MINOR-1 PROJECT

END TERM EVAUATION REPORT for

SECURE FILE STORAGE IN CLOUD ENVIRONMENT USING CRYPTOGRAPHY TECHNIQUES

Submitted by:

Name	Roll No	Branch
Ekanshu Dargan	R110216063	CSE CCVT
Babanjot Singh	R110216052	CSE CCVT
Manik Khurana	R110216092	CSE CCVT

Under the guidance of:

Ms. Avita Katal

Assistant Professor

Department of Virtualization,

School of Computer Science,

University Of Petroleum and Energy Studies.

Dehradun- 248007

Approved By

(Ms. Avita Katal) (Dr.Amit Agarwal)

Project Guide **Department Head**

School of Computer Science

University of Petroleum & Energy Studies, Dehradun

Synopsis Report (2018-19)

1 Project Title

SECURE FILE STORAGE IN CLOUD ENVIRONMENT USING CRYPTOGRAPHY TECHNIQUE.

2 Abstract

In the cloud condition, assets are shared among the majority of the servers, clients and people. So it is troublesome for the cloud supplier to guarantee record security. Thus it is simple for a gatecrasher to access, abuse and demolish the first type of information thus necessitating the use of cryptography to safeguard communication. In this project, the presentation is a comprehensive review of the cryptography algorithms. This project presents a design of data encryption and decryption in a network environment using RSA algorithm and AES algorithms with a specific message. This project provides a description of their encryption and decryption operations, points out their security on the basis of keys, areas of implementation, their strengths and weaknesses during operation. This project also presents an overview in developing a client-server based network between two systems using socket programming. The client server approach is used to display how files can be sent securely between the cloud provider and the client.

Keywords: Cryptography, RSA algorithm, AES algorithm, Socket programming

3 Introduction

Cryptography's aim is to construct schemes or protocols that can still accomplish certain tasks even in the presence of an adversary. A basic task in cryptography is to enable users to communicate securely over an insecure channel in a way that guarantees their transmissions' privacy and auth Assume, for example, that Alice wants to send a message to Bob over the Internet. Ideally, no attacker like Oscar should be able to obtain information about her message or modify it without Alice's notice. Providing privacy and authenticity remains a central goal for cryptographic protocols, but the field has expanded to encompass many others, including e-voting, digital coins, and secure auctions. [1]

- **Socket programming** is a way of connecting two nodes on a network to communicate with each other. One socket (node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.[a] [2]

- **RSA (Rivest - Shamir – Adleman)** is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and Private Key is kept private. The idea of RSA is based on the fact that it is difficult to factorize a large integer. First, we will generate a public key and then a private key. Once the key pair has been generated, the process of encryption and decryption are relatively straightforward and done mathematically.[3][4]

- **AES (Advanced Encryption Standard)**, on the other hand is a symmetric cryptography algorithm. AES is a replacement for DES as its key size was too small. Here, we have 128-bit data, 128/192/256-bit keys. AES is an iterative rather than Feistel cipher. AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix. The encryption process has different rounds that comprise of four sub-processes- **Byte Substitution, Shiftrows, MixColumns and Addroundkey.** And the decryption process consists of the same 4 processes in the reversed order.[5]

After the implementation of both algorithms separately, a comparison and review chart with pros and cons will be studied.

4 Literature Review

The socket programming concept has benefited many areas of computer networks today. The socket application developed in this work is based on the concept of distributed computing. This is relevant as to that it allows for resources to be distributed among several nodes in the network [2]. Socket programming helps to implement the bottom level of network communication, using Application Programming Interface (API). A similar application is built by other researchers as presented in [2], [6]. The idea of the RSA public key cryptosystem was from Diffie and Hellman, who introduced the method of the exponential key exchange. In their project, Ronald and Adi would develop ideas while

Leonard would attempt to bring the ideas down, by cracking them. Leonard was time and again successful at cracking them until one night when Ronald developed an algorithm that Leonard could not crack. The Algorithm was named RSA from their names Rivest, Shamir, and Adleman [3].

The core of RSA has withstood every attack from the best cryptographic minds. The robustness of the algorithm, the absence of rigorous proof notwithstanding, provides a sense of security. According to Dan Boney, a computer science professor at the Stanford University, "we kind to chip at the sides, but no one has figured out how to get at the heart of it" [4]. Security of RSA cryptosystem depends on the large prime numbers because it is difficult to break the large prime numbers. RSA algorithm provides the security and performance. Every element of the set is greater than all integer numbers. In this project, an RSA algorithm which is provided security against brute force attacks [3].

The National Institute of Standards and Technology (NIST) thus called for a proposal for a new advanced encryption standard. Selection of AES was an open process. In 2001 NIST declared the block cipher Rijndael as the new AES [8]. AES was developed in a way so that it could be implemented on both software are hardware. Hardware implementation of AES can be done using the reprogrammable device like FPGAs (Field programmable gate arrays) as they can provide better performance than software methods. The proposed design uses an efficient way of implementing the 128 bit key AES encryption by reusing the resources required for encrypting the data in each round. Thus, reducing the resources required for the encryption of data. This helps in reducing number of slices required in the FPGA and also helps in improving the performance. With a key length of 128 bits 10 rounds are required thus each round can be done in one clock cycle and thus a total of 10 clock cycles will be required to get the cipher text [8].

5 Problem Statement

The present scenario of cloud computing grossly requires the conveniences of network connectivity, thus have increased the risk of ever getting hacked. The problems of data falling into the wrong hands is hazardous and consequences can be devastating. So, to overcome this problem nowadays encryption becomes a crucial link in the security chain. It generally scrambles the plain text to make it unreadable by anyone other than those with the keys to decode it. Similar to hiding the cipher text from Oscar while Alice and Bob communicate safely.

6 Objective

The primary objective of this project is to compare the two Algorithms of Cryptography, RSA and AES. And to create simple, effective and secure network which could enable user to transfer data without the risk of any unauthorized access within the network taking care that the data is not compromised.

7 Methodology

Module 1: Deploying the Network Using Socket Programming

Firstly, the connection has to be established between a client program and a server program that is represented by sockets. Such programming is called as socket programming. This connection works on the TCP model because of its reliability and the order of packages that were sent are maintained properly.

For Server.c

1. socket() function creates a new socket inside kernel and returns an integer which used as socket descriptor.
2. For IP4 address we are sending first argument as AF_INET. You can also see we are assigning ipOfServer = AF_INET, represents that this argument related to Internet IP addresses.
3. SOCK_STREAM argument confirms that we are using TCP as a protocol in this communication, which is reliable communication.
4. Sending '0' as third argument is, we are saying to kernel that use default protocol
5. Next we have to bind the created socket to structure ipOfServer. For this we are calling bind() functional. Which includes port, ip addresses as details.
6. listen() also an inbuilt function the 2nd argument 20 says that maximum 20 number of clients can connect to that server. So maximum 20 queue process can be handled by server.
7. Upto now server started. Next server waits for client request by accept() function.
8. accept() runs infinite loop to keep server always running. But it may eat up all CPU processing, to avoid that we have written sleep(1), which server went to sleep for 1 sec.
9. When server hit by client, it prints date and time on clients socket through descriptor returned by accept().

For Client.c

1. Since this communication through socket, here also, we created socket.
2. Port number of the process and IP address both bundled in a structure. We connect these with socket
3. Once sockets are connected, the server sends the date and time to client socket through clients socket descriptor.

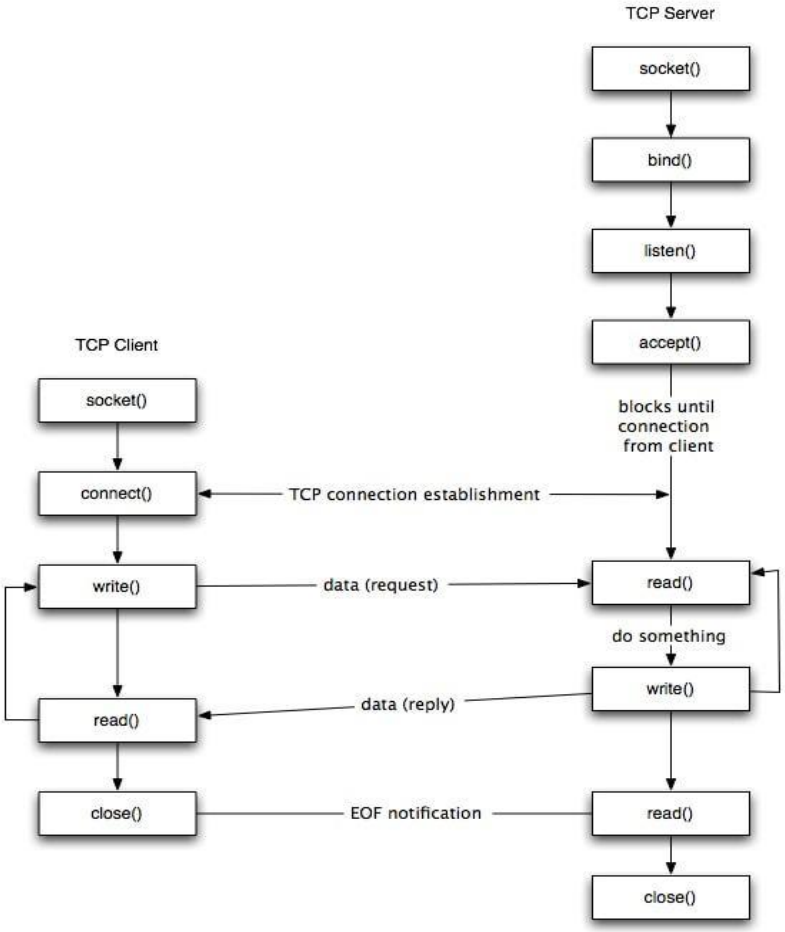


Figure 7.1 Client-Server Flow diagram[d]

After the Client and Server Programs are compiled using GCC. We will be able to send requests and receive responses on the client side while we can receive requests and send responses from the server to the client.

Module 2: Data Encryption Using RSA Algorithm

a. Generation of RSA modulus (n)

Select two large primes, p and q.

Calculate n=p*q. (Equation 1)

For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.

b. Find the Derived Number (e)

Number e must be greater than 1 and less than (p − 1)(q − 1). There must be no common factor for e and (p − 1)*(q − 1) except for 1. In other words two numbers e and (p – 1)(q – 1) are co-prime. Form the public key. The pair of numbers (n, e) form the RSA public key and is made public. Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n. This is strength of RSA.

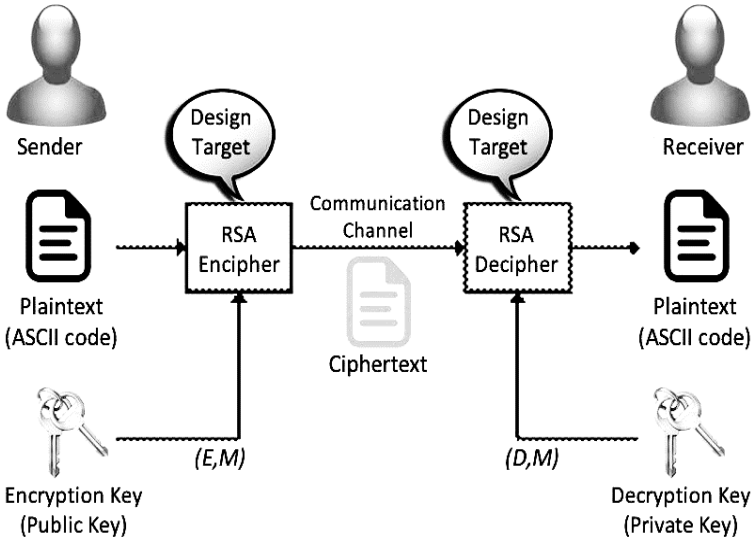


Figure 7.2 Flow of RSA algorithm (b)

c. Generation of Private Key

Private Key d is calculated from p, q, and e. For given n and e, there is unique number d. Number d is the inverse of e modulo (p - 1)(q – 1). This means that d is the number less than (p - 1)(q - 1) such that when multiplied by e, it is equal to 1 modulo (p - 1)(q - 1). This relationship is written mathematically as follows:

$$e.d = \mathbf{1 \bmod (p-1)(q-1)} \quad \text{(equation 2)}$$

The Extended Euclidean Algorithm takes p, q, and e as input and gives d as output

RSA Encryption

Suppose the sender wish to send some text message to someone whose public key is (n, e). The sender then represents the plaintext as a series of numbers less than n. To encrypt the first plaintext P, which is a number modulo n. The encryption process is simple mathematical step as:

$$\mathbf{C = Pe \bmod n} \quad \text{(equation 3)}$$

In other words, the ciphertext C is equal to the plaintext P multiplied by itself e times and then reduced modulo n. This means that C is also a number less than n.

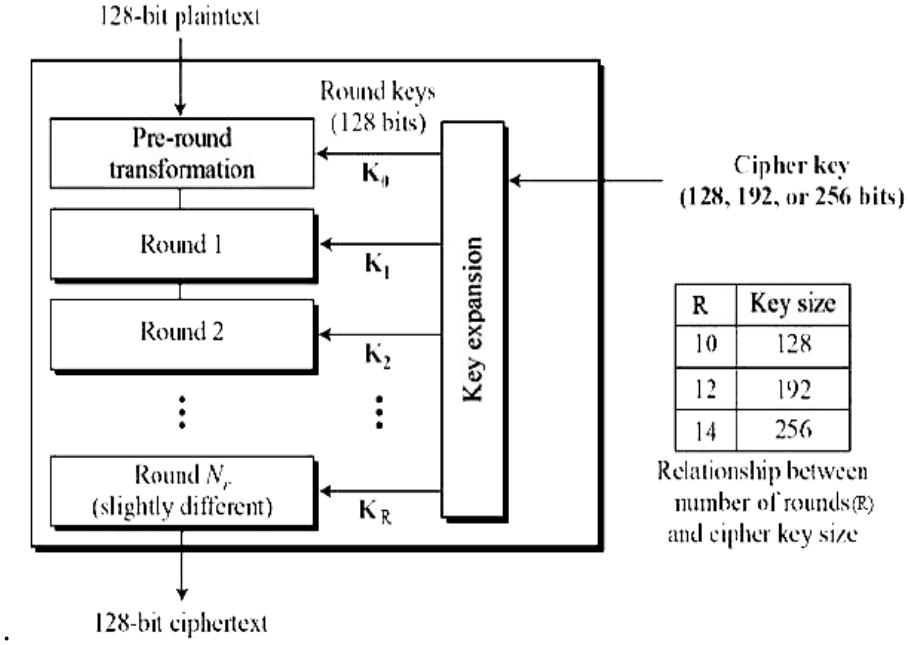
RSA Decryption

The decryption process for RSA is also very straightforward. Suppose that the receiver of public-key pair (n, e) has received a ciphertext C. Receiver raises C to the power of his private key d. The result modulo n will be the plaintext P.

$$\mathbf{Plaintext = Cd \bmod n} \quad \text{(equation 4)}$$

Module 3: AES Algorithm

AES basically repeats 4 major functions to encrypt data. It takes 128 bit block of data and a key [layman's term: password] and gives a ciphertext as output



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

Figure 7.3 AES Algorithm (c)

Encryption Process

The encryption process consists of 4 main functions:

- ByteSubstitution
- Shift Rows
- Mix Columns
- AddRoundKey

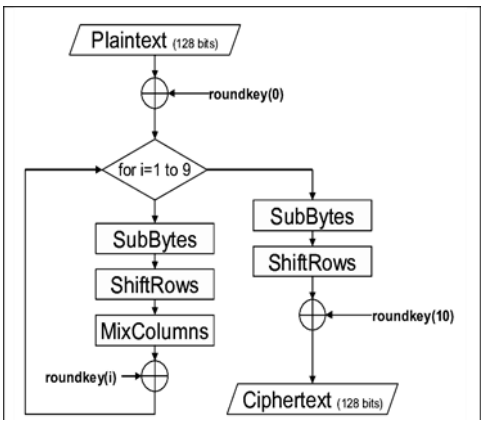


Figure 7.4: AES Encryption Process

Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns. .

Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows − .

1. First row is not shifted. .
2. Second row is shifted one (byte) position to the left .
3. Third row is shifted two positions to the left. .
4. Fourth row is shifted three positions to the left. .

The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

MixColumns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round. .

Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the **reverse order.** Each round consists of the four processes conducted in the reverse order: .

- Add round key
- Mix columns
- Shift rows
- Byte substitution

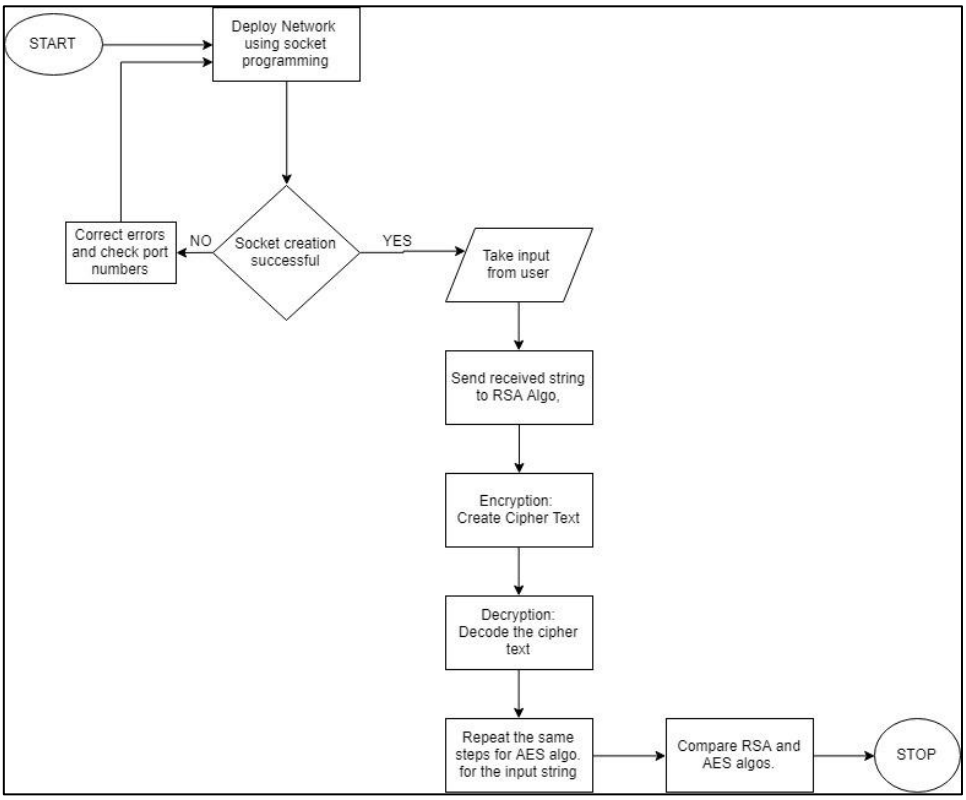
Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.

8 DESIGN

8.1 Algorithm for the working code

Step 1: Start
Step 2: Deploy the network using Socket Programming
Step 3: If not successful, then check errors and correct the port number associated.
Step 4: If successful, create Socket(), Accept() data, Connect() to server, and Close() the socket after the completion.
Step 5: Take input from the user for both RSA and AES Algorithm
Step 6: Send the received input form the user
Step 7: Send input to the RSA Module Code in first iteration and to AES Module in the second Iteration
Step 8: Perform the Encryption in the RSA format, for the first iteration and in the AES format for the second iteration
Step 9: Show the Cipher Text to user
Step 10: Decrypt the Cipher Text to get the plain text
Step 11: Repeat Steps 6, 7, 8, 9 and 10 for the AES Module (the second Iteration)
Step 12: Compare the working of two Algorithms
Step 13: Show results to the user
Step 14: Display “Thank You”
Step 15: End

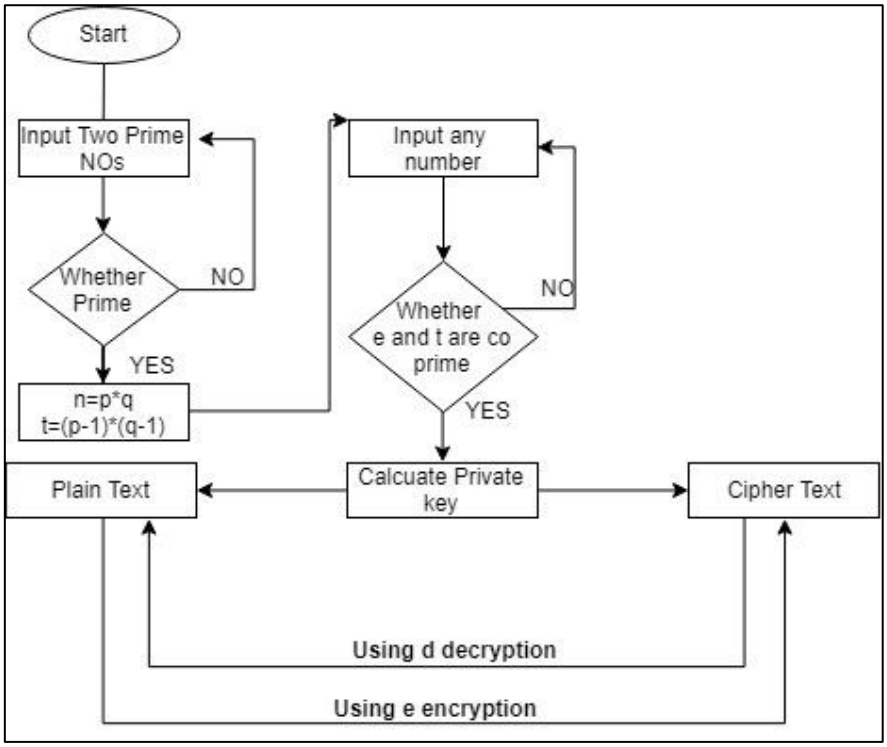
Flowchart of the System



8.2 Pseudo Code for RSA

1. Generate (Find) two Prime Numbers (P and Q)
2. Calculate N = P*Q.
3. Calculate M = Φ(N) = (P-1)*(Q-1).
4. Select any integer e, the rules to select E are:
   a. E is positive integer
   b. 0<E<M
   c. GCD(M,E) = 1
5. Calculate D a use Extended Euclid Theorem
   $ed = 1 \bmod (p-1)(q-1)$
6. At Encryption: C = P^e *modn
7. At decryption: C *d mod n

Flowchart of RSA Algorithm



8.3 Algorithm of AES Module

Encryption:

- Derive the set of round keys from the cipher key.
- Initialize the state array with the block data (plaintext).
- Add the initial round key to the starting state array.
- Perform nine rounds of state manipulation.
- Perform the tenth and final round of state manipulation.
- Copy the final state array out as the encrypted data (ciphertext).

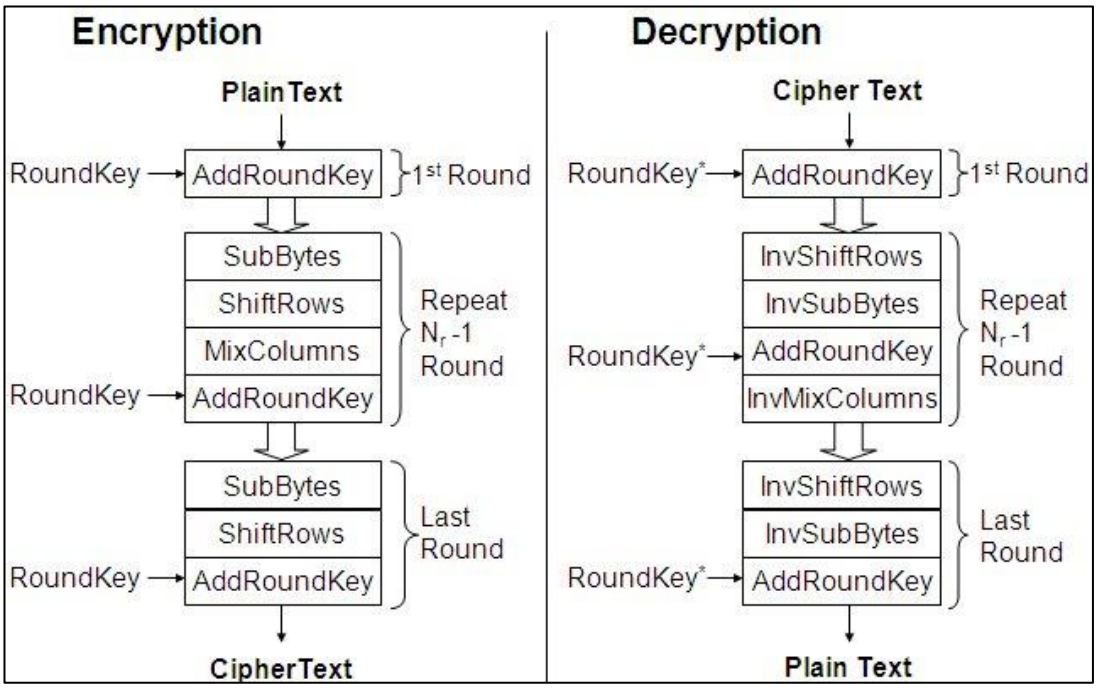
Decryption:

The order of operation in decryption is:

1. **Perform initial decryption round:**
   - XorRoundKey
   - InvShiftRows
   - InvSubBytes
2. Perform nine full decryption rounds:
   - XorRoundKey
   - InvMixColumns
   - InvShiftRows
   - InvSubBytes
3. Perform final XorRoundKey

The same round keys are used in the same order.

Flowchart of AES Algorithm



8.4 Algorithm for Socket Programming

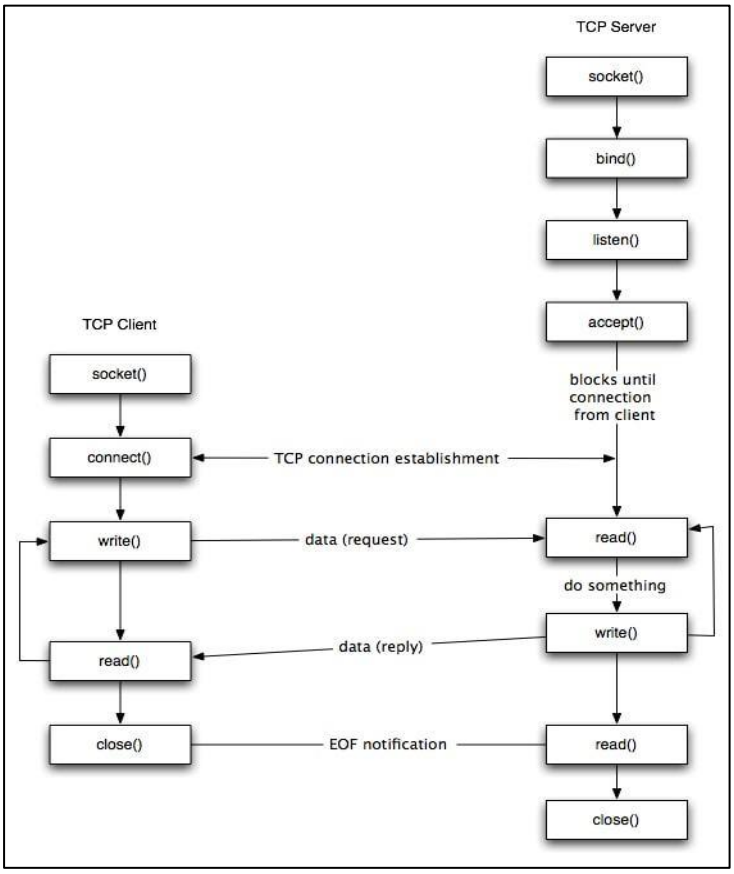
<u>Steps to create a client using TCP/IP</u>

- Create a socket using the socket() function in c.
- Initialize the socket address structure as per the server and connect the socket to the address of the server using the connect();
- Receive and send the data using the recv() and send() functions.
- Close the connection by calling the close() function.

Steps to create a server using TCP/IP

- Create a socket using the socket() function in c.
- Initialize the socket address structure and bind the socket to an address using the bind() function.
- Listen for connections with the listen() function.
- Accept a connection with the accept() function system call. This call typically blocks until a client connects to the server.
- Receive and send data by using the recv() and send() function in c.
- Close the connection by using the close() function.

Flowchart of Socket Programming Module



9 System Requirements (Software/Hardware)

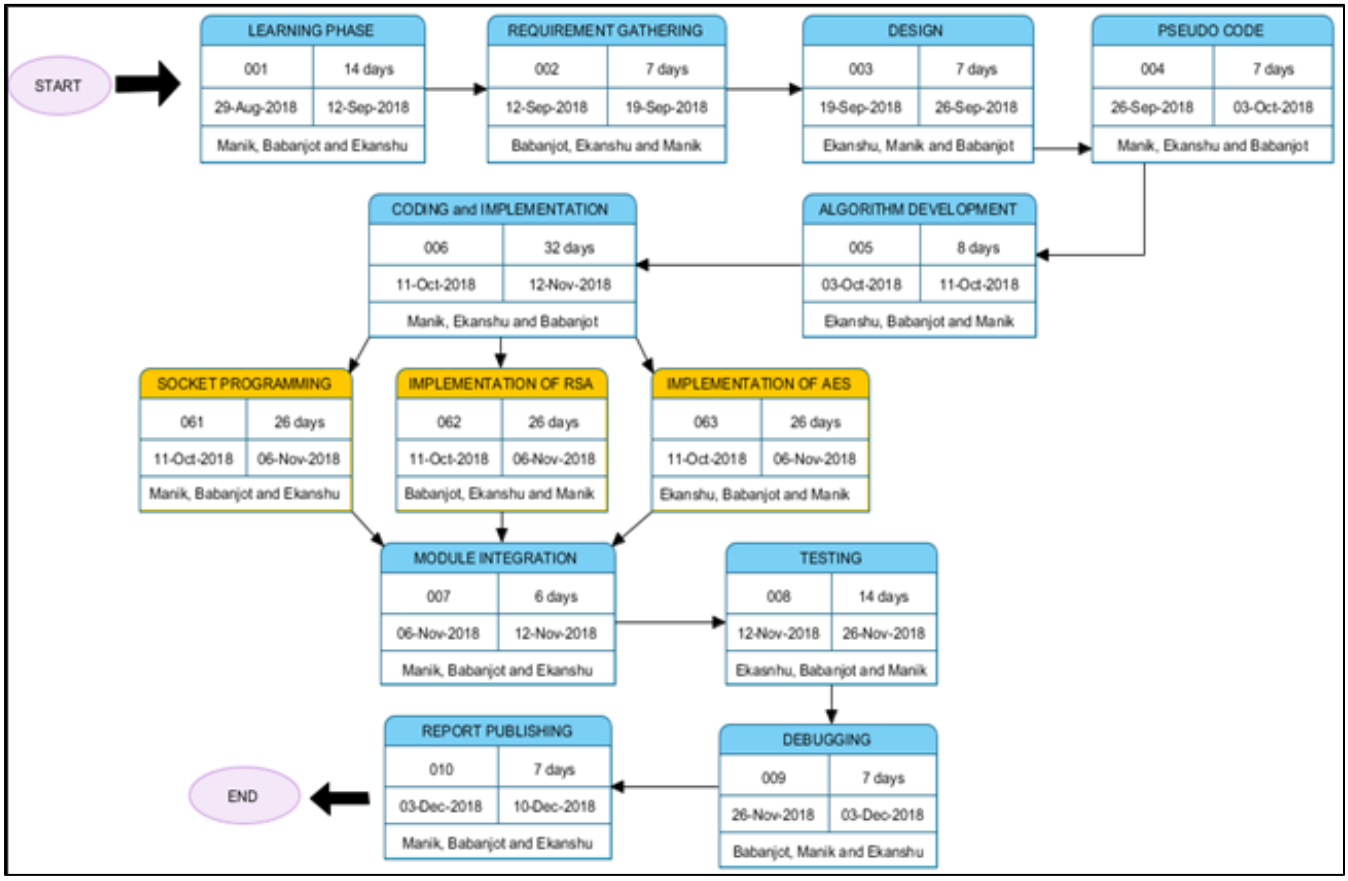
- **Hardware:**

a. Intel(R) Core(TM) i3-3200 CPU @ 1.5 GHz
b. 2 GB RAM
c. System type is 32/64-bit Operating System

- **Software:**

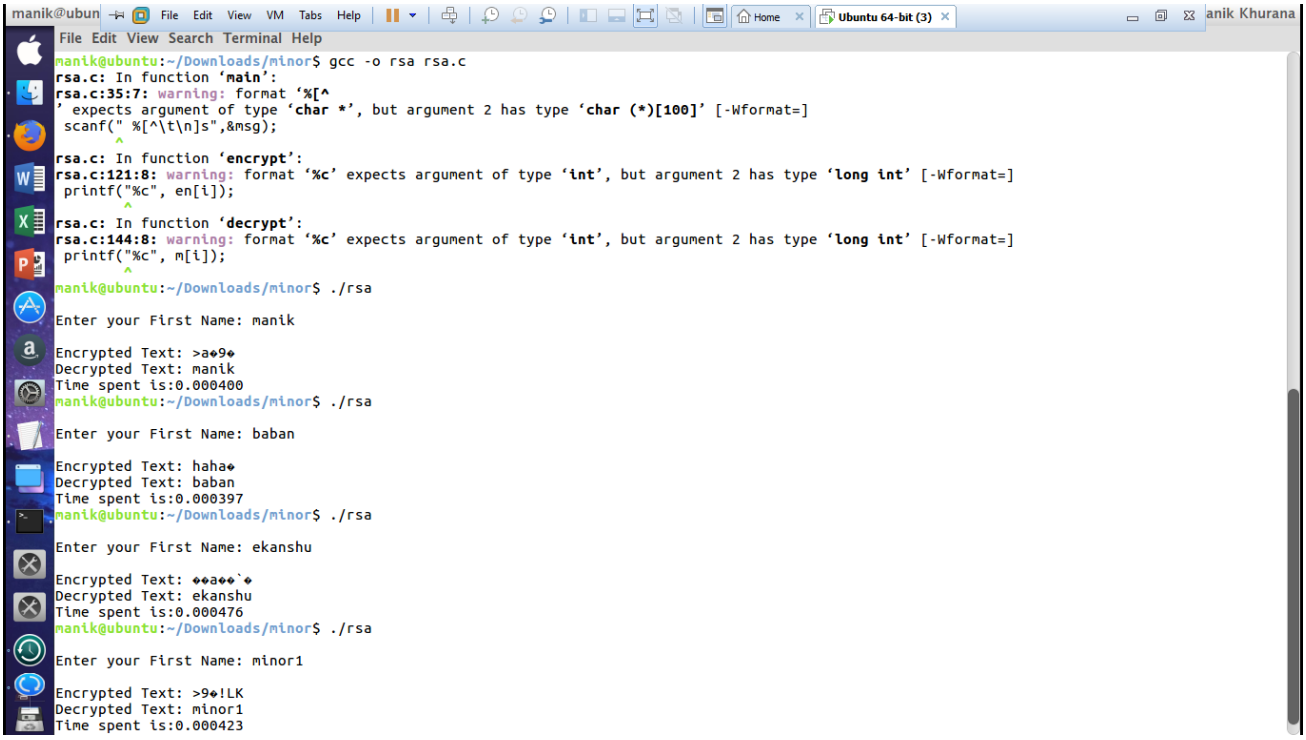
Linux based operating System – Ubuntu and GCC compiler.

10 Schedule (**P**roject **E**valuation and **R**eview **T**echnique Chart**)**

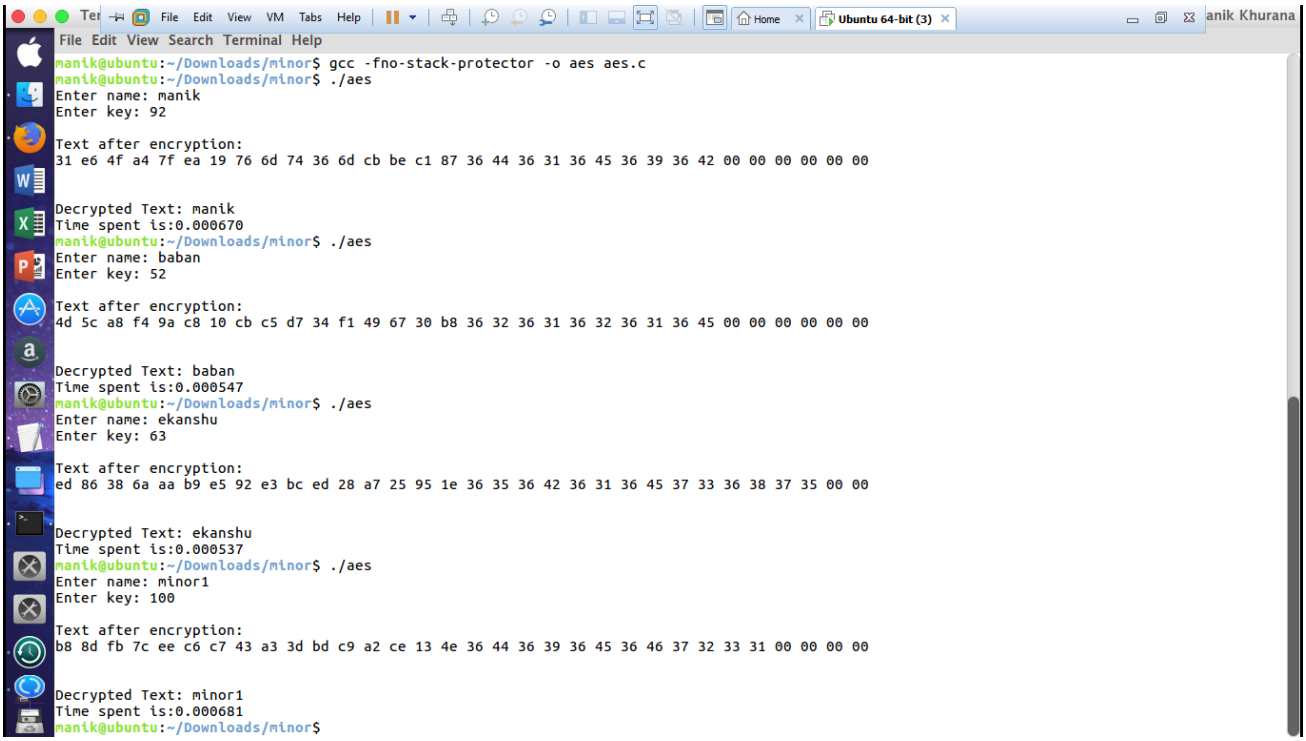


11 Working Code Outputs

File name: rsa.c

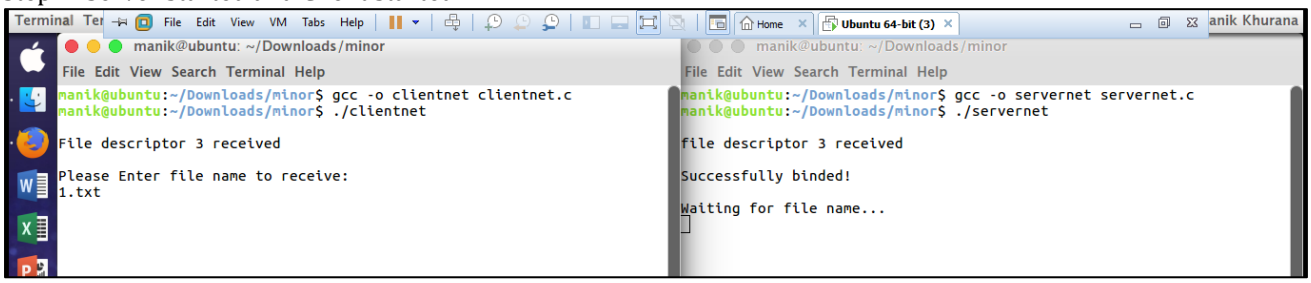


File name: aes.c

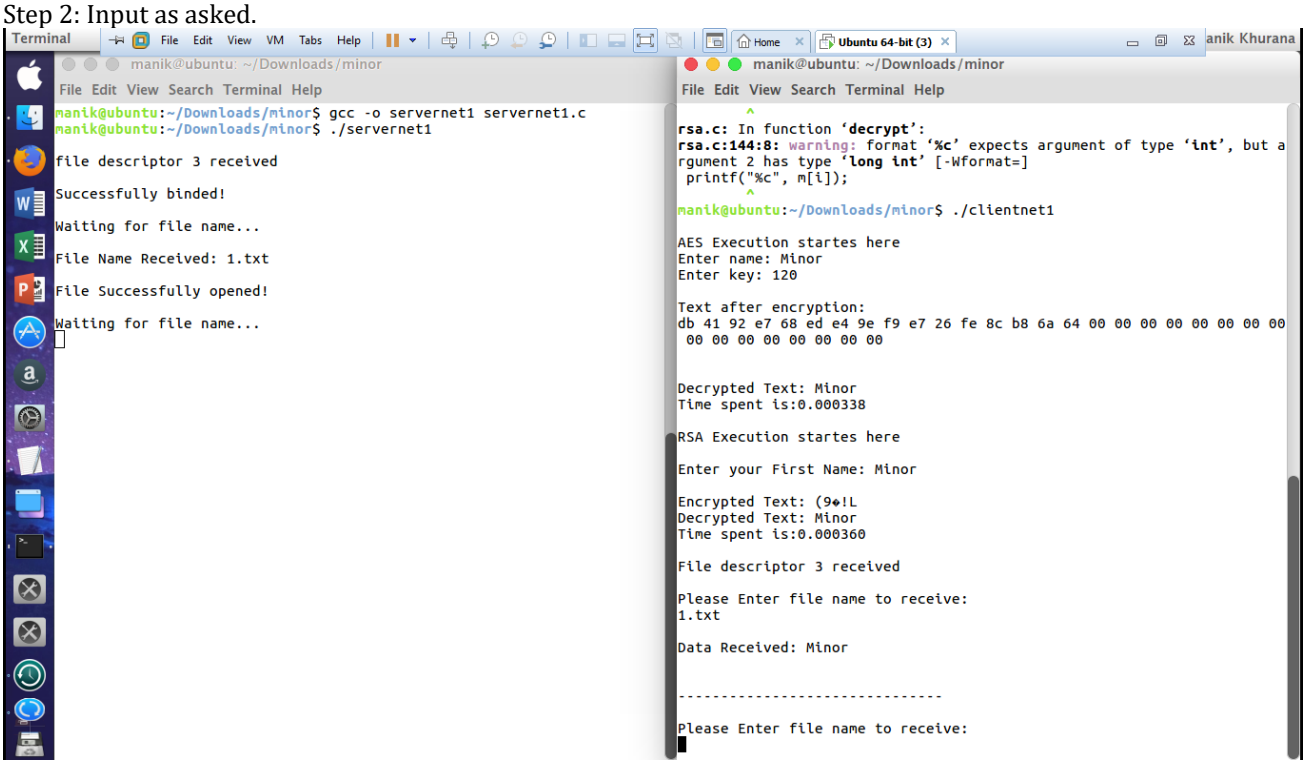


File name: clientnet.c and servernet.c (Interaction steps)

Step 1: Server started and Client started



Step 2: Input as asked.



12 Final Comparison

FEATURE	RSA	AES
DEVELOPED	1977	2000
KEY LENGTH	MORE THAN 1024 bits	128,192,256 bits
CIPHER TYPE	ASYMMETRIC BLOCK CIPHER	SYMMETRIC BLOCK CIPHER
SECURITY	LESS SECURE	MUCH SECURE
ENCRYPTION/DECRYPTION SPEED	SLOWER	FASTER
HARDWARE AND SOFTWARE IMPLEMENTATION	NOT EFFICIENT	EFFICIENT IMPLEMENTATION

13 References

[1] Jean Sébastien Coron “Crypto Corner”, *University of Luxembourg*, 2006.

[2] Divya Singh et al, *International Journal of Computer Science and Mobile Applications*, http://www.academia.edu/97169/Study_of_Data_Transmission_Using_Sockets.

[3] Hellman, M. and J. Diffie, 1976. New Directions in Cryptography. *IEEE Transactions on Information theory, vol. IT-22, pp:644-654.*

[4]. Jamgekar, R. S., & Joshi, G. S. (2013), File Encryption and Decryption Using Secure RSA, *International Journal of Emerging Science and Engineering (IJESE), 1(4), 11–14.*

[5] Fei Shao, Zinan Chang, *(2010) Second International Conference on Communication Software and Networks.*

[6]. Socket Programming Tutorials (http://www.binarytides.com/socket-programming-c-linux-tutorial)

[7]. M.Pitchaiah, Philemon Daniel, Praveen, Implementation of Advanced Encryption Standard Algorithm, *International Journal of Scientific & Engineering Research Volume 3, Issue 3, March -2012 1 ISSN 2229-5518*

[8] Abhinandan Aggarwal1 , Gagandeep singh2 , Prof. (Dr.) Neelam Sharma3 *Department of electronics and communication, Maharaja Agrasen Institute of Technology,New delhi ,India*

Links:

[a].http://mathworld.wolfram.com/RSAEncryption.html
(Visit Date: 26/09/18)
[b].http://www.academia.edu/97169/Study_of_Data_Transmission_Using_Sockets.
(Visit Date: 5/10/18)
[c].https://aticleworld.com/socket-programming-in-c-using-tcpip/
(Visit Date: 8/10/18)
[d].https://www.geeksforgeeks.com/socket-programming-in-c-using-tcpip/
(Visit Date: 15/10 /18)
[e].https://www.tutorialspoint.com/cryptography/public_key_encryption.htm
(Visit Date: 16/10/18)
[f]. https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm
(Visit Date: 18/10/18)