SECURE FILE TRANSACTION USING RSA CRYPTOSYSTEM AND MESSAGE AUTHENTICATION USING MD5

A PROJECT REPORT

Submitted by

AMIT MATHEW, 17BCI0079

ES SUJEETH, 17BCI0133

RAJESH KARUMANCHI, 17BCI0147

Course Code: CSE2008

Course Title: NETWORK SECURITY

Under the guidance of

Dr.ANIL KUMAR K

Associate professor, SCOPE,

VIT University, Vellore.

INDEX

ABSTRACT:-

In this project we have introduced secure RSA for secure file transmission. There are many cases where we need secure file transmission for example in banking transactions, e-shopping etc. In this project we present modified RSA algorithm for secure file transmission.Cryptography is the process of converting plain text into encrypted text and decrypt cipher text to plain text at other end. RSA algorithm is asymmetric key cryptography also called Public Key cryptography. Two keys are generated in RSA, one key is used for encryption and other key which is only known to authenticated receiver can decrypt message. No other key can decrypt the message. Every communicating party needs just a key pair for communicating with any number of other communicating parties. Once someone obtains a key pair, he /she can communicate with anyone else. RSA is a well known public key cryptography algorithm and was one of the first great advances in public key cryptography. Even if it is efficient algorithm it is vulnerable to attackers. With the help of all brute force attacks hacker can obtain private key. Many improvements has been done to improve RSA like BATCH RSA, MultiPrime RSA, MultiPower RSA, Rebalanced RSA, RPrime RSA etc. As craze of internet is increasing exponentially, it is used for email, chatting, transferring data and files from one end to other. It needs to be a secure communication among the two parties. This project focuses on file transfer using Secure RSA, which eliminates some loopholes of RSA that might prevent a hacker from stealing and misuse of data. This project also presents comparison between RSA file transfer and Secure RSA file transfer.

Here we use MD5 hashing algorithm for authentication purpose.

1.INTRODUCTION:-

In the current time, when the Internet provides essential communication between millions of people and is being increasingly used as a tool for ecommerce, security becomes a tremendously important issue to deal with. Internet is often used to upload web pages and other documents from a private development machine to public webhosting servers. Transfer of files like banking transactions e-shopping, tenders etc need special authenticated mechanism. As a communications and transmission of files over internet has increased exponentially since last few years, there is need of security in such file transfer. One of the solutions to secure file transmission by cryptographic ciphers. It is the process of converting plain text into encrypted text and decrypt cipher text to plain text at other end.

There are three ways in which we can achieve security

1.Encrypted file transfer.

2.Strong secure protocol for transmission of files.

3.Hasing programme for Athuentication.

1.1RSA (Rivest–Shamir–Adleman):

R.S.A is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and it is different from the decryption key which is kept secret (private).

A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret.

RSA algorithm consist of three phases, phase one is key generation which is to be used as key to encrypt and decrypt data, second phase is encryption, where actual process of conversion of plaintext to cipher text is being carried out and third phase is decryption, where encrypted text is converted in to plain text at other side.

It provides a method of assuring the confidentiality, integrity, authenticity and non-reputability of electronic communications and data storage.Many protocols like SSH, OpenPGP, S/MIME, and SSL/TLS rely on RSA for encryption and digital signature functions. It is also used in software programs -- browsers are an obvious example, which need to establish a secure connection over an insecure network like the Internet or validate a

digital signature. RSA signature verification is one of the most commonly performed operations in IT.

RSA is widely used in electronic ecommerce protocols. With sufficiently long keys and the use of up-to-date implementations; RSA is believed to be totally secure.

1.2 SECURE SOCKET LAYER(SSL):-

Secure Socket Layer (SSL) is an effective method of protecting data which is sent over a local or wide area network. It works by encrypting data sent over a network, It can be configured on both wired and wireless networks and will work with other forms of security such as WPA keys and firewalls.

The primary goal of the SSL protocol is to provide privacy and reliability between two communicating applications. The protocol is composed of two layers. At the lowest level, layered on top of some reliable transport protocol (e.g., TCP [RFC0593]), is the SSL record protocol. The SSL record protocol is used for encapsulation of various higherlevel protocols. One such encapsulated protocol, the SSL handshake protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. One advantage of SSL is that it is application protocol independent. A higherlevel protocol can layer on top of the SSL protocol transparently.

1.3 MD5 HASHING:-

The **MD5(message-digest)** is a widely used hash function producing a 128-bit hash value. The method used to do authentication is very simple. Each user has a file containing a set of keys that are used to salt an md5 hash. The information being transferred has its md5 checksum calculated using this salt, and is then transferred to the destination, along with the md5 hash result. At the destination the server will get the user id, obtain the salt value from a key file, and then calculate the md5 hash value. If the two are in agreement, authentication is successful.

2.BACKGROUND OF WORK:-

2.1Command prompt:-

Command Prompt interacts with the user through a command-line interface. Command Prompt has an advantage of features available to native programs of its own platform.

We actually store all the files in one folder in order to use the command prompt.

In this project we call many other files in one code while execution ,where command prompt has its own advantages.

3.Overview:-

3.1 RSA:-

The original Algorithm of RSA is as follows:-

original algorithm.

Key Generation

1. Generate two large random primes, p and q, of approximately equal size such that their product n=pq is of the required bit length, e.g. 1024 bits.
2. Compute n= pq and $\phi=(p-1)(q-1)\phi=(p-1)(q-1)$.
3. Choose an integer e, $1<e<\phi$, such that $\gcd(e,\phi)=1$.
4. Compute the secret exponent d, $1<d<\phi$, such that $ed\equiv1 \bmod \phi$.
5. The public key is (n,e) and the private key (d,p,q). Keep all the values d, p, q and ϕ secret. [Sometimes the private key is written as (n,d) because you need the value of n when using d. Other times we might write the key pair as ((N,e),d)]

   Where

- n is known as the modulus.
- e is known as the public exponent or encryption exponent or just the exponent.
- d is known as the secret exponent or decryption exponent.

Encryption:

Sender A does the following:-

Obtains the recipient B's public key (n,e).

Represents the plaintext message as a positive integer m with $1<m<n$ [see note 4].

Computes the ciphertext $c=m^e \bmod n$.

Sends the ciphertext c to B.

Decryption:-

Recipient B does the following:-

Uses his private key (n,d) to compute $m=c^d \bmod n$.

Extracts the plaintext from the message representative m.

3.2 MD5 Authentication:-

Step 1. Append Padding Bits

The message is "padded" (extended) so that its length (in bits) iscongruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long.

Step 2. Append Length:

A 64-bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2^64, then only the low-order 64 bits of b are used.

Step 3. Initialize MD Buffer

A four-word buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes

word A: 01 23 45 65

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 56 54 32 10

Step 4. Process Message in 16-Word Blocks

We first define four auxiliary functions that each take as inputthree 32-bit words and produce as output one 32-bit word.

F(X,Y,Z) = XY v not(X) Z

G(X,Y,Z) = XZ v Y not(Z)

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X v not(Z))

In each bit position F acts as a conditional: if X then Y else Z. The function F could have been defined using + instead of v since XY and not(X)Z will never have 1's in the same bit position.) It is interesting to note that if the bits of X, Y, and Z are independent and unbiased, the each bit of F(X,Y,Z) will be independent andunbiased.

The functions G, H, and I are similar to the function F, in that they act in "bitwise parallel" to produce their output from the bits of X, Y, and Z, in such a manner that if the corresponding bits of X, Y, and Z are independent and unbiased, then each bit of G(X,Y,Z), H(X,Y,Z), and I(X,Y,Z) will be independent and unbiased.

Step 5. Output

The message digest produced as output is A, B, C, D. That is, we begin with the low-order byte of A, and end with the high-order byte of D.

3.3 SOCKET PROGRAMMING

.pseudocode server

• Create socket

• Bind socket to a specific port where clients can contact you

• Loop (Receive UDP Message from client x)+ (Send UDP Reply to client x)* • Close Socket

Pseudo code client

• Create socket

• Loop (Send Message To Well-known port of server)+ (Receive Message From Server)

. Close Socket

4.WHY WE ARE USING MD5

*__MD5__ is faster than __SHA__ i.e it takes __lesser time__ to generate a __MD5__ hash for a given text comapared to its equivalent __SHA__ Hash.

*For long strings __SHA__ takes long time for Hash generation.

*Therefore we are using __MD5__ even though __SHA__ is more secure.

5. IMPLEMENTATION

5.1 ALGORITHM

1)START

2)party A is the server and is waiting for the client party B to request for connection.

3)party B establishes connestion with party A.

4)party A sends message encrypted using RSA cryptosystem to party B along with the MD5 Hash of the message.

5)party B receives the message,decrypts it using RSA and generates MD5 hash for the message.

6)It then compares the two message digest,and if both come out to be same then the message is accepted as authentic.

7)END

5.2 SOURCE CODE

partyA.py

```
import socket

import hashlib

import RSA

HOST = socket.gethostbyname(socket.gethostname())

PORT = 1234

print("IP of PartyA {}".format(HOST))

file = open('SampleText.txt', 'r')

datar = file.readlines()

datar = ''.join(datar)

H=hashlib.md5(datar.encode())
```

```
J=H.digest()
file1= open('HashA.txt', 'wb')
file1.write(J)
file1.close()
p = 17
q = 19
n = p*q
private_key, public_key = RSA.GenerateKeys(p, q)
encdata1 = []
for i in datar:
 encdata1.append(pow(ord(i), public_key, n))
encdata1 = ' '.join([str(i) for i in encdata1])
encdata = encdata1.encode()
datar = encdata
file= open('encdata.txt','w')
file.write(encdata1)
file.close()
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
 s.bind((HOST,PORT))
 s.listen(5)
 conn, addr = s.accept()
 with conn:
 print("Connected by Party: {}".format(addr))
 while True:
```

```
 conn.sendall(str(private_key).encode())
 conn.sendall(datar)
 break
```

partyB.py

```
import socket
import hashlib
import RSA
HOST = socket.gethostbyname(socket.gethostname())
PORT = 1234
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
 s.connect((HOST,PORT))
 key = s.recv(20)
 data = s.recv(1024)
data = data.decode()
data = data.split()
data = [int(i) for i in data]
decdata = []
p = 17
q = 19
n = p*q
private_key, public_key = RSA.GenerateKeys(p, q)
for i in data:
 decdata.append(pow(i, int(key.decode()), n))
data = ''.join([chr(i) for i in decdata])
```

```
H=hashlib.md5(data.encode())

J=H.digest()

file1= open('HashB.txt', 'wb')

file1.write(J)

file1.close()

file = open('recdata.txt', 'w')

print('\n Data received at revdata.txt')

file.write(data)

file.close()

count=0

with open('HashA.txt') as f1, open('HashB.txt') as f2:

 for l1, l2 in zip(f1, f2):

 if l1!=l2:

 count=1

if(count==0):

 print("\n ~MESSAGE RECIEVED IS AUTHENTIC AND HAS NOT

BEEN MODIFIED~")
```

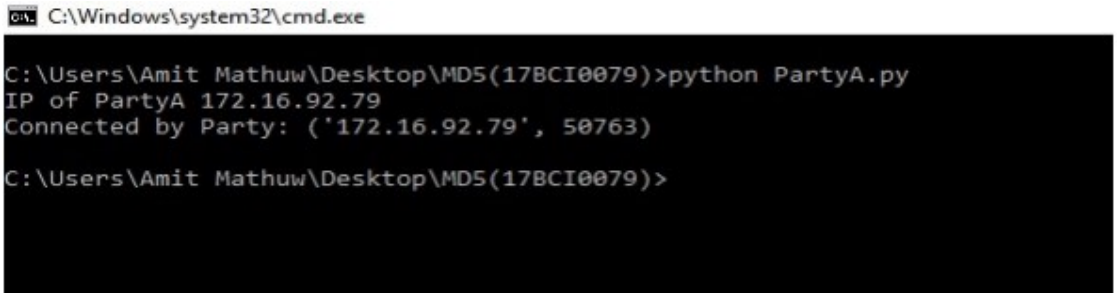
RSA.py

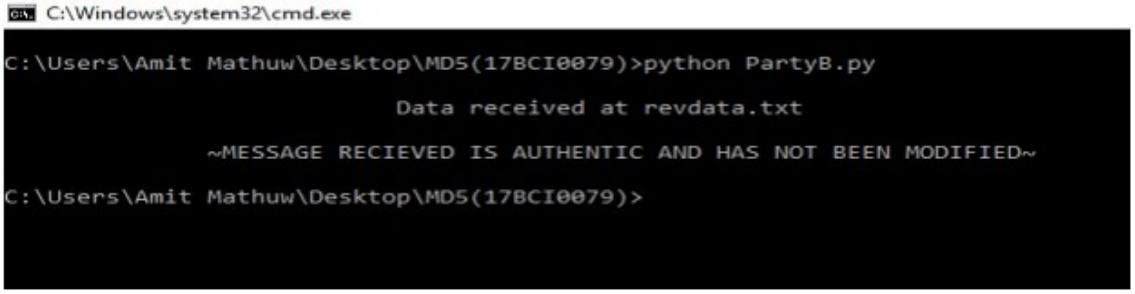
```
from random import *

def gcd(a, b):

 r1 = a

 r2 = b

 while r2 > 0:
```

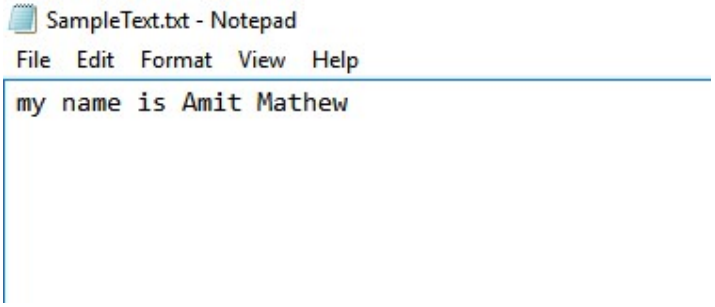
```
 q = r1 // r2
 r = r1 - q*r2
 r1 = r2
 r2 = r
 return r1
def GenerateKeys(p, q):
 n = p*q
 Qn = (p-1) * (q-1)
 e = 0
 d = 0
 Q = []
 for i in range(2, Qn):
 if gcd(Qn, i) == 1:
 if e == 0:
 e = i
 Q.append(i)
 for i in Q:
 if(i*e) % Qn == 1:
 d = i
 break
 return d, e
def miller_rabin(n, k=10):
 if n == 2:
 return True
```

```
if not n & 1:

return False

def check(a, s, d, n):

x = pow(a, d, n)

if x == 1:

return True

for i in range(s - 1):

if x == n - 1:

return True

x = pow(x, 2, n)

return x == n - 1

s = 0

d = n - 1

while d % 2 == 0:

d >>= 1

s += 1

for i in range(k):

a = randrange(2, n - 1)

if not check(a, s, d, n):

return False

return True
```

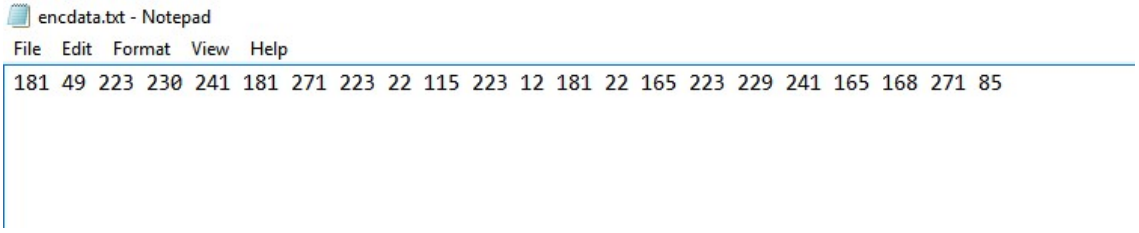
5.3 EXECUTION SNAPSHOTS



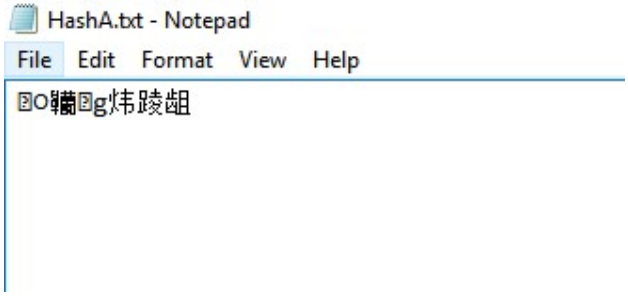




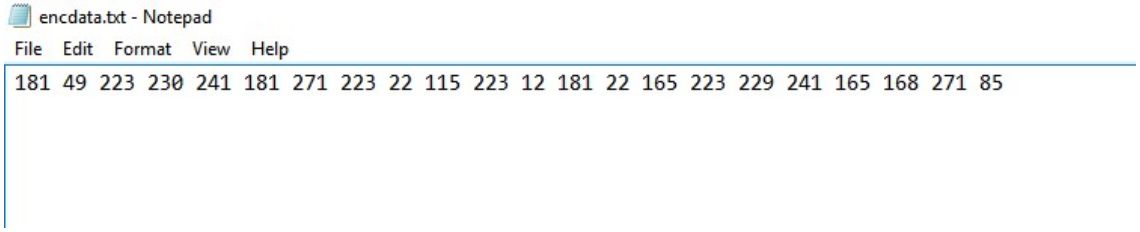
Message to be sent by party A



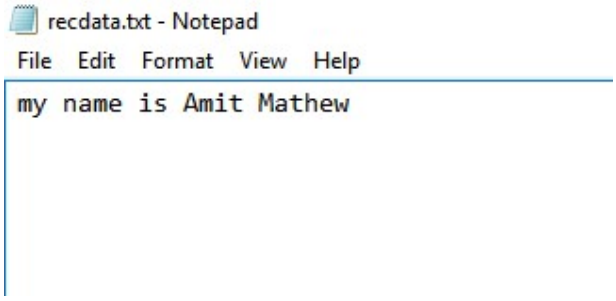
RSA Encrypted Message sent by party A



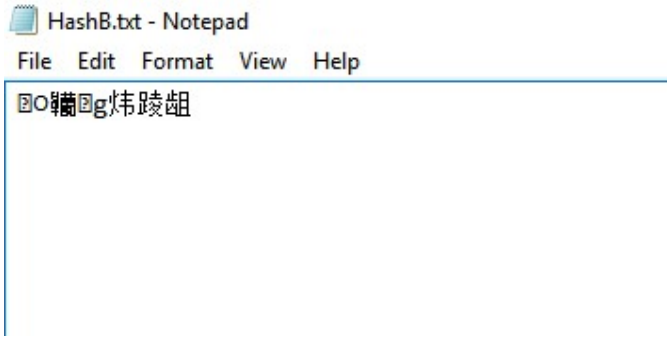
Hash of message generated by party A



RSA Encrypted Message received by party B



Message decrypted by party B



Hash of message generated by party B

6.CONCLUSION

In this project we are making use of RSA,MD5, socket programming for secure transmission of the message. We are using socket programming for providing communication between client and the server.MD5 hash function is used for authentication which means it makes sure that the received message is not modified or changed.Here sender(party A) first generates the MD5 hash of the message and sends it along with the message encrypted using RSA to party B.The receiver(party B) receives the message and hash and he first decrypts the message and also generates its MD5 hash and compares it with the received hash. This is how authentication is done .The file transmission through RSA is implemented with python coding. We created a sample.txt file which contains the data to be transferred .The RSA.py carries the coding part for encryption of RSA cipher. The PartyA.py carries the code to read the text from sample.txt and calls the RSA file to encrypt the data and creates a file encry.py which contains the encryption part, hash.txt file which contains the hash generated by MD5 for the encrypted data.Whereas the PartyB contains the code to read ,authenticate the encrypted data and then decryption if the authentication is successful.

7.REFERENCES

*Bao, F., Samarati, P. and Zhou, J. (2012). *Applied Cryptography and Network Security*. Berlin, Heidelberg: Springer Berlin Heidelberg.

*Oriyano, S. (2013). *Cryptography*. New York: McGraw-Hill Education.

*Stallings, W. (2017). *Cryptography & Network Security GE*. Pearson Australia Pty Ltd.

*It.slashdot.org. (2018). *RSA Released Into The Public Domain - Slashdot*. [online] Available at: https://it.slashdot.org/story/00/09/06/1252204/rsa-released-into-the-public-domain [Accessed 31 Oct. 2018].

*Rachmawati, D., Tarigan, J. and Ginting, A. (2018). A comparative study of Message Digest 5(MD5) and SHA256 algorithm. *Journal of Physics: Conference Series*, 978, p.012116.

* Eddie Law, K. and Leung, R. (2003). A design and implementation of active network socket programming. Microprocessors and Microsystems, 27(5-6), pp.277-284.

*Cole, E., Krutz, R., Conley, J. and Cole (2009). Network Security. Hoboken: Wiley [Imprint].

*Eddie Law, K. and Leung, R. (2003). A design and implementation of active network socket programming. Microprocessors and Microsystems, 27(5-6), pp.277-284.

*Gay, W. (2000). Linux socket programming by example. Indianapolis, Ind.: Que.

*LI, Y., LIU, Q., HAO, L. and ZHOU, B. (2010). Efficient variant of RSA cryptosystem. Journal of Computer Applications, 30(9), pp.2393-2397.

*Pandey, V. and K Mishra, V. (2013). Architecture based on MD5 and MD5-512 Bit Applications. International Journal of Computer Applications, 74(9), pp.29-33.