Using Biometrics to Generate Public and Private Keys

GRADUATE PROJECT REPORT

Submitted to the Faculty of
the Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Satrugna Pakala
Spring 2006

Committee Members

Dr. Mario Garcia
Committee Chairperson

Dr. David Thomas
Committee Member

Dr. Long-zhuang Li
Committee Member
ABSTRACT

In traditional cryptographic systems, user authentication is based on possession of secret keys. This system falls apart if the keys are not kept secret (i.e., shared with non legitimate users). Considering the need for heightened security all over the world, just having a secret key is not a viable solution. Most of the secret keys are shared, written down, forgotten, lost or stolen, or just not complex enough. This is where biometrics can help because biometrics isn’t tied to a secret key but to the physical body demographics or behavioral ways.

Current authentication systems based on physiological and behavioral characteristics of persons, such as fingerprints, inherently provide solutions to many of the above mentioned problems and may replace the authentication component of traditional cryptosystems. This project produces the cryptographic keys which are bonded with the fingerprint of a user. Such cryptosystems provide better security as the key cannot be compromised without an authenticated user.
# TABLE OF CONTENTS

Abstract .......................................................................................................................... ii

Table of Contents ........................................................................................................ iii

List of Figures ............................................................................................................... vi

List of Tables ............................................................................................................... viii

1. Background and Rationale ...................................................................................... 1
   1.1 Cryptography ...................................................................................................... 1
   1.2 Encryption and Decryption ............................................................................... 1
   1.3 Types of Cryptography Systems ..................................................................... 2
       1.3.1 Public-Key Cryptography ......................................................................... 2
       1.3.2 Private-Key Cryptography ....................................................................... 3
   1.4 Cryptography Goals ......................................................................................... 4
   1.5 Cryptographic Applications ............................................................................. 5
       1.5.1 Secure Communication ........................................................................... 5
       1.5.2 Identification and Authentication ............................................................ 5
       1.5.3 Secret Sharing ......................................................................................... 6
       1.5.4 Electronic Commerce ............................................................................ 6
       1.5.5 Secure Remote Access ........................................................................... 7
   1.6 Cryptographic Random Number Generation .................................................... 7

2. Narrative .................................................................................................................. 9
   2.1 Previous Work Done in the Field ...................................................................... 9
   2.2 Biometrics ......................................................................................................... 9
   2.3 Biometrics Applications .................................................................................. 11
LIST OF FIGURES

Figure 1.1 Asymmetric Key.................................................................2
Figure 1.2 Symmetric Key.................................................................3
Figure 2.1 Encryption Window............................................................12
Figure 2.2 Send Mail button Enabled....................................................13
Figure 2.3 Send Mail Window..............................................................13
Figure 2.4 Decryption Window............................................................14
Figure 3.1 Overall System Design.......................................................19
Figure 3.2 Main Window.................................................................21
Figure 3.3 Message Box Displayed when Required Fields are Empty...........22
Figure 3.4 Successful Encryption.........................................................22
Figure 3.5 Send Mail button enabled....................................................23
Figure 3.6 Send Mail Window..............................................................23
Figure 3.7 Configuring the Mail Server.................................................24
Figure 3.8 Decryption Window............................................................24
Figure 3.9 Message Box Displayed when Required Fields are Empty...........25
Figure 3.10 Successful Decryption.......................................................25
Figure 3.11 Encrypted Fingerprint Image File........................................26
Figure 3.12 Private Key.................................................................26
Figure 3.13 Public Key.................................................................27
Figure 3.14 Decrypted Image............................................................27
Figure 4.1 Fingerprint Image 1..........................................................30
Figure 4.1 Fingerprint Image 1 with a mark........................................30
LIST OF TABLES

Table 3.1 List of Java Programs.................................................................27
1. BACKGROUND AND RATIONALE

1.1 Cryptography

Cryptography is the science of using mathematics to encrypt and decrypt data. It enables the user to store sensitive information or transmit it across insecure networks so that it cannot be read by anyone except the intended recipient or user.

A cryptographic algorithm or cipher, a mathematical function, is used in the encryption and decryption process. A cryptographic algorithm works in combination with a key, a word, number, or phrase to encrypt the plaintext. The same plaintext encrypts to different cipher text with different keys. The security of encrypted data is entirely dependent on the secrecy of the key [Bishop 2003].

1.2 Encryption and Decryption

Encryption is the transformation of data into a form that is difficult for a person to read without the appropriate knowledge. Its purpose is to ensure privacy by keeping information hidden from anyone for whom it is not intended, even those who have access to the encrypted data. Decryption is the reverse process of encryption; it is the transformation of encrypted data back into an intelligible form [Young 2003].

Encryption and decryption will require the use of some secret information, referred to as a key. For some encryption mechanisms, the same key is used for both encryption and decryption and for other mechanisms, the key used for encryption and decryption is different [Bishop 2004].
1.3 Types of Cryptography Systems

A cryptographic system is a method of hiding data so that only certain people will have the access to view it. It typically consists of algorithms, keys, and key management facilities [Stallings 2003].

There are two basic types of cryptography systems:

- Private Key Cryptography and
- Public Key Cryptography

1.3.1 Public Key Cryptography

In public key cryptography, also called asymmetric cryptographic systems, each user has a public key that can be made public to anyone and a private key to avoid unauthorized users. Both the public key and the private keys are linked mathematically; data encrypted with the public key can be decrypted only by the private key, and data signed with the private key can only be verified with the public key [Young 2003]. Figure 1.1 illustrates the asymmetric cryptographic system work flow.

The RSA public key cryptosystem is the most popular form of public key cryptography. RSA is a cipher based on the concept of a trapdoor function [Bishop 2003]. This is a function which is easily calculated, but whose inverse is extremely difficult to calculate.

![Figure 1.1 Asymmetric Key](image)
The Digital Signature Algorithm (DSA) is also a popular public key technique, though it can only be used for signatures and not for encryption [Young 2003]. It is the first digital signature scheme recognized by any government. It consists of a private key and a public key. If anything at all is changed in the document after the digital signature is attached to it, it changes the value that the digital signature compares to, rendering the signature invalid.

1.3.2 Private Key Cryptography

In private key cryptography, also called symmetric key system or secret key cryptosystem, both the sender and the recipient must have the same key [Alfred 1996]. This key is used by the sender to encrypt the data, and again by the recipient to decrypt the data. Symmetric cryptography algorithms are typically fast and are suitable for processing large streams of data. The working of symmetric key system is shown in Figure 1.2.

![Figure 1.2 Symmetric Key](image)

The most popular secret key cryptosystem in use today is the Triple-Data Encryption Standard (DES) which is safer than plain DES. The plain DES is easily
breakable with special hardware by criminal organizations, governments or major corporations. It is getting weak, and should not be used in new designs [Kenneth 1996].

1.4 Cryptography Goals

As cryptography has developed, the number of goals addressed has expanded. Some of the goals that can be accomplished are explained below:

Confidentiality is a service used to keep the content of information from all but those authorized to have it. Secrecy is a term synonymous with confidentiality and privacy. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms which render data unintelligible [Bishop 2003].

Data integrity is a service which addresses the unauthorized alteration of data. To assure data integrity, a user must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes such things as insertion, deletion, and substitution [Bishop 2004].

Authentication is a service related to identification. This function applies to both entities and information itself. Two parties entering into a communication should identify each other. Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent, etc. For these reasons the authentication aspect is usually subdivided into two major classes: entity authentication and data origin authentication. Data origin authentication implicitly provides data integrity (for if a message is modified, the source has changed) [Young 2003].

Non-repudiation is a service which prevents an entity from denying previous commitments or actions. When disputes arise due to an entity denying that certain actions
were taken, a means to resolve the situation is necessary. For example, one entity may authorize the purchase of property by another entity and later deny such authorization was granted. A procedure involving a trusted third party is needed to resolve the dispute [Kenneth 1996].

1.5 Cryptographic Applications

Cryptography is extremely useful; there are a multitude of applications, many of which are currently in use [Kenneth 1996]. Cryptography applications include:

- Secure communication
- Identification
- Authentication
- Secret sharing
- Electronic commerce
- Secure remote access

The cryptographic applications are described briefly in what follows.

1.5.1 Secure Communication

Secure communication is the most straightforward use of cryptography. Messages can be sent between two persons securely by encryption. It is difficult to the third person to decipher the messages. This is important for a large scale network of people to communicate securely [Alfred 1996].

1.5.2 Identification and Authentication

Identification and authentication are two widely used applications of cryptography. Identification is the process of verifying the identity of person or things. Let us consider ATM machines. After inserting the ATM card it will ask the user to enter
the PIN. If the correct PIN is entered, the machine identifies the cardholder and grants
access otherwise it will not [Alfred 1996].

Another important application of cryptography is authentication. In
authentication, it is not necessary to identify a person or entity. Authentication determines
whether that person or entity is authorized for whatever is in question [IEEE 2000].

1.5.3 Secret Sharing

Secret sharing is another application of cryptography that allows the trust of a
secret to be distributed among a group of people. In some implementations of secret
sharing schemes, each participant receives the secret after it has been generated. In other
implementations, the actual secret is never made visible to the participants, although the
purpose for which they sought the secret (for example, access to a building or permission
to execute a process) is allowed [Stalling 2003].

1.5.4 Electronic Commerce

E-commerce is comprised of online banking, online brokerage accounts, internet
shopping etc. A person can buy products, make hotel reservations, transfer money from
one account to another and so on, all while sitting in front of computer. All these tasks
require that the user enters his credit card details in the internet application, thus leaving
it open to fraud. One cryptographic solution to this problem is to encrypt the credit card
number. When the person enters the credit card number the computer encrypts the
number and sends it out on the internet, it is incomprehensible to a third party viewer.
The server (receiver) receives the encrypted information, decrypts it, and proceeds with
the sale without fear that the credit card number slipped into the wrong hands
[Kenneth 1996].
1.5.5 Secure Remote Access

Secure remote access is another important application of cryptography. The basic system of passwords certainly gives a level of security for secure access, but it may not be enough in some cases. For instance, passwords can be eavesdropped, forgotten, stolen, or guessed. Many products supply cryptographic methods for remote access with a high degree of security [Kenneth 1996].

1.6 Cryptographic Random Number Generation

Another major concern when applying cryptography is the problem of generating good random numbers. Traditional symmetric cryptography utilizes one key to encrypt the plain text and decrypt the cipher text. Public key cryptography utilizes two asymmetric keys to encrypt and decrypt the text [Peyravian 1999].

A random number generation is used in cryptographic operations, such as key generation and challenger/response protocols. A random number generator is a function that outputs a sequence of 0s and 1s such that at any point, the next bit cannot be predicted based on the previous bits.

Since public key algorithms rely on random numbers, generation of good random numbers is critical to acceptable security. Unfortunately, predictable computers are poor generators of unpredictable numbers. So, pseudo-random number generators (PRNGs) are used. If the user uses a "bad" PRNG, which doesn't generate unpredictable numbers, the bit-length of the encryption algorithm is reduced.

Pseudorandom number generators (PRNG) are used to produce symmetric cryptographic keys. These random keys are difficult to remember. Thus the keys must be stored in some fashion, such as on a computer or smartcard. Since the entire basis of
symmetric cryptography relies on the secrecy of the cipher key, the storage of the key can compromise the security of the encryption. However, with biometric based keys storage is not necessary since the key is part of the individual who performed the encryption [Peyravian 1999].

Biometrics can provide a way to generate keys for asymmetric encryption. Since biometric identifier is not completely consistent for each capture of the biometric data, such as a fingerprint, a template can be used in conjunction with other distortion tolerance methods to provide a way to ensure very similar identifiers will correspond to the same cipher key [Uludag 2004]. Biometric data is much more difficult to steal or capture making the encryption inherently more secure than storing a key produced from a PRNG. In addition to biometric keys being more secure, they also provide a way to offer verification of identity since only the individual with matching biological identifiers would be able to produce the biometric key [Ratha 2001].

Even though cryptographically strong random number generators are not very difficult to build, they are often overlooked. The importance of the random number generator must be emphasized. If done badly, it will easily become the weakest point of the system.
2. NARRATIVE

2.1 Previous Work Done In the Field

Over the past several years, there have been a number of research efforts aimed at addressing the issues related to integration of biometrics into cryptosystems. Even though with limited number of research efforts, the underlying biometric characteristics do not represent all the currently available pool of modalities, they convey the challenges related to this concept.

Soutar et al. [Soutar 1998] proposed a key binding algorithm in an optical correlation-based fingerprint matching system. This algorithm binds a cryptographic key with the user’s fingerprint images at the time of enrollment. The key is then retrieved only upon a successful authentication.

Davida et al. proposed an algorithm based on the iris biometric. They consider binary representation of iris texture, called iris code, which is 2048 bits in length. The biometric matcher computes the hamming distance between the input and database template representations and compares it with a threshold to determine whether the two biometric samples are from the same person or not [Davida 1998].

In their “fuzzy commitment” scheme, Juels and Wattenberg generalized and significantly improved Davida et al.’s methods to tolerate more variation in the biometric characteristics and to provide stronger security [Juels 1999].

2.2 Biometrics

Biometrics is a technology that is used in the security industry and is integrated with other authentication applications and technologies, such as domain access, single
sign-on, smart cards, encryption, remote access, and digital signatures. Biometrics deals with identification of individuals based on their biological or behavioral characteristics. It is the most secure and convenient authentication tool [Jain 1999].

Today, fingerprint identification systems are most popular and widely used form of biometric technology. Forensic applications have used fingerprints to identify people [Das 2004].

Biometrics comes in several different security solutions. It is divided based on whether the biometrics solution is a physical attribute or a characteristic attribute [Wiley 2003]. Physical attributes include the following items.

- Ear Recognition
- Face Recognition
- Finger Geometry Recognition
- Fingerprint Recognition
- Hand Geometry Recognition
- Iris Recognition
- Retina Recognition

On the other hand characteristic attributes include the items which follow.

- Gait Recognition
- Odor Recognition
- Signature Recognition
- Typing Recognition
- Voice Recognition
2.3 Biometric Applications

There are an endless number of applications to which biometric technology can be applied. Biometric technology and its applications have existed longer than people believe. The oldest ongoing general application of biometrics is the hand-scanning system [Pankanti 2002]. The number of biometric applications instituted around the world is increasing. Approximately 10,000 biometric devices were in use worldwide. Most of the companies have been investing in higher security measures using biometric technology [Jain 1999].

Today, fingerprint identification systems are most popular and widely used from of biometric technology. Forensic applications have used fingerprints to identify people [Das 2004]. Fingerprint biometrics became very familiar when Veridicom Inc. announced the development of a stamped-sized fingerprint reader. The reader-on-a-chip, which is smaller than optical fingerprint readers, can be built into a computer keyboard or mouse, allowing verified users to gain access to a PC or notebook [Maltoni 2003].

Credit card companies want to eliminate billions of dollars in annual losses with help of biometric technology. As well, many other businesses and institutions, such as healthcare centers and prisons, are looking to control records and regulate personnel movement [Richards 1995].

2.4 Steps Involved in the Application Design

This application is comprised of generating the public and private keys from a biometric fingerprint, encryption process, decryption process and mailing process. This project makes use of RSA algorithm, which is one of the most popular algorithms used for public key cryptographic system, to generate the public and private keys. Once the
fingerprint image is loaded it is processed to generate the public and private keys. The application is being designed to work with the following image formats viz., .jpg, .bmp, .gif and .png, hence it is important for the user to make sure that the correct image format is selected.

2.5 Application Interface

Figure 2.1 illustrates the initial user interface that appears at the start of the application. This window allows the user to select a fingerprint image and the destination folders to store the resulting files using the Browse buttons. After selecting the required data user has to hit the Encrypt button in order to initiate the key generation and encryption process. A Help button is placed in this window to launch a HTML page, which contains the information about the application. The window also has a Decrypt button which allows user to go directly to the decryption window for decryption process.

![Encryption Window](image)

Figure 2.1 Encryption Window
Once the encryption is done the *Send Mail* button is enabled as shown in the Figure 2.2

![Send Mail button Enabled](image)

**Figure 2.2 Send Mail button Enabled**

The mailing process window is shown in the following Figure 2.3. It contains *Configure* button to configure the mail server. The *Configure* button allows the user to enter the name of mail server and the email id of the user. This window also contains the *Decrypt* button which helps the user to go to the decryption window.

![Send Mail Window](image)

**Figure 2.3 Send Mail Window**
The window that accepts the decryption process parameters is shown in Figure 2.4 below. The decryption window allows user to select the encrypted file, private key and the folder to store the decrypted file with the help of the *Browse* buttons. This window has an *Encrypt* button which will take the user to the encryption window when hit by the user.

![Figure 2.4 Decryption Window](image)

**2.6 RSA Algorithm**

The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977. In cryptography, RSA is an algorithm for public key encryption. It was the first algorithm known to be suitable for signing as well as encryption, and one of the first great advances in public key cryptography. RSA is supported by Java 1.4 [Riikonen 2002].

The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers [Bishop 2004]. RSA algorithm makes use of two keys: public key and private key. Public key is used in order to encrypt the message to be sent. Everyone who wants to send a message in a
secured manner encrypts the message with the help of public key. Once the message is encrypted, a normal person can not understand what it is. In order to view the message the encrypted message must be decrypted after receiving the message or file. The recipient after receiving the encrypted message or file must decrypt the message with the private key [Johnston 2004].

The RSA cryptosystem i.e. public key cryptosystem is currently used in a wide variety of products, platforms, and industries around the world. It is found in many commercial software products and is planned to be in many more. In hardware, the RSA algorithm can be found in secure telephones, on Ethernet network cards, and on smart cards. In addition, the algorithm is incorporated into all of the major protocols for secure internet communications, including S/MIME, SSL, and S/WAN. It is also used internally in many institutions, including branches of the U.S. government, major corporations, national laboratories, and universities.
3. PROJECT REQUIREMENTS, SYSTEM DESIGN, AND IMPLEMENTATION

3.1 System Requirements

The project has been implemented on a PC running Microsoft Window XP operating system. A Windows XP with Java SDK (Software Development Kit) installed on the local machine was used to develop the project. The Java Runtime Environment (JRE) is also installed on the operating system to run the application. This project also works fine on a Red Hat Linux operating system with out any modifications.

Minimum requirements for running Java are:

- Processor: 2.60 GHz Celeron Processor, 450-Megahertz Pentium Processor and higher are recommended.
- Operating System: Microsoft Windows XP Professional or Red Hat Linux.
- Memory: Minimum 256 MB of RAM.
- Hard Disk: 300 MB of available space is required on system drive

3.2 Java 1.4

The Java programming language is robust and versatile, enabling developers to:

- Write software on one platform and run it on another.
- Create programs to run within a web browser.
- Develop server-side applications for online forums, stores, polls, processing HTML Forms and more.

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:
• Simple,
• Object oriented,
• Portable,
• Distributed,
• High performance,
• Dynamic,
• Robust, and
• Secure.

In Java, the source code is written in plain text files and the files are saved with the .java extension. Then the source files are compiled into .class files by the java compiler (javac). A .class file does not contain code that is native to the processor; it instead contains bytecodes—the machine language of the Java Virtual Machine. The java launcher tool (java) then runs the application with an instance of the Java Virtual Machine. Because the Java Virtual Machine is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris operating system, Linux, or MacOS.

3.3 The Java Platform

A platform is the hardware or software environment in which a program runs.

The java platform has two components:

• The Java Virtual Machine (JVM)
• The Java Application Programming Interface (API)

JVM is a virtual machine that runs compiled java programs i.e., java bytecodes.

This code is generated by the java compiler. The JVM is a crucial component of the java
Platform. All java programs are compiled for the JVM. Therefore, the JVM must be
implemented on a particular platform before compiled java programs will run on that
platform. The JVM plays a central role in making java portable. It provides a layer of
abstraction between the compiled java program and the underlying hardware platform
and operating system. The JVM is central to java's portability because compiled java
programs run on the JVM, independent of whatever may be underneath a particular
JVM implementation.

The API is a large collection of ready-made software components that provide
many useful capabilities, such as Graphical User Interface (GUI) widgets. It is grouped
into libraries of related classes and interfaces; these libraries are known as packages.

3.4 The Java 2 Runtime Environment (JRE)

The Java 2 Runtime Environment allows the user to run applications written in
the java programming language. Like the Java 2 SDK, it contains the Java Virtual
Machine, classes comprising the Java 2 Platform API, and supporting files. Unlike the
Java 2 SDK, it does not contain development tools such as compilers and debuggers. In
order to run the application, the JRE must be installed on the operating system in which
the user is working on.

3.5 Java Swing

Swing is a Graphical User Interface toolkit for java. Swing is one part of the java
Foundation Classes (JFC). Swing includes GUI widgets such as text boxes, buttons, split-
panes, and tables.

Swing widgets provide more sophisticated GUI components. Since they are
written in pure java, they run the same on all platforms. Swing has been included as part
of the Java Standard Edition since release 1.2. The swing classes are contained in the 
javafx.swing package hierarchy.

3.6 System Design

This project has been developed using Java 2 Platform 1.4 and java security 
package which is a java implementation of cryptographic algorithms. The Graphical User 
Interface for the project has been designed using java swing. Figure 3.1 illustrates the 
overall design of the application.

![Figure 3.1 Overall System Design](image)

The system accepts a fingerprint image in any of the four formats viz., .jpg, .bmp, 
.gif and .png to generate the public and private keys. Key generation process is aided by 
the java security package. Encryption process encrypts inputted image with the public 
key generated using the RSA/ECB/PKCS1 Padding. The mailing process of the system 
design has been implemented using the Java Mail API. This process incorporates both the 
server configuration process and the process of attaching encrypted file and private key 
file. Decryption process decrypts the encrypted file using the attached private key to 
retrieve the original file.

3.7 Implementation

As a first step, this application takes a fingerprint image source file, destination 
folder to store the encrypted file and a folder to store a private key. This application
generates both public and private keys using the fingerprint image and stores the
generated keys in the string format in the specified folder. In the encryption process, the
image to be encrypted is converted to a byte array. The byte array is then encrypted with
the public key generated to obtain the cipher text. The cipher text is Base64 encoded to
obtain the final encrypted text. This encrypted text is stored in a string format in the
specified destination folder.

In the Send Mail process, the user is allowed to configure the mailing process
(i.e., configuring the mail server and senders email id) through a Configure button. Once
the user enters the To, Subject and Body fields and hits the Send button a message with
two attached files (encrypted file and private key file) is sent to the recipient.

Decryption takes place in exactly the reverse process to encryption. Encoded text
is Base64 decoded to get back the cipher text. Cipher text is decrypted using the private
key that is sent as an attachment to retrieve the byte array. Image can be constructed back
from the byte array obtained.

3.8 Padding

In public key cryptography, padding is the process of preparing a message for
encryption with a primitive such as RSA. The padding ensures that the message does not
fall into the range of insecure plaintexts, and that a given message, once padded, will
encrypt to one of a large number of different possible ciphertexts. A modern padding
scheme aims to ensure that the attacker cannot manipulate the plaintext to exploit the
structure of the primitive. Breaking the padding scheme is as hard as solving the hard
problem underlying the primitive
3.9 Graphical User Interface

The user interface has been designed to aid the user to understand and use the application without any difficulty. The GUI has been constructed using Java Swing. The access to the fingerprint image, folders to store the files, has been provided by Java Swing components and enables the application user to manipulate the data. Swing supports pluggable look and feel—not by using the native platform's facilities, but by roughly emulating them. This means that the user can get any supported look and feel on any platform. The advantage is uniform behavior on all platforms.

3.10 Sample Demonstration of the Developed Application

Figure 3.2 below illustrates the initial screen that is presented to the user. Here the user has to input a fingerprint image to be encrypted, a destination folder to store the encrypted file, and a folder to store the private key.

![Figure 3.2 Main Window](image)

If the user clicks the Encrypt button without entering the required fields, the user is prompted to proceed in a right way. Figure 3.3 illustrates one such scenario.
Figure 3.3 Message Box Displayed when required Fields are Empty

Figure 3.4 illustrates the screen that is displayed once the encryption is done successfully.

Figure 3.4 Successful Encryption

After the successful completion of the encryption process, *Send Mail* button is enabled as can be observed from Figure 3.5.
Figure 3.5 Send Mail button enabled

Figure 3.6 illustrates the Send Mail window with Encrypted file and Private Key file added as attachments.

Figure 3.6 Send Mail Window
Figure 3.7 illustrates the configuration process invoked by the user.

![Figure 3.7 Configuring the Mail Server](image)

The decryption window is illustrated in Figure 3.8.

![Figure 3.8 Decryption Window](image)

If the user clicks the Decrypt button without entering the required fields, the user is prompted with a message box. Figure 3.9 illustrates one such scenario.
Figure 3.9 Message Box Displayed when required Fields are Empty

Figure 3.10 illustrates the screen that appears once the decryption is completed successfully.

Figure 3.10 Successful Decryption
Encrypted fingerprint image file is shown in Figure 3.11.

![Figure 3.11 Encrypted Fingerprint Image File](image)

Generated private key is shown in Figure 3.12

![Figure 3.12 Private Key](image)
Generated public key is shown in Figure 3.13

![Public Key](image)

**Figure 3.13 Public Key**

Figure 3.14 is the original image retrieved by the decryption process.

![Decrypted Image](image)

**Figure 3.14 Decrypted Image**

### 3.11 List of Java Programs

Table 3.1 shows the listing of java programs and their brief description

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Java Program Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FileChooserEncrypt.java</td>
<td>Designs the main window</td>
</tr>
<tr>
<td>2</td>
<td>EncryptLocation.java</td>
<td>Stores the files after key generation and encryption process</td>
</tr>
<tr>
<td>3</td>
<td>EncryptFingerprint.java</td>
<td>Encrypts the fingerprint image</td>
</tr>
<tr>
<td></td>
<td>Class Name</td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>4</td>
<td><code>SendMail.java</code></td>
<td>Designs the send mail window</td>
</tr>
<tr>
<td>5</td>
<td><code>ConfigureMail.java</code></td>
<td>Configures the mail server</td>
</tr>
<tr>
<td>6</td>
<td><code>MailingProcess.java</code></td>
<td>Sends email with attachments</td>
</tr>
<tr>
<td>7</td>
<td><code>FileChooserDecrypt.java</code></td>
<td>Designs decryption window</td>
</tr>
<tr>
<td>8</td>
<td><code>DecryptLocation.java</code></td>
<td>Stores the file after decryption process</td>
</tr>
<tr>
<td>9</td>
<td><code>DecryptFingerprint.java</code></td>
<td>Decrypts the encrypted file</td>
</tr>
</tbody>
</table>
4. TESTING AND EVALUATION

This chapter describes the usability and user interface design of the project. A set of testing schemes has been used to see if the whole project is performing the tasks as expected. The project was evaluated and debugged at every stage of development.

4.1 Testing the Designed Application

The application designed for the user has been tested for the fingerprint image input, folder selection and usability of the application.

4.1.1 Interface testing

The user interface is tested for fault tolerance, in that if the user populates any of the fields in the interface with inappropriate data, the user is prompted for a right option. For example if the user inputs a finger print image in “.” format a message box pops up listing out the allowable image formats. It is also tested that it will display a message box when the user tries to encrypt the fingerprint image without selecting the fingerprint image and the folders to store the encrypted message and the keys.

4.1.2 Application Testing

This is performed to check if the code written is generating the expected results. In this testing the input is supplied to the designed application and to check if the application is generating the expected output. If the application generates the same encrypted file for two or more different fingerprint images it defeats the sole purpose of the project by allowing duplicate identities. Hence the application has been tested for the occurrence of duplicate encrypted files, if generated, for different user inputs (fingerprint images).
The application has been tested for two different scenarios to evaluate its efficiency.

Scenario 1: Using the same fingerprint image but with a "mark" differentiating it from the original image when applied the second time to the system.

Keys generated using the two images are stored and compared to prove the authenticity of the system. If the system generates the same keys for an image without a mark and the one with a mark, it proves that the system is not fault tolerant. The following figures illustrate the keys generated from the fingerprint images i.e. with a mark and without a mark.

![Figure 4.1 Image1](image1.png)  
![Figure 4.2 Image1 with a “mark”](image2.png)

![Figure 4.3 Privatekey of the Fingerprint Image1](privatekey.png)

30
Figure 4.4 Privatekey of the Fingerprint Image1 with a “mark”.

Figure 4.5 Publickey of the Fingerprint Image1.

Figure 4.6 Publickey of the Fingerprint Image1 with a “mark”.

Scenario 2: Using two different fingerprint images to generate the keys each time the application is tested.

In this scenario the fingerprint images are not of the same identity. In this case the keys generated from the image must be different in order not to compromise the sole
purpose of the application. The public and private keys generated from these two images are stored and compared against each other to check for equality.

The results of the key comparison module show that the application generates different keys each time a different fingerprint image is applied as an input.

This is illustrated in the following figures that show the keys generated from two different fingerprint images.

![Figure 4.7 Fingerprint Image1](image1)

![Figure 4.8 Fingerprint Image2](image2)

![Figure 4.9 Privatekey of the Fingerprint Image1](image3)
Black Box and White Box test design methods are used for testing the application. Black box test is also known as functional testing. This testing technique treats the system as black box or closed box. Black box test design doesn’t examine the internal structure of the system. It tests a program from the external view, i.e., the internal structure of the system is ignored, and mainly focused on relationship between inputs and outputs.
outputs. Black box tests find bugs such as incorrect functions, interface problems, and database errors. White box testing is also known as glass box. White box testing uses specific knowledge of the internal structure of systems to guide development of tests.

The following are the list of tests that were performed on this project:

1. Unit testing is performed to test particular functions or code modules. It is done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. The three major modules of the project, key generation module, encryption module and decryption module have been tested in this step.

   *Generate Key*: This module is tested primarily as the following steps of encryption and decryption are based on the correct output of this module. It is tested to check if the specified destination folders contain the files that store the generated keys in expected format.

   *Decrypt file*: This particular code block has been tested to check if it decrypts the file using the private key that is generated by the application and outputs the fingerprint image provided by the user.

   *Encrypt file*: This module has been tested to see if it generates an encrypted file based on the public key generated and that the file is not in readable form.

2. Incremental integration testing is the continuous testing of an application as new functionality is added. The code development process has been carried out in a step by step process to ensure proper functionality of the application at every independent step. For instance, the implementation of key generation procedure has fallen under two consecutive code modules. First a code snippet has been
developed to produce byte array from any image file (that conforms to the specified formats) provided as input. Next a module that generates public and private keys given the byte array is carried out. The two modules are then integrated to perform the required operation. On integration the later module takes in the generated byte array as input. In a similar way the encryption and decryption modules have been tested in the development process to comply with each other.

3. System testing is a type of Black box testing that is based on overall requirements specifications; covers all combined parts of a system.

4. Compatibility testing is the testing of how well the software performs in a particular hardware/software/operating system environment. The application has been tested on various platforms such as Windows, LINUX, and UNIX.

4.1.3 Usability of the Application Testing

Usability testing is used to identify how the users actually interact with the application developed. The application has been tested for the ease of navigation through the windows, appearance, consistency so that the output is displayed as expected.

Appearance: All the windows in the application have been designed using the same color, font and style. All the windows have been checked for the uniformity in the appearance.
5. FUTURE WORK

The project has some scope for enhancements in the future. The following are some of the ideas that can be incorporated into this project in future.

1. Fingerprint reader can be used as an input reader instead of taking fingerprint image from the storage media.

2. Maintain a database of the authorized user’s fingerprint images to authenticate the input read from the fingerprint reader. Authentication can be implemented by developing a module that compares the user inputted fingerprint with the database entries.
6. CONCLUSION

The design of this project is aimed at generating the public and private keys from the biometric fingerprint, which is somewhat difficult to produce than the standard random symmetric key generation. The security gained by the performance of biometric systems is greater than the pseudo-random number generation systems. Biometrics makes it more convenient for end-users to communicate with the other users in a secure method.
BIBLIOGRAPHY AND REFERENCES


APPENDIX A: Data Dictionary

API: An Application Program Interface is a set of routines, protocols, and tools for building software applications. It defines how programmers utilize a particular computer feature. APIs exist for windowing systems, file systems, database systems, and, of course, networking systems.

Asymmetric key: In cryptography, an asymmetric key algorithm uses a pair of cryptographic keys to encrypt and decrypt. The two keys are related mathematically; a message encrypted by the algorithm using one key can be decrypted by the same algorithm using the other. In a sense, one key "locks" a lock (encrypts); but a different key is required to unlock it (decrypt).

Authentication: In computer security, authentication is the process by which a computer, computer program, or another user attempts to confirm that the computer, computer program, or user from whom the second party has received some communication is, or is not, the claimed first party.

Biometrics: Biometrics is the science and technology of authentication (i.e. establishing the identity of an individual) by measuring the subject person's physiological or behavioral features.

Black box: is used in computer programming, software engineering and software testing to check that the outputs of a program, given certain inputs, conform to the functional specification of the program.

Confidentiality: Confidentiality is the guarantee that data is not shared with unauthorized entities.
Cryptography: The process of protecting information by transforming it into an unreadable format. The information is encrypted using a "key" that makes the data unreadable. It is later decrypted, making the information readable again.

Data Integrity: The property that data have not been altered or destroyed in an unauthorized manner or by unauthorized users; it is a security principle that protects information from being modified or otherwise corrupted either maliciously or accidentally.

Decryption: The process of converting encrypted content back into its original form, often the process of converting cipher text to plaintext.

DES: A Data Encryption Standard algorithm used for encrypted which was the official algorithm of the United States Government. It was developed by IBM with assistance from the NSA.

DSA: The Digital Signature Algorithm is used in creating the digital signature for a given message or transaction. DSA is only used for digital signatures but is not used for key exchange.

Encryption: Encryption is the conversion of data into a form, called a cipher text that cannot be easily understood by unauthorized people.

GUI: A Graphical User Interface is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use.

Java: Java is a high-level, object-oriented programming language developed by Sun Microsystems. These java applications can execute on any platform--Mac, PC, etc.

Java byte code: Java byte code is the code the java compiler produces from java source code. Java byte code is interpreted (executed) by a Java Virtual Machine (JVM).
Java Mail API: The JavaMail API provides a platform-independent and protocol-independent framework to build mail and messaging applications.

Javac: A java compiler is a computer program that translates programs in java to java byte-code.

Java Swing: Java swing is a GUI toolkit for java. It includes Graphical User Interface widgets such as text boxes, buttons, split-panes, and tables.

JDK: Java Development Kit is a software development package from Sun Microsystems that implements the basic set of tools needed to write, test and debug java applications and applets.

JRE: Java Runtime Environment is the package of software that must be installed on a machine in order to run java applications. It does not include development tools.

JVM: The Java Virtual Machine interprets the bytecodes that are written by the java compiler.

Non-repudiation: Method by which the sender of data is provided with proof of delivery and the recipient is assured of the sender's identity, so that neither can later deny having processed the data.

Private Key: The private key is used to decrypt messages that were encrypted by the corresponding public key.

Public key: A public key is used to encrypt messages that only the corresponding private key can decrypt.

RSA Algorithm: The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman. In cryptography, RSA is an algorithm for public-key encryption. It was the first
algorithm known to be suitable for signing as well as encryption, and one of the first
great advances in public key cryptography.

**Padding:** In public key cryptography, padding is the process of preparing a message for
encryption or signing with a primitive such as RSA.

**Symmetric key:** A symmetric key is an encryption system in which the sender and
receiver of a message share a single, common key that is used to encrypt and decrypt the
message.

**White box:** White box is used in computer programming, software engineering and
software testing to check that the outputs of a program, given certain inputs, conform to
the structural specification of the program.