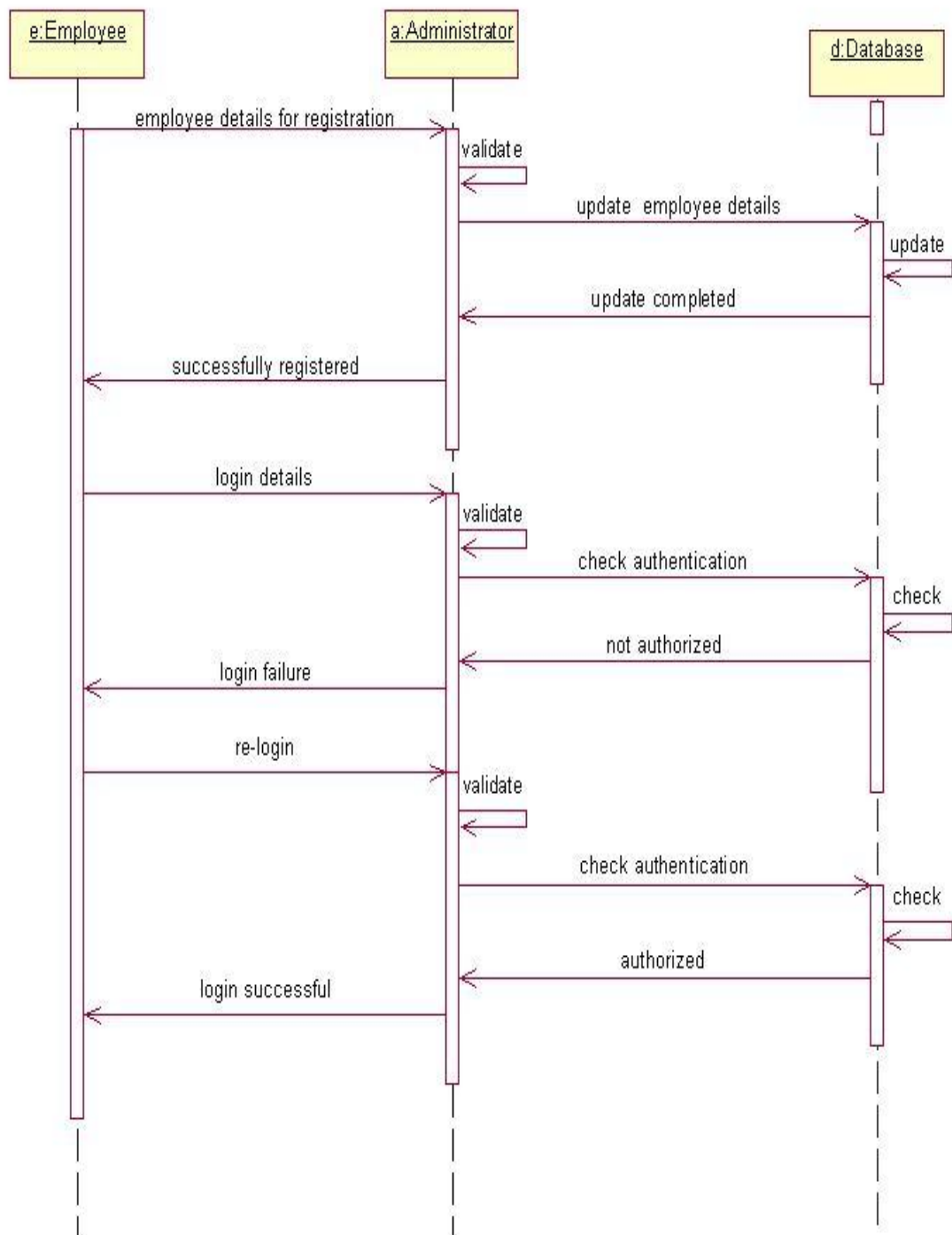


Fig. 5.8 Sequence Diagram for Login

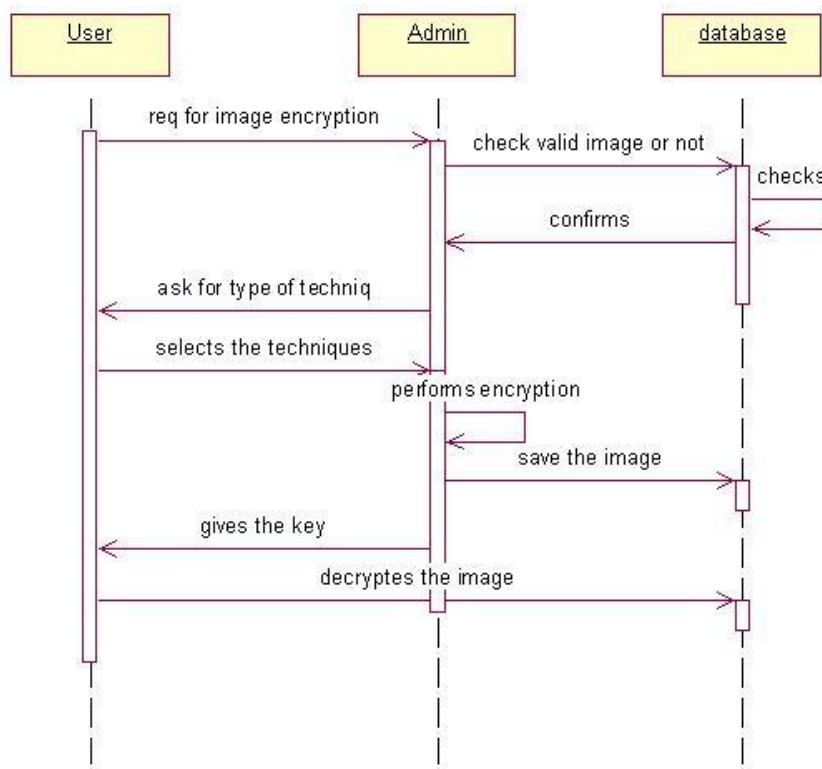


Fig. 5.9 Sequence Diagram for Encryption Mechanism

5.2.2.2. Collaboration Diagram

Collaboration or communication diagram models the interaction between objects or parts in terms of sequenced messages.

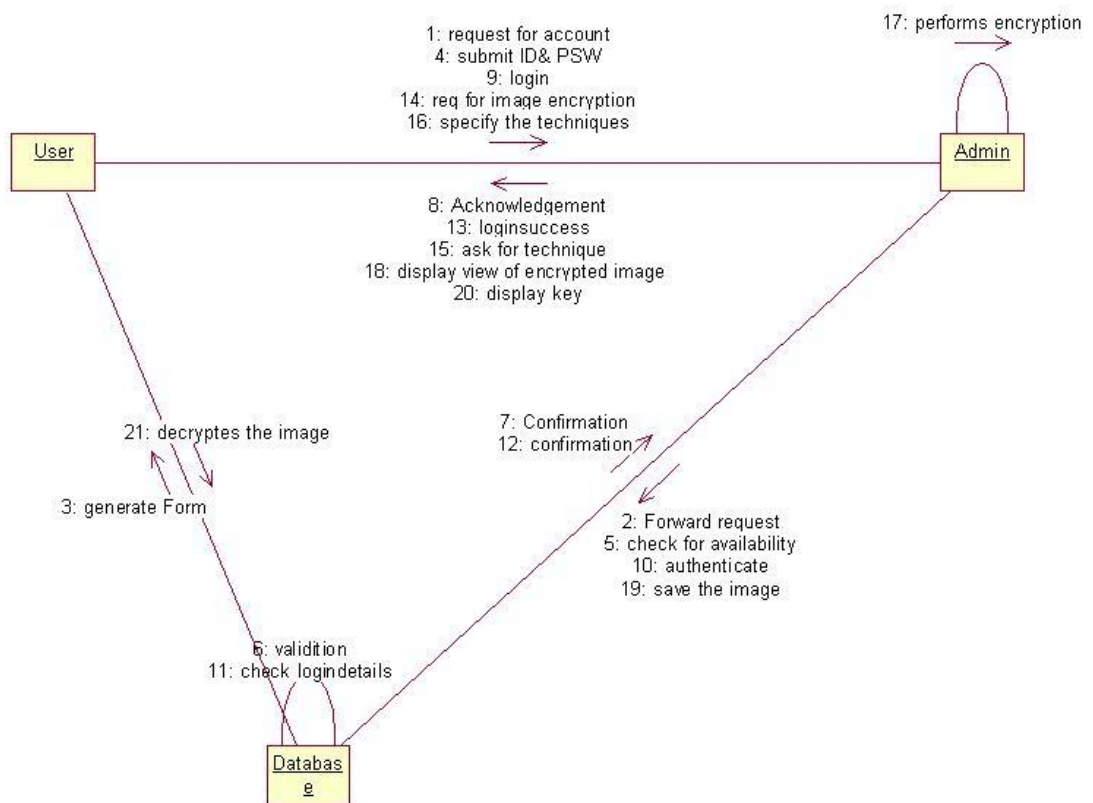


Fig. 5.10 Overall Collaboration Diagram

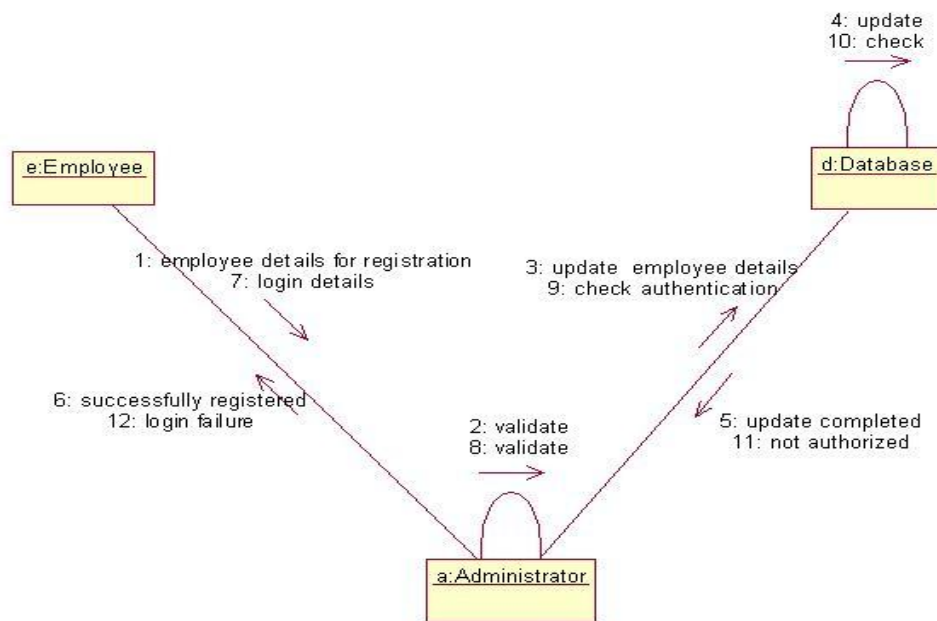


Fig. 5.11. Login Collaboration Diagram

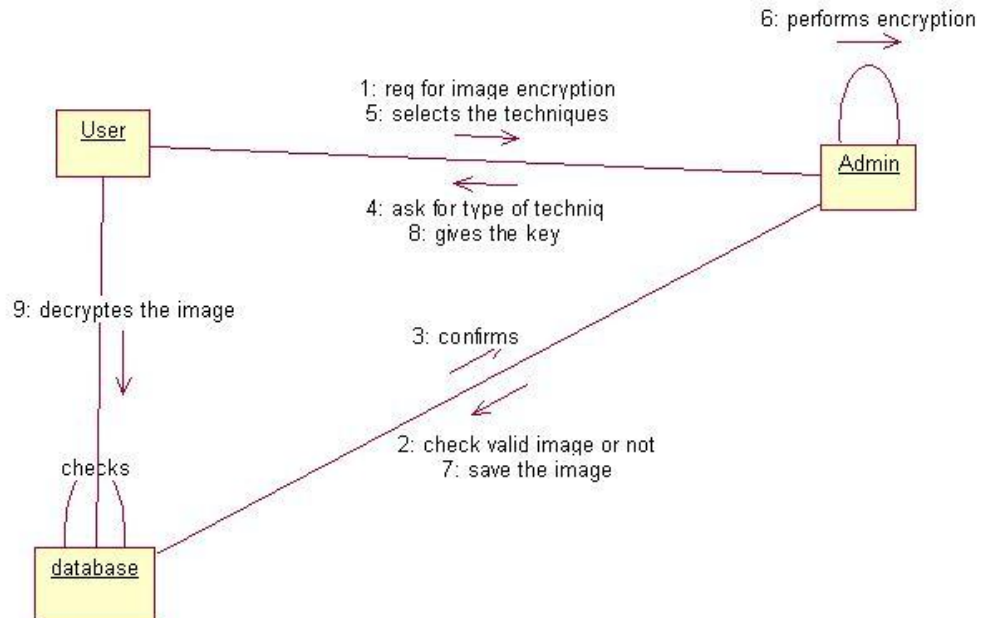


Fig 5.12 Encryption Collaboration Diagram

5.2.3. Use Case Diagram

Use case diagram is a behavioral diagram which is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals, and any dependencies between those use cases.

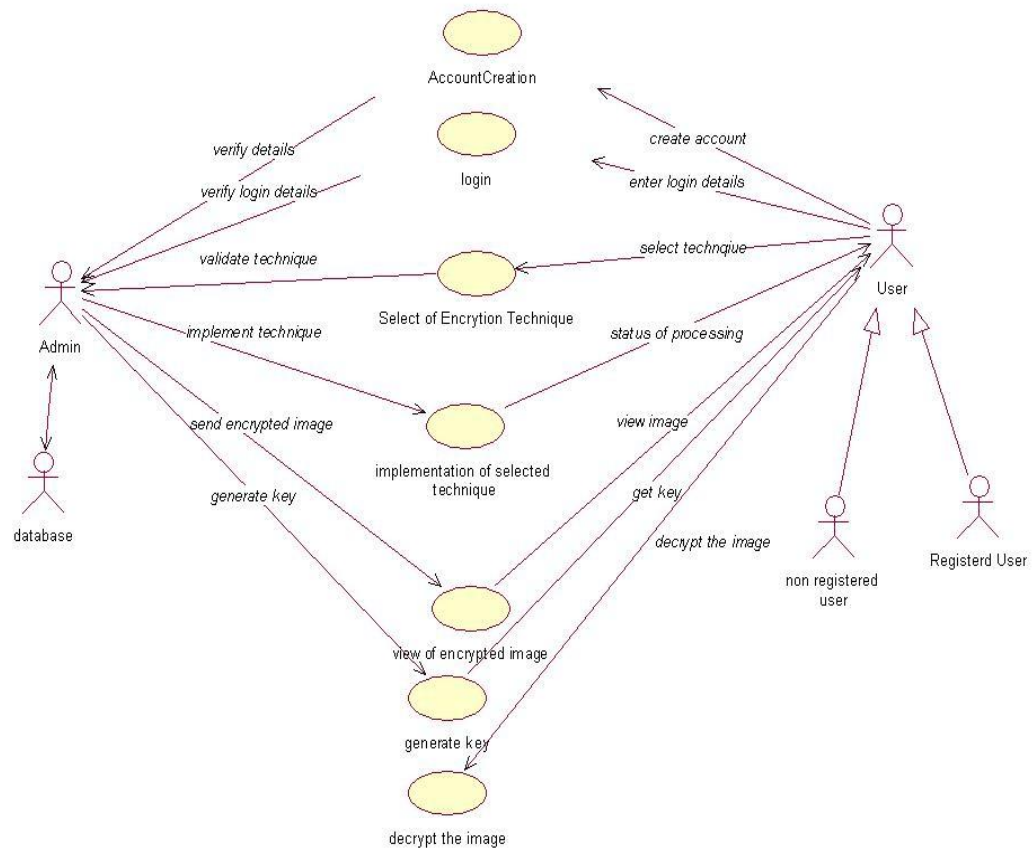


Fig. 5.13. Overall Use case Diagram

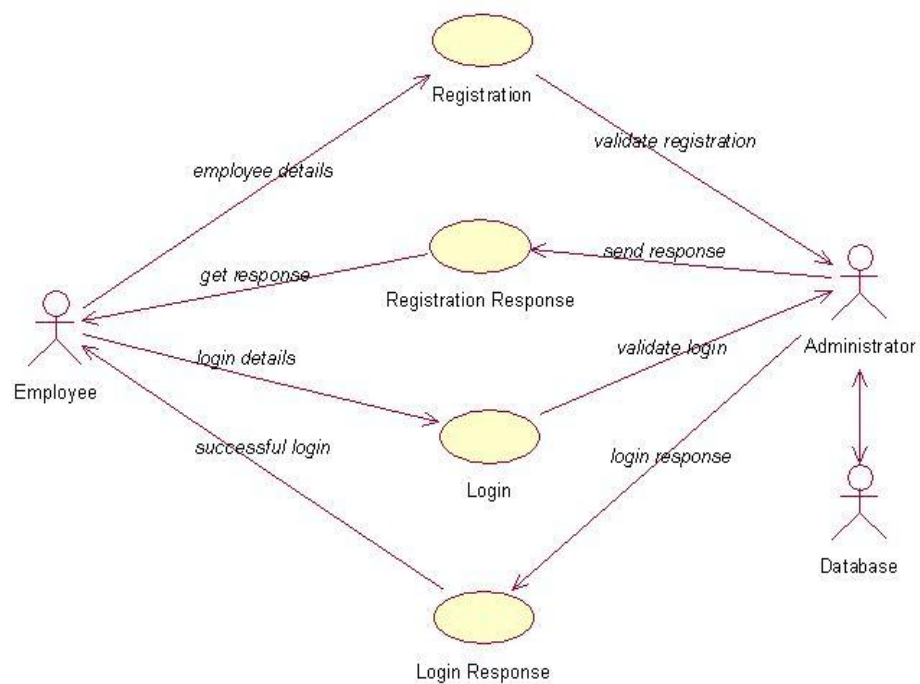


Fig. 5.14. Login Use case

1. User Registration is the initial Step.
2. Administrator will verify the details.
3. Data Base Manager will store the detail into the database.
4. Database Manager will acknowledge the Administrator.
5. Now the Administrator will send Register successfully Message to the user.
6. Now the user can login to the system.

5.3. Control Flow Diagrams

A control flow diagram consists of a subdivision to show sequential steps, with if-then-else conditions, repetitions, and/or case conditions. The activity diagram is a control flow diagram.

5.3.1. Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities of stepwise activities and actions with support for choice, iteration and concurrency.

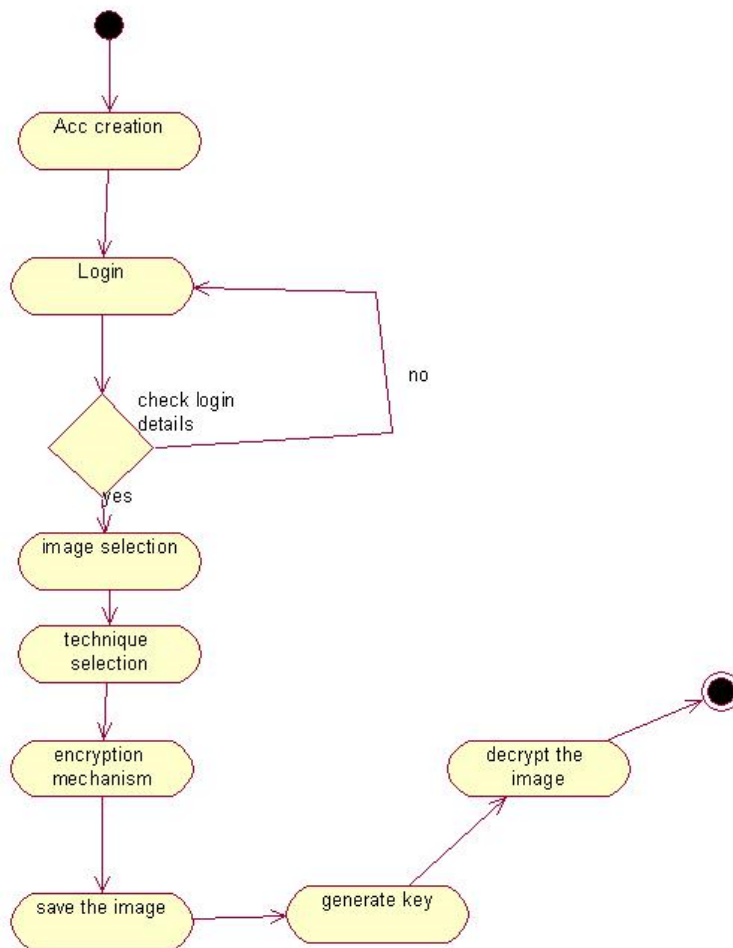


Fig 5.14. Overall Activity Diagram

5.4. Database Design

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database.

5.4.1. E-R Diagram

An entity-relationship model is an abstract and conceptual representation of data. This is a database modeling method, used to produce a type of conceptual schema or schematic data model of a system.

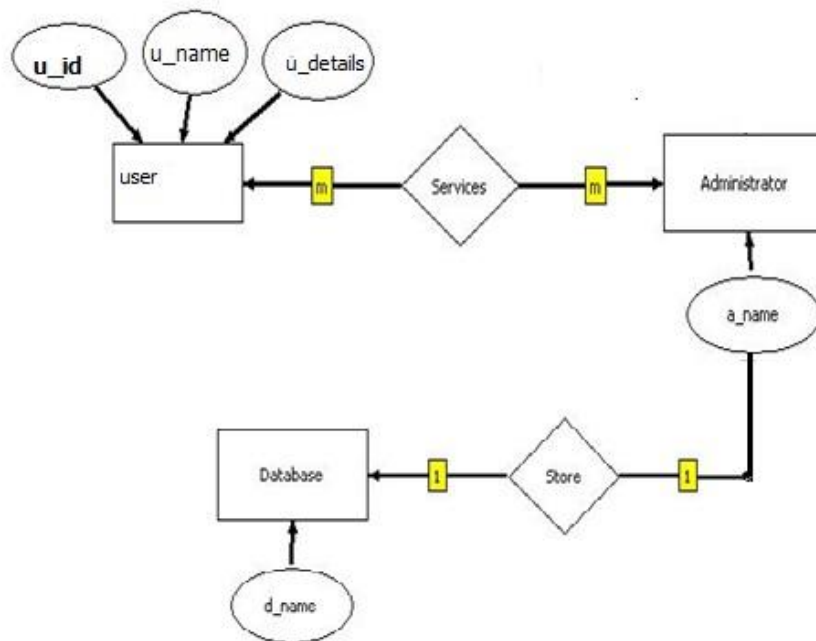


Fig. 5.15. E-R Diagram

5.4.2. Tables

The format of the database tables created for this project is:

emp_login

Column Name	Data Type	Constraint Type
usr_id	Varchar(10)	Primary Key
usr_pwd	Varchar(10)	Not Null
usr_name	Varchar(20)	Not Null
usr_dob	Varchar(20)	Not Null
usr_sex	Varchar(6)	Not Null
usr_mar_stat	Varchar(15)	Not Null
usr_email	Varchar(30)	Not Null
usr_phone	Varchar(12)	Not Null
usr_add	Varchar(100)	Not Null
usr_city	Varchar(20)	Not Null
usr_state	Varchar(20)	Not Null
usr_zip	Varchar(6)	Not Null

Table 5.1 User Details

CHAPTER-6

Coding/Code Templates

Java Server Pages

JSP not only enjoys cross-platform and cross-Web-server support, but effectively melds the power of server-side Java technology with features of static HTML pages.

JSP pages typically comprise of:

- Static HTML / XML components.
- Special JSP tags.
- Optionally, snippets of code written in the java programming language called “script lets.”

JSP Advantages

- **Separation of static from dynamic content:** In JSP, the logic to generate the dynamic content is kept separate from the static presentation templates by encapsulating it within external Java beans components. When a page designer makes any changes to the presentation template, the JSP page is automatically recompiled and reloaded into the web server by the JSP engine.
- **Write Once Run Anywhere:** JSP technology brings the “Write Once, Run anywhere” paradigm to interactive Web pages.
- **Dynamic content can be served in a variety of formats:** There is nothing that mandates the static template data within a JSP page to be of a certain format.

JSP Architecture

The purpose of JSP is to provide a declarative, presentation-centric method of developing servlets. JSP pages are subject to a translation phase and a request-processing phase. The translation phase is carried out only once, unless

the JSP page changes, in which case it is repeated. The JSP engine itself typically carries out the translation phase, when it receives a request for the JSP page for the first time.

Life Cycle of A JSP Life cycle of a JSP consists of the following three methods:

- `_jspInit`
- `_jspService`
- `_jspDestroy`

JSP Syntax

Directives

JSPs can define information for the container with directives. Here is what directives look like in a general form:

```
<%@ directive attribute="some Value" attribute="another Value" ...
%>
```

There are three directives:

- `<%@ page ... %>` specifies information that affects the page
- `<%@ include ... %>` includes a file at the location of the include directive (parsed)
- `<%@ taglib ... %>` allows the use of custom tags in the page

Declarations

Declarations are used to specify supplemental methods and variables. You can think of these as the page's private functions; they can only be called by the JSP where they are defined, or by another JSP that includes it (using the `<%@ include %>` directive).

Here is a sample declaration:

```
<%! // this integer can be used anywhere in this JSP page
private int myVariable = -1; // this function can be called from anywhere in this JSP
page
public boolean isPositive() {
    return ( myVariable > 0 );
}
%>
```

Scriptlets

Scriptlets are bits of Java code. They can do anything but they will most likely concentrate on generating HTML code or setting up variables to be part of later expressions.

Expressions

Expressions are special-purpose mini-Scriptlets used for evaluating expressions. This could be something as simple as outputting the value of a variable, or a more complicated Java expression, like calling a function and outputting the result. `<%= counter %>` Note that `counter` is defined as an `int`, but we do not need to explicitly convert it to a string.

Servlets

A servlet is a java programming language class that is used to extend the capabilities of servers that host applications access via a request-response programming mode. Servlets are Java technology's answer to Common Gateway Interface (CGI) Programming. They are programs that run on a Web server, acting as middle layer between request coming from a Web browser or other HTTP client and databases of applications on the HTTP server.

Read any data sent by the user: This data usually entered in a form on a Web page, but could also come from a java applet or a custom HTTP client program.

Look up any other information about the request that is embedded in the HTTP request: This information includes details about browser capabilities, cookies, the host name of the requesting client, and so forth.

Generate the results: This process may require talking to a database, executing an RMI or CORBA call, invoking a legacy application, or computing the response directly.

Format the results inside a document: In most cases, this involves embedding the information inside an HTML page.

Set the appropriate HTTP response parameters: This means telling the browser what type of document is being returned (e.g.HTML), setting cookies and caching parameters, and other such tasks.

Send the document back to the client: This document may be sent in text format (HTML), binary format (GIF images), or even in a compressed format like gzip that is layered on top of some other underlying format.

The `Javax.servlet` and `javax.servlet.http` packages provide interfaces and classes for writing servlets. All servlets must implement the `Servlet` interface, which defines life-cycle methods. When implementing a generic service, you can use or extend the `GenericServlet` class provided with the `java Servlet API`. The `HttpServlet` classes provide methods, such as `doGet` and `do Post`, for handling HTTP-specific services.

To be a servlet, a class should extend `HttpServlet` and override `doGet` or `do Post` (or both), depending on whether the data is being sent by `GET` or by `POST`. These methods take two arguments: An `HttpServletRequest` and an `HttpServletResponse`. The `HttpServletRequest` have methods that let you find out about incoming information such as FORM data, HTTP request headers, and the like. Finally, note that `doGet` and `do Post` are called by the `service` method, and sometimes you may want to override `service` directly.

Servlet Life Cycle: The life cycle of a servlet is controlled by the container in which the servlet has been deployed. When a request is mapped to a servlet, the container performs the following steps.

1. If an instance of the servlet does not exist, the Web container:
 - Loads the servlet class.
 - Creates an instance of the Servlet class.
 - Initializes the servlet instance by calling the `init` method.
2. Invokes the service method, passing request and response objects.
 - If the container needs to remove the servlet, it finalizes the servlet by calling the servlet's `destroy` method.

Session Management

Many applications require that a series of requests from a client be associated with one another. Sessions are represented by an `Http Session` object. A session can be accessed by calling the `getSession()` method of a request object. This method returns the current session associated with this request, or, if the request does not have a session, it creates one. The timeout period can be accessed by using a session's `getMaxInactiveInterval()` method.

Session Tracking

A Web container can use several methods to associate a session with a user, all of which involve passing an identifier between the client and the server. The identifier can be maintained on the client as a cookie, or the Web component can include the identifier in every URL that is returned to the client.

In fact, on many servers, they use cookies if the browser supports them, but automatically revert to URL-rewriting when cookies are unsupported or explicitly disabled.

The Session Tracking API

Using sessions in servlets is quite straightforward, and involves looking up the session object associated with the current request, creating a new session object when necessary, looking up information associated with a session, storing information in a session, and discarding completed or abandoned sessions.

Algorithm Implementation

Blowfish

Blowfish is a variable-length key block cipher. It does not meet all the requirements for a new cryptographic standard discussed above: it is only suitable for applications where the key does not change often, like a communications link or an automatic file encryptor. It is significantly faster than DES when implemented on 32-bit microprocessors with large data caches, such as the Pentium and the PowerPC.

DESCRIPTION OF THE ALGORITHM

Blowfish is a variable-length key, 64-bit block cipher. The algorithm consists of two parts: a key-expansion part and a data- encryption part. Key expansion converts a key of at most 448 bits into several subkey arrays totaling 4168 bytes.

Data encryption occurs via a 16-round Feistel network. Each round consists of a key-dependent permutation, and a key- and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

Subkeys:

Blowfish uses a large number of subkeys. These keys must be precomputed before any data encryption or decryption.

1. The P-array consists of 18 32-bit subkeys:
P1, P2,..., P18.
2. There are four 32-bit S-boxes with 256 entries each:
S1,0, S1,1,..., S1,255;
S2,0, S2,1,..., S2,255;
S3,0, S3,1,..., S3,255;
S4,0, S4,1,..., S4,255.

The exact method used to calculate these subkeys will be described later.

Encryption:

Blowfish is a Feistel network consisting of 16 rounds (see Figure 1). The input is a 64-bit data element, x .

Divide x into two 32-bit halves: x_L , x_R

For $i = 1$ to 16:

$x_L = x_L \text{ XOR } P_i$

$x_R = F(x_L) \text{ XOR } x_R$

Swap x_L and x_R

Swap x_L and x_R (Undo the last swap.)

$x_R = x_R \text{ XOR } P_{17}$

$x_L = x_L \text{ XOR } P_{18}$

Recombine x_L and x_R

Function F

Divide x_L into four eight-bit quarters: a , b , c , and d

$F(x_L) = ((S1,a + S2,b \bmod 232) \text{ XOR } S3,c) + S4,d \bmod 232$

Decryption is exactly the same as encryption, except that P_1, P_2, \dots, P_{18} are used in the reverse order. Implementations of Blowfish that require the fastest speeds should unroll the loop and ensure that all subkeys are stored in cache.

Generating the Subkeys:

The subkeys are calculated using the Blowfish algorithm. The exact method is as follows:

1. Initialize first the P -array and then the four S -boxes, in order, with a fixed string. This string consists of the hexadecimal digits of π (less the initial 3). For example:

$P1 = 0x243f6a88$

$P2 = 0x85a308d3$

$P3 = 0x13198a2e$

$P4 = 0x03707344$

2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.)
3. Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).
4. Replace P1 and P2 with the output of step (3).
5. Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.
6. Replace P3 and P4 with the output of step (5).
7. Continue the process, replacing all entries of the P- array, and then all four S-boxes in order, with the output of the continuously-changing Blowfish algorithm.

In total, 521 iterations are required to generate all required subkeys. Applications can store the subkeys rather than execute this derivation process multiple times.

MINI-BLOWFISH

The following mini versions of Blowfish are defined solely for cryptanalysis. They are not suggested for actual implementation. Blowfish-32 has a 32-bit block size and subkey arrays of 16-bit entries (each S-box has 16 entries). Blowfish-16 has a 16-bit block size and subkey arrays of 8-bit entries (each S-box has 4 entries).

DESIGN DECISIONS

The underlying philosophy behind Blowfish is that simplicity of design yields an algorithm that is both easier to understand and easier to implement. Through the use of a streamlined Feistel network--a simple S-box substitution and a simple P-box substitution--I hope that the design will not contain any flaws.

A 64-bit block size yields a 32-bit word size, and maintains block-size compatibility with existing algorithms. Blowfish is easy to scale up to a 128-bit block, and down to smaller block sizes. Cryptanalysis of the mini-Blowfish variants may be significantly easier than cryptanalysis of the full version.

The fundamental operations were chosen with speed in mind. XOR, ADD, and MOV from a cache are efficient on both Intel and Motorola architectures. All subkeys fit in the cache of a 80486, 68040, Pentium, and PowerPC.

The Feistel network that makes up the body of Blowfish is designed to be as simple as possible, while still retaining the desirable cryptographic properties of the structure. Figure 3 is round i of a general Feistel network: $R_{n,i}$ are reversible functions of text and key, and N_i is a non-reversible function of text and key. For speed and simplicity, I chose XOR as my reversible function. This let me collapse the four XORs into a single XOR, since:

$$R_{-1,i+1} = R_{1,i+1} \text{ XOR } R_{2,i-1} \text{ XOR } R_{3,i} \text{ XOR } R_{4,i}$$

This is the P-array substitution in Blowfish. The XOR can also be considered to be part of the non-reversible function, N_i , occurring at the end of the function. (Although equivalent, I chose not to illustrate them in this way because it simplifies description of the subkey-generation process.) There are two XORs that remain after this reduction: R_1 in the first round and R_2 in the last round. I chose not to eliminate these in order to hide the input to the first non-reversible function.

I considered a more complicated reversible function, one with modular multiplications and rotations. However, these operations would greatly increase the algorithm's execution time. Since function F is the primary source of the algorithm's security, I decided to save time-consuming complications for that function.

Function F , the non-reversible function, gives Blowfish the best possible avalanche effect for a Feistel network: every text bit on the left half of the round affects every text bit on the right half. Additionally, since every subkey bit is affected by every key bit, the function also has a perfect avalanche effect between the key and the right half of the

text after every round. Hence, the algorithm exhibits a perfect avalanche effect after three rounds and again every two rounds after that.

I considered adding a reversible mixing function, more complicated than XOR, before the first and after the last round. This would further confuse the entry values into the Feistel network and ensure a complete avalanche effect after the first two rounds. I eventually discarded the addition as a time-consuming complication with no clear cryptographic benefits.

The non-reversible function is designed for strength, speed, and simplicity. Ideally, I wanted a single S-box with 232 32-bit words, but that was impractical. My eventual choice of 256-entry S-boxes was a compromise between my three design goals. The small-number of bits to large-number of bits may have weaknesses with respect to linear cryptanalysis, but these weaknesses are hidden both by combining the output of four S-boxes and making them dependent on the key.

We used four different S-boxes instead of one S-box primarily to avoid symmetries when different bytes of the input are equal, or when the 32-bit input to function F is a bitwise permutation of another 32-bit input. I could have used one S-box and made each of the four different outputs a non-trivial permutation of the single output, but the four S-box design is faster, easier to program, and seems more secure.

The function that combines the four S-box outputs is as fast as possible. A simpler function would be to XOR the four values, but mixing addition mod 232 and XOR combines two different algebraic groups with no additional instructions. The alternation of addition and XOR ends with an addition operation because an XOR combines the final result with xR.

If the four indexes chose values out of the same S-box, a more complex combining function would be required to eliminate symmetries. I considered using a more complex combining function in Blowfish (using modular multiplications, rotations, etc.), but chose not to because the added complication seemed unnecessary.

The key-dependent S-boxes protect against differential and linear cryptanalysis. Since the structure of the S-boxes is completely hidden from the cryptanalyst, these attacks have a more difficult time exploiting that structure. While it would be possible to replace these variable S-boxes with four fixed S-boxes that were designed to be resistant to these attacks, key-dependent S-boxes are easier to implement and less susceptible to arguments of "hidden" properties. Additionally, these S-boxes can be created on demand, reducing the need for large data structures stored with the algorithm.

Each bit of x_L is only used as the input to one S-box. In DES many bits are used as inputs to two S-boxes, which strengthen the algorithm considerably against differential attacks. I feel that this added complication is not as necessary with key-dependent S-boxes. Additionally, larger S-boxes would take up considerably more memory space.

Function F does not depend on the iteration. I considered adding this dependency, but did not feel that it had any cryptographic merit. The P -array substitution can be considered to be part of this function, and that is already iteration-dependent.

The number of rounds is set at 16 primarily out of desire to be conservative. However, this number affects the size of the P -array and therefore the subkey-generation process; 16 iterations permits key lengths up to 448 bits. I expect to be able to reduce this number, and greatly speed up the algorithm in the process, as I accumulate more cryptanalysis data.

In algorithm design, there are two basic ways to ensure that the key is long enough to ensure a particular security level. One is to carefully design the algorithm so that the entire entropy of the key is preserved, so there is no better way to crypt analyze the algorithm other than brute force. The other is to design the algorithm with so many key bits that attacks that reduce the effective key length by several bits are irrelevant. Since Blowfish is designed for large microprocessors with large amounts of memory, I chose the latter.

The subkey generation process is designed to preserve the entire entropy of the key and to distribute that entropy uniformly throughout the subkeys. It is also designed to distribute the set of allowed subkeys randomly throughout the domain of possible subkeys. I chose the digits of pi as the initial subkey table for two reasons: because it is a random sequence not related to the algorithm, and because it could either be stored as part of the algorithm or derived when needed. There is nothing sacred about pi; any string of random bits--digits of e, RAND tables, and output of a random number generator--will suffice. However, if the initial string is non-random in any way (for example, ASCII text with the high bit of every byte a 0), this non-randomness will propagate throughout the algorithm.

In the subkey generation process, the subkeys change slightly with every pair of subkeys generated. This is primarily to protect against any attack of the subkey generation process that exploits the fixed and known subkeys. It also reduces storage requirements. The 448 limit on the key size ensures that the every bit of every subkey depends on every bit of the key. (Note that every bit of P15, P16, P17, and P18 does not affect every bit of the cipher text, and that any S-box entry only has a .06 probability of affecting any single cipher text block.)

The key bits are repeatedly XORed with the digits of pi in the initial P-array to prevent the following potential attack: Assume that the key bits are not repeated, but instead padded with zeros to extend it to the length of the P-array. An attacker might find two keys that differ only in the 64-bit value XORed with P1 and P2 that, using the initial known subkeys, produce the same encrypted value. If so, he can find two keys that produce all the same subkeys. This is a highly tempting attack for a malicious key generator.

To prevent this same type of attack, I fixed the initial plaintext value in the subkey-generation process. There is nothing special about the all-zeros string, but it is important that this value be fixed.

The subkey-generation algorithm does not assume that the key bits are random. Even highly correlated key bits, such as an alphanumeric ASCII string with the bit of every

byte set to 0, will produce random subkeys. However, to produce subkeys with the same entropy, a longer alphanumeric key is required.

The time-consuming subkey-generation process adds considerable complexity for a brute-force attack. The subkeys are too long to be stored on a massive tape, so they would have to be generated by a brute-force cracking machine as required. A total of 522 iterations of the encryption algorithm are required to test a single key, effectively adding 29 steps to any brute-force attack.

POSSIBLE SIMPLIFICATIONS

Here we are exploring several possible simplifications, aimed at decreasing memory requirements and execution time. These are outlined below:

Fewer and smaller S-boxes. It may be possible to reduce the number of S-boxes from four to one. Additionally, it may be possible to overlap entries in a single S-box: entry 0 would consist of bytes 0 through 3, entry 1 would consist of bytes 1 through 4, etc. The former simplification would reduce the memory requirements for the four S-boxes from 4096 bytes to 1024 bytes, the latter would reduce the requirements for a single S-box from 1024 bytes to 259 bytes. Additional steps may be required to eliminate the symmetries that these simplifications would introduce. Additionally, four different 10- or 12-bit indexes into a single large S-box could be used instead of the current series of S-boxes.

Fewer iterations. It is probably safe to reduce the number of iterations from 16 to 8 without compromising security. The number of iterations required for security may be dependent on the length of the key. Note that with the current subkey generation procedure, an 8-iteration algorithm cannot accept a key longer than 192 bits.

On-the-fly subkey calculation. The current method of subkey calculation requires all subkeys to be calculated advance of any data encryption. In fact, it is impossible to calculate the last subkey of the last S-box without calculating every subkey that comes before. An alternate method of subkey calculation would be preferable: one where every

subkey can be calculated independently of any other. High-end implementations could still precompute the subkeys for increased speed, but low-end applications could only compute the required subkeys when needed.

Encryption

Encryption is the process of transforming information from an unsecured form ("clear" or "plaintext") into coded information ("cipher text") that cannot be easily read by outside parties. An algorithm and a key control the transformation process. The process must be reversible so that the intended recipient can return the information to its original, readable form, but reversing the process without the appropriate encryption information should be impossible. This means that details of the key must also be kept secret.

Encryption is generally regarded as the safest method of guarding against accidental or purposeful security breaches. The strength of the encryption method is often measured in terms of work factor - the amount of force that is required to 'break' the encryption. A strong system will take longer to break, although applying greater force can reduce this (the more effort that is put into the attack, the less time required to break the code).

The main characteristics of private key cryptosystem are as follows:

- 1) In private key encryption, the same key is used for both encryption and decryption. The key must be kept secret so that unauthorized parties cannot, even with knowledge of the algorithm, complete the decryption process.
- 2) Once the encryption part is carried out, the main next part is the decryption, where the cipher text has to be converted back into the original text, so that whole process of file transfer is implemented. The receiver is interested in receiving only the original text, therefore decryption plays a vital role in this project.

CHAPTER-7

7.1 TESTING

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected.

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

7.1.1. Unit Testing

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

Each module can be tested using the following two strategies:

i. Black Box Testing:

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been used to find errors in the following categories:

- a) Incorrect or missing functions
- b) Interface errors
- c) Errors in data structure or external database access
- d) Performance errors
- e) Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

ii. White Box testing

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases.

It has been uses to generate the test cases in the following cases:

- a) Guarantee that all independent paths have been executed.
- b) Execute all logical decisions on their true and false sides.
- c) Execute all loops at their boundaries and within their operational bounds.
- d) Execute internal data structures to ensure their validity.

7.1.2. Integrating Testing

Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

The system was tested successfully for the integration between different modules like Encryption Module, Decryption Module and File Transfer module. It was found that there was an good integration between all the modules.

7.1.3. System Testing

Involves the in house testing of the entire system before delivery to the user. Its aim is to satisfy the user the system meets all requirements of the client's specifications.

7.1.4. Acceptance Testing

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

7.1.5 Validation Testing

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirement specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed. The system was successfully tested for wrong password while decryption and it does not display the original image and will generate an error message about password.

7.1.6 Compilation Testing

It was a good idea to do our stress testing early on, because it gave us time to fix some of the unexpected deadlocks and stability problems that only occurred when components were exposed to very high transaction volumes.

7.1.7 Execution Testing

This program was successfully loaded and executed. Because of good programming there were no execution errors.

7.1.8 Output Testing

The output of the project was up to the expectations and the successful output screens are placed in the output screens section below.

CHAPTER-8

Output Screens

8.1 Home Page

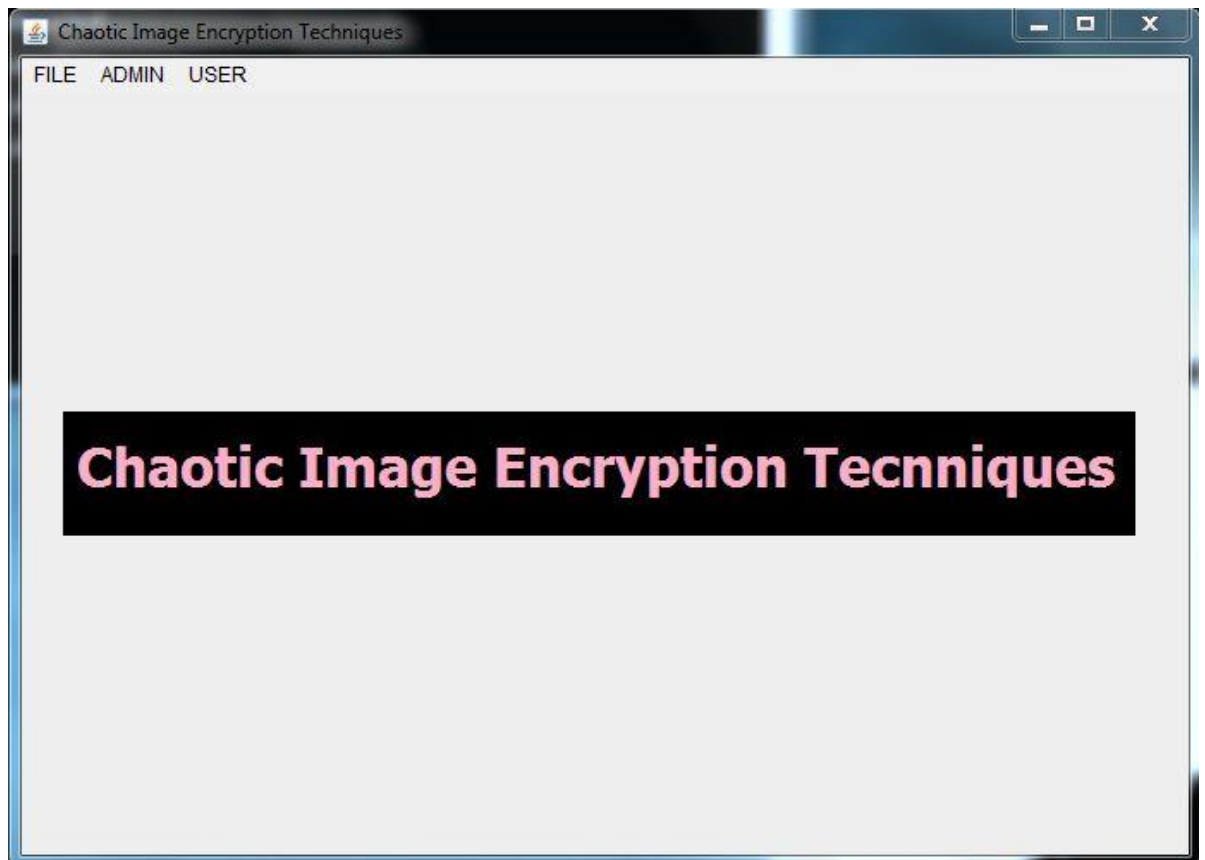


Fig. 8.1 Home page

Description:

This home page contains description login page and services provided by the project.

8.2 Selection of input and output files

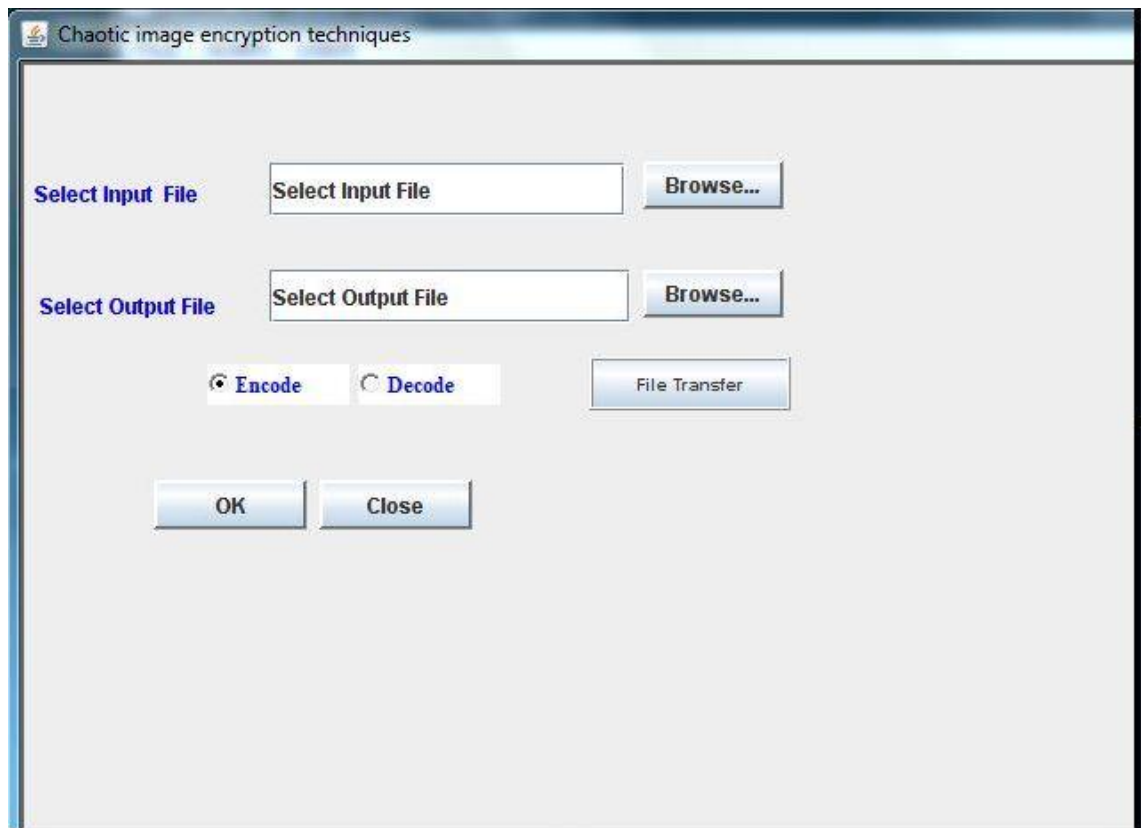


Fig. 8.2 Selection of input and output files

Description:

This page provides the view of the selection of both the input and output file.

8.3 Browse the input file

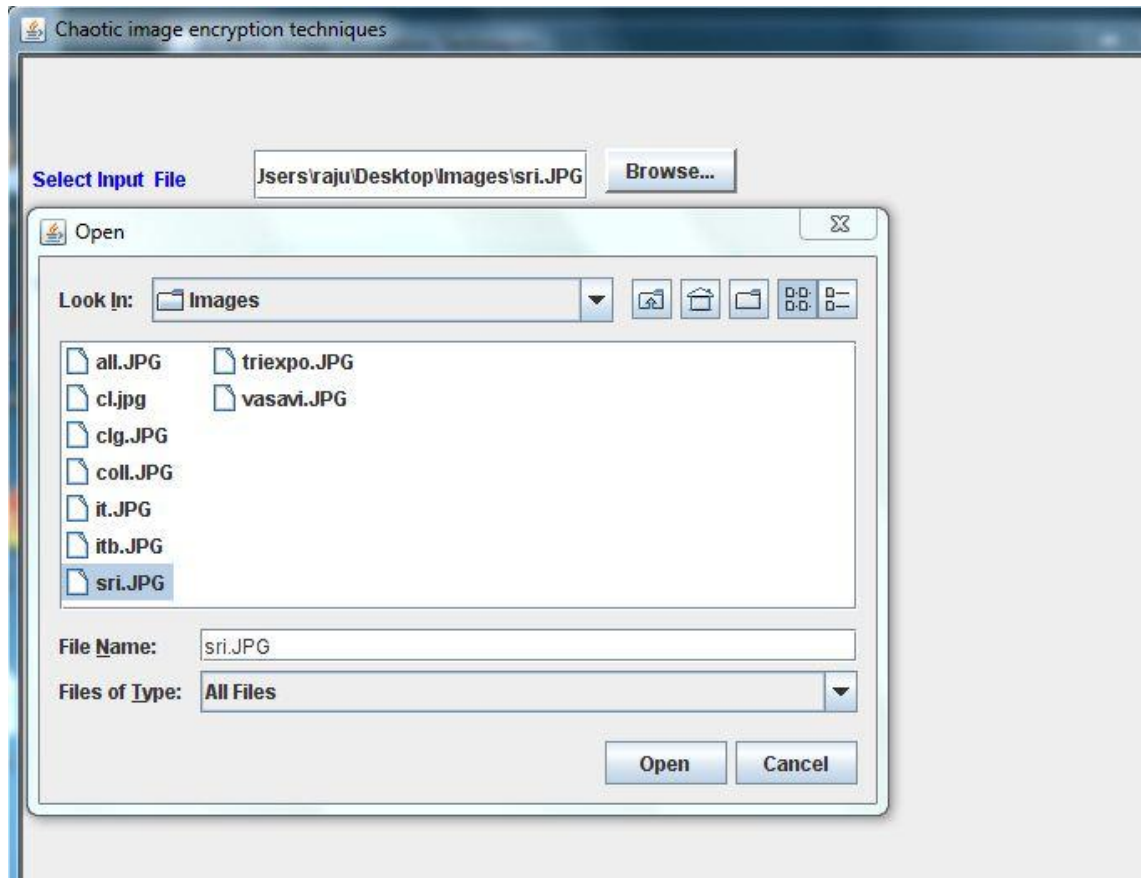


Fig. 8.3 Browse the Input file

Description:

This page provides the view of the browsing for the input file.

8.4 Browse the output folder

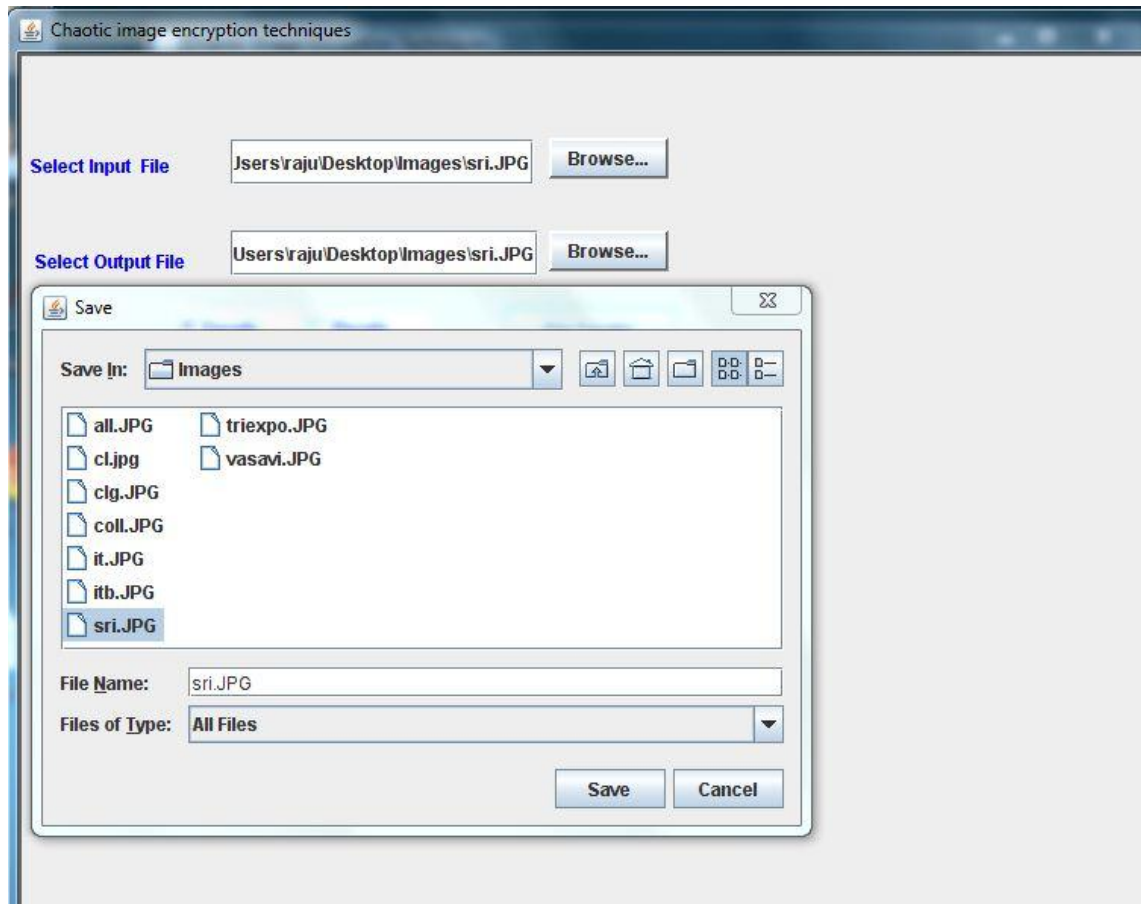


Fig. 8.4 Browse the Output file

Description:

This page provides the view of the browsing for the output folder.

8.5 Encoding the image with password

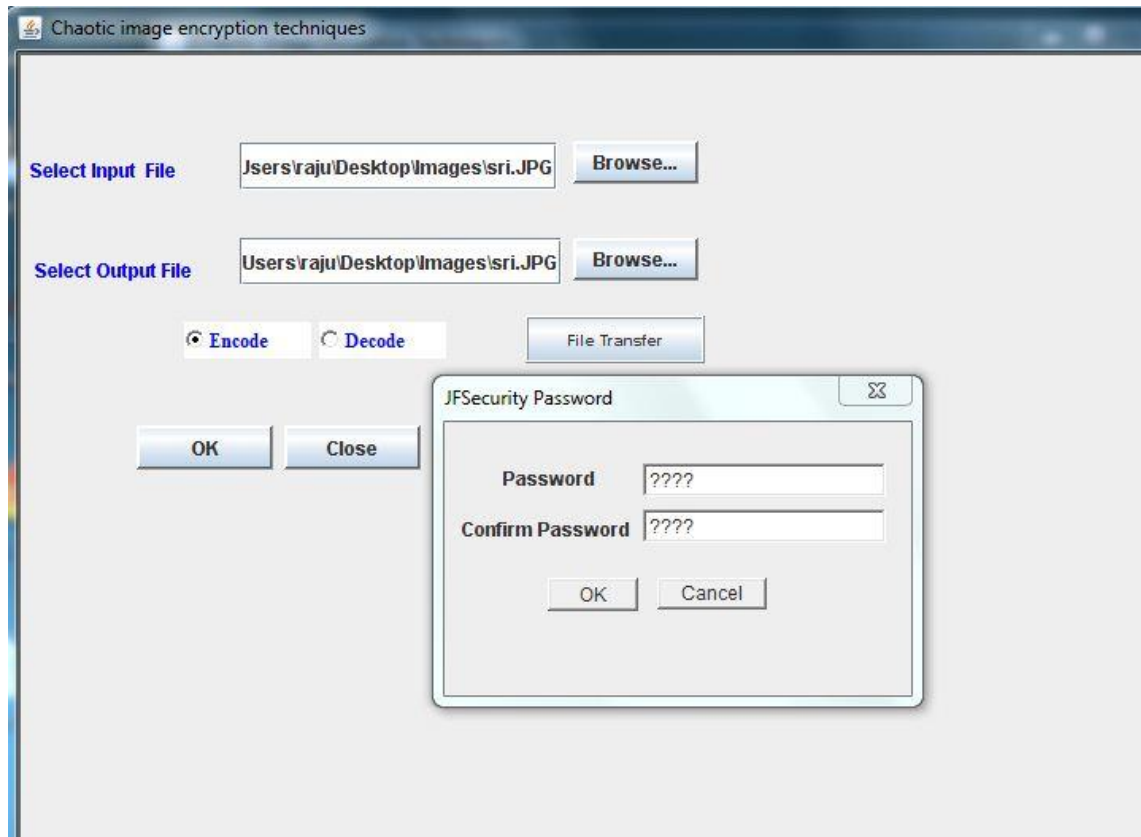


Fig. 8.5 Encoding Image

Description:

To encrypt the image select the image from list and click on encrypt button. It will generate an encryption technique window, in which the output path location is specified. The user can change the output location. It also contains the encryption password.

8.6 Status of the Encryption method

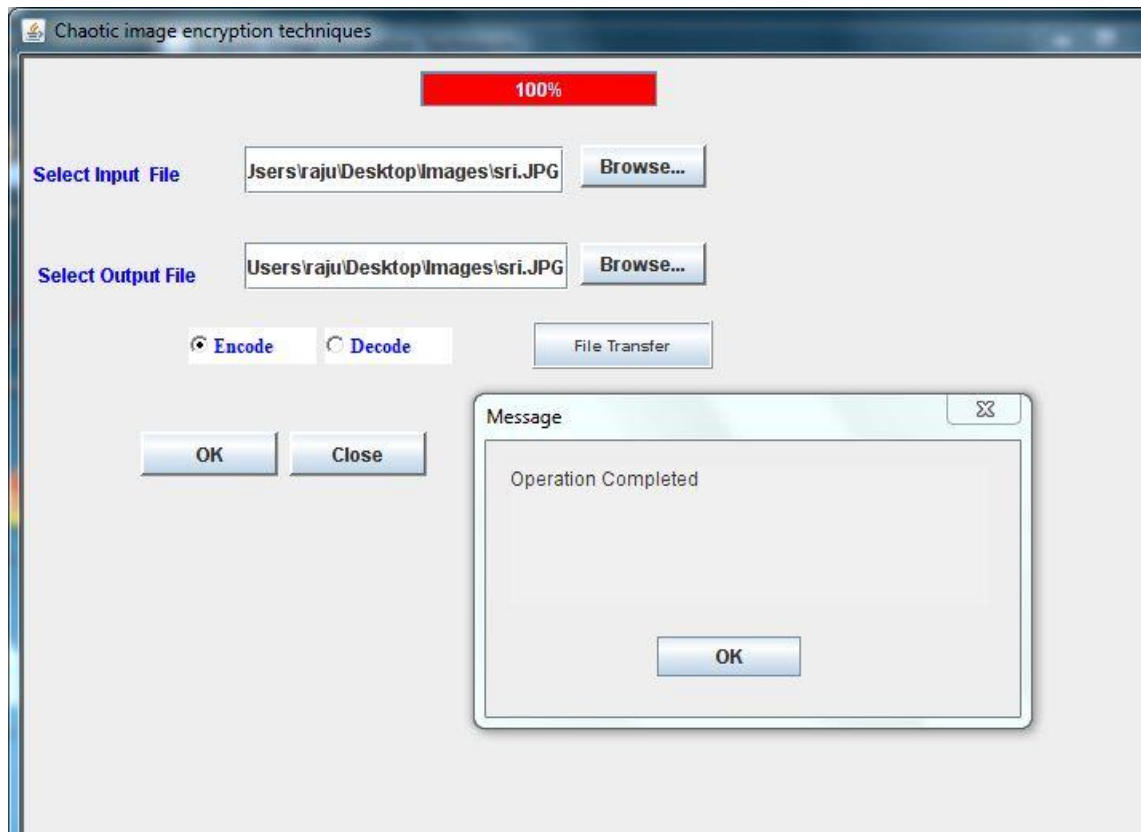


Fig. 8.6 Status of Encoding Mechanism

Description:

This page provides the view status of the Encryption Mechanism.

8.7 File Transfer Module

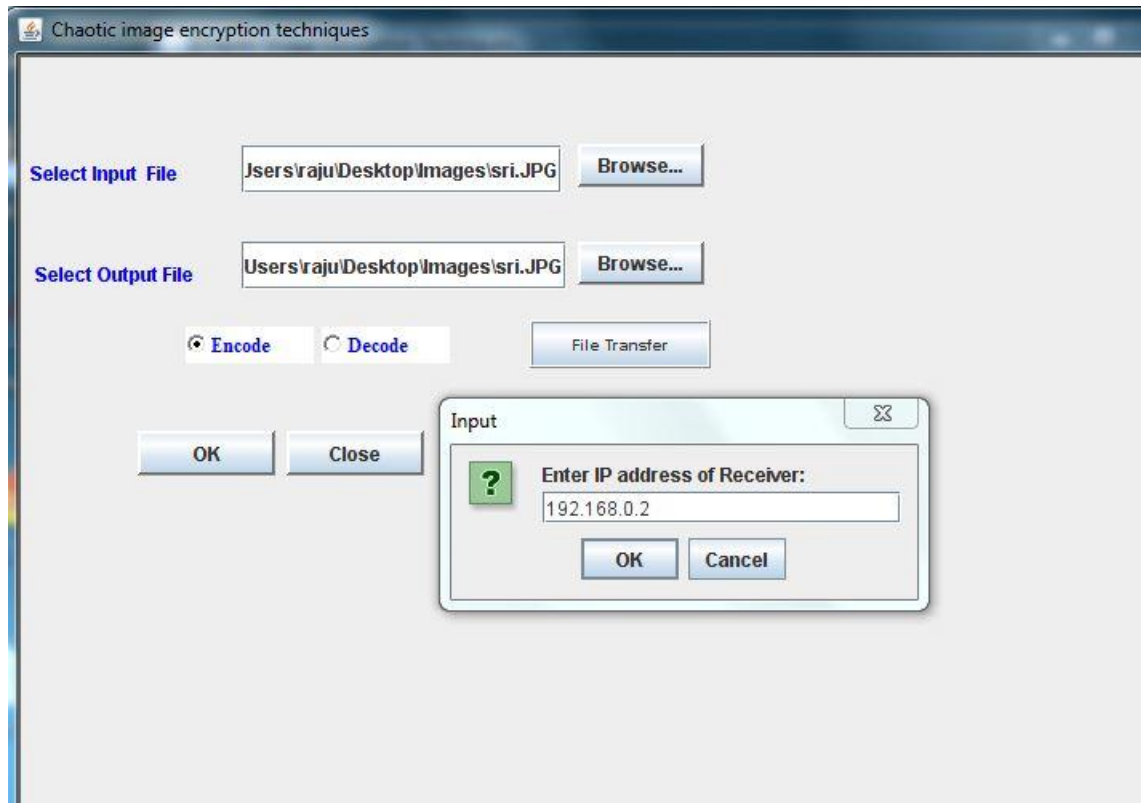


Fig.8.7 File Transfer Module

Description:

This page provides the view of the entering IP address for file transfer module.

8.8 Conforming the Host System

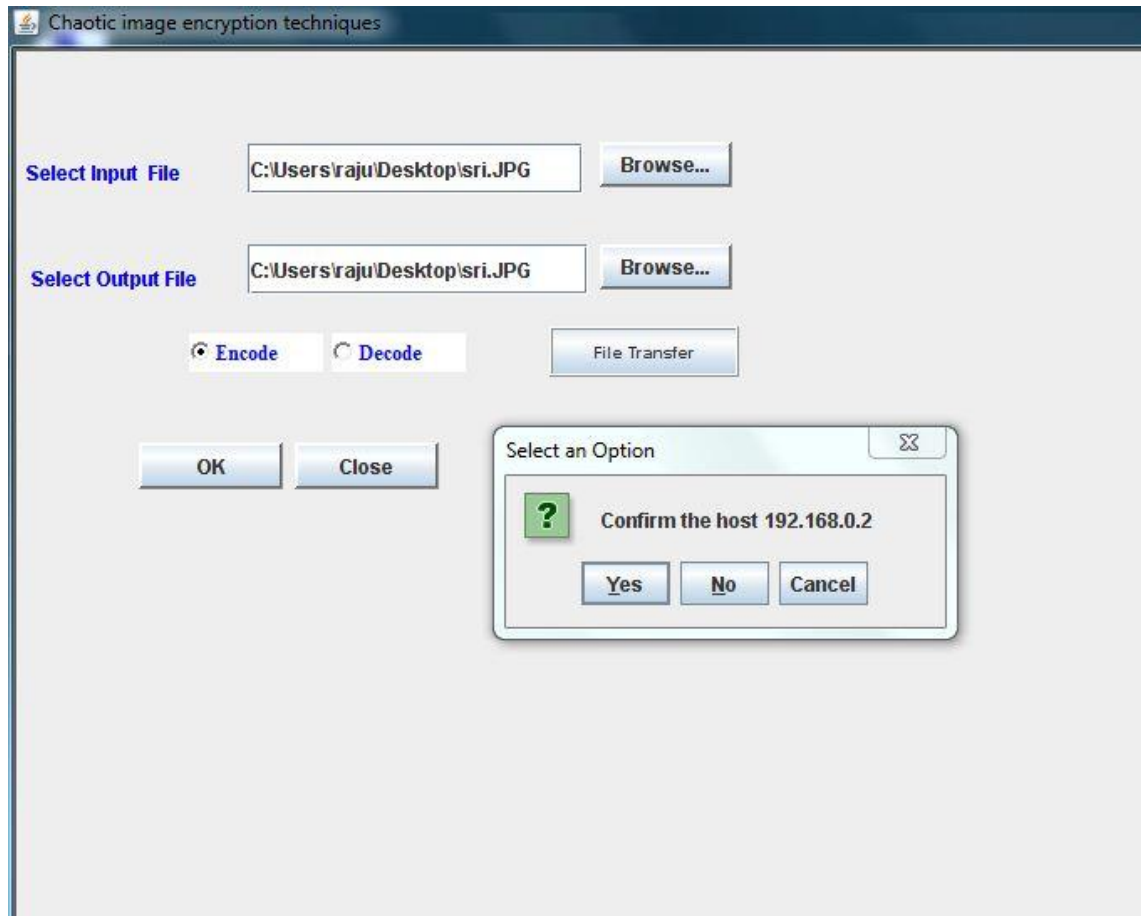
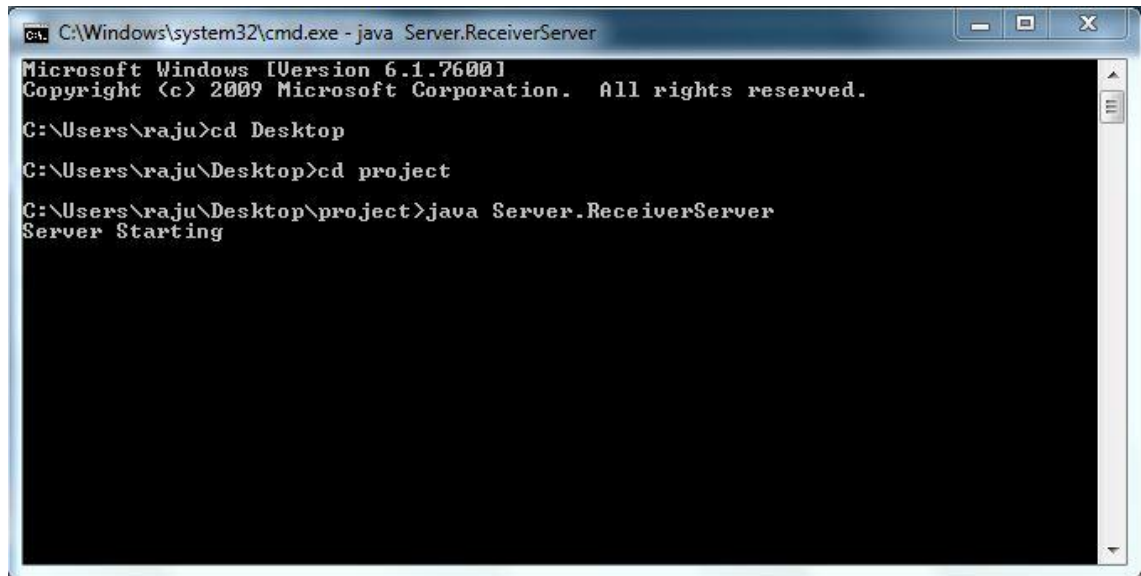


Fig. 8.8 Conforming the Host System

Description:

This page provides the view of conforming the host system.

8.9 Starting Server at Client Side



```
C:\Windows\system32\cmd.exe - java Server.ReceiverServer
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\raju>cd Desktop
C:\Users\raju\Desktop>cd project
C:\Users\raju\Desktop\project>java Server.ReceiverServer
Server Starting
```

Fig. 8.9 Starting Server

Description:

This page provides the view of starting server at the receiver side.

8.10 Status of the file in Network

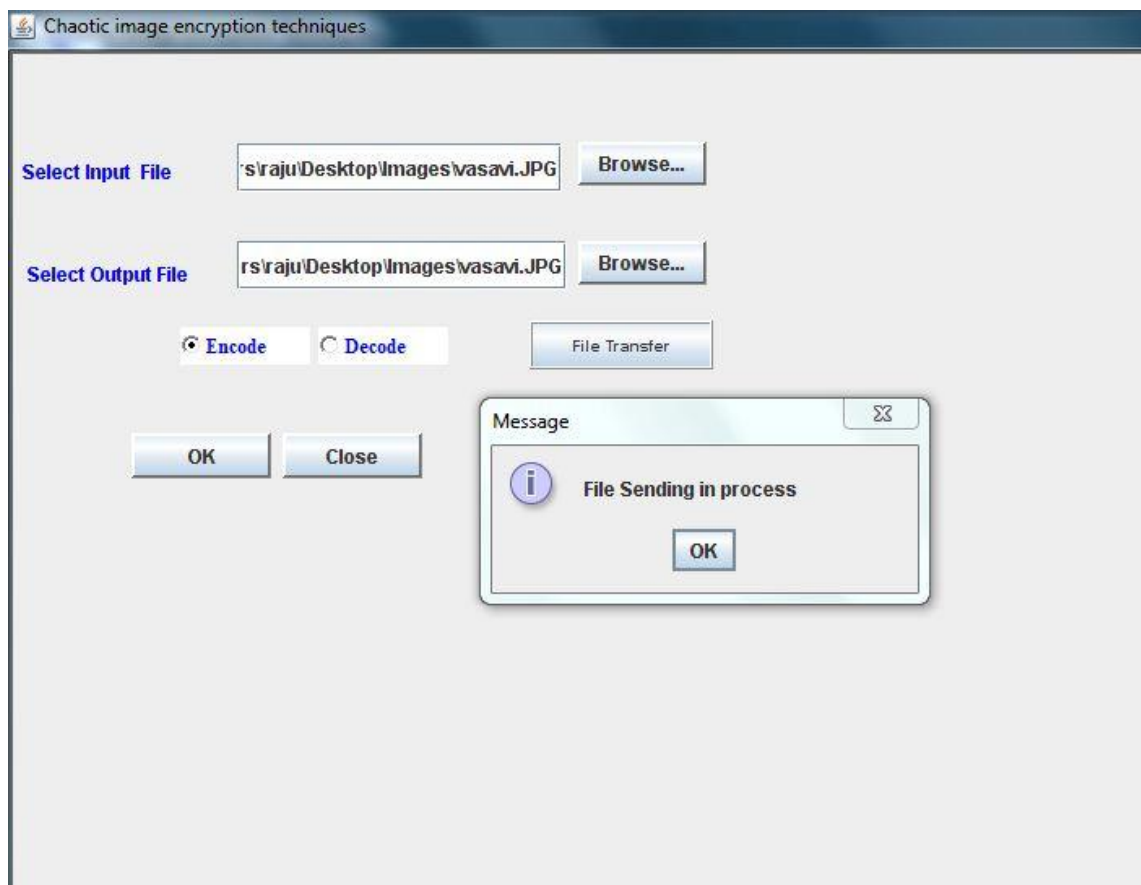


Fig. 8.10 Status of Image in Network

Description:

This page provides the view of status of the sending image in network.

8.11 File Received Status

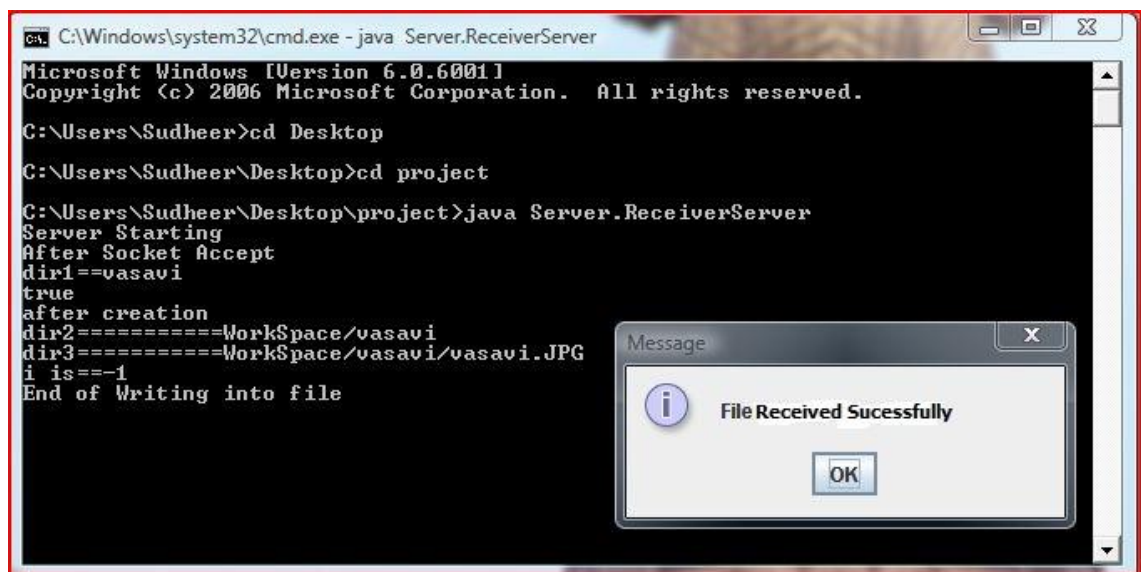


Fig. 8.11 File Status at Received Side

Description:

This page provides the view of sent image received at the host side successfully.

8.12 Decrypting the received Encrypted Image

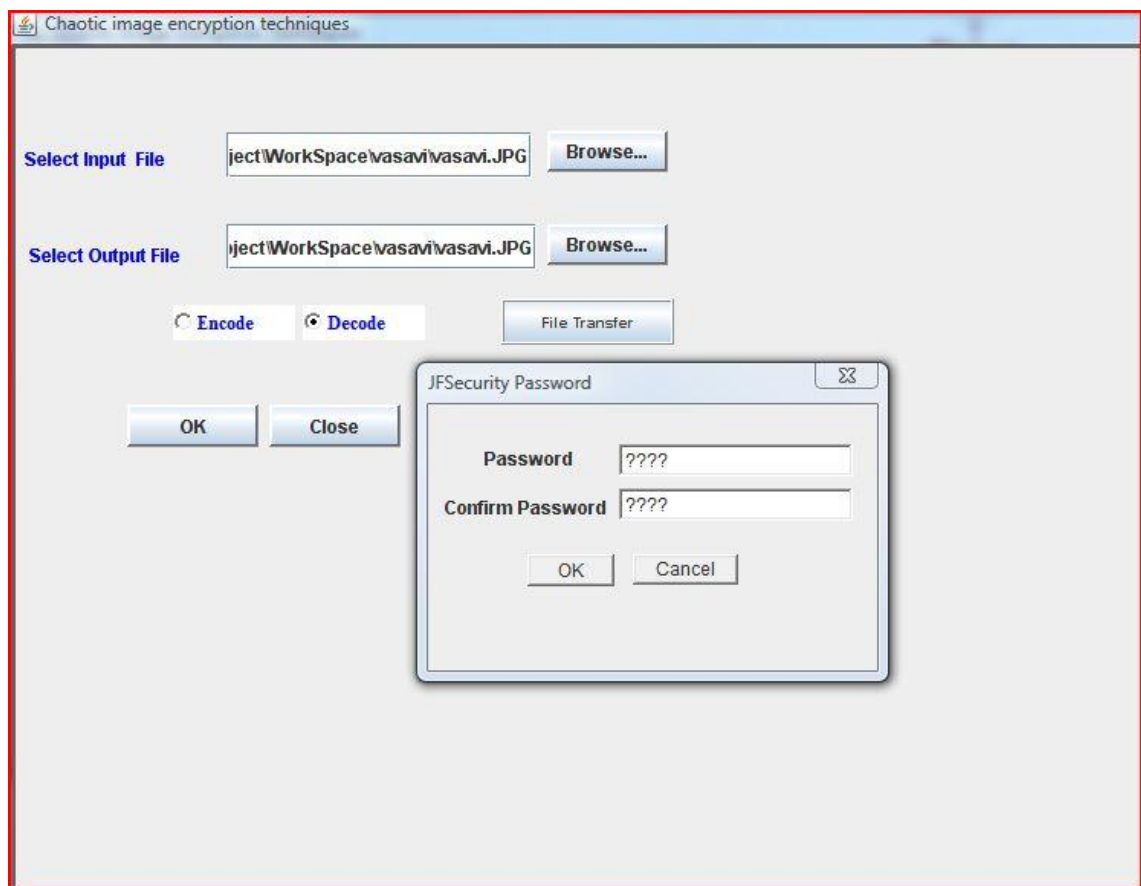


Fig. 8.12 Decrypting the Image

Description:

This page provides the view of the decrypting the image by entering the password for the image.

8.13 View of the Original Image



Fig. 8.13 View of the Original Image

Description:

This page provides the view of the decrypted original image at the received side.

CHAPTER-9

Conclusion

Advances in space science, data analysis, and communication technologies present new opportunities for users to increase productivity, reduce cost, facilitate innovation and create virtual collaborative environments for addressing new challenges. In such processes data sharing is usually based on CD/DVD-ROM hardcopy or on shared network environment (Internet, LAN, WAN etc) , so there exists inherent security risk of unauthorized access or use of the product. To fulfil such security and privacy needs in various applications, encryption of such data is very important to minimize malicious attacks from unauthorized parties and to safeguard sensitive data.

From the study of the above traditional and chaos-based image encryption techniques; it is quit clear that traditional image encryption techniques DES, Triple-DES and IDEA have some limitations such as these algorithms have high security level under CBC mode but requires large data size, long computational time and high computing power.

On the other hand chaos-based cryptographic scheme provides high security level, less computational time and power in reliable and efficient way to deal with balky, difficult and intractable data that why many researchers recommends that it is more suitable for multimedia data, especially for images . Chaos-based system have many properties to achieve high security level, such as sensitivity to change initial conditions and parameters, ergodicity (a system that tends in probability to a limiting form that is independent of the initial conditions), random behaviour and unstable periodic orbits with long periods. It has very high diffusion and confusion properties that are desirable for cryptosystem.

CHAPTER-10

Future Enhancements

This project represents a preliminary study of different mechanisms used for image protection highlighting comprehensive comparative overview of existing traditional image encryption techniques like DES, Triple-DES and IDEA algorithms. Some existing chaos-based image encryption schemes have also been briefly reviewed by studying and analyzing multiple algorithms properties, such as, encryption speed, compatibility to image format and compression standards, and real-time implementation etc.

Now a day's chaos-based encryption technique is getting more and more popularity worldwide to deal with multimedia data sets especially for images and is recommended by many researchers [7 -12]. The similar technique of chaos-based image encryption is planned to be applied on satellite imageries since chaos-based techniques have many proper-ties to achieve high security level in efficient and reliable manner, like sensitivity, ergodicity, randomness, and no recurring or reappearing orbits for long periods with high diffusion and confusion. The summary of the planed future work is the following:

Continue to study other proposed encryption techniques, theories and algorithms, such as chaos-based or chaotic cryptosystem, number theory, AES, DES, RSA etc., need to be understood.

Investigate and compare the security and robustness of some chaos-based and traditional encryption schemes for large data sets e.g. huge satellite images.

In-depth study general insecurity properties of satellite imagery and remotely sensed data, and try to propose some corresponding countermeasures, furthermore, design some new schemes that have good security properties and can meet the real application requirements at the same time.

CHAPTER-11

Bibliography

1. Mauro Barni, Franco Bartolini, Enrico Magli and Gabriella Olmo, "Watermarking techniques for electronic delivery of remote sensing images", *Optical Engineering*, 41, No. 9, pp. 2111-2119, September 2002.
2. Ray A. Williamson, "REMOTE SENSING AND TRANSPORTATION SECURITY", Pecora 15/Land Satellite Information IV/ISPRS Commission I/FIEOS Conference Proceedings, 2002
3. Klaus Holtz, "Advanced Data Compression promises the next big Leap in Network Performance", IS&T / SPIE EUROPTO Conference, Zurich, Switzerland, May 1998.
4. R. J. Anderson and F. A. P. Petitcolas, "On the limits of steganography," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 474-481, 1998
5. Mohammad Zakir Hossain Sarker and Md. Shafiul Parvez, "A Cost Effective Symmetric Key Cryptographic Algorithm for Small Amount of Data", *Proceedings of the 9th IEEE International Multi topic Conference*, pp. 1-6, December 2005
6. Xun Yi Chik How Tan Chee Kheong Slew Rahman Syed, M., "Fast encryption for multimedia," *IEEE Transactions on Consumer Electronics*, vol. 47, no. 1, pp. 101-107, 2001.
7. Yen J. C. and Guo J. I., "A new chaotic image encryption algorithm," *Proceeding of National Symposium on Telecommunications*, pp. 358-362, December 1998.
8. Jui-Cheng Yen and J. I. Guo, "A New Chaotic Mirror-Like Image Encryption Algorithm and its VLSI Architecture", *Pattern Recognition and Image Analysis*, vol.10, no.2, pp.236-247, 2000.
9. Jui-Cheng Yen and J. I. Guo, "Efficient Hierarchical Chaotic Image Encryption Algorithm and Its VLSI Realization". *IEEE Proceeding Vis. Image Signal Process*, vol. 147, no. 2, pp. 167-175, 2000