Date : 2079-01-10

Samriddhi College
Lokanthali, Bhatapur

SUPERVISOR'S RECOMMENDATION

I hereby recommend that the report on "**Text File Sharing with Digital Signature in Intranet**" by Aayush Bhattarai, Deepesh Oli and Parajanya Shrestha in partial fulfillment of the requirement of the degree of Bachelor of Science in Computer Science and Information Technology (B. Sc. CSIT) has been prepared under my supervision. In my best knowledge, this is an original work in Computer Science by them.

……

Mr. Bikash Balami

Supervisor

Department of Computer Science & Information Technology

Samriddhi College

Lokanthali, Bhaktapur

Date:2078-12-09

Tribhuvan University

Institute of Science and Technology

Samriddhi College

Lokanthali, Bhaktapur

CERTIFICATE OF APPROVAL

The undersigned certify that this project is prepared under my supervision, a project report entitled **"Text File Sharing with Digital Signature in Intranet"** submitted by **Aayush Bhattarai**, **Deepesh Oli** and **Parajanya Shrestha** in partial fulfillment for the Degree of Bachelor in Computer Science and Information Technology. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

Evaluation Committee

……..

Mr. Bikash Balami

Supervisor/Professor

Department of Computer Science & Information Technology

Samriddhi College

Lokanthali, Bhaktpur

……

External

Tribhuvan University

Institute of Science and Technology

A Final Year Project Report

on

"Text File Sharing with Digital Signature in Intranet"

Submitted to:

Department of Computer Science and Information Technology

Samriddhi College

In Partial Fulfillment of the Requirements

For the Bachelor's Degree in Computer Science and Information Technology

Submitted by:

Aayush Bhattarai (TU Roll No.: 16133)

Deepesh Oli (TU Roll No.: 16147)

Parajanya Shrestha(TU Roll No.: 16152)

29 April, 2022

Acknowledgement

We are thankful for the proper guidance and assistance from every individual supporting the project directly or indirectly. We would like to thank our respected supervisor Mr.Bikash Balami, for supervising us in this project and clearing our doubts and problems when we faced them. The problems were tackled by providing deeper knowledge on various aspects of the project.

We would like to express our sincere gratitude to our teacher Mr. Mohan Bhandari, for providing us with a valuable and necessary guideline to bring up with a good project. Also, we are glad to express our sincere gratitude to the Department of Computer Science, College Administration for motivating us to bring forth our project. Likewise, all the results that we care for now would not have been successful without the involvement of our respected sir, Mr. Lok Nath Regmi. All his experience and advice helped us to complete the project effectively on time. Moreover, we would like to thank all the friends and teachers who helped us by contributing their suggestions and ideas.

Lastly, we would like to express our sincere thanks to Samriddhi College for providing us the opportunities and space to work on our ideas into a valuable project. Also, we thank the department for teaching us the value of teamwork and co-operation.

Thanking You,

Name

Aayush Bhattarai[TU Roll No. : 16133]

Deepesh Oli[TU Roll No. : 16147]

Parajanya Shrestha[TU Roll No. : 16152

Abstract

This application makes use of **Digital Signature Scheme using RSA** along with a **SHA-256** hash function. The hash code is provided as input to a signature function along with a random number generated for this particular signature. The signature function also depends on the sender's private key and a set of parameters known to a group of communicating parties. This set constitutes a global public key. The result is a signature consisting of two components.

At the receiving end, verification is performed. The receiver generates a quantity that is a function of the public-key components, the sender's public key, and the hash code of the incoming message. If this quantity matches with one of the components of the signature, then the signature is validated.

Keywords: *Encryption, Decryption, Conventional signature, Digital signature, Java, Swing, Sockets, RSA and Hash function.*

Table of Contents

Acknowledgement i
Abstract ii
List of Figures v
List of Tables vi
List of Abbreviations vii
CHAPTER 1
INTRODUCTION 1
1.1 Introduction 1
1.2 Problem Statement 2
1.3 Objectives 2
1.4. Scope and Limitation 2
1.5. Development Methodology 3
1.6. Report Organization 3
CHAPTER 2
BACKGROUND STUDY AND LITERATURE REVIEW 4
2.1 Background Study 4
2.2 Literature Review 5
CHAPTER 3
SYSTEM ANALYSIS 8
3.1 System Analysis 8
3.1.1 Requirement Analysis 8
i. Functional Requirements 8
ii. Non-Functional Requirements 9
3.1.2 Feasibility Analysis 9
i. Technical Feasibility 9
ii. Operational Feasibility 9
iii. Economic Feasibility 9
iv. Schedule Feasibility 10
3.1.3 Analysis 11

CHAPTER 4

SYSTEM DESIGN..... 14

4.1 Design.....14

4.2 Algorithm Details.....16

4.2.1 RSA 16

4.2.2 SHA256 17

CHAPTER 5

IMPLEMENTATION AND TESTING..... 21

5.1 Implementation.....21

5.1.1 Tools Used.....21

5.2 Testing.....22

5.2.1 Test cases for Unit Testing 22

5.2.2 Test Case for System Testing 23

5.3 Result Analysis.....24

CHAPTER 6

CONCLUSION AND FUTURE RECOMMENDATION..... 26

6.1 Conclusion.....26

6.2 Future Recommendation 26

References

Appendix

List of Figures

Figure 3.1: Use case diagram 8

Figure 3.2 : PERT Diagram 10

Figure 3.3. : Class diagram 11

Figure 3.4: Sequence Diagram 12

Figure 4.1: Block Diagram 14

Figure 4.2: Activity Diagram 15

List of Tables

Table 5.1: Unit testing of SHA256 22

Table 5.2: Unit testing of SHA256 23

Table 5.3: Unit testing of RSA 23

Table 5.4: Test case for system testing 24

Table 5.3.1: Result analysis of SHA256 25

List of Abbreviations

SHA	Secure Hash Algorithm
RSA	Rivest-Shamir-Adleman
DSA	Digital Signature Algorithm
S/MIME	Secure/Multipurpose Internet Mail Extensions
SSL	Secure Socket Layer
PGP	Pretty Good Privacy
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
JVM	Java Virtual Machine
PuK	Public Key
PK	Private Key

CHAPTER 1
Introduction

1.1 Introduction

Digital signature authentication schemes provide secure communication with minimum computational cost for real time applications such as electronic commerce, electronic voting etc. The sender generates the signature of a given message using his secret key; the receiver then verifies the signature by using sender's public key[1]

Before the concept of digital signature, the security of information available to an organization was primarily provided through physical and administrative means. For example : Rugged file cabinets with a key lock were used for storing sensitive documents and the trust was very low between the employees.

Many organizations prefer going paperless by using electronic forms of sending and receiving data. In this context, it is essential that not only the sender needs to authenticate the receiver, the receiver should also authenticate the sender and ascertain himself from whom the message was received [2].

A digital signature or digital signature scheme is a mathematical scheme for demonstrating the authenticity of digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered in transit. Digital signatures are commonly used for software distribution, financial transactions and in other cases where it is important to detect forgery and tampering.[3]

The digital signature provides three main security services for the data sent namely authentication, integrity and non-repudiation. The identity of the sender is ensured by the private key of sender. So, the receiver can verify the identity of the sender. The author of the message is really who they claim to be. Digital signature ensures that the data is not tampered during the transfer. It ensures the message or a document cannot be altered while transmitting. As it ensures the authenticity of the sender. The falsely deny of the sender is also not possible. The author of the message can't later deny that they were the source.

1.2 Problem Statement

These days almost all organizations around the globe uses Intranet to transfer data and documents among their employees. But the security provided is not of high standards. More and more unauthorized people are gaining access to confidential data.

The validity of sender is not known. The sender may deny sending a message that he/she has actually sent and similarly the receiver may deny the receipt that he/she has actually received. Unauthorized people can gain access to classified data. Intruders can modify the messages or the receiver himself may modify the message and claim that the sender has sent it.

1.3 Objectives

This project has been developed keeping in view the security features that need to be implemented in the Local Area Network(LAN) following the fulfillment of these objectives:

- To enable the end-users come out with a file transfer using digital signature.

1.4 Scope and Limitations

1.4.1 Scope

The system aim is broad in terms of other file sharing system where there is no feature of signing the sent file. It solves the problem of sending the file with authenticity, as RSA algorithm has better security triad. This system can be used as an solution for the intra-office file transfer. Easy transmission i.e. uploading, and downloading of documents or text files can be performed. User can send chunk of files at a time.

1.4.2 Limitation

The limitations of our System are as follows:

- It cannot be used to share file across network with different public IP address.
- Although the system can allow to share the multimedia files, only text files can be digitally signed.
- An infrastructural barrier such as internet connectivity, electrical connection, etc. might interfere with how the system will work.
- The encryption facility is not yet provided by our system.
- Although client can send multiple files at a same time but only single file can be digtially signed at a time.

1.5 Development Methodology

Software development is the process of developing software product in a planned and structured process. It involves creating a computer program, or a set of programs to perform two various tasks, as well as maintaining and improving these programs

The system is developed following the object oriented design principle. All the phases of the software development life cycle was being followed during the creation of this project. The absence of digital signature in most of the day to day operation was the requirement which was analyzed to undertake the project. The project was initially done using the Java API for generating and validating the digital signature. Once the prototype was constructed, the algorithm implementation was done in the Implementation phase. The system was tested thoroughly to detect the defects in the software.

1.6 Report Organization

This report is divided into six chapters. Each chapter is further divided into different headings.

- **Chapter 1** gives introduction about Digital Signature System. The problem definition, objectives, scopes and limitation of this system are discussed here.
- **Chapter 2** It contains literature review section where the research works done in the field of the Document Signing and Verification System are enlisted.
- **Chapter 3** focuses on the analysis part. This chapter also includes feasibility study, requirement analysis and diagram like Class, Sequence.
- **Chapter 4** discusses in details about the design of the system. This chapter also discusses about interface design and flowchart of the system built.
- **Chapter 5** gives information about implementation and testing process. It discusses about how the system is implemented and what tools and software are used to implement this system. The testing process is also included in detail in this chapter.
- **Chapter 6** includes conclusion of whole project. This chapter shows major achievements in the system and also shows how it can be enhanced later in the future so that it can be made more productive and useful than prevailing system.

CHAPTER 2
Background Study and Literature Review

2.1 Background Study

Digital signature is an electronic signature used to authenticate digitally transferred data and to ensure that the content of the message or the document sent has not been tempered. Digital signatures and handwritten signatures both rely on the fact that it is very hard to find two people with the same signature. People use public-key cryptography also called asymmetric cryptography to compute digital signatures by associating something unique with each person. When public – key cryptography is used to encrypt a message, the sender encrypts the message with a public key of the intended recipient. When public-key cryptography is used to calculate a digital signature, the sender encrypts the "digital finger print" of the document with his or her own private key. Anyone with access to the public key of the signer may verify the signature.

The digital signature of a document is a piece of information based on both the document and the signer's private key. It is typically created through the use of a hash function and a private signing function (encryption with the signer's private key).

The key-pair is derived from a very large number(n) that is the product of two prime numbers chosen according to special rules; each prime are very large in length, yielding an n with roughly twice as many digits as the prime factors. The public key information includes n and a derivative of one of the factors of n; an attacker cannot determine the prime factors of n (and, therefore, the private key) from this information alone and that is what makes the RSA algorithm so secure. Regardless, one presumed protection of RSA is that users can easily increase the key size to always stay ahead of the computer processing curve.

A one-way hash is a function (usually mathematical) that takes a variable-length string, a message, and compresses and transforms it into a fixed-length value referred to as a hash value. A hash value is also called a message digest. Just as fingerprints can be used to identify individuals, hash values can be used to identify a specific message.

The hashing one-way function takes place without the use of any keys. This means that anyone who receives the message can run the hash value and verify the message's integrity. However, if a sender only wants a specific person to be able to view the hash value sent with the message, the value would be encrypted with the key. This is referred to as the message authentication code [4].

2.2 Literature Review

RSA is the first algorithm known to be suitable for signing as well as encryption, and one of the first great advances in public key cryptography. It is named for the three MIT mathematicians who developed it — Ronald Rivest, Adi Shamir, and Leonard Adleman. RSA today is used in hundreds of software products and can be used for key exchange, digital signatures, or encryption of small blocks of data.

RSA is very widely used today for secure Internet communication (browsers, S/MIME, SSL, S/WAN, PGP, and Microsoft Outlook), operating systems (Sun, Microsoft, Apple, Novell) and hardware (cell phones, ATM machines, wireless Ethernet cards, Mondex smart cards, Palm Pilots).

Pretty Good Privacy in short PGP, a freeware created by Phil Zimmerman, providing encryption and authentication for e-mail and file storage applications across multiple platforms uses RSA algorithm for its key transportation. If client machines are slow then DSA is used and if slower server then RSA is used for verification, yet again difference of speed is at the start of the process.[5]

Digital signature schemes based on public-key cryptosystem are vulnerable to existential forgery attack which can be prevented by use of one-way hash function and message redundancy. In this paper the authors have proposed an forgery attack over the digital signature scheme proposed by Chang and Chang in 2004. The authors have additionally proven stepped forward scheme the use of new key agreement protocol over the Chang and Chang model which honestly lacks the usage of one way hash function and redundancy padding [6]

The problem of fair exchange is one of the major threats in the field of secure electronic transactions. In this paper the authors have presented a multi signature scheme based on DSA which describes a method of constructing efficient fair-exchange protocols based on improved DSA signatures.[7]

In this paper, the authors compare the computational times of RSA and DSA with some bits and choose which bits are better used. Then combine both RSA and DSA algorithms to improve data security. From the simulation results, the authors chose RSA 1024 for the encryption process and added digital signatures using DSA 512, so the messages sent are not only encrypted but also have digital signatures for the data authentication process.[8]

This paper proposed an implementation of a complete and practical RSA encrypt/decrypt solution based on the study of RSA public key algorithm. In addition, the encrypt procedure and code implementation is provided in details.[9]

The authors have proposed an enhance algorithm for the RSA cryptosystem. This new proposed cryptosystem uses a third prime number in calculating the value of n. This additional third prime number increases the factor complexity of n. It will provide more security to the RSA. [10]

The authors have proposed a scheme of digital signature in electronic government to settle some specific problems such as spilling out secret, forging or denial and so on. Apart from this, a brief analysis regarding security issues of digital signature is also mentioned in this paper. [11]

The public key cryptosystem RSA is the first and most popular cryptosystem for performing encryption and decryption of data, to keep data secret, to transfer data from one location to another. Also it is known that the security of RSA depends on large factorization. If the factorization is possible then the whole algorithm can become breakable. Authors proposed a new methodology to change the original modulus with the fake modulus. Therefore if the hacker factorizes this new modulus value then he will not be able to locate the original decryption key.[12]

In this paper the authors have presented an efficient approach of encryption where the openness of the encryption key does not thereby reveal the corresponding decryption key. In this technique the message M is enciphered using the publicly available encryption key which is in-turn deciphered only by the intended recipient using the decryption key which is privately owned by the actual receiver.[13]

CHAPTER 3
System Analysis

3.1 System Analysis

3.1.1 Requirement Analysis

The functional and non-functional requirements are necessary to analyze the system requirements before developing and implementing. The functional requirements specify the documentation of the system and activities that a system must be able to perform.

i) Functional Requirement

- The system should allow user to send file to another user.
- The system should have the functionality of signing & verifying the digital signature.
- The system should alert the user if file has been tempered.

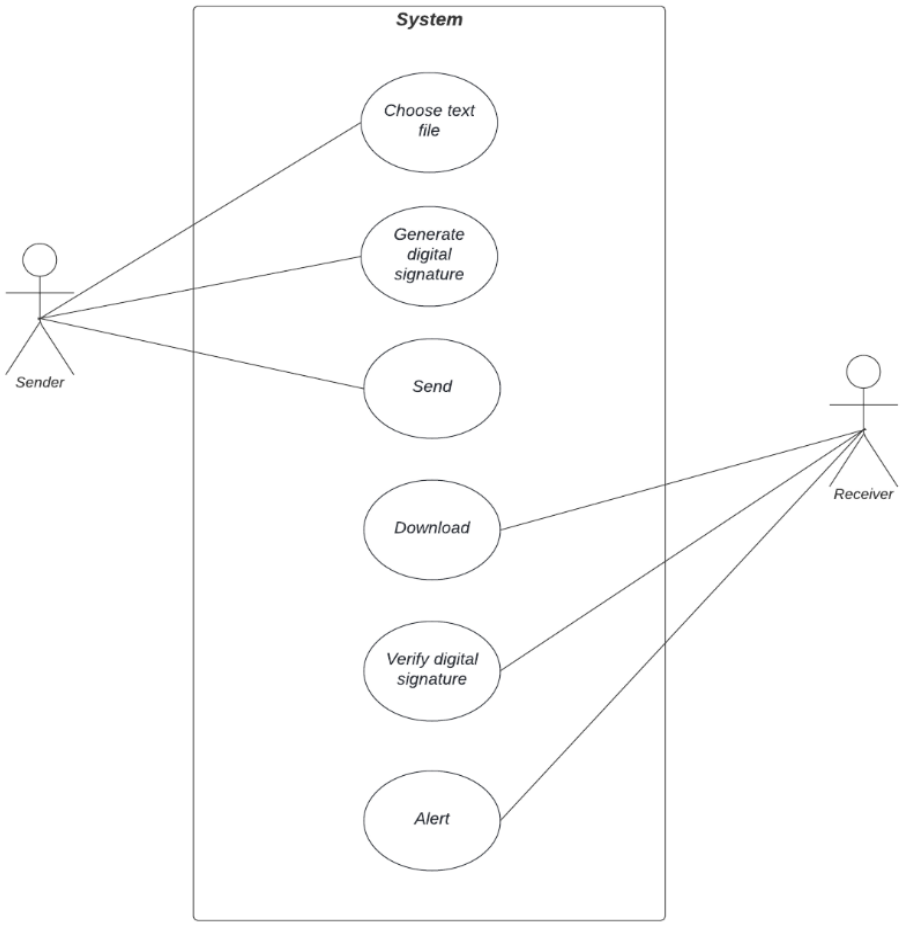


Figure 3.1:Use Case Diagram of client and server communication

ii) Non-functional Requirement

In addition to the obvious features and functions that are provided in this system, there are other requirements that don't actually do anything, but are important characteristics nevertheless which are called as non-functional requirement or sometimes Quality Attributes. They are those type of requirements which is not directly concerned with the system functionality but in absence of it reduces the quality of the system process.

For example: attributes such as performance, security, usability, compatibility.

3.1.2 Feasibility Analysis

i) Technical

It is desktop based application that uses Java Swing as front-end and core Java as backend. It will be based on client-server architecture provided by Java socket programming. Java is rich language making it technically feasible to develop proposed system.

ii) Operational

The operational feasibility of project deals with user's requirements and usefulness. The project is confined to the intranet in an organization. This application makes sure that security services such as authentication, integrity and non-repudiation are provided to the communicating parties. It can be easily operated once properly guided.

iii) Economic

The purpose of economic feasibility is to analyze a project's costs and revenues in an effort to determine whether or not it is logical and possible to complete. Economically it has very little cost for development of our software. Anyone with core Java knowledge can understand the ins and outs. For the deployment, it requires at least 2 PC with Internet connection. The operational and training cost is minimal because in most cases the user will be IT professional or the people with knowledge of how to operate computer programs.

iv) Schedule

It is the most important for the completion of the project on time. The project that we are proposing will too be completed within time constraints.

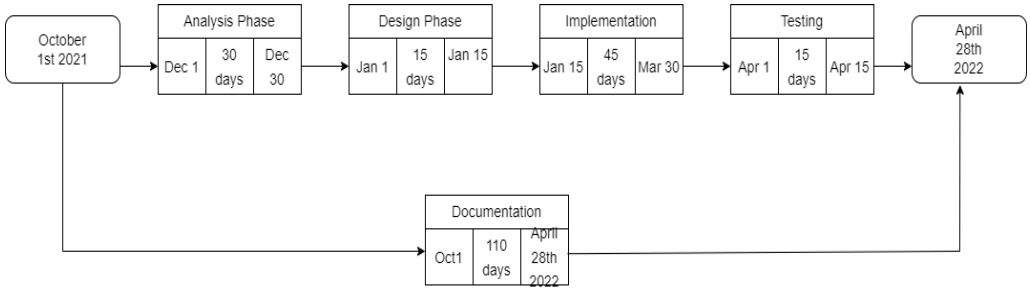


Figure 3.2: PERT Diagram for system development

3.1.3 Analysis

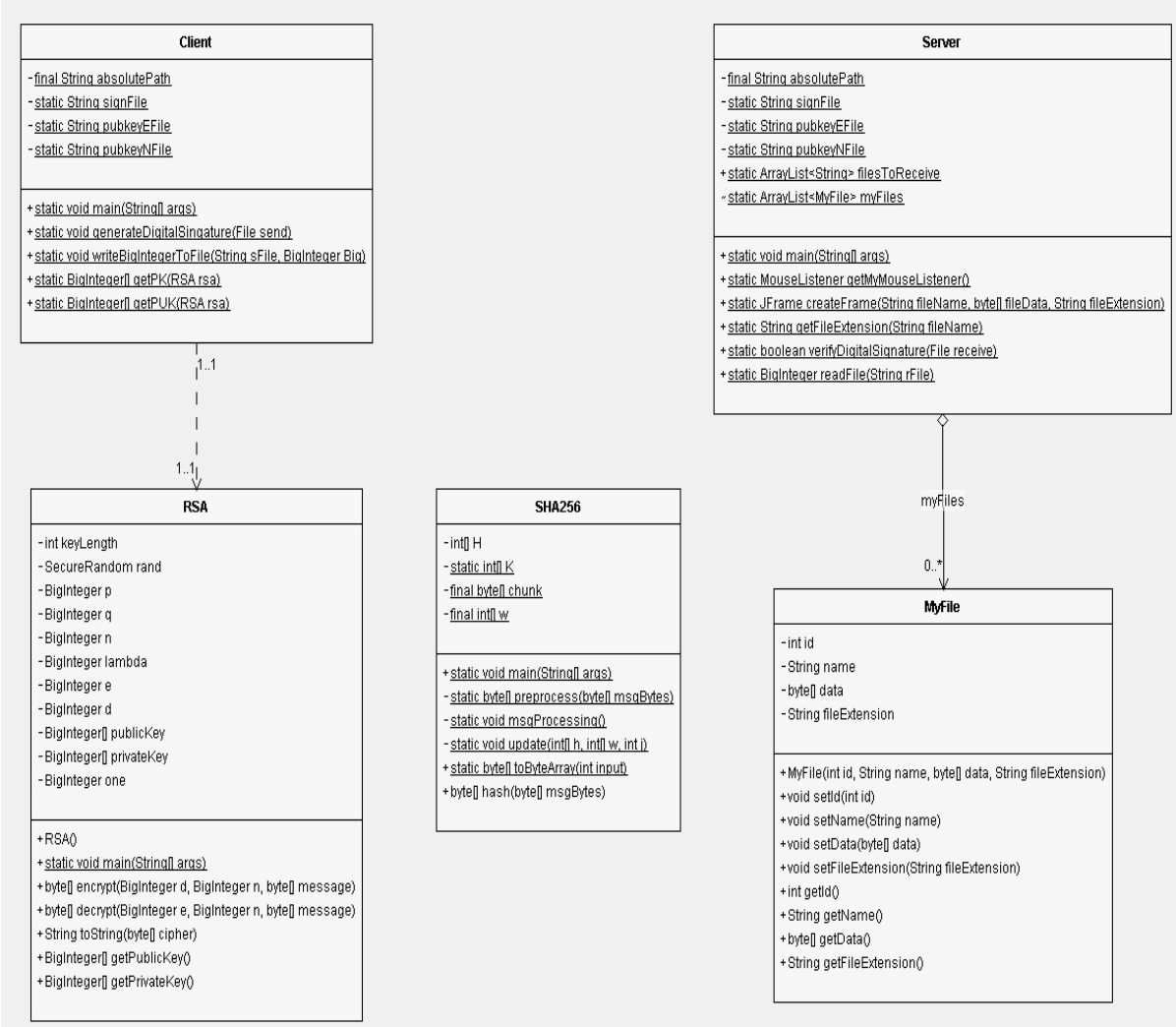


Figure 2.3: Class Diagram of client and server communication

Client wants to send a signed document or message to Server which is a text file. The first step is generally to apply a hash function to the message, creating what is called a message digest. The message digest is usually considerably shorter than the original message. In fact, the job of the hash function is to take a message of arbitrary length and shrink it down to a fixed length. The Client then uses RSA algorithm to generate private & public key for itself. To create a digital signature, one usually signs (encrypts) the message digest as opposed to the message itself. This saves a considerable amount of time.

Client sends Server the encrypted message digest(digital signature), the message and public key. In order for Server to authenticate the signature, it applies the same hash function as Client to the message it sent, decrypt the encrypted message digest using Client's public key and compare the two. If the two are the same, the signature is successfully authenticated. If the two do not match there are few possible explanations. Either someone is trying to impersonate Client, the message itself has been altered since Client signed it or an error occurred during transmission.

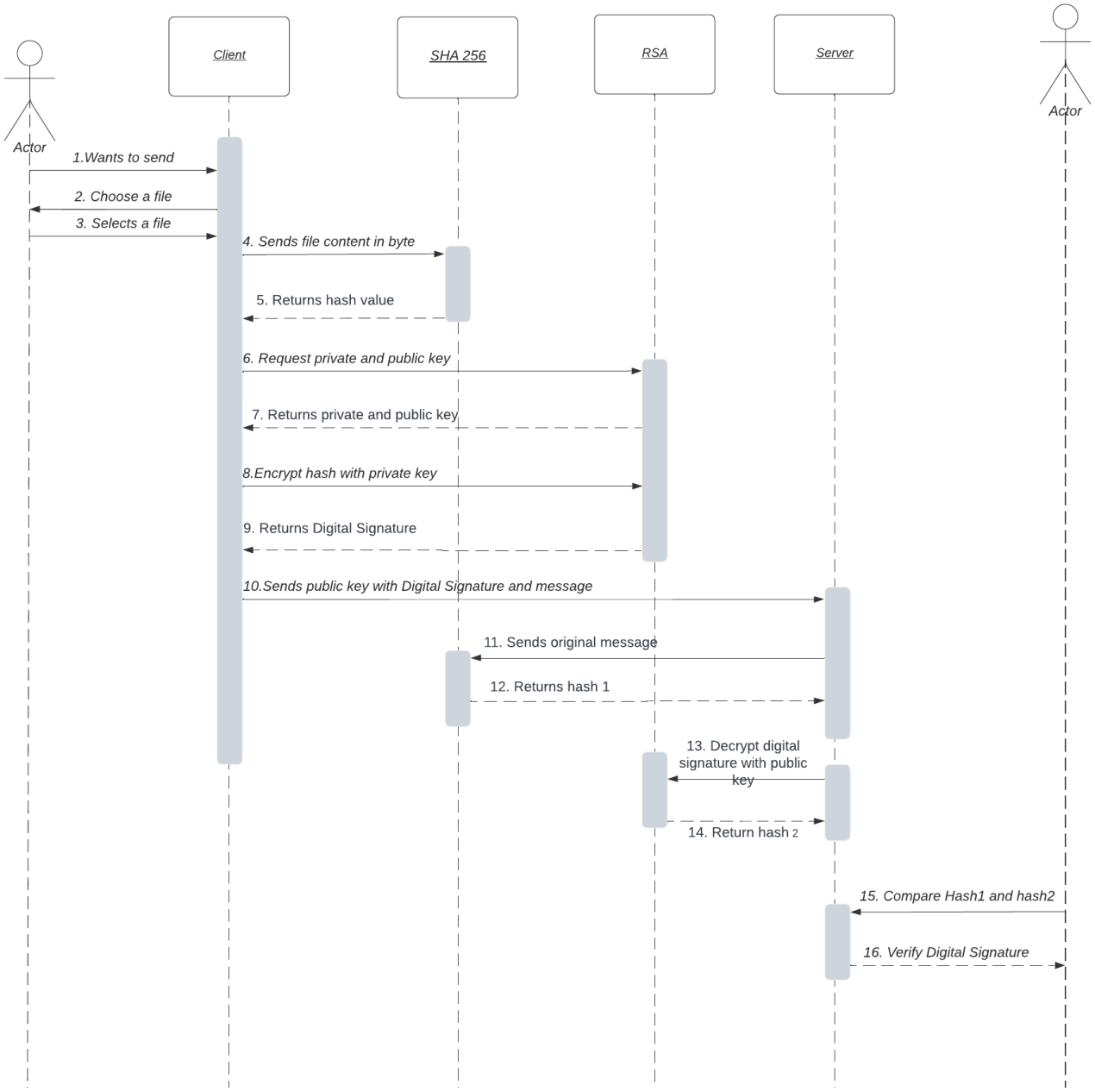


Figure 3.3: Sequence Diagram of client and server communication

CHAPTER 4
System Design

4.1 Design

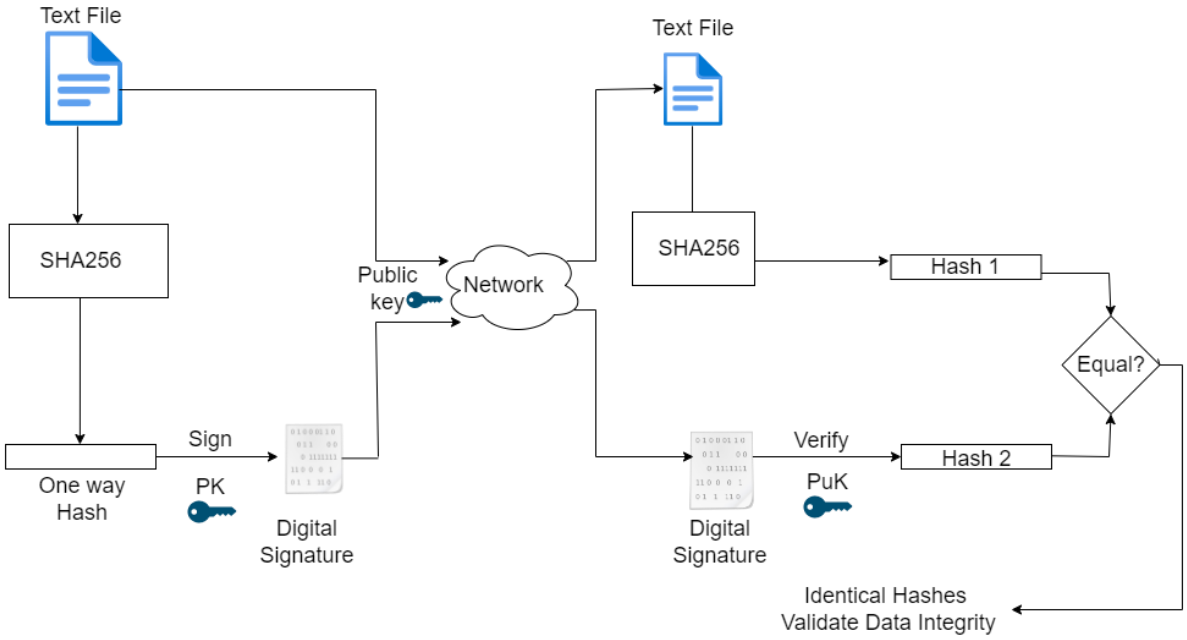


Figure 4.1: Block diagram of Client and Server communication

The client & server both uses the same hashing algorithm. The client's activity is represented on the left hand side of the network. The client chooses a text file and uses SHA256 to generate the hash which is signed with the RSA signing algorithm using the client's private key. The client then sends the digital signature, public key & the plain text file to server whose activities are represented on right side of the network.

The server generates hash1 from the received plain text file using same SHA256 algorithm. Hash2 is obtained after using the RSA verifying algorithm on received digital signature using the public key of client. The hash1 & hash2 are compared to check if the file is tempered during transmission or not.

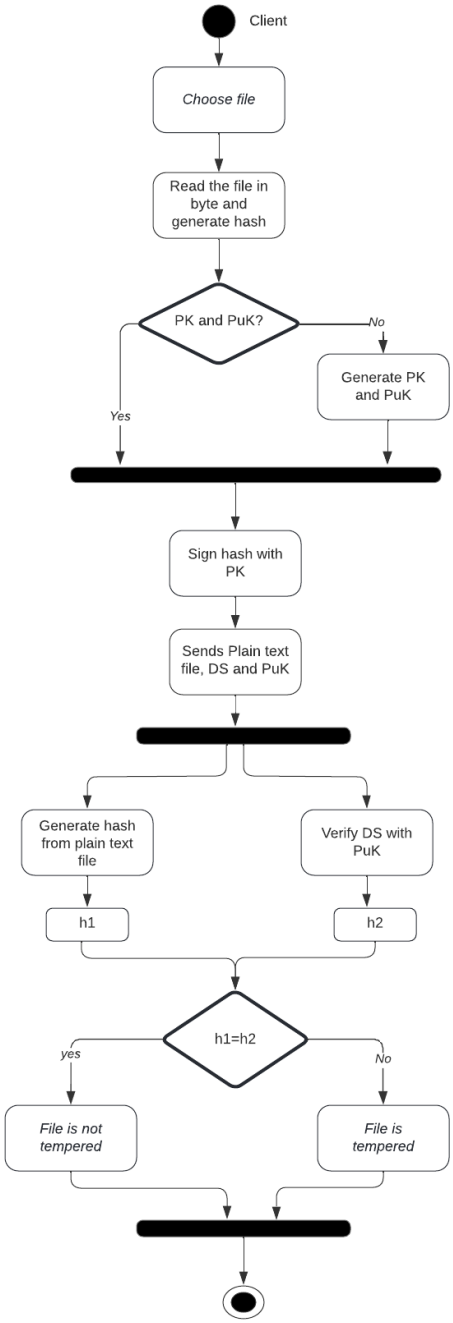


Figure 4.2: Activity Diagram of client and server communication

4.2 Algorithm Details

4.2.1 RSA Algorithm

The RSA idea is used to sign and verify a message called RSA digital signature scheme. It changes the role of the private and public keys. In RSA encryption, the public key of sender was used but in RSA digital signature scheme, the private key of the sender is used to sign the document and the receiver uses the sender's public key to verify it.
Currently, there are three Federal Information Processing Standard (FIPS) approved digital signature algorithms: Digital Signature Algorithm (DSA), RSA and Elliptic Curve Digital Signature Algorithm (ECDSA). RSA is used in this project.

The RSA algorithm is a public-key signature algorithm developed by Ron Rivest, Adi Shamir, and Leonard Adleman. Their paper was first published in 1977, and the algorithm uses logarithmic functions to keep the working complex enough to withstand brute force and streamlined enough to be fast post-deployment
Key Generation The required public and private keys are generated using following steps :

- *Two large prime numbers (p and q) are chosen.*
- *Compute n = p*q and φ(n) = (p-1)(q-1)*
- *Public key(e) is chosen where 1 < e < φ(n) and gcd(φ(n), e) = 1. e is relative prime of φ(n)*
- *Private key(d) is calculated such that d is multiplicative inverse of e as below,*

  $de \equiv 1 \bmod \varphi(n)$

  $de \bmod \varphi(n) = 1$

  $d \equiv e^{-1} \bmod \varphi(n)$
- *Private key pair is (n,d) is kept secret.*
- *Public key pair is (n,e) is publicly announced*

1. Signing Algorithm

$S = M^d \bmod n$

Client uses its private key or we can say it uses its private exponent "d" to create signature.

2. Verification Algorithm

$M = S^e \bmod n$

Sever uses client's public exponent(e) to the signature to create a copy of the message.

3. Euclidean Algorithm

It is used for determining GCD of two large positive integers.

while(b != 0)

gcd(a,b) = gcd(b, mod b)

gcd(a,0) = a

4. Extended Euclidean Algorithm

To find the multiplicative inverse.

$r_i = s_i * r_0 + t_i * r_1$

where, $s_i = s_i{-}2 - q_i{-}1 * s_i{-}1$

$t_i = t_i{-}2 - q_i{-}1 * t_i{-}1$

$i >= 2, s_0 = 1, s_1 = 0$

$t_0 = 0, t_1 = 1$

s and t are Bezout's identity, r1 and r2 are two integer

4.2.2 SHA Algorithm

The secure hash algorithm are a family of cryptographic hash functions that are published by the National Institute of Standards and Technology (NIST) along with the NSA. It was passed as a federal information processing standard also known as FIPS. SHA-256 uses 32-bit words

To be considered cryptographically secure the hash function should meet two requirements,

1. It is impossible for an attacker to generate a message that matches a specific hash value
2. It should be impossible for an attacker to create two messages producing the exactly same hash value. Even a slight change in the plaintext should trigger a drastic difference in the two digest.

A message is processed by blocks of 512 = 16 × 32 bits, each block requiring 64 rounds.

4.2.2.1 Basic operations

• Boolean operations AND, XOR and OR, denoted by $\wedge$, $\oplus$ and $\vee$, respectively.
• Bitwise complement, denoted by $\bar{}$
• Integer addition modulo 2^{32}, denoted by A + B.

Each of them operates on 32-bit words. For the last operation, binary words are interpreted as integers written in base 2.

• Rot R(A, n) denotes the circular right shift of n bits of the binary word A.
• Sh R(A, n) denotes the right shift of n bits of the binary word A.
• A||B denotes the concatenation of the binary words A and B.

4.2.2.2 Functions and constants

The algorithm uses the functions:

$Ch(X, Y, Z) = (X \wedge Y) \oplus (\bar{X} \wedge Z)$,
$Maj(X, Y, Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$,
$\Sigma 0(X) = RotR(X, 2) \oplus RotR(X, 13) \oplus RotR(X, 22)$,
$\Sigma 1(X) = RotR(X, 6) \oplus RotR(X, 11) \oplus RotR(X, 25)$,
$\sigma 0(X) = RotR(X, 7) \oplus RotR(X, 18) \oplus ShR(X, 3)$,
$\sigma 1(X) = RotR(X, 17) \oplus RotR(X, 19) \oplus ShR(X, 10)$,

and the 64 binary words Ki given by the 32 first bits of the fractional parts of the cube roots of the first 64 prime numbers:

0x428a2f98	0x71374491	0xb5c0fbcf	0xe9b5dba5
0x3956c25b	0x59f111f1	0x923f82a4	0xab1c5ed5
0xd807aa98	0x12835b01	0x243185be	0x550c7dc3
0x72be5d74	0x80deb1fe	0x9bdc06a7	0xc19bf174
0xe49b69c1	0xefbe4786	0x0fc19dc6	0x240ca1cc
0x2de92c6f	0x4a7484aa	0x5cb0a9dc	0x76f988da
0x983e5152	0xa831c66d	0xb00327c8	0xbf597fc7
0xc6e00bf3	0xd5a79147	0x06ca6351	0x14292967
0x27b70a85	0x2e1b2138	0x4d2c6dfc	0x53380d13
0x650a7354	0x766a0abb	0x81c2c92e	0x92722c85
0xa2bfe8a1	0xa81a664b	0xc24b8b70	0xc76c51a3
0xd192e819	0xd6990624	0xf40e3585	0x106aa070
0x19a4c116	0x1e376c08	0x2748774c	0x34b0bcb5
0x391c0cb3	0x4ed8aa4a	0x5b9cca4f	0x682e6ff3
0x748f82ee	0x78a5636f	0x84c87814	0x8cc70208
0x90befffa	0xa4506ceb	0xbef9a3f7	0xc67178f2

4.2.2.3 Padding

To ensure that the message has length multiple of 512 bits:
• first, a bit 1 is appended,
• next, k bits 0 are appended, with k being the smallest positive integer such that $l + 1 + k \equiv 448 \bmod 512$, where l is the length in bits of the initial message,
• finally, the length $l < 2^{64}$ of the initial message is represented with exactly 64 bits, and these bits are added at the end of the message.
The message shall always be padded, even if the initial length is already a multiple of 512.

4.2.2.4 Block decomposition

For each block $M \in \{0, 1\}^{512}$, 64 words of 32 bits each are constructed as follows:

• the first 16 are obtained by splitting M in 32-bit blocks

$M = W1||W2|| \cdots ||W15||W16$

• the remaining 48 are obtained with the formula:

$Wi = \sigma1(Wi-2) + Wi-7 + \sigma0(Wi-15) + Wi-16, 17 \leq i \leq 64.$

4.2.2.5 Hash computation

• First, eight variables are set to their initial values, given by the first 32 bits of the fractional part of the square roots of the first 8 prime numbers:

$H_1^{(0)} = 0x6a09e667$ $H_2^{(0)} = 0xbb67ae85$ $H_3^{(0)} = 0x3c6ef372$ $H_4^{(0)} = 0xa54ff53a$

$H_5^{(0)} = 0x510e527f$ $H_6^{(0)} = 0x9b05688c$ $H_7^{(0)} = 0x1f83d9ab$ $H_8^{(0)} = 0x5be0cd19$

• Next, the blocks $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$ are processed one at a time:

For t = 1 to N

– construct the 64 blocks W_i from $M^{(t)}$, as explained above

– set

$(a, b, c, d, e, f, g, h) = (H_1^{(t-1)}, H_2^{(t-1)}, H_3^{(t-1)}, H_4^{(t-1)}, H_5^{(t-1)}, H_6^{(t-1)}, H_7^{(t-1)}, H_8^{(t-1)})$

– do 64 rounds consisting of:

$T_1 = h + \Sigma_1(e) + Ch(e, f, g) + K_i + W_i$
$T_2 = \Sigma_0(a) + M a j(a, b, c)$
$h = g$
$g = f$
$f = e$
$e = d + T_1$
$d = c$
$c = b$
$b = a$
$a = T_1 + T_2$

1

It is assumed that the length of the message can be represented by a 64-bit integer.

- compute the new value of $H_j^{(t)}$

$$H_1^{(t)} = H_1^{(t-1)} + a$$
$$H_2^{(t)} = H_2^{(t-1)} + b$$
$$H_3^{(t)} = H_3^{(t-1)} + c$$
$$H_4^{(t)} = H_4^{(t-1)} + d$$
$$H_5^{(t)} = H_5^{(t-1)} + e$$
$$H_6^{(t)} = H_6^{(t-1)} + f$$
$$H_7^{(t)} = H_7^{(t-1)} + g$$
$$H_8^{(t)} = H_8^{(t-1)} + h$$

End for

The hash of the message is the concatenation of the variables H_1^N after the last block has been processed

$$H = H_1^{(N)}||H_2^{(N)}||H_3^{(N)}||H_4^{(N)}||H_5^{(N)}||H_6^{(N)}||H_7^{(N)}||H_8^{(N)}$$

4.2.2.6 Implementation: signatures

Implement the cryptographic hash function just described. Define the class sha256 with the method:

public static BigInteger hash(byte[] M)

input: M is a chain of bytes of arbitrary length;

output: a positive integer in the interval [0, 2256), the value of the hash of M.

Chapter 5
Implementation and Testing

5.1 Implementation

5.1.1 Tools Used

For making the UML diagrams, different online diagramming tools were used like

- Draw.io
- Lucid chart.

After the concept was formalized, it was time to think about the programming language in which it can be implemented. It was decided to implement our project work using Java programming language. Various features of Java were utilized like

- Java socket programming
- Java swing for GUI

Apache NetBeans was used as the code editor for running as well as debugging our application

Overleaf was used to prepare the project report.

5.1.2 Implementation Details of Modules

In the implementation, SHA256 algorithm was used to generate the message digest value using java software version 8.0. The message digest value needs to be expressed as 64 bit hexadecimal number. Since SHA-256 returns a hexadecimal representation, 4 bits are enough to encode each character (instead of 8, like for ASCII), so 256 bits would represent 64 hex characters as the length is always the same, not varying at all.

Client & Server

Java Socket programming is used for communication between the applications running on different JRE. Socket and ServerSocket classes are used for connection-oriented socket programming

The system implements one-way client and server communication. In this application, client sends a text file to the server, server reads the content of text file message and can download it. In Server two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, the message was read and written. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.

FileInputStream is used to obtain input bytes from a file. DataOutputStream to write from client to server. The client will choose a file to send then generate digital signature and send 4 files namely:

i) The original plain text file
ii) Signature file

iii) Public key files(E)
iv) Public key file(N)

The server will download those 4 files and verify the digital signature.

RSA.java

Keylength is usually 1024-4096 bits. The recommended key length today is 2048 bits, but the longer the key length is, the more time running the program takes, so we set the key- length to 1024.

Java provides some primitives, such as int or long, to perform integer operations. But sometimes, we need to store numbers, which overflow the available limits for those data types. Thus BigInteger class was used for our mathematical operations as it involved very big integer calculations larger than the primitive long type. It represents immutable arbitrary-precision integers.

We need to use probablePrime() method of BigInteger because of the size of the integers. It shouldn't be a problem though, because the chance of probablePrime not generating a prime is extremely low. we used java.security.SecureRandom class that provides a cryptographically strong random number generator. p & q were generated using probablePrime() passing keyLength & instance of SecureRandom.

5.2 Testing

Testing is one of the critical phases to know if there ar nay errors in an application. There are test cases create to validate applications, to know whether they are working or not. Test cases provide a brief knowledge about the working process of any system. The tsting phase consitst of Unit testing and System testing.

5.2.1 Test Cases for Unit Testing

Test case No. 1:

Objective : To check a slight change in the plaintext should trigger a drastic difference in the two digest.

Input : "Hello" "Helo"

Expected Result : Two hashes should be different.

Actual Result : There was drastic difference while changing a single letter.

Table 5.1: Unit testing for SHA256

Input	Output
Helo	185F8DB32271FE25F561A6FC938B2E264306EC304EDA518007D1764826381969
Helo	375738319E86099FE081FABEE238C40D6F038959DA383C99CA3FE146E5CC8B7E

Test case No. 2:
Objective : To check whether same plaintext generate same digest every time
Input : "passw0rd"
Expected Result : Two hashes should be same
Actual Result : The same plaintext generates the same digest every time

Table 5.2: Unit testing of SHA256

Input	Output
passw0rd	8F0E2F76E22B43E2855189877E7DC1E1E7D98C226C95DB247CD1D547928334A9
passw0rd	8F0E2F76E22B43E2855189877E7DC1E1E7D98C226C95DB247CD1D547928334A9

Test case No. 3:
Objective : To check whether encrypted hash can be decrypted back to original hash.
Input : "Hello"
Expected Result : The hash string of plain text should match with hash string after decryption.
Actual Result: The hash before encryption and after decryption are same.

Table 5.3 : Unit testing of RSA

Before encryption	After decryption
185F8DB32271FE25F561A6FC938B2E26 4306EC304EDA518007D1764826381969	185F8DB32271FE25F561A6FC938B2E26 4306EC304EDA518007D1764826381969

5.2.2. Test Case for System Testing

System testing is conducted on a complete, integrated system to evaluate the system with its specified requirements. In this testing the whole system is tested to check the errors.

The system testing of this system is given below:

Table 5.4:Test cases for System Testing

Test case No	Name	Input	Expected Output	Obtained Output	Remarks
1	Client sends text file with its own digital signature	Server clicks on Verify button	Successful verification	The file is not tempered	True
2	Client sends just the text file without its digital signature	Server clicks on Verify button	Unsuccessful verification	The file is tempered	True

5.3 Result Analysis

Analysis is a problem-solving technique that improves the system and ensures that all the component of system work efficiently to accomplish their purpose. After testing the system, we also analyzed the system in various aspects :

Byte array was used to store the content of file. Since the size of byte array is limited to size of Int which is 65,535. Due to this, the system was only able to hash the text file that has less than that number of characters. But when Java library SHA256 hash was used, the system was able to hash text file of any length.

If the size of text file is very large, it takes longer time to process and apply all features. Accuracy of the system can only be tested by the feedback of user. It is not possible to say that system is 100% accurate unless user gives feedback but the system was able to tell whether the file was modified accurately.

5.3.1 Sender changing single letter in a file

At first the file name [Hell1.txt] with content "Hell one" was sent along with its own digital signature. Later when the content was modified to "Hell One" with capital O and the signature for that was not sent. The Server tries to verify "Hell One" with the digital signature of "Hell one" and couldn't verify. The system alerts the recipient with the message "The file if tempered". Thus the data integrity is provided by the system.

5.3.2 Changing file name

The necessary condition for the system to correctly label whether the file has been tempered or not is that the file content shouldn't be changed even though the file name can be changed. At first the file name [Hell1.txt] with content "Hell one" was sent along with its own digital signature. Later the file was renamed to [Hell2.txt] without modifying its content and sent it to Server. When the server clicked on "Verify" button, the system alerted, "The file is not tempered". Since only the file name was modified not its content.

5.3.3 Receiver changing the file content

Client sent the file along with the signature associated with that file. When the Server changed the content of the file to claim that sender has sent it, the system alerted file is tempered when trying to verify the signature bringing the truth forward.

Analysis of SHA Output :

To check the implementation the following values were used, given in hexadecimal notation.

Table 5.3.1: Result Analysis of SHA256

Input hash	61 62 63 1DCCE1A3487E6F500BA1717A74375462C4612B5D9A2D356D69103263FCEA478B
input hash	Sixtyone Sixtytwo Sixtythree 8FF76397CA93EE44D352FD1E4DE1530358EBF8C9E8A55DD93ECF283DFC230232
Input hash	Six hundred sixteen thousand two hundred sixty three 82C55757ECF9433BC941FFAD368D3856939700CED722CC9B9072E18964EA9301

CHAPTER 6
Conclusions and Future Recommendations

6.1 Conclusions

The desktop based application named "Deeps File Sharer" is developed. The system is designed to provide the simple text file sharing from client to server with digital signature functionality. It was just the modality of what can be achieved using digital signature. The problem of whether the file has been tempered or not was solved.

RSA uses public and private keys and so files signed with the Private key can be correctly verified by the Public Key. Also, the difficulty in factoring out these keys makes RSA secured. Apart from the security advantage, this system has been proved to maintain the integrity of the information secured since the length of the message is preserved during any of the operations of signature and verification. The system has therefore eliminated the fear of losing vital information to an eavesdropper or an enemy at any point in time.

But at the end of the day it is up to the Network Administrator to make sure that his network is out of danger. Also it is possible that anyone with access to client's computer can easily imposter the client's identity.

6.2 Future Recommendations

The algorithm used in the system can be extended to sign the files other than with .txt extension. The encryption and decryption facility can be incorporated to make the sharing of information more confidential. There is just one way communication between client and server although they can exchange file as long as they want. But the sharing of file can be made two way in the future version.

References

[1] Ramasamy, R. R. and Prabakar, M.A. (2009): Digital signature scheme with message recovery using knapsack-based ECC, International Journal of Network Security, Vol. 12 No. 1, pp. 15-20.

[2] Roja, P. P. and Avadhani, P.S. (2007): Digital signature development using truncated polynomials, International Journal of Computer Science and Network Security vol. 7 No. 7.

[3] Application of Digital Signature for Securing Communication Using RSA Scheme based on MD5, Stephen Fashoto, University of Swaziland Kwaluseni Swaziland

[4] Harrisx retrieved from www.cccure.org/documents/cryptography/cisspallinone.pdf (2001)

[5] L. C. B. Vagner Schoaba, Felipe Eduardo Gomes, "Digital signature for mobile devices: A new implementation and evaluation," 2011.

[6] Z. Y. Qiao-yan Wen, Hua Zhang, "A digital signature schemes without using one-way hash and message redundancy and its application on key agreement," (Beijing, China), 2007.

[7] Z. X. Wang Shaobin, Hong Fan, "Optimistic fair-exchange protocols based on dsa sig natures," in *Handbook of Applied Cryptography*, (College of Computer, Huazhong Uni versity of Science and Technology, Wuhan, China), 2004.

[8] A. A. Farah Jihan Aufa, Endroyono, "Security system analysis in combination method: Rsa encryption and digital signature algorithm," (Indonessia), 2018.

[9] X. Z. X. Tang, "Research and implementation of rsa algorithm for encryption and de cryption," (China), 2011.

[10] V. F. J. Padhye and D. Towsley, "A stochastic model of tcp reno congestion avoidance and control," (University Of Massachusetts, Amherst, MA), 1999.

[11] G. X. Na Zhu, "The application of a scheme of digital signature in electronic govern ment," (Hebei Univ. of Technol., Tianjin), 2008.

[12] "Wireless lan medium access control(mac) and physical layer (phy) specification," 1997.

[13] A Method for Obtaining Digital Signatures and Public-Key Cryptosystems32
R.L. Rivest, A. Shamir, and L. Adleman

Appendix-I

```
public class RSA {
   private int keyLength = 1024;
   private SecureRandom rand;
   private BigInteger p;
   private BigInteger q;
   private BigInteger n;
   private BigInteger lambda;
   private BigInteger e;
   private BigInteger d;
   private BigInteger[] publicKey = new BigInteger[2];
   private BigInteger[] privateKey = new BigInteger[2];
   private final BigInteger one = new BigInteger("1");
   public RSA() {
      rand = new SecureRandom();
      p = BigInteger.probablePrime(keyLength, rand);
      q = BigInteger.probablePrime(keyLength, rand);
      n = p.multiply(q);
      lambda = p.subtract(one).multiply(q.subtract(one));
      e = BigInteger.probablePrime(keyLength / 2, rand);
         while(e.intValue() < lambda.intValue() && (lambda.gcd(e).intValue() -
one.intValue()) > 0) {
         e.add(one);}
      d = e.modInverse(lambda);
      publicKey[0] = e;
      publicKey[1] = n;
      privateKey[0] = d;
      privateKey[1] = n;
   }
```

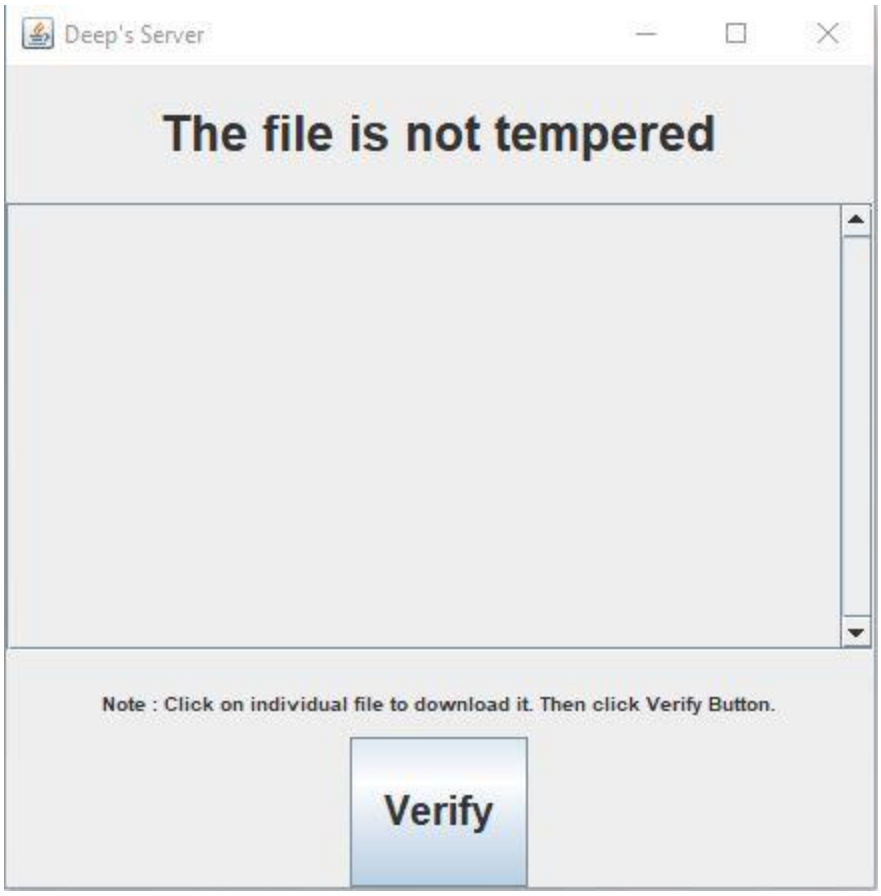
```
  public byte[] encrypt(BigInteger d, BigInteger n, byte[] message) {
     return (new BigInteger(message)).modPow(d, n).toByteArray();
   }
   public byte[] decrypt(BigInteger e, BigInteger n, byte[] message) {
     return (new BigInteger(message)).modPow(e, n).toByteArray();
   }
   //To Convert HASH/CIPHER bytes into String for printing
   public String toString(byte[] cipher) {
     String s = "";
     for (byte b : cipher) {
        s += Byte.toString(b);
     }
     return s;
   }

   public BigInteger[] getPublicKey() {
     return publicKey;
   }

   public BigInteger[] getPrivateKey() {
     return privateKey;
   }

}
```

Appendix II


deep's client
deep's File Sender
choose a file to send
1. Choose File
2. Generate Signature
3. Send File


Deep's Server
Deep's File Receiver
Note : Click on individual file to download it. Then click Verify Button.
Verify


Deep's File Downloader
Deep's File Downloader
Are you sure you want to download file Hello.txt
Yes
No


Deep's Server
The file is not tempered
Note : Click on individual file to download it. Then click Verify Button.
Verify