IMPLEMENTATION OF S/MIME

P-1218

Project Report

Submitted in partial fulfillment of the
Requirement for the award of the degree of the

Bachelor of Computer Science and Engineering
Of
Bharathiar University, Coimbatore.

Submitted by

Z. Althaf Ahamed
M. Pradeep
T. Thangavel

Under the guidance of

Mrs.M.S. Hema B.E.,
Lecturer
Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE – 641006.

MARCH 2004.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University, Coimbatore)

CERTIFICATE

This is to certify that the project entitled

IMPLEMENTATION OF S/MIME

is done by

Z. Althaf Ahamed
0027K0155

M. Pradeep
0027K0189

T. Thangavel
0027K0207

and submitted in partial fulfillment of the
requirement for the award of the degree of the

Bachelor of Computer Science and Engineering
Of
Bharathiar University, Coimbatore.

[illegible]

Professor & Head of the department
(Dr.S.THANGASAMY)

[illegible]

Guide
(Mrs.M.S. HEMA B.E.)

Certified that the candidates were examined by us in the project work
Viva voce examination held on 23.3.2004.

[illegible] 23/3/'04

Internal Examiner

[illegible]

External Examiner

Declaration

We,

Z. Althaf Ahamed	**0027K0155**
M. Pradeep	**0027K0189**
T. Thangavel	**0027K0207**

declare that the project entitled "Implementation of S/MIME", is done by us and to the best of our knowledge, a similar work has not been submitted earlier to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

This project report is submitted on the partial fulfillment of the requirement for all awards of the degree of Bachelor of Computer Science and Engineering of Bharathiar University.

Place : Coimbatore.

Date : 22.3.2004.

[illegible]

Z. Althaf Ahamed

[illegible]

M. Pradeep

[illegible]

T. Thangavel

Countersigned : [illegible]

Guide : **Mrs.M.S. Hema B.E.**

DEDICATED TO OUR BELOVED PARENTS AND FRIENDS

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, we take this opportunity to thank the management of our college for having provided us excellent facilities to work with. We express our deep gratitude to our Principal Dr.K.K. Padmanabhan B.Sc(Engg),M.Tech., for ushering us in the path of triumph.

We are always thankful to our beloved Professor and the Head of the Department, Prof.S. Thangasamy B.E.(HONS)., whose consistent support and enthusiastic involvement helped us a great deal.

We are greatly indebted to our beloved guide Mrs.M.S. Hema B.E., Lecturer, Department of Computer Science and Engineering for her excellent guidance and timely support during the course of this project. As a token of our esteem and gratitude, we honour her for her assistance towards this cause.

We also thank our project coordinator Mrs. S. Chandrakala M.E., and our beloved class advisor Mrs.M.S. Hema B.E., for their invaluable assistance.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the Department of Computer Science and Engineering.

We feel proud to pay our respectful thanks to our Parents for their enthusiasm and encouragement and also we thank our friends who have associated themselves to bring out this project successfully.

SYNOPSIS

The project entitled “Implementation of S/MIME “is developed to provide security for the MIME messages that are sent during communication.

This project aims at providing data security across the Public key Infrastructure Standards. This project provides security in terms of various cryptographic techniques such as Encryption-Decryption, Digital signatures and Data enveloping. The project is key based i.e. the public key and private key. The complete security resides in the key because the algorithm is standardized. This system can be applied in both Internet and Intranet.

Digital signatures are used to provide authentication and data integrity during the message transfer. Digital signature is a signed document generated and sent to the designated host. Digital signature verification is done in the case of non-repudiation.

Data enveloping is a method of wrapping the digital signature. By enveloping the content is completely hidden. Enveloping provides confidentiality for the messages transferred.

CONTENTS

1. INTRODUCTION

1.1 EXISTING SYSTEM & LIMITATIONS

The existing system does not have any security for the information transferred during the communication. The messages are sent through the MIME protocol which doesn't have any Standardized techniques or algorithm for encryption and decryption. The security lies in the algorithm to be implemented.

The system follows the Group-id technique i.e. whenever the message is sent from the source machine, the message will reach all the team members and the person having the key will decrypt the message, it is really cumbersome and less secured.

These were some of the drawbacks met in the existing system.

The existing system doesn't provide privacy for the messages. The major drawback is the non-repudiation that is the sender should not be able to falsely deny later that he sent a message. The existing system lack data integrity where the possibility for the receiver of a message to verify that it has not been modified in transit.

1.2 PROPOSED SYSTEM AND ADVANTAGES

The proposed system is developed, owing to the number of drawbacks in the existing system. The proposed system aims to remove most of the drawbacks found extensively in the existing system which can be thought of as maintenance friendly. The proposed system maintains proper flow of control and relationships.

The security for the information is given by encryption and decryption algorithm which is key based i.e. the public key and private key. The authentication and data integrity are provided by digital signature by making use of hashing algorithm. Confidentiality of information is assured through data enveloping.

The public key cryptography is chosen for security over messages mainly to avoid the key distribution problem. Authentication and data integrity for secured messages are provided through hashing algorithms. The verification option available in digital signature serves the non-repudiation activity.

The digitally signed message can be enveloped which acts as a wrapper for the message. Enveloping is mainly done for confidentiality purpose. In rare cases attacks are also possible on cipher text. So enveloping is done to prevent it.

Thus the project provides security by addressing all the four major cryptographic pillars namely, authentication, data integrity, confidentiality and non-repudiation of the cryptographic field.

The user has the option of selecting the modules depending on the requirements. The proposed system is developed to be user-friendly. The proposed System provides complete security to the data that are being handled by the users.

Features of the proposed system are

- Privacy
- Authentication
- Signature
- Data integrity
- Non repudiation

SYSTEM REQUIREMENT ANALYSIS

2. SYSTEM REQUIREMENT ANALYSIS

2.1 PRODUCT DEFINITION

This project is for providing security for the MIME messages that are sent during communication. This project provides security in terms of various cryptographic techniques such as Encryption-Decryption, Digital signatures and Data enveloping. The project is key based i.e. the public key and private key.

2.2 PROJECT PLAN

The project titled "Implementation of S/MIME" is to develop a standardized method which can be used for providing data security across the Public key Infrastructure Standards.

The following are the key steps involved

- Obtaining the message digest from the original text.
- Attaching the digest with the original text and encrypting.
- Enveloping the encrypted document.
- The receiver unwrapping the document from the envelope to retrieve the message and a new digest.
- Comparing the message digests to check the authenticity of the message.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 PURPOSE

The purpose of this project is to create a standardized method to create security for the messages sent during communication.

3.2 SCOPE

The scope of the proposed system is to develop and establish complete security to the data that is being sent via net and intranet for all the users.

3.3 PRODUCT OVERVIEW AND SUMMARY

The project is developed to provide security, authentication, data integrity, nonrepudiation and privacy for the messages sent during communication. The information is secured by encryption-decryption techniques, using RSA algorithm. The Digital signature is used to provide authentication and data integrity. MD5 hashing algorithm is used to obtain the message digest in digital signature. Data enveloping is implemented using symmetric algorithm.

This project provides security in terms of three modules namely,
1. Encryption and Decryption
2. Digital signatures
3. Data Enveloping.

The first module Encryption and Decryption using RSA, is a type of asymmetric public key cryptography. RSA was developed by Ron Rivest, Adi Shamir, and Leu Adleman at MIT. Public key cryptography was mainly developed to overcome the key distribution difficulty faced in symmetric algorithms. RSA works on a pair of keys - Public key and the Private Key. The Public key and the private key are generated using key generation algorithms. Among the pair of keys, one key is made public and the other private. Encryption is done using the public key and decryption is done using the private key.

The RSA consists of three sub-modules:

1. Key generation

2. Encryption

3. Decryption

The second module is Digital Signatures which is generated using the RSA and a hashing algorithm. In cases where the communication becomes unreliable, digital signatures play a vital role. A digital signature is a signed document which is generated by the sender and sends to the designated party. The digital signatures involves two processes namely,

a. Digital signature creation
b. Digital signature verification

In digital signature creation, MD5 hashing algorithm is used to obtain the message digest. The message digest is then encrypted using the sender's private key and the result is the signature. The signature is then appended with the original data which now becomes the signed document, is send to the receiver.

Digital signature verification is done in cases where the sender or the receiver denies the communication that has taken place. In such cases, the receiver applies the hashing algorithm for the original data and obtains the message digest. The receiver also applies the public key for the signature and gets the message digest. Both the message digest is then compared.

The last module is Data Enveloping. Enveloping acts as a wrapper for the signed document. The signed document is the output of the digital signature module. In enveloping, the signed document is given as input and data enveloping is done to the signed document by encrypting it.

The encryption involves the data encryption standard (DES) for encrypting and decrypting the data. The symmetric key is encrypted and sent to the receiver. The symmetric key is encrypted using RSA private key. The enveloped data follows the encrypted symmetric key. Data enveloping is done to obtain privacy in the communication.

3.4 DEVELOPING AND OPERATING ENVIRONMENT

HARDWARE ENVIRONMENT

It describes the environment in which the system has been developed. The hardware interfaces used for development are listed below.

Processor	: Pentium II
Primary memory	: 128 Mb or more
Secondary memory	: More than 6GB
User interface	: keyboard\mouse

SOFTWARE ENVIRONMENT

The following software products are required to run this software at the implementation environment

Operating system	: windows 9x/NT/XP
Language	: C.
Complier	: Borland C.

3.5 FUNCTIONAL SPECIFICATION

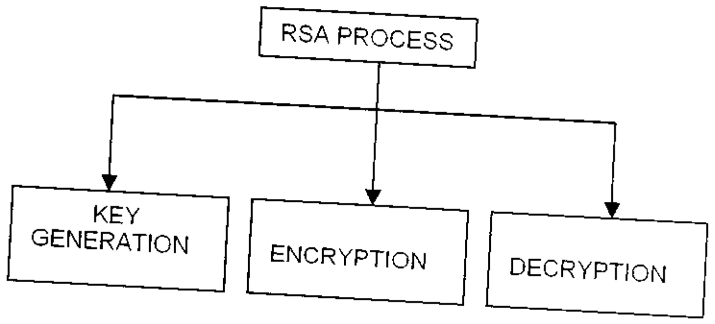
MODULE DESCRIPTION

The project involves three modules namely,

a. Encryption-decryption
b. Digital signatures
c. Data Enveloping

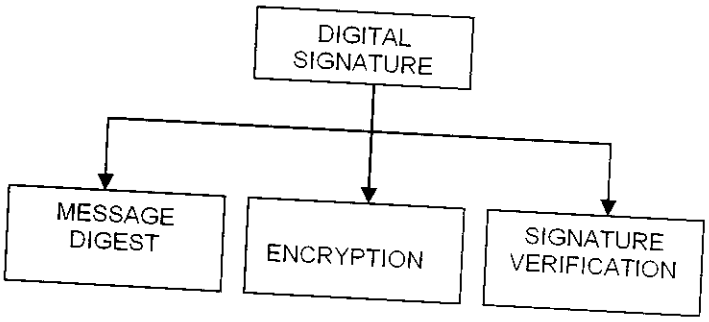
ENCRYPTION – DECRYPTION

This module consists of key generation, encryption and decryption process using the RSA algorithm. This is a type of asymmetric public key cryptography. RSA works on a pair of keys - Public key and the Private Key. The Public key and the Private key are generated using key generation algorithms. Among the pair of keys, one key is made public and the other private. Encryption is done using the public key and decryption is done using the private key.



DIGITAL SIGNATURES

A Cryptographic primitive which is fundamental in authentication, authorization, and non-repudiation is the digital signature. The purpose of digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of signing details transforming the message some secret information held by the entity into a tag called a signature.



Module diagram for Digital signature

The digital signature is a number dependent on some secret known only to the signer and additionally on the content of the message being signed. Signatures are verifiable; if a dispute arises whether a party signed a document, the third party must be able to resolve the matter equitably, without requiring access to the signer's secret information. The digital signatures offer many advantages in terms of functionality and implementation.

In situations where there is not complete trust between sender and receiver, some thing more than authentication is needed. The most attractive solution to this problem is digital signature.

· It must be able to authenticate the contents of the signature
· The signature must be verifiable by third parties to resolve disputes
· It must be relatively easy to produce the digital signature
· It must be relatively easy to recognize and verify the signature
· It must be computationally infeasible to forge a digital signature either by constructing a new message form an existing digital signature or by constructing a fraudulent digital signature for the given message.

The output of the hash function (MD5) is the message digest. The message digest is then encrypted using RSA private key. This yields the signature. The signature is then appended with the original message to produce the signed document. The signed document is sent to the other party.

Digital signature verification is done in cases where the sender or the receiver denies the communication that has taken place. In such cases, the receiver applies the hashing algorithm for the original data and obtains the message digest. The receiver also applies the public key for the signature and gets the message digest. Both the message digest is then compared.

DATA ENVELOPING

The last module is Data Enveloping. Enveloping acts as a cover for the signed document. The signed document is the output of the digital signature module. In enveloping, the signed document is given as input and data enveloping is done to the signed document by encrypting it using DES algorithm.

4. DESIGN DOCUMENT

The system design is the high level strategy for solving the problem and building a solution. System design includes decisions about the organization of the system into subsystems, allocation of subsystems to hardware and software components, and major conceptual and policy decisions that form the framework for the detailed design.

RSA ALGORITHM

The Rivest-Shamir-Adleman (RSA) algorithm is one of the most popular and secures public-key encryption methods.

ALGORITHM DESCRIPTION

To generate the two keys, choose two random large prime numbers, p and q. for maximum security, choose p and q of equal length. Then compute the following

$$n = p*q$$

Then randomly choose the encryption key, e such that e and [p-1][q-1] are relatively prime. Finally, use the extended Euclidean algorithm to compute the decryption key, d, such that

$$e*d=1 \bmod [p-1][q-1]$$

in other words

$$d=e-1 \bmod [p-1][q-1]$$

note that d and n should be relatively prime .

Using an encryption key (*e*,*n*), the algorithm is as follows:

1. Represent the message (plain text P) as an integer between 0 and (*n*-1). Large messages can be broken up into a number of blocks. Each block would then be represented by an integer in the same range.

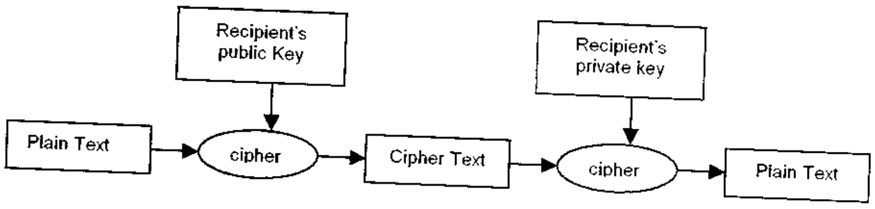
2. Encrypt the message by raising it to the *e*th power modulo *n*. The result is a cipher text message C.

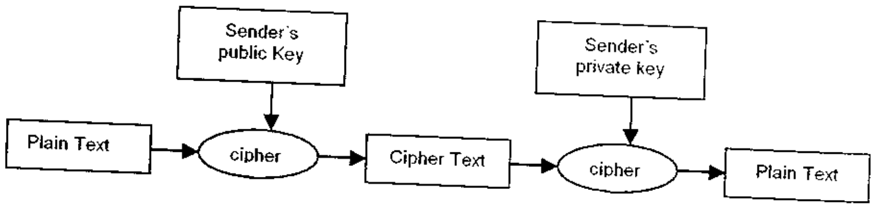
The encryption key (*e*,*n*) is made public. The decryption key (*d*,*n*) is kept private by the user. Using a decryption key (d,*n*), the algorithm is as follows:

1. Represent the cipher text message C as an integer between 0 and (*n*-1). Large messages can be broken up into a number of blocks. Each block would then be represented by an integer in the same range.

2. Decrypt the message by raising it to the dth power modulo *n*. The result is a plain text message.

Rather than using the same key to both encrypt and decrypt the data, the RSA system uses a matched pair of encryption and decryption keys. Each key performs a one way transformation upon the data.





Security of RSA

Cryptographic methods cannot be proven secure. The RSA method's security rests on the fact that it is extremely difficult to factor very large numbers. If 100 digit numbers are used for *p* and *q*, the resulting *n* will be approximately 200 digits. Any cryptographic technique, which can resist a concerted attack, is regarded as secure. At this point in time, the RSA algorithm is considered secure.

MD5 ALGORITHM

Ron Rivest at MIT developed the MD5 message-digest algorithm. This algorithm takes as input a message of arbitrary length and produces as output a 128-bit message digest. The input is processed in 512-bit blocks.

The processing consists of the following steps:

Step1: Append Padding bits.

The message is padded so that its length in bits is congruent to 448 modulo 512. That is, the length of the padded message is 64 bits less than an integer multiple of 512 bits. Padding is always added, even if the message is already of the desired length. The padding consists of a single 1-bit followed by the necessary number of 0-bits.

Step 2: Append Length.

A 64-bit representation of the length in bits of the original message is appended to the result of the first step. If the original length is greater than 264, then only the low order 64 bits of the length are used. Thus the field contains the length of the original message, modulo 264 .The outcome of the first two steps yields a message that is an integer multiple of 512 bits in length.

Step 3: Initialize MD buffer.

A 128-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as four 32-bit registers (A, B, C, and D). These registers are initialized to the following 32-bit integers (hexadecimal values):

A = 67453201
B = EFCDAB89
C = 98BADCFE
D = 10235476
These values are stored in little-endian format, which is the least significant byte of a word in the low-address byte position. As 32-bit strings, the initialization values (in hexadecimal) appear as follows:

Word A: 01 23 45 67
Word B: 89 AB CD EF
Word C: FE DC BA 98
Word D: 76 54 32 10

Step 4: Process message in 512-bit (16-word) blocks.

The heart of the algorithm is a compression function that consists of four rounds of processing. The four rounds have a similar structure, but each uses a different primitive logical function. Each round takes as input the current 512-bit block being processed and the 128-bit buffer value ABCD and updates the contents of the buffer. Each round also makes use of one-fourth of a 64-element table, constructed from the sine function. The output of the fourth round is added to the input of the first round. The addition is done independently for each of the four words in the buffer wit each of the corresponding words, using addition modulo 232.

Step 5: Output.
After all 512-bit blocks have been processed; the output is the 128-bit message digest.

DATA ENCRYPTION STANDARD

DES is a block cipher it encrypts data in 64 bit block of plaintext goes in one end of the algorithm and 64 bit block of cipher text comes out the other end. Des is the symmetric algorithm the same algorithm and key is used for both encryption and decryption .The key length is 56 bits usually the key is expressed as the 64 bit number but every 8th bit is used for parity checking and is ignored. Theses parity bits are the least significant bits of the key bytes. The key can be any 56 bit number and can be changed at any time.

ALGORITHM

Des operates on a 64 bit block of plain text. After the initial permutation the block is broken into right half and left half each 32 bit long. Then there are 16 rounds of operation in which the data are combined with the key. After the sixteenth round the right and the left halves are joined and a final permutation finishes the algorithm.

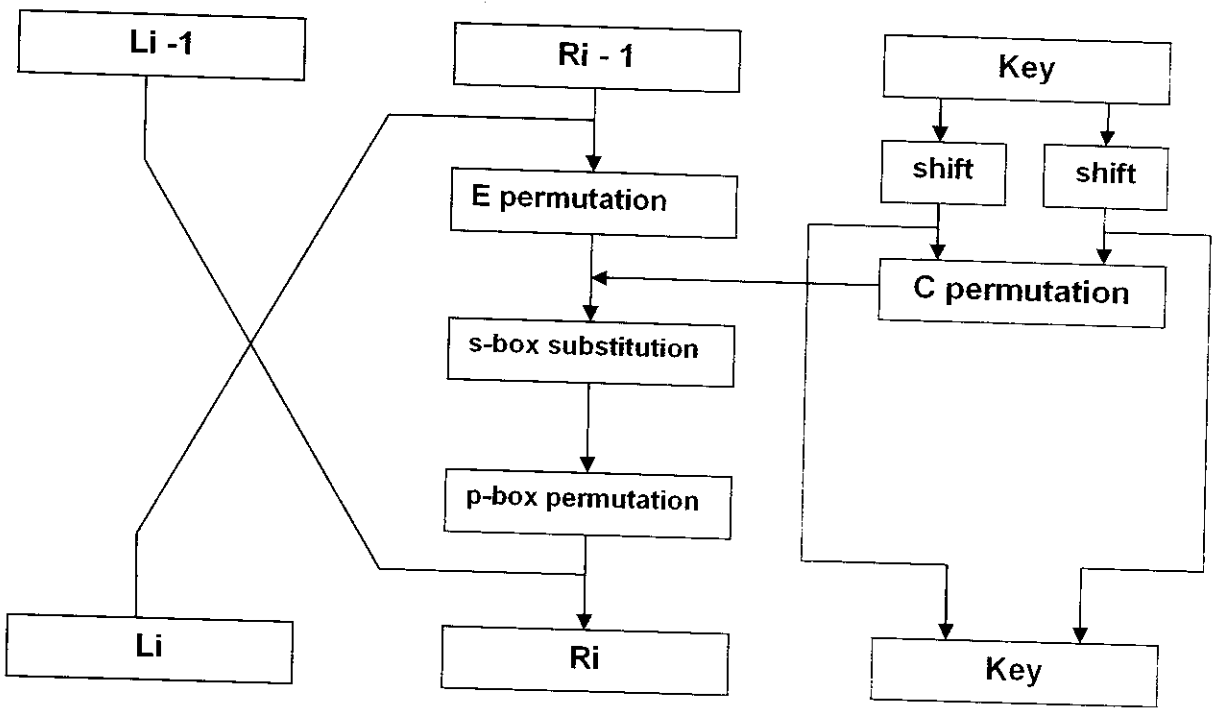
In each round the key bits are shifted then 48 bits via an expansion permutation, combined with 48 bits if shifted and permuted key via an n XOR, send through 8 s-boxes producing 32 new bits and permuted again. The output function is combined with the left half via another XOR. The result of these operations becomes the new right half and the old right half becomes the new left half. These operations are repeated 16 times making16 rounds of DES.

a. INITIAL PERMUTATION

The initial permutation occurs before the first round. Which will transpose the input block the initial permutation and the final permutation will not affect the des security.

b. KEY TRANSFORMATION

Initially the 64 bit des key is reduced to 56 bit key by ignoring every 8th bit. These bits can be used as parity check to ensure the key is error free. After the 56 bit is extracted different 48 bit sub key is generated for each round of 16 rounds of des. First the 56 bit key is divided into two halves. Then the half is circularly shifted left by either one or two bits depending on the round after being shifted 48 out of 56 bits are selected, it is called as compression permutation.



One round of DES

c. EXPANSION PERTMUTATION

This operation expands the right half of the data from 32 bits to 48 bits. Because this operation changes the order of the bits as well as repeating certain bits this is known as expansion permutation. This operation has two purposes: it makes the right half the same size as the key for XOR operation and it provides a longer result that can be compressed during the substitution operation. Des is designed to reach the condition of having every bit of the cipher text depend on every bit of the plain text and every bit of the key as quick as possible.

After the compressed key is XORed with the expanded block, the 48 bit result moves to a substitution operation. The substitutions are performed by eight substitution boxes or s-boxes. Each s-box has the 6 bit input and a 4 bit output and there are eight different s-boxes. Each s-box is a table of 4 rows and 16 columns. The six bit input of the s-box specifies under which row and column number to look for the output.

d. THE P-BOX PERMUTATION

The 32 bit block of the s-box substitution is permuted according to a p-box. This permutation maps each input bit to an output box position. No bits are used twice and no bits are ignored. This is called a straight permutation. Finally the result of the p-box permutation is XORed either the left half of the initial 64 bit block.

e. FINAL PERMUTATION

The final permutation is the inverse of the initial permutation. The left and the right halves are not exchanged after the last round of des, instead the concatenated block is used as the input to the final permutation.

The encryption involves the data encryption standard (DES) for encrypting and decrypting the data. The symmetric key is encrypted and sent to the receiver.

The symmetric key is encrypted using RSA private key. The enveloped data follows the encrypted symmetric key. Data enveloping is done to obtain privacy in the communication.

5. PRODUCT TESTING

TESTING AND TEST PLAN

i) TEST PLAN

- Testing is done with one primary objective to ensure the quality of software before live operations.
- Testing is a process of executing of a program with intent of finding an error.
- A good test is not redundant that is there is no point in conducting a test that has the same purpose as another test.

a. UNIT TESTING

Unit testing focuses verification effort on the smallest unit of the software design- The software component or a module. Important control paths have been tested comprehensively to uncover errors within the boundary or the module.

b. INTERGARTION TESTING

Integration testing is a systematic technique for constructing program structure while at the same time conducting tests to uncover errors associated with interfacing. All components have been combined in advance and the entire program has been tested as a whole.

c. STRESS TESTING

Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency or volume. Test cases that require maximum memory or other resources have been executed and the system has been performing consistently without crashing.

d. SENSITIVITY TESTING

Sensitivity testing has been performed to check if a very small range of data contained within the bounds of valid data for a program may cause extreme and even erroneous processing.

ii) TEST CASE

TESTING	ACTION	EXPECTED RESULT	OBSERVED RESULT
Unit testing	Checking for loops eg. Choice : 1 Choice :2	Creation of digital signature computed Verification of digital signature	Signature created Signature verified
Sensitivity testing	Range of value for key should be 64 bit	Program should execute properly	Key generation executed properly
Stress testing	Avoiding very long path name for files	If exceeds 40 characters should be invalid	Characters less than forty accepted
Integration testing	Modules tested individually and also whole system	Linking between the three modules should be proper	The modules individually and linking are proper
Dataflow testing	Testing flow of data by printing value of data frequently	Data stored in variable should be correct	If data stored is wrong then error is shown
Unit testing	Clicking mouse at appropriate position within the range	Correct action should be taken	By clicking the Mouse appropriate modules are linked and values got for processing.

6. FUTURE ENHANCEMENTS

Our project is so flexible that it can be enhanced so easily. The primary objective of the project is to provide security to the data during data transformation. The system met the initial primary requirement but there is ample scope for the system to be improved. Some of the features and facilities can be incorporated in the system are listed below.

- Increasing the key length from 64 bits to 128 bits
- Increase the security by using three keys(Triple DES)

7. CONCLUSION

This software provides maximum security to the data during transmission. It protects the data from the intruders and unauthorized persons. Specifically, digital signatures provide authentication and data integrity, and in rare case of intruders the verification process is also included. The system has user friendly features. It is possible for any novice user to use the system.

The programming techniques used in the design of the system provide a scope for further expansion and implementation of any changes, which may occur in the future.

Data enveloping serves as the more secure process which provides more security and flexibility and it is the option of the users to choose. The plain text is encrypted and decrypted quickly and accurately then transmitted across the network.

This system developed will be useful for the organization to maximize the security of important data during transmission very effectively and securely. The system is also proven to be more flexible to future enhancements.

Since, the requirements of any organization and their standards are changing day by day, the system has been designed in such a way that its scope and boundaries could be expanded in the future with little modifications.

This system has been developed using turbo-C. The system has been tested and performances found satisfactory.

8. REFERENCE

1. SCHNEIER. B, 1992, "**Applied Cryptography**", Replika Press Pvt Ltd, Second Edition

2. STALLINGS. W, 1997, "**Cryptography and Network Security**", Pearson Education Inc, Second Edition

3. RHEE.M.Y, 1995, " **Cryptography and security Communications**".

4. MENEZES. A, OOORSCHOT. P, VANSTONE. S, 1998, " **Applied Cryptography**", CRC Press.

5. AWARD.E.M,1987, "**System Analysis and Design**" , Galgotia Publications, Second Edition

6. HAWRYSZKIEWYCZ.I.T,1992, "**System Analysis and Design**", Prentice Hall of India Pvt Ltd, Third Edition

7. PRESSMAN.R.S, 1990 "**Software Engineering**", McGraw Hill, Fifth Edition

Web Sites:

8. http://courses.cs.vt.edu/~cs5204/fall99/protection/rsa.html

9. http://www.hull.ac.uk/Hull/CC_Web/docs/cnotes/c3.html#c3-5

10. http://pardus-larus.student.utwente.nl/DOC/Lang/C/ ctut/ctutorial/Advantages-of-C.html

11. http://www.specs.de/users/danni/compars/

12. http://gandalf-library.sourceforge.net/

13. http://www.verisign.com/repository/crptintr.html

14. http://www-Ibm.comsoftware/webservers/appservdoc/ v40/ae/infocenter/was/050501.html

15. http://docs.sun.com/source/816-6154-10/contents.htm#1016376

16. http://www.xml.com/pub/a/2001/08/08/xmldsig.html

9. ANNEXURE

9.1 SAMPLE CODE

```
#include <dos.h>
#include "desh.h"
#include <stdio.h>
#include <process.h>
#include <conio.h>
#include "digest.h"

void init_mouse();
void show_mouse();
void hide_mouse();
void rsamain(void);
void move_mouse(int x,int y);
void getmousepos(int *button,int *x,int *y);
void env(int);
void shadow(int ,int ,int ,int );
void textbar(int ,int ,int ,int ,int );
void square(int);
void screen(void);
void rsakey(void);
int findkey(void);
int encrypt(void);
int decrypt(void);
union REGS ii,o;
long double max;

void choice1(void);
void choice2(void);
void digital(void);
void envelope(void);
void quit(void);
void combine(char *,char *,char *);
void extract(char *,char *,char *);
int compare(char *,char *);

void main()
      {
      int cho;
      clrscr();
      env(1);
      printxy(16,5,"Enter 1 Digital/ 2 Envelope/ 3 RSA : " ,11);
      cscanf("%d",&cho);
      getch();
      switch(cho)
      {
      case 1:
```

```
                digital();
                break;


        case 2:
                envelope();
                break;
        case 3:
                rsamain();
                break;
        default:
                printxy(16,7,"Invalid choice " ,11);
                delay(1000);
                exit(1);

        }

        getch();
        }
        void envelope(void)
        {
        char select;
        clrscr();
        printxy(26,3,"MODULE 3 - ENVELOPING SIGNATURE",11);
        printxy(16,8,"Enter ur choice : 1-Tx ,2-Rr : ",3);
        textcolor(15+BLINK);
        cscanf("%c",select);
        select=getche();
        if(select=='1')
        choice1();
        else if(select=='2')
        choice2();
        else
        quit();
        return;
        }
void choice1(void)
        {
        long double pub,modu;
        char *keyip;
        printxy(5,11,"SENDER SIDE ENVELOPE CREATION",2);
        printxy(30,15,"DES KEY     : ",12);
        textcolor(15);
        cscanf("%s",keyip);
        printxy(30,17,"PUBLIC KEY : ",12);
        textcolor(15);
        cscanf("%Lf",&pub);
        printxy(30,19,"MODULUS     : ",14);
        textcolor(15);
        cscanf("%Lf",&modu);
        getch();
        rsaemain(keyip,"keyip1.txt",pub,modu);
        findenvelope(keyip);
        printxy(12,23,"ENVELOPE WAS SUCCESSFULLY CREATED,PRESS ANY KEY TO
QUIT",10+BLINK);
        return;
        }
```

```
void choice2(void)
      {
      long double pri,modu;
      printxy(5,11,"RECEIVER SIDE ENVELOPE EXTRACTION",2);
      printxy(30,17,"PRIVATE KEY : ",12);
      textcolor(15);
      cscanf("%Lf",&pri);
      printxy(30,19," MODULUS     : ",14);
      textcolor(15);
      cscanf("%Lf",&modu);
      getch();
      rsadmain("keyip1.txt","key.txt",pri,modu);
      extractenvelope();
      printxy(12,23,"ENVELOPE WAS SUCCESSFULLY EXTRACTED,PRESS ANY KEY
      TO QUIT",10+BLINK);
      return;
      }
void quit(void)
      {
      printxy(30,17,"Inavalid Choice",12);
      printxy(28,23,"PRESS ANY KEY TO QUIT",10+BLINK);
      return;
      }

void combine(char *ipfile,char *sign,char *fin)
{
FILE *fp,*fp1,*fp2;
char c1;

fp=fopen(fin,"w+");
fp1=fopen(ipfile,"r");
fp2=fopen(sign,"r");

      while((c1=fgetc(fp1))!=EOF)
      {
      fputc(c1,fp);
      }
      fputc(1,fp);
      fclose(fp1);
      while((c1=fgetc(fp2))!=EOF)
      {
      fputc(c1,fp);
      }
      fclose(fp);
      fclose(fp2);
return;
}

void extract(char *file3,char *file1,char *file2)
{
FILE *fp,*fp1,*fp2;
int i;
char c1;

fp=fopen(file3,"r");
```

```
  fp1=fopen(file1,"w");
  fp2=fopen(file2,"w");
        while((i=fgetc(fp))!=1)
        {
        fputc(i,fp1);
        }
        fclose(fp1);
        while((c1=fgetc(fp))!=EOF)
        {
        fputc(c1,fp2);
        }
        fclose(fp2);
        fclose(fp);
 return;
 }

 int compare(char *file1,char *file2)
 {
 FILE *fp,*fp1;
 char c1,c2;
 int k=1;

 fp=fopen(file1,"r");
 fp1=fopen(file2,"r");

       while((c1=fgetc(fp1))!=EOF)
       {
       c2=fgetc(fp);
       if(c1!=c2) k=0;
       }
       fclose(fp1);
       fclose(fp);
return k;

}
void digital(void)
{
      char ch;
      char *ipfile,*opfile,*sign;
      char *fin,*fflag="No";
      int flag=0;
      fin="final.txt";
      clrscr();
      env(1);

      printxy(23, 5,"DIGITAL SIGNATURE VERIFICATION",15);
      printxy(13, 8,"MESSAGE.. FILE NAME :> ",10);
      textcolor(13);
      gotoxy(38,8);
      cscanf("%s",ipfile);
      getch();
      opfile="digest.txt";
      sign="sign.txt";
      printxy( 5,16,"ENTER YOUR CHOICE 1-Sender 2-Recepient :> ",11);
      textcolor(14+BLINK);
      ch=getche();
      delay(3000);
```

```
    if(ch=='1')
          {
          finddigest(ipfile,opfile);
          printxy(13,18,"RSA Private Key :> ",12);
          printxy(13,20,"Modulus...       :> ",12);
          textcolor(10);
          gotoxy(34,18);
          cscanf("%Lf",&e);
          gotoxy(34,20);
          cscanf("%Lf",&n);
          rsaemain(opfile,sign,e,n);
          combine(ipfile,sign,fin);
          }
          else if(ch=='2')
          {
          extract("final.txt","newm.txt","news.txt");
          finddigest("newm.txt","digest2.txt");
          printxy(13,18,"RSA Public  Key :> ",12);
          printxy(13,20,"Modulus...       :> ",12);
          textcolor(10);
          gotoxy(34,18);
          cscanf("%Lf",&d);
          gotoxy(34,20);
          cscanf("%Lf",&n);
          rsadmain("news.txt","digest1.txt",d,n);
          flag=compare("digest1.txt","digest2.txt");
          if(flag==1)
          fflag="Yes";
          gotoxy(13,24);
          textcolor(12);
          cprintf(fflag);
          }
          else
          {
          printxy(25,22,"------<-Invalid Choice->-------",6+BLINK);
          }

          delay(1000);
          getch();
          clrscr();
          return;
}
void rsamain(void)
{
    int i, j;
    char choice;
    int button,tx,ty;
    int idno=143;
    int key;
    FILE *stream,*temp;
    char ch;
    static char a[10][10],b[10][10];
    int accno,acc,no,count=0;
    char *name,*newname,*pword;
    float amount,amt=10.0;
```

```
        struct  time t;
        int h,m,s;
        float hh,mm;
        unsigned int size;
        aa:
        clrscr();
        init_mouse();
        show_mouse();
        screen();
        delay(2000);
        while(1)
           {
           getmousepos(&button,&tx,&ty);
           if(tx>=72 && tx<=184 && ty>=96 && ty<=112 && button==1)
           {
           encrypt();
           goto aa;
           }
           if(tx>=264 && tx<=376 && ty>=96 && ty<=112 && button==1)
           {
           decrypt();
           goto aa;
           }
           if(tx>=472 && tx<=584 && ty>=96 && ty<=112 && button==1)
           {
           findkey();
           goto aa;
           }
           if(tx>=272 && tx<=368 && ty>=144 && ty<=160 && button==1)
           {
           clrscr();
           exit(0);
           }
           }
           return;
}
int findkey(void)
{
         int button,tx,ty;
         rsakey();
         square(7);
         textcolor(7);
         textbackground(1);
         gotoxy(28,6);
         cscanf("%Lf",&max);
         rsak(max);
         textcolor(WHITE);
         textbackground(1);
         gotoxy(45,12);
         cprintf("%.2Lf",d);
         gotoxy(45,14);
         cprintf("%.2Lf",n);
         gotoxy(45,16);
         cprintf("%.2Lf",e);
         shadow(58,24,10,1);
         textbar(60,23,10,1,11);
         textcolor(11+BLINK);
        gotoxy(62.0,23);
        cprintf("O.K...");
        x1:
        getmousepos(&button,&tx,&ty);
        if(tx>=472 && tx<=544 && ty==176 && button==1)
        return 0;
        goto x1;
        //getch();
 }
 int encrypt(void)
 {
        char *ipf,*opf;
        int button,tx,ty;
        clrscr();
        env(1);
        shadow(20,4,38,3);
        textbar(18,3,38,3,11);
        textcolor(WHITE);
        textbackground(11);
        gotoxy(22,4);
        cprintf("RSA ENCRYPTION-PLAIN TO CIPHER");
        square(8);
        square(24);
        textcolor(WHITE);
        textbackground(1);
        gotoxy(13,13);
        cprintf("ENTER THE PUBLIC   KEY :");
        gotoxy(13,15);
        cprintf("ENTER THE MODULUS VALUE:");
        gotoxy(13,17);
        cprintf("ENTER THE  SOURCE  FILE:");
        gotoxy(13,19);
        cprintf("ENTER THE  OUTPUT  FILE:");
        textcolor(9);
        gotoxy(38,13);
        cscanf("%Lf",&e);
        gotoxy(38,15);
        cscanf("%Lf",&n);
        gotoxy(38,17);
        cscanf("%s",ipf);
        gotoxy(38,19);
        textcolor(14);
        cscanf("%s",opf);
        rsaemain(ipf,opf,e,n);
        shadow(58,24,10,1);
        textbar(60,23,10,1,11);
        textcolor(11+BLINK);
        gotoxy(62.0,23);
        cprintf("O.K...");
        x2:
        getmousepos(&button,&tx,&ty);
        if(tx>=472 && tx<=544 && ty==176 && button==1)
        return 0;
        goto x2;
        //getch();
}
```

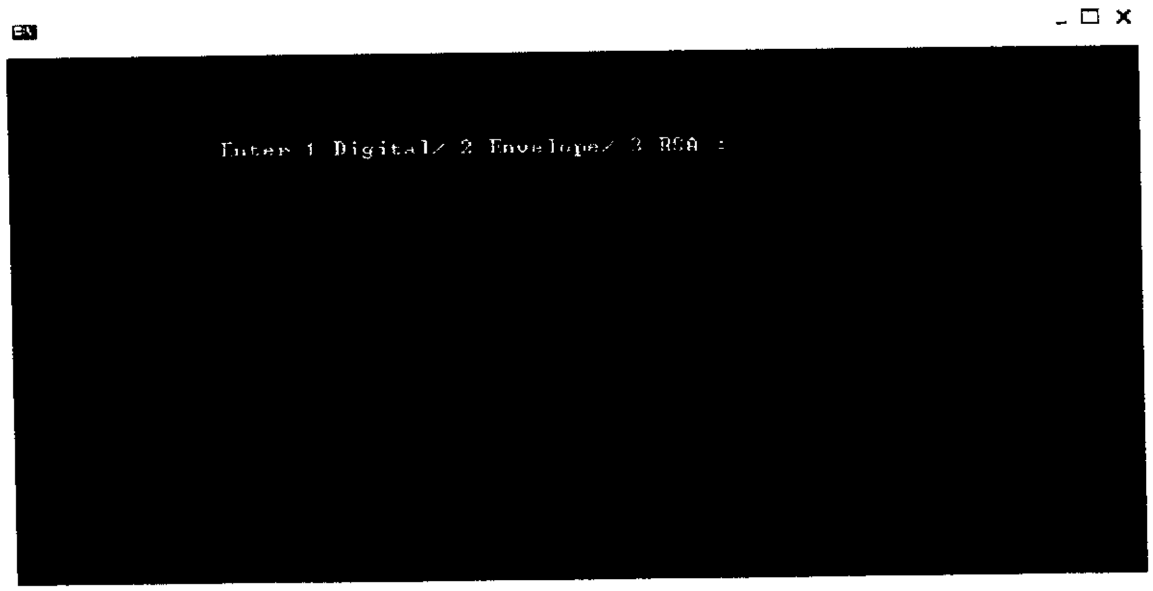
```
    int decrypt(void)
    {
           char *ipf,*opf;
           int button,tx,ty;
           clrscr();
           env(1);
           shadow(20,4,42,3);
           textbar(18,3,42,3,11);
           textcolor(WHITE);
           textbackground(11);
           gotoxy(22,4);
           cprintf("RSA ENCRYPTION-CIPHER TO PLAIN TEXT");
           square(8);
           square(24);
           textcolor(WHITE);
           textbackground(1);
           gotoxy(13,13);
           cprintf("ENTER THE PRIVATE   KEY :");
           gotoxy(13,15);
           cprintf("ENTER THE MODULUS VALUE:");
           gotoxy(13,17);
           cprintf("ENTER THE  SOURCE  FILE:");
           gotoxy(13,19);
           cprintf("ENTER THE  OUTPUT  FILE:");
           textcolor(9);
           gotoxy(38,13);
           cscanf("%Lf",&d);
           gotoxy(38,15);
           cscanf("%Lf",&n);
           gotoxy(38,17);
           cscanf("%s",ipf);
           gotoxy(38,19);
           textcolor(14);
           cscanf("%s",opf);
           rsadmain(ipf,opf,d,n);
           shadow(58,24,10,1);
           textbar(60,23,10,1,11);
           textcolor(11+BLINK);
           gotoxy(62.0,23);
           cprintf("O.K...");
           x3:
           getmousepos(&button,&tx,&ty);
           if(tx>=472 && tx<=544 && ty==176 && button==1)
           return 0;
           goto x3;
           //getch();
 }

 void rsakey(void)
 {

    env(1);
    textcolor(WHITE);
    textbackground(1);
    gotoxy(3,6);
    cprintf("ENTER THE MAXIMUM RANGE");
    textcolor(11);
    gotoxy(25,12);
    cprintf("PRIVATE KEY");

    gotoxy(25,14);
    cprintf("MODULUS");

    gotoxy(25,16);
    cprintf("PUBLIC KEY");

return;
}

void shadow(int p,int q,int r,int s)
{
    int i,j;
    textcolor(0);
    textbackground(0);
    for(i=q;i<q+s;i++)
    {
    gotoxy(p,i);
    for(j=0;j<r;j++)
    {
    cprintf(" ");
    }
    }
return;
}
void textbar(int p,int q,int r,int s,int c)
{
    int i,j;
    //textcolor(d);
    textbackground(c);
    for(i=q;i<q+s;i++)
    {
    gotoxy(p,i);
    for(j=0;j<r;j++)
    {
    cprintf(" ");
    }
    }
return;
}
void square(int v)
{
    int i,j;
    int d=196;
    textcolor(WHITE);
    textbackground(1);
    for(i=2;i<80;i++)
    {
    gotoxy(i,v);
    cprintf("%c",d);
    }
return;
}
```

```
void screen(void)
{
   highvideo();
   env(1);
   shadow(8,3,62,3);
   textbar(10,2,62,3,11);
   textcolor(WHITE+BLINK);
   gotoxy(29,3);
   cprintf("IMPLEMENTATION OF S\\MIME");

   shadow(25,8,28,3);
   textbar(27,7,28,3,11);
   textcolor(11);//+BLINK);
   gotoxy(34,8);
   cprintf("RSA ENCRYPTION");

   shadow(8,14,15,3);
   textbar(10,13,15,3,11);

   textcolor(11);
   gotoxy(12,14);
   cprintf("ENCRYPT...");
   shadow(32,14,15,3);
   textbar(34,13,15,3,11);
   textcolor(11);
   gotoxy(36,14);
   cprintf("DECRYPT...");
   shadow(58,14,15,3);
   textbar(60,13,15,3,11);
   textcolor(11);
   gotoxy(62,14);
   cprintf("NEW KEY...");
   shadow(33,18,13,3);
   textbar(35,19,13,3,11);
   textcolor(11);
   gotoxy(38,20);
   cprintf("EXIT...");
return;
}

void init_mouse()
{
      char status;
      _AX=0;
      geninterrupt(0X33);
      status=_AX;
      if (status!=-1)
      return;
}

void show_mouse()
{
      _AX=1;
      geninterrupt(0X33);
      return;
}
```

```
void hide_mouse()
{
      _AX=2;
      geninterrupt(0X33);
      return;
}

void move_mouse(int x,int y)
{
      _AX=4;
      _CX=x;
      _DX=y;
      geninterrupt(0X33);

      return;
}

     void getmousepos(int *button,int *x,int *y)
     {
     ii.x.ax=3;
     int86(0x33,&ii,&o);
     *button=o.x.bx;
     *x=o.x.cx;
     *y=o.x.dx;
     return;
     }
```

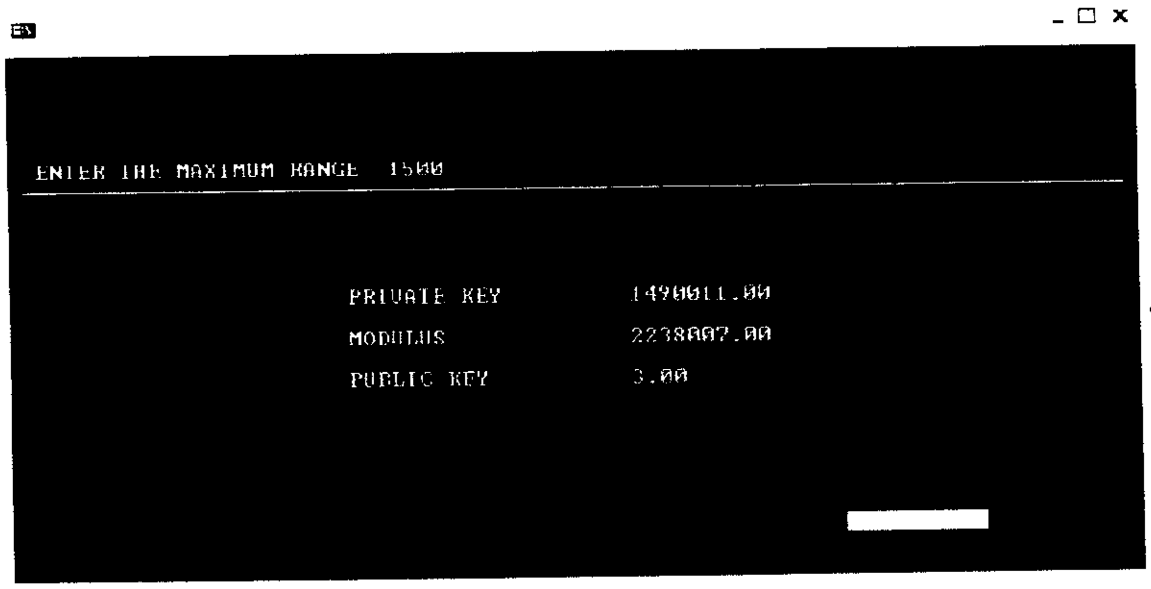
9.2 OUTPUT SCREENS

The Main Screen is shown below



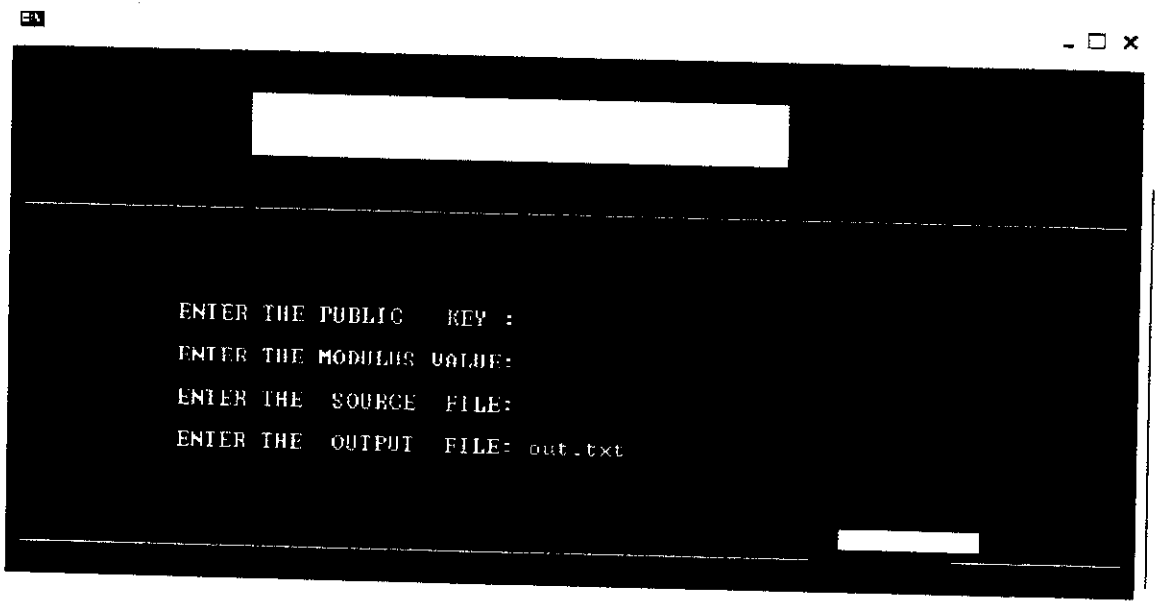
```
Enter 1 Digital/ 2 Envelope/ 3 RSA :
```

The below Screen shows the Key Generation Process



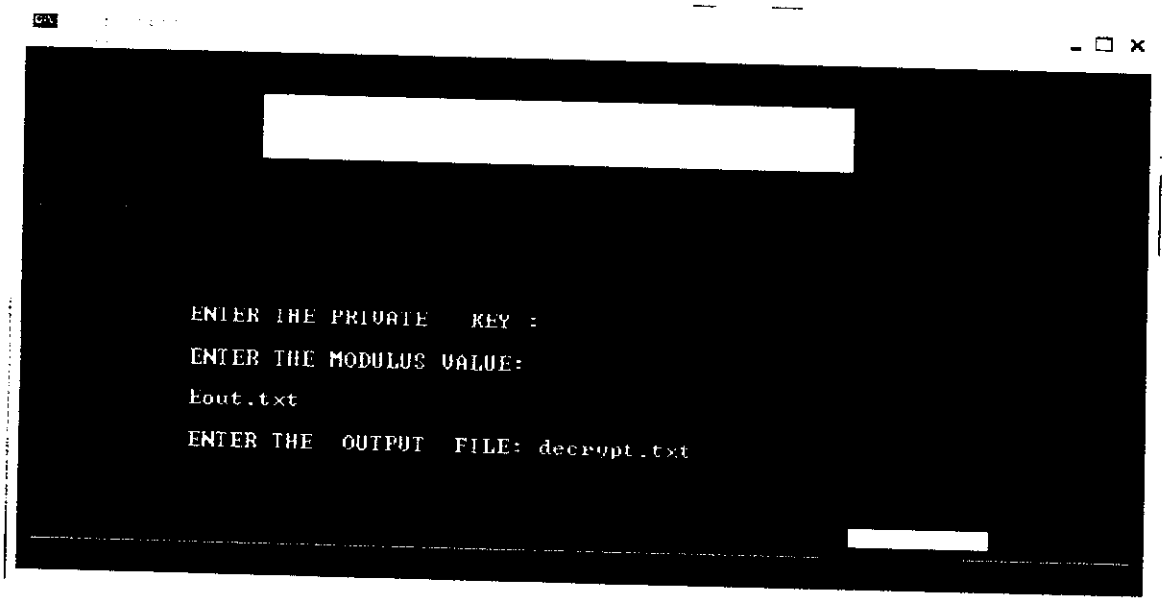
```
ENTER THE MAXIMUM RANGE  1500

                     PRIVATE KEY            1490011.00
                     MODULUS                2238007.00
                     PUBLIC KEY             3.00
```

The below Screen shows the RSA Encryption Process



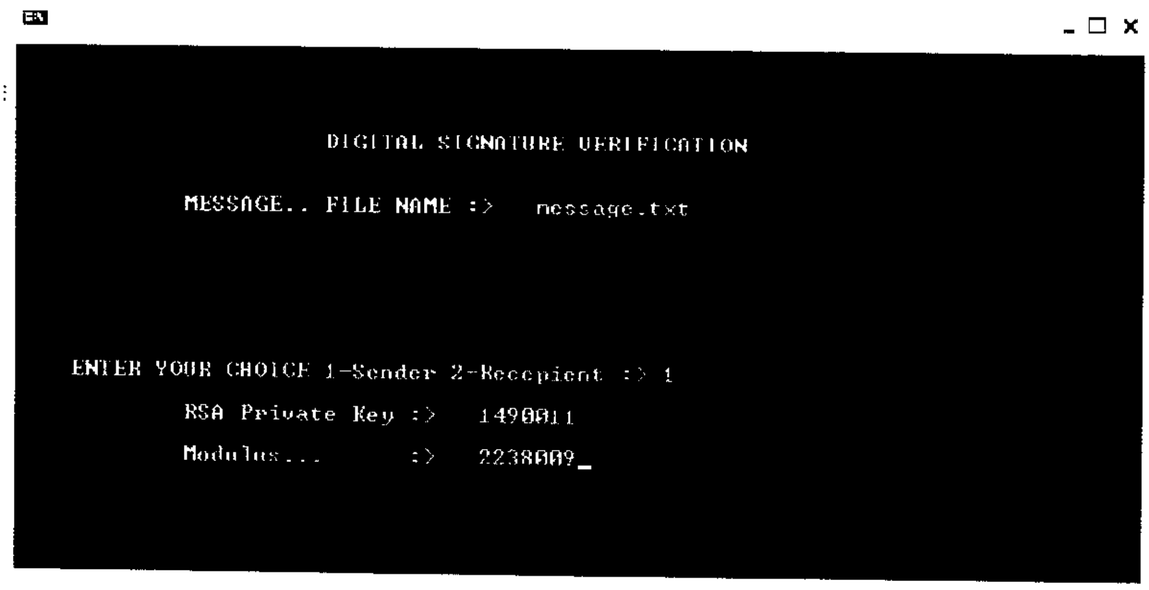
```
ENTER THE PUBLIC   KEY :
ENTER THE MODULUS VALUE:
ENTER THE  SOURCE  FILE:
ENTER THE  OUTPUT  FILE: out.txt
```

The below Screen shows the RSA Decryption Process



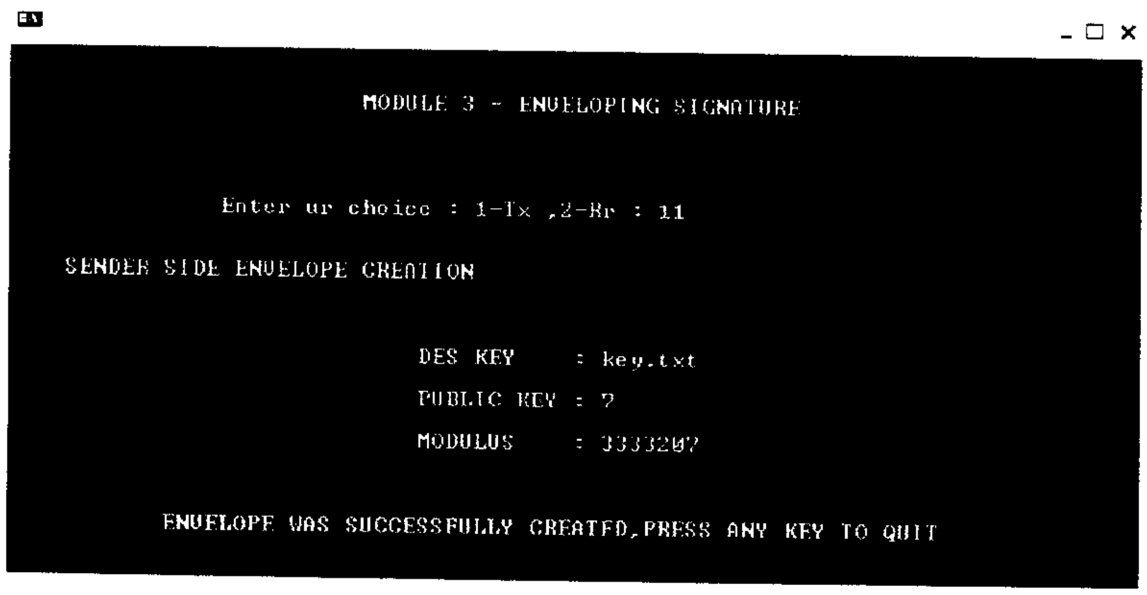
```
ENTER THE PRIVATE   KEY :
ENTER THE MODULUS VALUE:
Eout.txt
ENTER THE  OUTPUT  FILE: decrypt.txt
```

The below Screen Shows the Digital Signature Verification



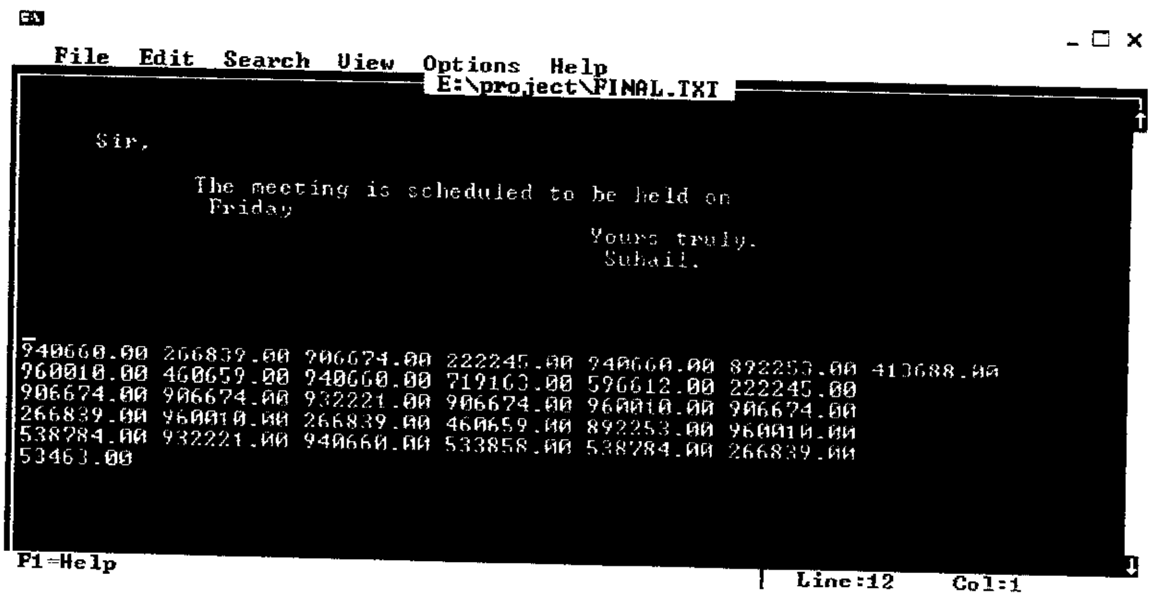
```
                    DIGITAL SIGNATURE VERIFICATION

        MESSAGE.. FILE NAME :>   message.txt



ENTER YOUR CHOICE 1-Sender 2-Reecpient :> 1
        RSA Private Key :>    1490011
        Modulus...      :>    2238009_
```

The below Screen shows Enveloping using DES



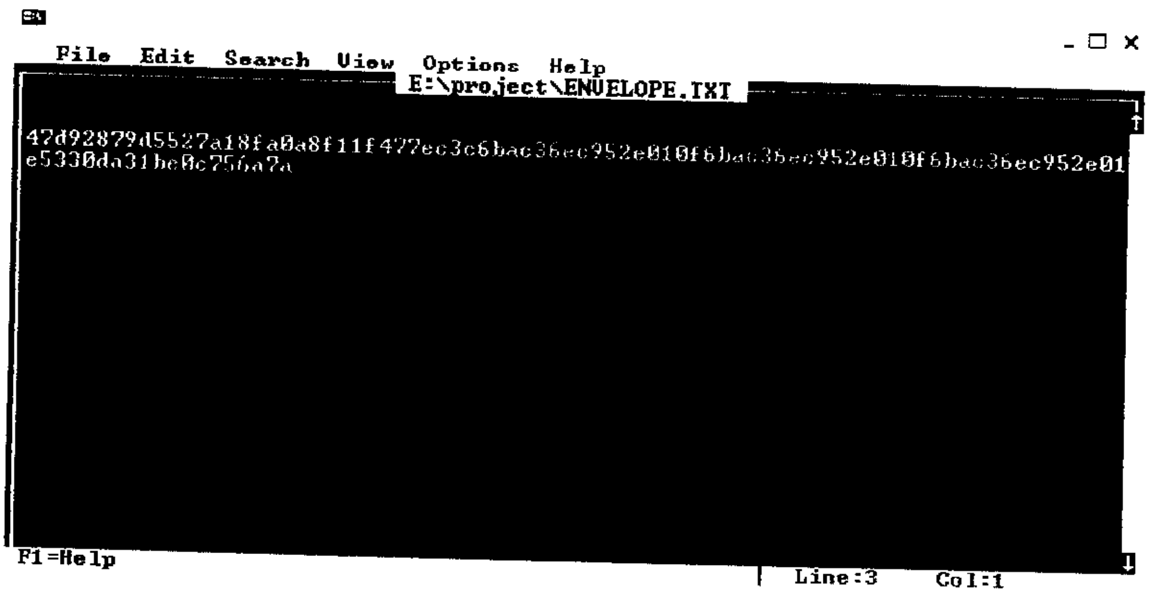
```
                    MODULE 3 - ENVELOPING SIGNATURE


          Enter ur choice : 1-Tx ,2-Rr : 11

 SENDER SIDE ENVELOPE CREATION

                        DES KEY     : key.txt
                        PUBLIC KEY  : 7
                        MODULUS     : 3333207

      ENVELOPE WAS SUCCESSFULLY CREATED,PRESS ANY KEY TO QUIT
```

The below Screen shows the Message along with the encrypted Digest



```
    Sir,

            The meeting is scheduled to be held on
             Friday
                                              Yours truly.
                                               Suhail.


940660.00 266839.00 906674.00 222245.00 940660.00 892253.00 413688.00
960010.00 460659.00 940660.00 719163.00 596612.00 222245.00
906674.00 906674.00 932221.00 906674.00 960010.00 906674.00
266839.00 960010.00 266839.00 460659.00 892253.00 960010.00
538784.00 932221.00 940660.00 533858.00 538784.00 266839.00
53463.00
```

The below Screen shows the Enveloped Document



```
47d92879d5527a18fa0a8f11f477ec3c6bac36ec952e010f6bac36ec952e010f6bac36ec952e01
e5330da31be0c756a7a
```