

# **CryptCom: Secure and Fast communication using encryption and compression technique**

**A**

## ***Project Report***

*submitted in partial fulfillment of the  
requirements for the award of the degree of*

## **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING**

**by**

<b>Name</b>	<b>Roll No.</b>
<b>Prakash Tiwari</b>	<b>R100217049</b>
<b>Amrit Kumar</b>	<b>R100217007</b>
<b>Akshit Chauhan</b>	<b>R100217004</b>
<b>Gaurav Singh</b>	<b>R100217026</b>

*Under the guidance of*

**Pushpendra Kumar Rajput**

Assistant Professor, SoCSE

Department of Cybernetics



**25**

**Department of Cybernetics**

**School of Computer Science and Engineering**

**Bidholi, Via Prem Nagar, Dehradun, UK**

**December – 2019**

## **CANDIDATE’S DECLARATION**

I/We hereby certify that the project work entitled “**CryptCom: Secure and Fast Communication using encryption and Compression Technique**” in partial fulfillment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in Open Source and Open Standard and submitted to the Department of Computer Science & Engineering at Center for Information Technology, University of Petroleum & Energy Studies, Dehradun, is an authentic record of our work carried out during a period from August, **2019** to **December, 2019** under the supervision of **Pushpendra Kumar Rajput, Assistant Professor SoCSE Department Of Cybernetics.**

The matter presented in this project has not been submitted by me/ us for the award of any other degree of this or any other University.

<b>Name</b>	<b>Roll No.</b>
<b>Prakash Tiwari</b>	<b>R100217049</b>
<b>Amrit Kumar</b>	<b>R100217007</b>
<b>Akshit Chauhan</b>	<b>R100217004</b>
<b>Gaurav Singh</b>	<b>R100217026</b>

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 18 Dec 2019

**Dr. Monit Kapoor**  
Head of the Department  
Department Of Cybernetics  
School of Computer Science and Engineering  
University of Petroleum & Energy Studies  
Dehradun – 248 007 (Uttarakhand)

---

## **ACKNOWLEDGEMENT**

We wish to express our deep gratitude to our guide **Pushpendra Kumar Rajput**, for all advice, encouragement and constant support he has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our respected Program Head of the Department, **Monit Kapoor**, for his great support in our project.

We are also grateful to **Dr. Manish Prateek**, Dean CoES, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

<b>Name</b>	<b>Prakash Tiwari</b>	<b>Amrit Kumar</b>	<b>Akshit Chauhan</b>	<b>Gaurav Singh</b>
<b>Roll No.</b>	<b>R100217049</b>	<b>R100217007</b>	<b>R100217004</b>	<b>R100217026</b>

## ABSTRACT

CryptCom is a chatting platform which uses Data Compression and Data Encryption Techniques to optimize the Communicating Experience of the Users. The chief focus of the Project is to provide and maintain a Secure and Data/Network Convenient path between two or more users so they can use it for communicating. Low speed internet will not restrict in chatting platform between two users.

For a Communicating Server, there should be a connection established first between the hosts. Communication between individual users is established via Socket Programming in C. Data Encryption is a must be present in the system, nowadays as everyone wants their data to be secure. For data Encryption there are many Techniques/Algorithms we can use. And we are using RSA.

For Data Compression we are Using Huffman Code as it was taught us in our DAA Course in 2<sup>nd</sup> Semester.

Encryption is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot.[2] Encryption does not itself prevent interference, but denies the intelligible content to a would-be interceptor. In an encryption scheme, the intended information or message, referred to as plaintext, is encrypted using an encryption algorithm – a cipher – generating cipher text that can be read only if decrypted. Decryption uses the decryption key to convert cipher text to plain text i.e. the original data. [1] For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, considerable computational resources and skills are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients but not to unauthorized users.

In computer science and information theory, a Huffman coding is the famous greedy algorithm that is used for the lossless compression of data. [4] The output from Huffman's algorithm can be viewed as a variable-length code table for encoding a source symbol (such as a character in a file). The algorithm derives this table from the estimated probability or frequency of occurrence (weight) for each possible value of the source symbol.

Socket programming is a way of connecting two nodes on a network to communicate with each other.[3] 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.

# TABLE OF CONTENTS

<b>S.No.</b>	<b>Contents</b>	<b>Page No</b>
<b>1.</b>	<b>Introduction</b>	6
<b>2.</b>	<b>Literature Review</b>	6
<b>3.</b>	<b>Problem Statement</b>	6
<b>4.</b>	<b>Objective</b>	7
	4.1. Objective Achieved	7
<b>5.</b>	<b>Methodology</b>	7
	5.1 Introduction	7
	5.2 Planning	7
	5.3 Implementation	7
	5.4 Testing and Analysis	8
<b>6.</b>	<b>Implementation</b>	8
	6.1 Socket Connectivity	8
	6.2 Huffman Algorithm	8
	6.3 RSA (Rivest Shamir Aldeman) Algorithm	8
	6.4 CryptCom Algorithm	9
<b>7.</b>	<b>Output screens</b>	9
	7.1 Socket Connectivity	9
	7.2 Huffman Code	10
	7.3 Project Output	11
<b>8.</b>	<b>Limitations and Future Enhancements</b>	12
<b>9.</b>	<b>Conclusion</b>	13
<b>10.</b>	<b>Diagrams</b>	14
	10.1 Huffman Algorithm	14
	10.2 RSA Algorithm	15
	10.3 Data Flow Diagram	16
	10.4 Flow Chart	17
<b>11.</b>	<b>System Requirement</b>	18
<b>12.</b>	<b>Schedule (PERT Chart)</b>	19
<b>13.</b>	<b>References</b>	20

## **1) Introduction:**

For a Communicating Server, there should be a connection established first between the hosts. Communication between individual users is established via Socket Programming in C. Data Encryption is a must be present in the system, nowadays as everyone wants their data to be secure. For data Encryption there are many Techniques/Algorithms we can use. And we are using RSA.

For Data Compression we are Using Huffman Code as it was taught us in our DAA Course in 2<sup>nd</sup> Semester.

## **2) Literature Review:**

Encryption is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot.[2] Encryption does not itself prevent interference, but denies the intelligible content to a would-be interceptor. In an encryption scheme, the intended information or message, referred to as plaintext, is encrypted using an encryption algorithm – a cipher – generating cipher text that can be read only if decrypted. Decryption uses the decryption key to convert cipher text to plain text i.e. the original data. [1] For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, considerable computational resources and skills are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients but not to unauthorized users.

In computer science and information theory, a Huffman coding is the famous greedy algorithm that is used for the lossless compression of data. [4] The output from Huffman's algorithm can be viewed as a variable-length code table for encoding a source symbol (such as a character in a file). The algorithm derives this table from the estimated probability or frequency of occurrence (weight) for each possible value of the source symbol.

Socket programming is a way of connecting two nodes on a network to communicate with each other.[3] 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.

**3) Problem Statement:** Nowadays there are many chatting platforms, which offer a variety of services. But are they Secure? Is your chat Secure? Does it work with low speed Internet or even No Internet?

#### **4) Objective:**

To establish a connection between multiple users, enable them to chat with one another with a decentralized encryption and compression technique, for secure and fast communication.

- To establish a Connection between 2 or more hosts with server.
- To apply Huffman Code for Data Compression.
- To apply RSA for Data Encryption
- At the receiver's side: Decompress the message and decrypt for the desired output.

##### **4.1) Objective Achieved:**

- Socket Connection is achieved.
- We have achieved compression using Huffman Encoding/Decoding algorithm.
- We have achieved Encryption and Decryption of message using RSA.
- Now we have achieved our main objective by integrating above objectives to form Secure and Fast chatting platform.

#### **5) Methodology:**

##### **5.1) Introduction**

This module covers details of planning, implementation, testing and analysis of our project based on the requirement.

##### **5.2) Planning**

In planning phase we study information from various research paper, article, blogs related to encryption and compression, also we identifies the hardware and software requirements of the project, software requirement like operating system(Ubuntu 18.04), Programming language(C), GCC compiler. We will use Ubuntu terminal to run client side and server side program, so that client can easily communicate with server.

##### **5.3) Implementation**

First Server will create the socket using create () and then bind to the port after server start listening request. Now, client create socket and connect to the server using TCP. Our server can also send the message. Client/Server can send the message as String, first it will encrypted (using RSA) and compressed (using Huffman) then send to the receiver side. Receiver (client/server) will decompress and decrypt into the original message. Compression technique will increase the performance and also we can send data fast, by using encryption message will be secured from other client.

#### **5.4) Testing & Analysis**

In testing phase we will use Black Box Testing in which focus is only on input and desired output. Our input format will be alphanumeric and every input is checked for both valid and invalid inputs and this technique is known as Equivalence partitioning technique. In our project we will specify length of message. Related to the length of our message we will make test cases to check the validity of the input. In last we analyze the performance of our project on the basis of test cases

### **6) Implementation**

#### **6.1) CryptCom Algorithm:**

1. Connection Created
- Sender Side:
  2. Input message
  3. Message compressed using Huffman
  4. Message Encryption
  5. Message send
- Receiver Side:
  6. Message received
  7. Message Decrypt
  8. Decompress and show
9. Repeat 2-8 until "Exit" encountered

#### **6.2) Socket Connectivity:**

1. Server side: Server Create socket using socket (AF\_INET, SOCK\_STREAM, 0). Bind socket to a specific port where clients can contact the server.
2. listen(int sockfd, int backlog); It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection.
3. Client side: Create socket using socket (AF\_INET, SOCK\_STREAM, 0).
4. Client connect to the server using connect () method in local host ip.
5. Client and server can send the message.
6. Type exit to terminate the chat.

#### **6.3) Huffman Algorithm:**

1. Input: Number of message with frequency count.
2. Output: Huffman merge tree.
3. Create a leaf node for each unique character and build a min heap of all leaf nodes.
4. Extract two nodes with the minimum frequency from the min heap.
5. Create a new internal node with a frequency equal to the sum of the two nodes frequencies.
6. Repeat steps#4 and #5 until the heap contains only one node. The remaining node is the root node and the tree is complete.

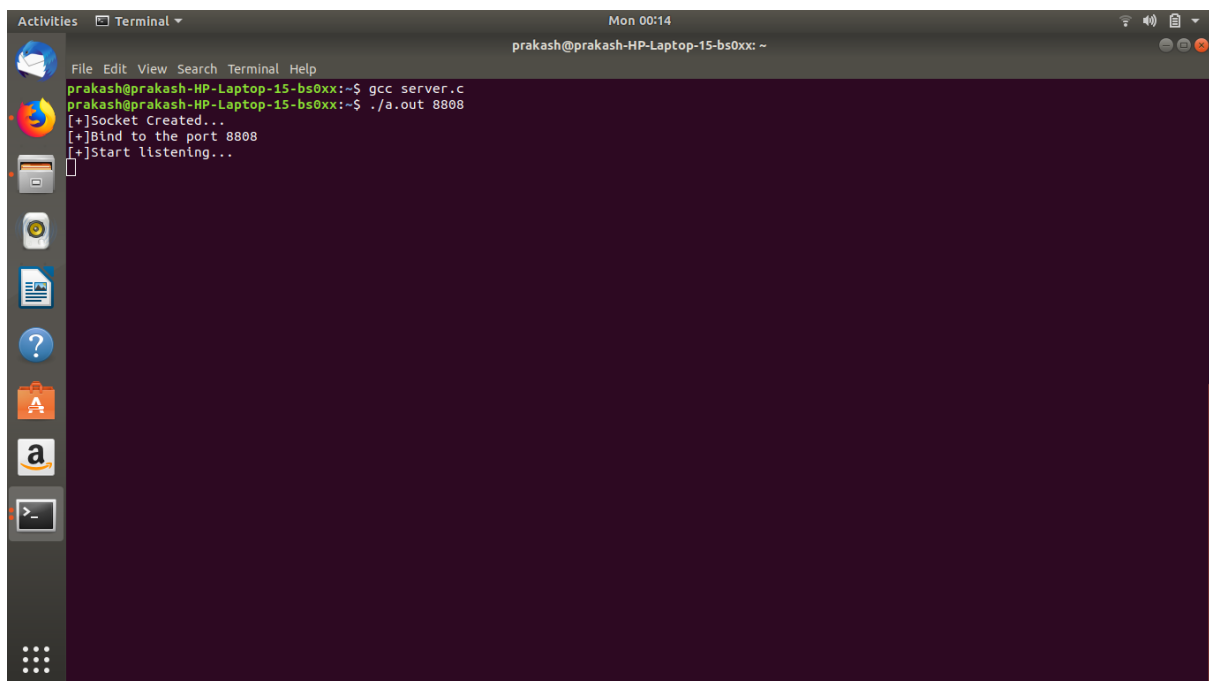


#### 6.4) RSA (Rivest Shamir Aldeman) Algorithm:

1. Choose two large prime no:  $p$  and  $q$
2. Calculate  $n$ :  $n = p * q$
3. Calculate Euler's totient function  $\phi(x)$ :  $\phi(x) = (p-1) * (q-1)$
4. Choose an integer  $e$ ,  $1 < e < \phi(x)$  such that  $\gcd(e, \phi(x)) = 1$
5. Calculate  $d$ ,  $1 < d < \phi(x)$  such that:  $d * e = 1 \bmod \phi(x)$
6. Public key is  $(n, e)$
7. Private key is  $(n, d)$
8. Encryption: The Cypher text  $C$  is found by the equation  $C = M^e \bmod n$   
(Where  $M$  is the original message)
9. Decryption: The message  $M$  can be found from the cypher text  $C$  by the  
Equation  $M = C^d \bmod n$

#### 7) Output Screen:

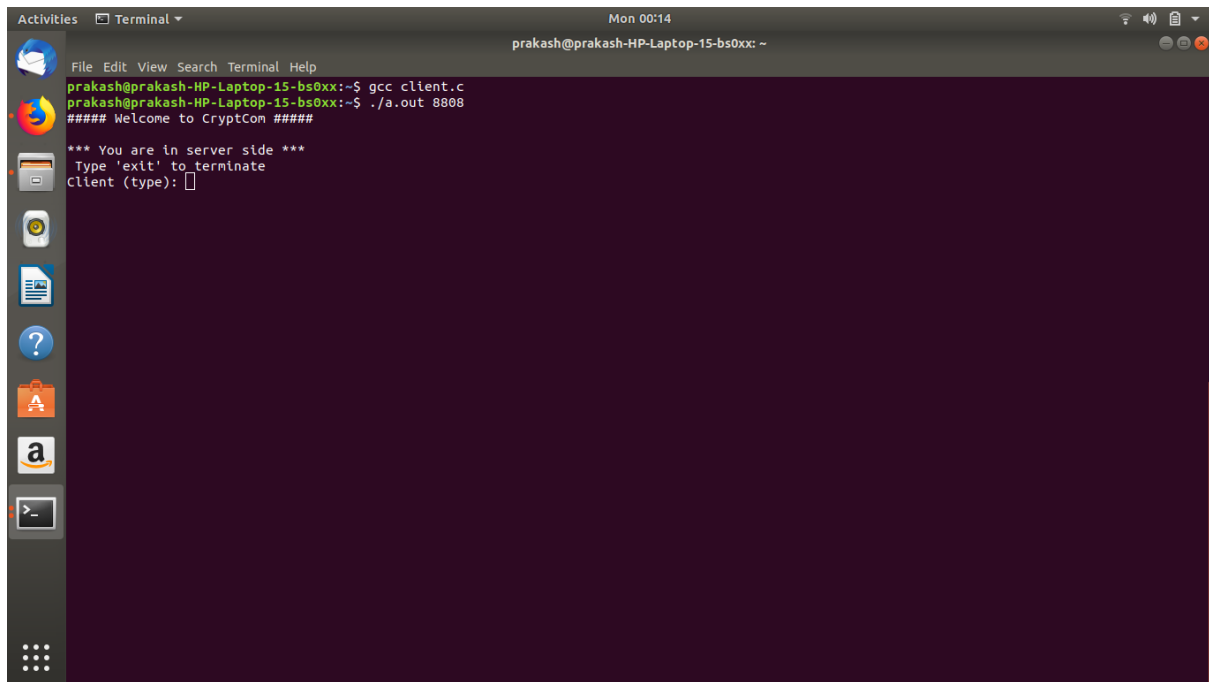
##### 7.1) Socket Connectivity:



The screenshot shows a terminal window titled "Terminal" with the user "prakash" on a system named "prakash-HP-Laptop-15-bs0xx". The terminal output shows the compilation and execution of a C program named "server.c". The program successfully creates a socket, binds it to port 8808, and starts listening for connections. The output is as follows:

```
prakash@prakash-HP-Laptop-15-bs0xx:~$ gcc server.c
prakash@prakash-HP-Laptop-15-bs0xx:~$ ./a.out 8808
[+]Socket Created...
[+]Bind to the port 8808
[+]Start listening...
```

Fig 1.0

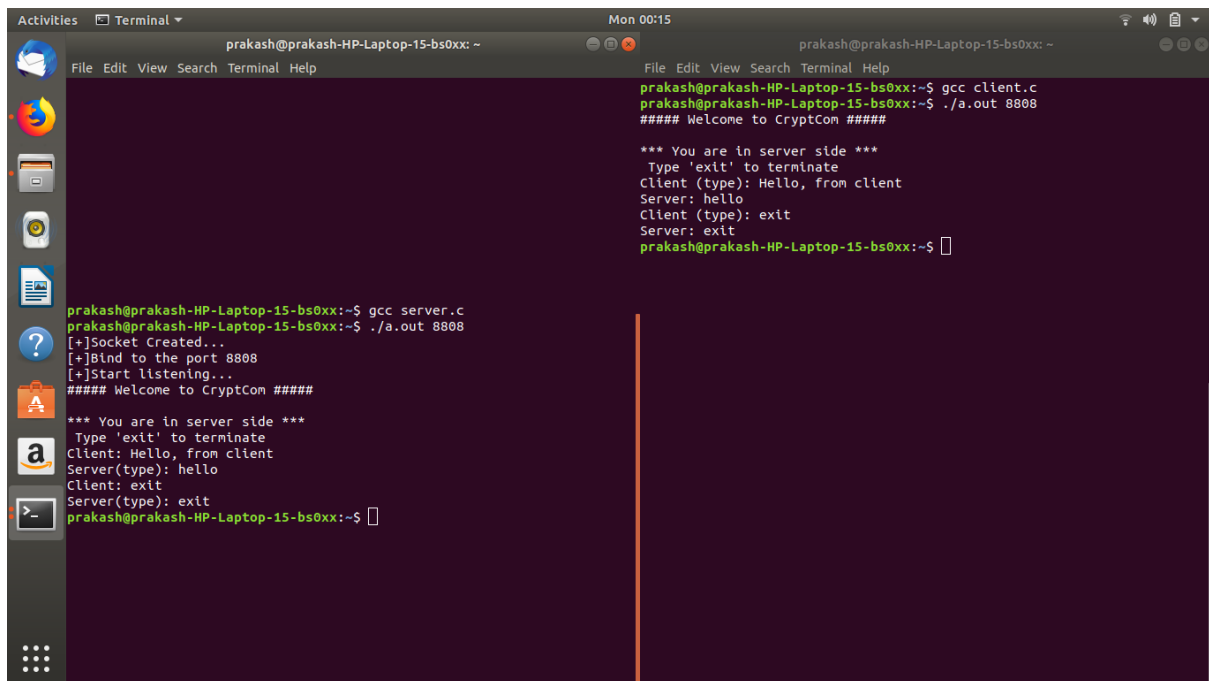


A terminal window titled "prakash@prakash-HP-Laptop-15-bs0xx: ~" showing the execution of a client program. The user enters the command `gcc client.c` and then `./a.out 8808`. The program outputs a welcome message and prompts the user to type 'exit' to terminate. The user has not yet entered anything.

```
prakash@prakash-HP-Laptop-15-bs0xx:~$ gcc client.c
prakash@prakash-HP-Laptop-15-bs0xx:~$ ./a.out 8808
##### Welcome to CryptCom #####

*** You are in server side ***
Type 'exit' to terminate
Client (type):
```

Fig 2.0



A terminal window titled "prakash@prakash-HP-Laptop-15-bs0xx: ~" showing the execution of a server program. The user enters the command `gcc server.c` and then `./a.out 8808`. The program outputs a welcome message and prompts the user to type 'exit' to terminate. The user has not yet entered anything.

```
prakash@prakash-HP-Laptop-15-bs0xx:~$ gcc server.c
prakash@prakash-HP-Laptop-15-bs0xx:~$ ./a.out 8808
[+]Socket Created...
[+]Bind to the port 8808
[+]Start listening...
##### Welcome to CryptCom #####

*** You are in server side ***
Type 'exit' to terminate
Client: Hello, from client
Server(type): hello
Client: exit
Server(type): exit
prakash@prakash-HP-Laptop-15-bs0xx:~$
```

Fig 3.0

### 7.2) Huffman Code:

```
E:\PROJECT\Minor\huffmaaaaa.exe
Enter the message : HELLOWORLDIAMCRYPTCOM
E: 0000
I: 0001
H: 0010
T: 0011
R: 010
M: 011
O: 100
L: 101
D: 1100
Y: 11010
P: 11011
A: 11100
W: 11101
C: 1111
COMPRESSION PERCENT: 50.000000
-----
Process exited after 10.21 seconds with return value 0
Press any key to continue . . .
```

**Fig 4.0**

### 7.3) Project Output

```
prakash@prakash-HP-Laptop-15-b50xx: ~/Minor/Test/Final
File Edit View Search Terminal Help
prakash@prakash-HP-Laptop-15-b50xx:~/Minor/Test/Final$ gcc server.c
prakash@prakash-HP-Laptop-15-b50xx:~/Minor/Test/Final$ ./a.out 8807
[+]Socket Created...
[+]Bind to the port 8807
[+]Start listening...
##### Welcome to CryptCom #####

*** You are in server side ***
Instructions:
1.Type 'exit' to terminate
2.First Client Send Message

New Message Received..
Encrypted Message:
'99'
Decrypted Message:Hi!,
Server Type >Welcome

Message Compressed :67.000000%

Encrypted Message:
♦♦ {*** Message Sent..
New Message Received..
Encrypted Message:
♦ '9'+9+♦♦9♦♦
Decrypted Message:This is Client

Server Type >exit

Message Compressed :68.000000%

Encrypted Message:
♦ '9♦ Message Sent..prakash@prakash-HP-Laptop-15-b50xx:~/Minor/Test/Final$
```

## **8) Limitation and Future Enhancement:**

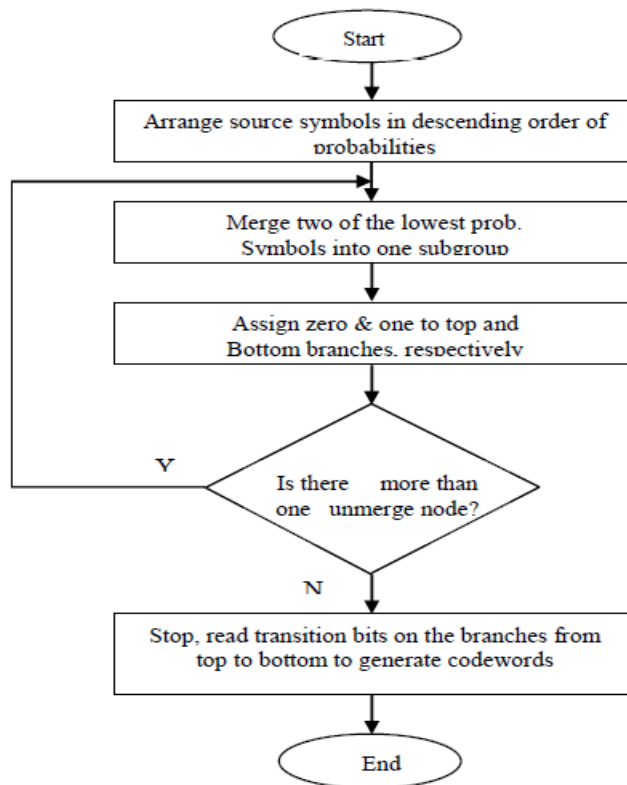
1. History of user chat is maintained so that it can be accessed later.
2. In future we can achieve the multiple user management in which multiple user can send or receive data simultaneously.

## **9) Conclusion:**

Now we conclude that first we establish a connection between client and server, now client sends message to the server and sever will get encrypted and compressed message now this message gets decrypted and decompressed at server side and finally server will receive the original message.

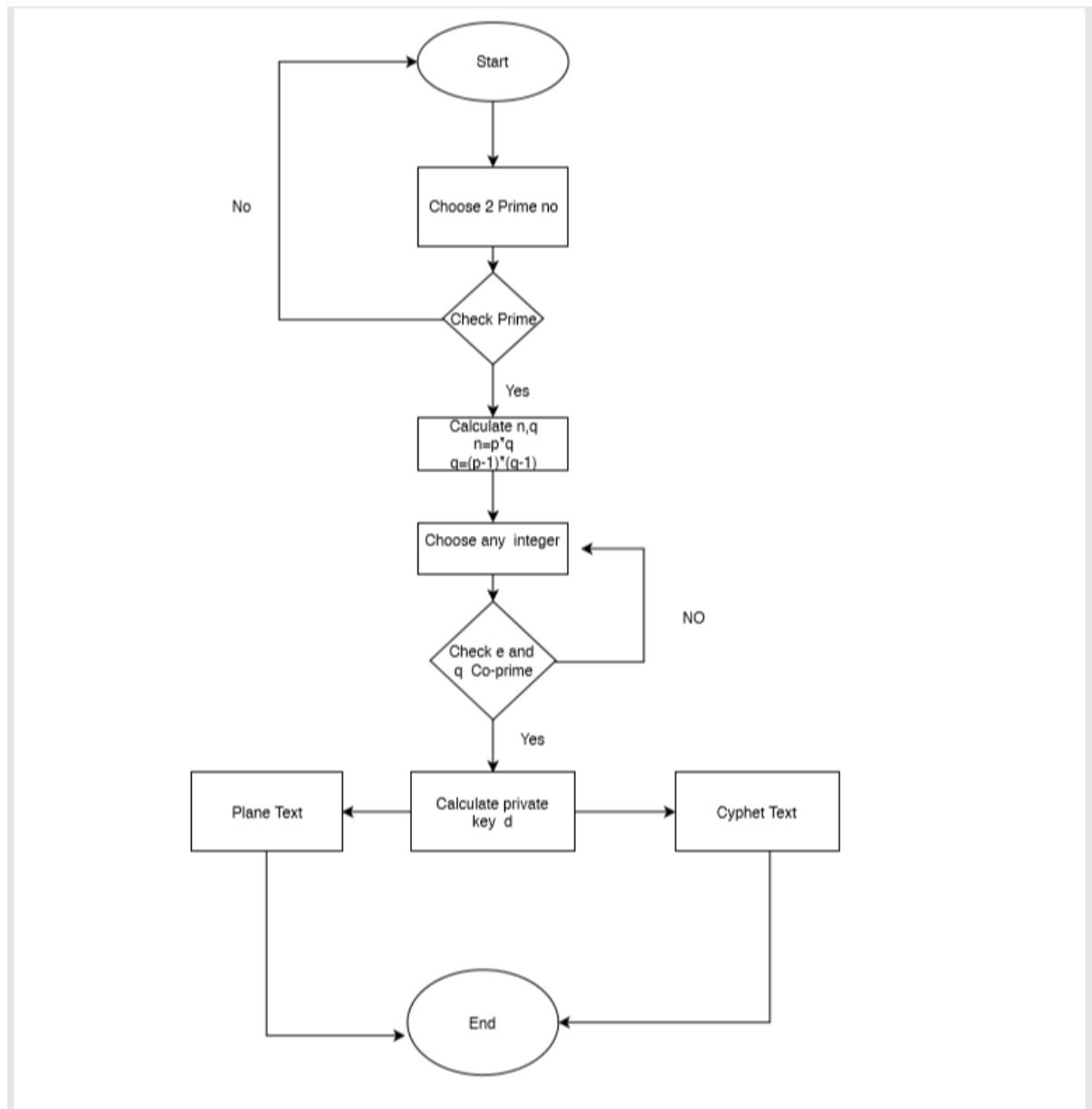
## 10) Diagrams:

### 10.1) Huffman Algorithm



**Fig10.1**

## 10.2) RSA Algorithm



**Fig10.2**



### 11) System Requirement:

Operating System	:	Ubuntu 18.04
Programming Language	:	C
Compiler	:	GCC
Processor	:	Pentium IV
Disk Drive	:	Floppy or Hard Disk Drive
RAM	:	512 MB (min)
Monitor	:	with 80 columns

### 12) Schedule (PERT Chart):

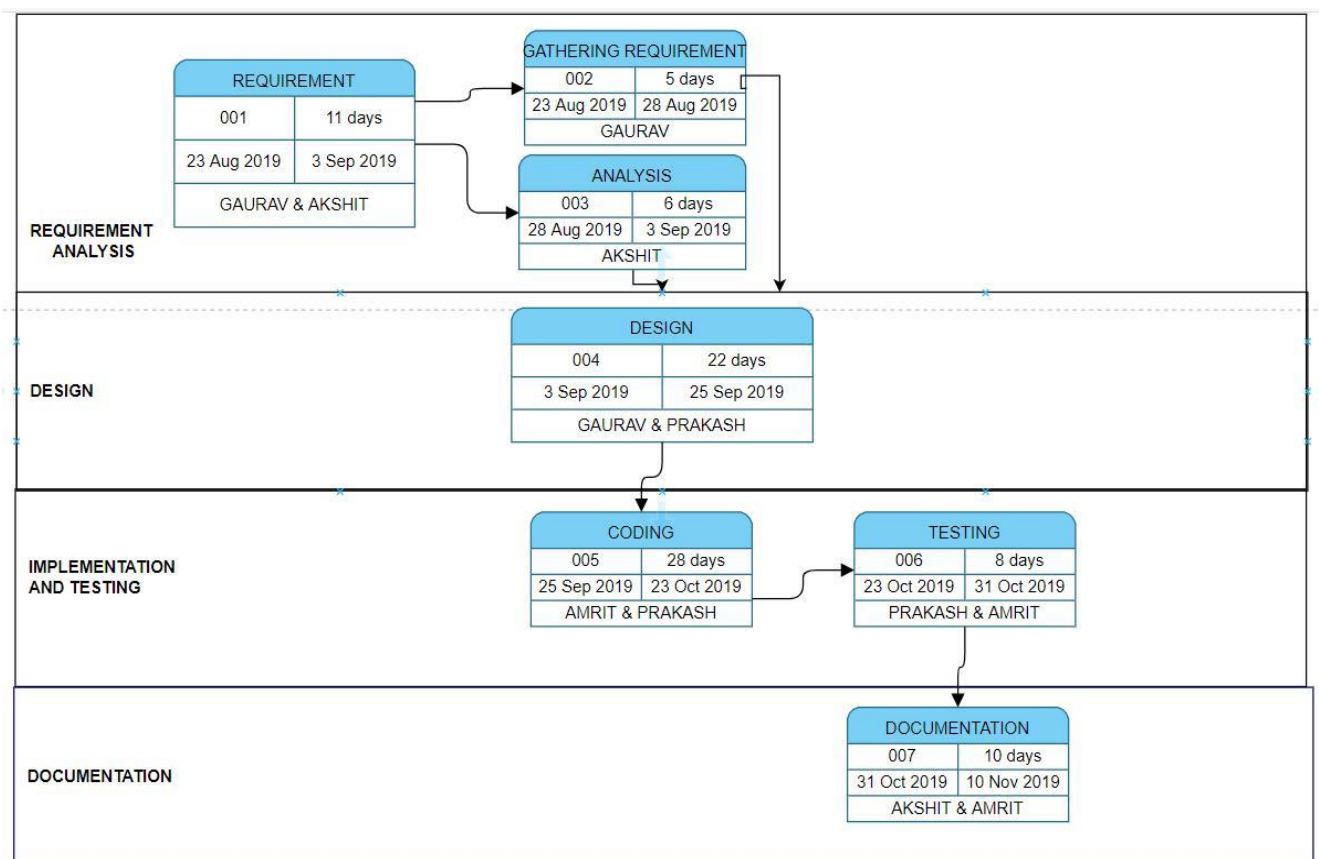


Fig 10.0



**13) References:**

1. M.Prettha,M.Nithya “A study and performance analysis of RSA Algorithm
2. <http://www.networksorcery.com/enp/data/encryption.htm/>
3. <https://www.geeksforgeeks.org/socket-programming-cc/>
4. <https://www.gatevidyalay.com/huffman-coding-huffman-encoding/>
5. [https://www.youtube.com/watch?v=IQ3S4fJ0U&list=PLPyaR5G9aNDvs6TdpLcVO43\\_jvxp4eml](https://www.youtube.com/watch?v=IQ3S4fJ0U&list=PLPyaR5G9aNDvs6TdpLcVO43_jvxp4eml)
6. <https://www.ijcsmc.com/docs/papers/June2013/V2I6201330.pdf>
7. [http://paper.ijcsns.org/07\\_book/201307/20130702.pdf](http://paper.ijcsns.org/07_book/201307/20130702.pdf)

**Approved By**

**Signature**

**Mr. Pushpendra Kumar Rajput  
Mentor**

**Signature**

**Dr. Monit Kapoor  
Head of Department**