MINOR PROJECT 1

MID SEM REPORT

ON

CrpytCom: Secure and Fast communication using encryption and compression technique

Submitted By

Prakash Tiwari	Amrit Kumar	Akshit Chauhan	Gaurav Singh
500062116	500062268	50006244 4	500062611

Under the guidance of

Pushpendra Kumar Rajput

Assistant Professor, SoCSE

Department of Cybernetics,
School of Computer Science
UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

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.

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 2nd 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.

5) Objective Achieved:

- Socket Connection is achieved.
- We have achieved compression using Huffman Encoding/Decoding algorithm.

6) Methodology:

6.1) Introduction

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

6.2) 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.

6.3) 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.

6.4) 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 partioning 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 analyse the performance of our project on the basis of test cases

7. Implementation

7.1) Socket Connectivity

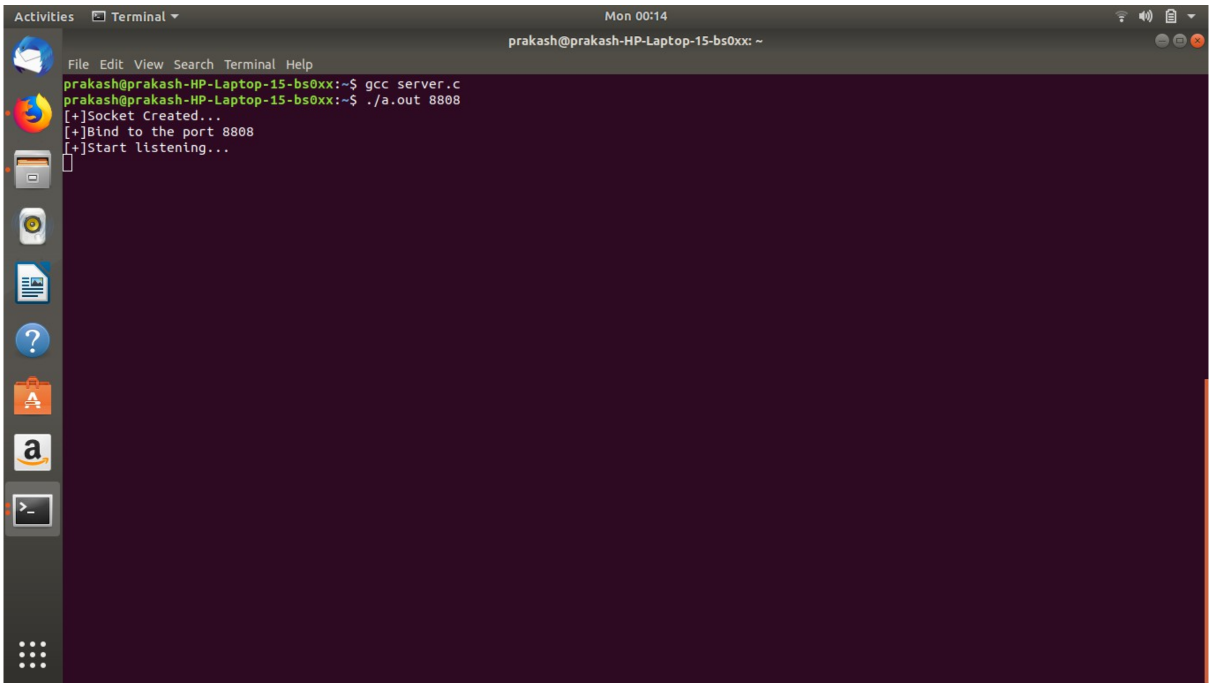
1. Server side: Server Create socket using socket(AF_INET, SOCK_STREAM, 0).
2. Bind socket to a specific port where clients can contact the server .
3. listen(int sockfd, int backlog); It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection.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

7.2) 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.

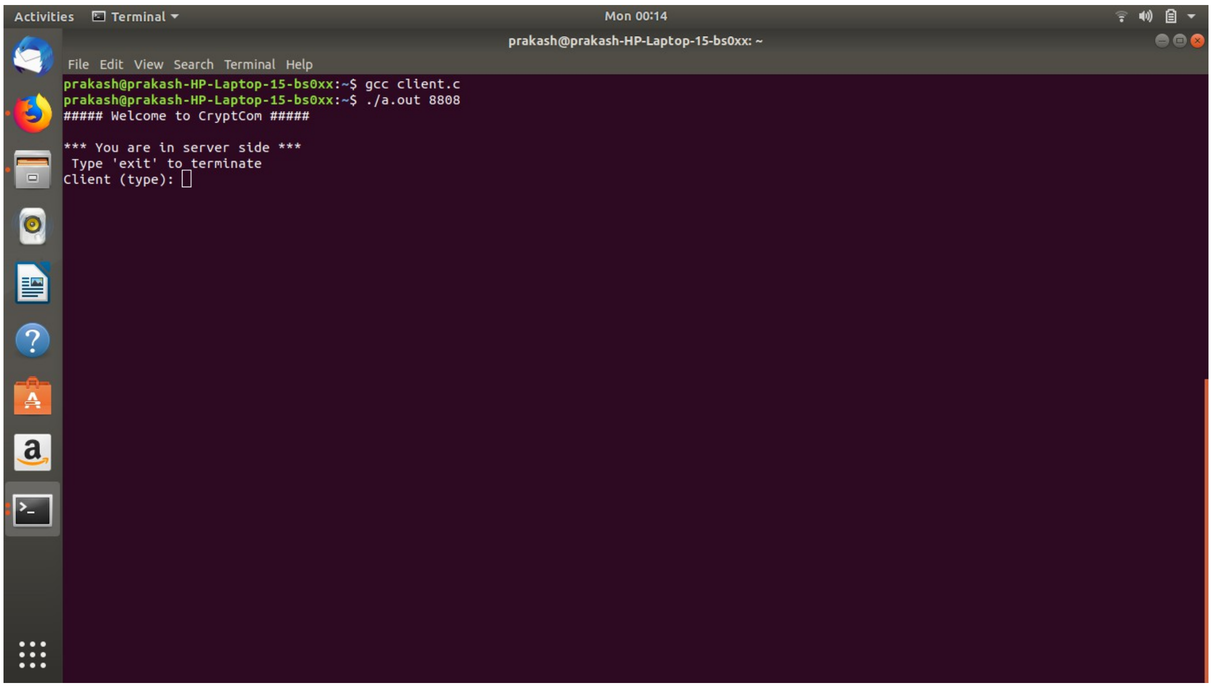
8) Output Screen:

8.1) Socket Connectivity:


Activities
Terminal
Mon 00:14
prakash@prakash-HP-Laptop-15-bs0xx: ~
File Edit View Search Terminal Help

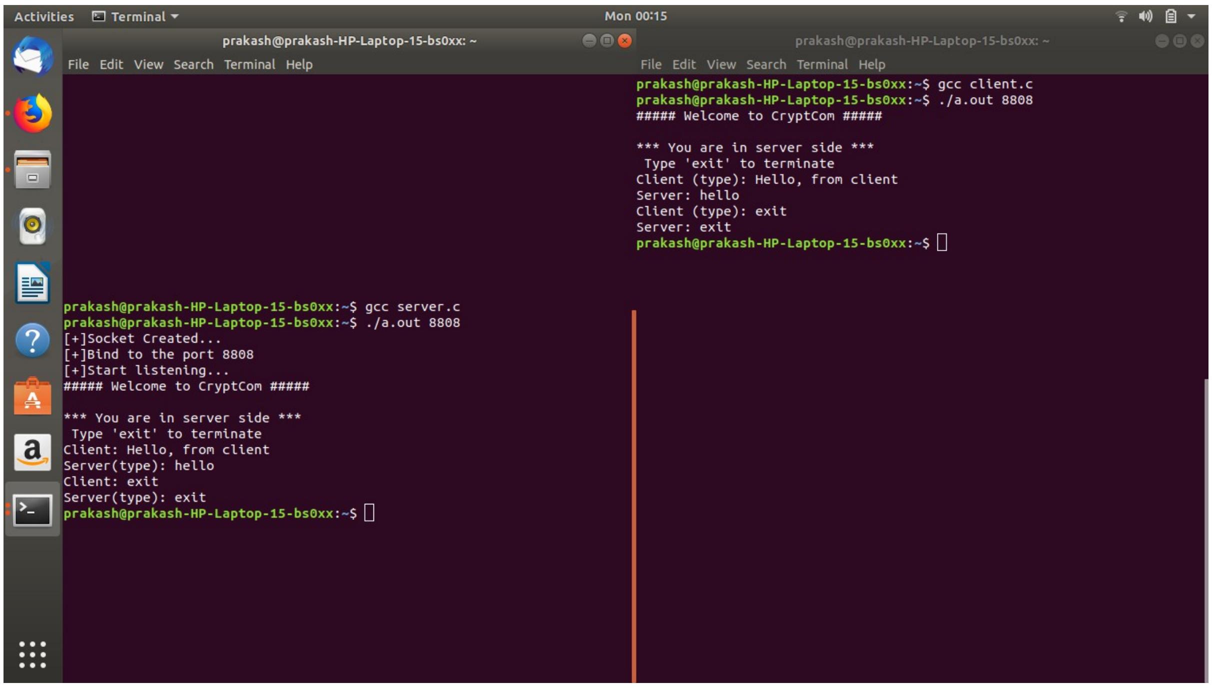
```
prakash@prakash-HP-Laptop-15-bs0xx:~$ gcc server.c
prakash@prakash-HP-Laptop-15-bs0xx:~$ ./a.out 8808
[+]Socket Created...
[+]Bind to the port 8808
[+]Start listening...
```

Fig 1.0


Activities
Terminal
Mon 00:14
prakash@prakash-HP-Laptop-15-bs0xx: ~
File Edit View Search Terminal Help

```
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

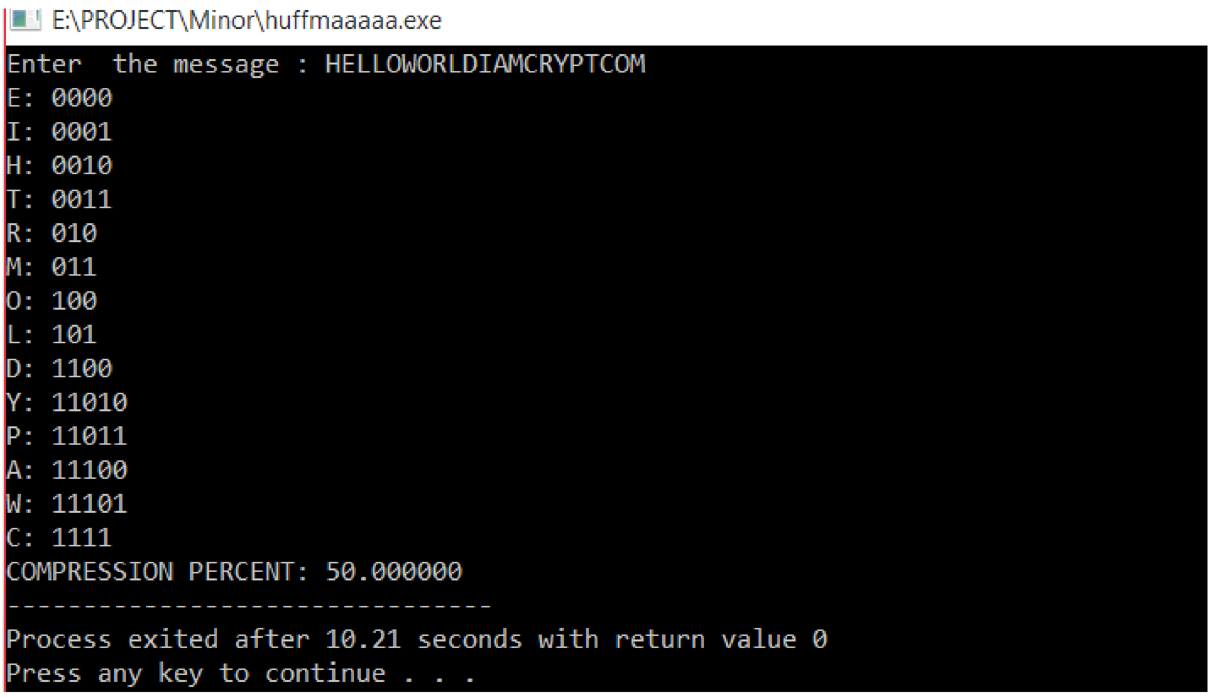

Activities
Terminal
Mon 00:15
prakash@prakash-HP-Laptop-15-bs0xx: ~
File Edit View Search Terminal Help
prakash@prakash-HP-Laptop-15-bs0xx: ~
File Edit View Search Terminal Help

```
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): Hello, from client
Server: hello
Client (type): exit
Server: exit
prakash@prakash-HP-Laptop-15-bs0xx:~$
```

```
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

8.2) 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

<u>Flowchart:</u>

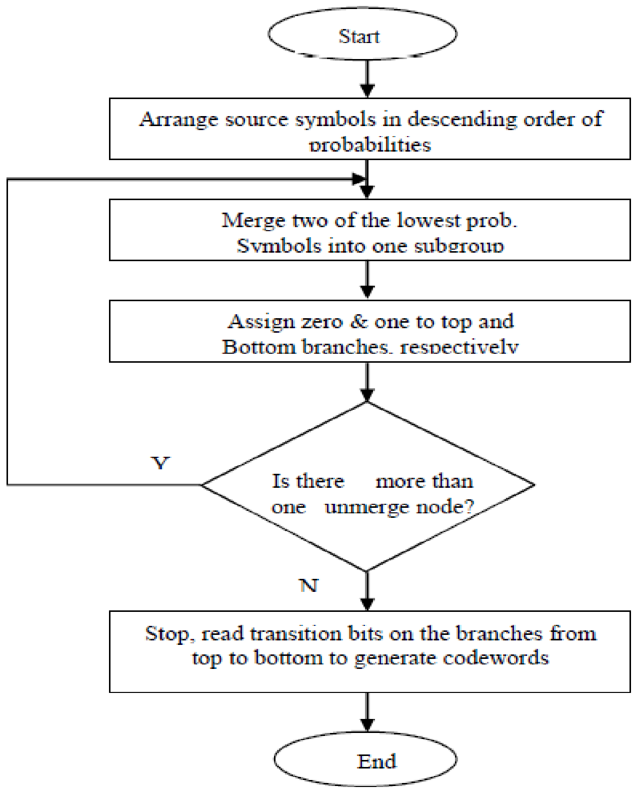

Start
Arrange source symbols in descending order of probabilities
Merge two of the lowest prob. Symbols into one subgroup
Assign zero & one to top and Bottom branches, respectively
Y
Is there more than one unmerge node?
N
Stop, read transition bits on the branches from top to bottom to generate codewords
End

Fig5.0

6) 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

7) Schedule (PERT Chart):

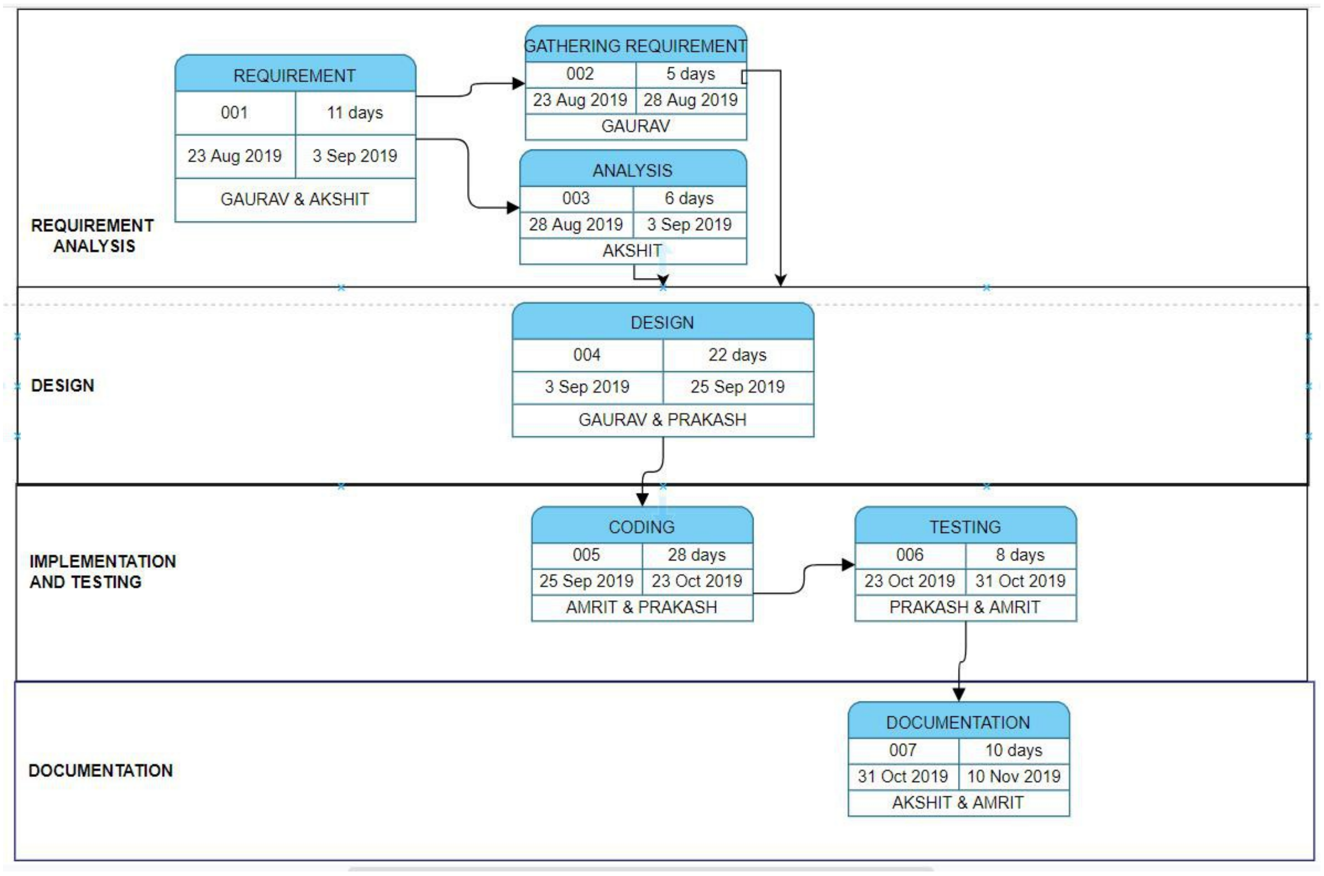


Fig 6.0

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=_lQ3S4fJ0U&list=PLPyaR5G9aNDvs6TtdpLcVO43_jvxp4emI***

Approved By

Signature	**Signature**
Mr. Pushpendra Kumar Rajput	**Dr. Monit Kapoor**
Mentor	**Head of Department**