JAMIA MILLIA ISLAMIA

COMPUTER NETWORK LAB FILE 2023-24

Submitted by:

Group 11

Md Kashiful Haque (21BCS029)

Bushra Shahzad (21BCS046)

Tabinda Hoda (21BCS055)

Ramsha Anwar (21BCS066)

Mohammad Rayyan Ahmad (21BCS078)

Submitted to:

Dr. Mohammad Amjad

Mr. Hannan Mansoor

INDEX

S. No	Lab Assignments	Date	Remarks	Signature
1.	Substitution Cipher	2nd Aug, 2023		
2.	Columnar Transposition Cipher	9th Aug, 2023		
3.	Play Fair Cipher	16th Aug, 2023		
4.	Vigenère Cipher	23rd Aug, 2023		
5.	TCP Implementation	6th Sept, 2023		
6.	UDP Implementation	20th Sept, 2023		
7.	Group Project *Features Implemented (changes recommended were also included)–* • Multiple clients can connect to the server using selectors • Encryption of the messages sent • Time stamp of the message • Bit-wise checking of message	18th Oct, 2023		
8.	Word Replacement *Features Implemented –* • Multi-clients are connecting to server and can send messages to each other through multi-threading • Abusive words checked in message • Log file is being maintained at the server's end • Search for a keyword logged in the log file • Replace any word with a new word in the log file	8th Nov, 2023		

Substitution Cipher

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
string encrypt(string text, int s)
{
    string result = "";
    for (int i = 0; i < text.length(); i++)
    {
        if (isupper(text[i]))
            result += char(int(text[i] + s - 65) % 26 + 65);
        else
            result += char(int(text[i] + s - 97) % 26 + 97);
    }
    return result;
}
string decrypt(string text, int s)
{
    string result = "";
    for (int i = 0; i < text.length(); i++)
    {
        if (isupper(text[i]))
            result += char(int(text[i] +26 - s - 65) % 26 + 65);
        else
            result += char(int(text[i] +26 - s - 97) % 26 + 97);
    }
    return result;
}
int main()
{
    string text;
    cout << "Enter the following text: ";
    cin >> text;
    int a;
    fflush(stdin);
    cout << "Enter the key: ";
    cin >> a;
    string p = encrypt(text, a);
    cout << "The Encrypted text: " << p << endl;
    cout << "The Decrypted text: " << decrypt(p, a) << endl;
    return 0;
}
```

```
PS C:\Users\DELL\OneDrive\Documents\Ramsha's Work\semester 5\comp_network> cd "c:\Users\DELL\OneDrive\Documents\Ramsha's Wo
rk\semester 5\comp_network\" ; if ($?) { g++ subst_cipher.cpp -o subst_cipher } ; if ($?) { .\subst_cipher }
Enter the following text: hello world
Enter the key: 2
The Encrypted text: jgnnq
The Decrypted text: hello
```

Transposition Columnar Cipher

```
#include<bits/stdc++.h>
#include <cstdlib>
using namespace std;
map<char,int>mp;
void keyOrder(string key)
{
    for(int i=0;i<key.length();i++)
    {
        mp[key[i]]=i;
    }
}
string encryption(string text,string key)
{
    string enc="";
    int keyLength=key.length();
    int textLength=text.length();
    int row,col;
    if(textLength%keyLength)
    {
        row=textLength/keyLength;
        row++;
    }
    else row=textLength/keyLength;
    col=keyLength;
    char matrix[row][col];
    int index=0;
    // encryption matrix
    for(int i=0;i<row;i++)
    {
        for(int j=0;j<col;j++)
        {
            if(text[index]=='\0')
            {
                matrix[i][j]='*';
                break;
            }
            else if((text[index]>=65 && text[index]<=90) || (text[index]>=97 &&
text[index]<=122)  || text[index]==' ')
            {
                matrix[i][j]=text[index];
                index++;
            }
        }
```

```
    }
    // print encrypted matrix
        for(int i=0;i<row;i++)
            {
            for(int j=0;j<col;j++)
            {
                cout<<matrix[i][j]<<" ";
            }
            cout<<"\n";
        }

    // encryting string
    for(auto &it:mp)
    {
        int x=it.second;
        for(int i=0;i<row;i++)
        {
            enc+=matrix[i][x];
        }
    }
    //cout<<"\n";
//  cout<<enc<<" ";
    return enc;
}
string decryption(string enc,string key)
{
    string dec="";
    int keyLength=key.length();
    int encLength=enc.length();
    int row2,col2;
    if(encLength%keyLength)
    {
        row2=encLength/keyLength;
        row2++;
    }
    else row2=encLength/keyLength;
    col2=keyLength;
    char matrix2[row2][col2];
    int index2=0;

    for(auto &it:mp)
    {
        int x=it.second;
        for(int i=0;i<row2;i++)
        {
```

```
            matrix2[i][x]=enc[index2];
            index2++;
        }
    }
        // print decrypted matrix
        for(int i=0;i<row2;i++)
            {
            for(int j=0;j<col2;j++)
            {
                cout<<matrix2[i][j]<<" ";
            }
            cout<<"\n";
        }

        //decryting string using map
            for(int i=0;i<row2;i++)
            {
                for(int j=0;j<col2;j++)
                {
                    if(matrix2[i][j]!='*')
                    dec+=matrix2[i][j];
                }
            }
            cout<<"\nDecrypted : ";
    cout<<dec<<" ";

    return dec;
}
int main()
{
    string str="";
    string key="";
    cout<<"Enter the text you want to encrypt : ";
    getline(cin,str);
    cout<<"Enter key : ";
    getline(cin,key);
    int counter=0;
    int choice;
    string str1="";
    string str2="";
    while(counter==0)
    {
        cout<<"\nEnter 1 to encrypt or 2 to decrypt and 3 to quit.\n";
        cin>>choice;
        keyOrder(key);
```

```
        switch(choice)
        {
            case 1:
                str1= encryption(str,key);
                cout<<"\nEncrypted text : ";
                cout<<str1;
                break;
            case 2:
                str2=decryption(str1,key);
                break;
            case 3:
                counter=1;
                break;
            default:
                cout<<"Invalid input\n";
        }
    }
    return 0;

}
```

```
PS C:\Users\DELL\OneDrive\Documents\Ramsha's Work\semester 5\comp_network> cd "c:\Users\DELL\OneDrive\Documents\Ramsha's Wo
rk\semester 5\comp_network\" ; if ($?) { g++ trans_column.cpp -o trans_column } ; if ($?) { .\trans_column }
Enter the text you want to encrypt : hello world
Enter key : hack

Enter 1 to encrypt or 2 to decrypt and 3 to quit.
1
h e l l
o   w o
r l d *

Encrypted text : e llwdhorlo*
Enter 1 to encrypt or 2 to decrypt and 3 to quit.
2
h e l l
o   w o
r l d *

Decrypted : hello world
Enter 1 to encrypt or 2 to decrypt and 3 to quit.
3
```

Play Fair Cipher

```
def create_matrix(key: str):
    unique_key = set(key.lower())
    if "j" in unique_key:
        unique_key.remove("j")
        unique_key.add("i")

    alpha = "abcdefghiklmnopqrstuvwxyz"
    mat_elements2 = [ch for ch in alpha if ch not in unique_key]
    mat_elements1 = []
    for ch in key:
        if ch in unique_key:
            mat_elements1.append(ch)
            unique_key.remove(ch)
    mat = [[0] * 5 for _ in range(5)]
    mat_elements = mat_elements1 + mat_elements2

    p = 0
    for row in range(5):
        for col in range(5):
            mat[row][col] = mat_elements[p]
            p += 1
    return mat

def create_pairs(input_string: str):
    if len(input_string) % 2:
        input_string += "x"

    pairs = []
    c1, c2 = "", ""
    for i in range(0, len(input_string) - 1, 2):
        c1, c2 = input_string[i], input_string[i + 1]
        if c1 != c2:
            pairs.append((c1, c2))
        else:
            pairs.append((c1, "x"))
    return pairs

def encrypt(input_string: str, matrix: list[list[str]]):
    pairs = create_pairs(input_string)
    j = 0
    output = ""
    for i in range(len(pairs)):
        c1, c2 = pairs[i]
```

```
        indices = {c1 : "", c2: ""}
        for row in range(5):
            for col in range(5):
                if matrix[row][col] == c1:
                    indices[c1] = (row, col)
                if matrix[row][col] == c2:
                    indices[c2] = (row, col)

        dec1 = matrix[indices[c1][0]][indices[c2][1]]
        output += dec1
        dec2 = matrix[indices[c2][0]][indices[c1][1]]
        if dec2 == "x":
            output += dec1
        else:
            output += dec2
    if dec2 == "x":
        return output[:-1]
    return output

flag = 1
while flag:
    print("Choose 1. encryption, 2. decryption, 3. exit")
    choice = int(input())
    print("Enter the key:")
    key = input()
    mat = create_matrix(key)

    if choice == 1:
        print(mat)
        print("Enter the input string")
        inp_s = input()
        print(encrypt(inp_s, mat))

    if choice == 2:
        print("Enter the input string")
        inp_s = input()
        print(encrypt(inp_s, mat))

    if (choice == 3):
        flag = 0
```

```
PS C:\Users\DELL\OneDrive\Documents\Ramsha's Work\semester 5\comp_network> python -u "c:\Users\DELL\OneDrive\Documents\Rams
ha's Work\semester 5\comp_network\play_fairr.py"
Choose 1. encryption, 2. decryption, 3. exit
1
Enter the key:
hack
[['h', 'a', 'c', 'k', 'b'], ['d', 'e', 'f', 'g', 'i'], ['l', 'm', 'n', 'o', 'p'], ['q', 'r', 's', 't', 'u'], ['v', 'w', 'x'
, 'y', 'z']]
Enter the input string
welcome
wenhmofw
Choose 1. encryption, 2. decryption, 3. exit
2
Enter the key:
hack
Enter the input string
wenhmofw
welcome
Choose 1. encryption, 2. decryption, 3. exit
3
```

Vigenère Cipher

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
void Table(char table[26][26])
{
    for (int i = 0; i < 26; i++)
    {
        for (int j = 0; j < 26; j++)
        {
            table[i][j] = (i + j) % 26 + 'A';
        }
    }
}
void Enycrypt(string &text, string key, char table[26][26])
{
    cout << "Given String:-  " << text << endl;
    for (int k = 0; k < text.size(); k++)
    {
        cout << k << " ";
        int i = text[k] - 'A', j = key[k] - 'A';
        if (text[k] >= 'A' && 'Z' >= text[k])
        {
            text[k] = table[i][j];
        }
    }
    cout << "Encrypted String:-  " << text << endl;
}
void Decrypt(string &text, string key, char table[26][26])
{
    cout << "Encrypted String:-  " << text << endl;

    for (int k = 0; k < text.size(); k++)
    {
        int j = key[k] - 'A';
        if (text[k] >= 'A' && 'Z' >= text[k])
        {
            for (int i = 0; i < 26; i++)
            {
                if (table[i][j] == text[k])
                {
                    text[k] = i + 'A';
                    break;
                }
```

```
            }
        }
    }
    cout << "Decrypted String:-  " << text << endl;
}
int main()
{
    string text;
    cout << "Enter the string:-  ";
    getline(cin, text);
    string key;
    cout << "Enter the key:-  ";
    cin >> key;
    string temp = key;
    while (key.size() < text.size())
        key += temp;
    bool flag = false;
    char table[26][26];
    Table(table);
    for(int i=0;i<text.size();i++){
        if(text[i]>='a'&&text[i]<='z'){
            text[i]=text[i]-'a'+'A';
        }
    }
    for(int i=0;i<key.size();i++){
        if(key[i]>='a'&&key[i]<='z'){
            key[i]=key[i]-'a'+'A';
        }
    }
    while (1)
    {
        int option;
        cout << "1. Encrypt\n2. Decrypt\n3. Exit\n";
        cin >> option;
        if (option == 1)
        {
            flag = true;
            Enycrypt(text, key, table);
        }
        else if (option == 2)
        {
            if (!flag)
            {
                cout << "Please Encrypt the string first\n";
                continue;
```

```
            }
            Decrypt(text, key, table);
            flag = false;
        }
        else if (option == 3)
        {
            break;
        }
        else
        {
            cout << "Invalid Option\n";
        }
    }
}
```

```
PS C:\Users\DELL\OneDrive\Documents\Ramsha's Work\semester 5\comp_network> cd "c:\Users\DELL\Downloads\" ; if ($?) { g++ Vi
genereCipher.cpp -o VigenereCipher } ; if ($?) { .\VigenereCipher }
Enter the string:-  hello world
Enter the key:-  hack
1. Encrypt
2. Decrypt
3. Exit
1
Given String:-  HELLO WORLD
0 1 2 3 4 5 6 7 8 9 10 Encrypted String:-  OENVV YYYLF
1. Encrypt
2. Decrypt
3. Exit
2
Encrypted String:-  OENVV YYYLF
Decrypted String:-  HELLO WORLD
1. Encrypt
2. Decrypt
3. Exit
3
```

TCP Protocol

Client –

```
import socket

HOST = "127.0.0.1"
PORT = 10977

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as clt:
    clt.connect((HOST, PORT))
    clt.sendall("hello".encode())
    data = clt.recv(1024)

print(f"Data status is {data.decode()}")
```

Server –

```
import socket

HOST = "127.0.0.1"
PORT = 10977
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as serv:
    #v4, TCP Protocol
    serv.bind((HOST, PORT))
    serv.listen()
    connection, address = serv.accept()
    with connection:
        print(f"Connected to the adress {address}")
        while True:
            data = connection.recv(1024) #max amount of data that can be received
            print(f"The data is - {data.decode()}")
            if not data:
                break
            connection.sendall("received".encode())
```

```
PS C:\Users\DELL\OneDrive\Documents\Ramsha's Work\semester 5\comp_network> python3 server.py
Connected to the adress ('127.0.0.1', 53669)
The data is - hello
The data is -
```

```
PS C:\Users\DELL\OneDrive\Documents\Ramsha's Work\semester 5\comp_network> python3 client.py
Data status is received
```

UDP Protocol

Client –

```
import socket

HOST = socket.gethostname()
PORT = 20987

with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as clt:
    data = "Sending data"
    clt.sendto(data.encode(), (HOST, PORT))
    data, addr = clt.recvfrom(1024)
    print(f"Data received is {data.decode()}")
```

Server –

```
import socket

HOST = socket.gethostname()
PORT = 20987

with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as serv:
    #UDP is Socket type
    serv.bind((HOST, PORT))
    data, addr = serv.recvfrom(1024) #connection request incoming from client
side
    print(f"Address is - {addr} and data is {data.decode()}")
    message = "Hello!".encode()
    serv.sendto(message, addr)
```

```
PS C:\Users\DELL\OneDrive\Documents\Ramsha's Work\semester 5\comp_network> python3 server_udp.py
Address is - ('192.168.56.1', 63189) and data is Sending data
```

```
PS C:\Users\DELL\OneDrive\Documents\Ramsha's Work\semester 5\comp_network> python client_udp.py
Data received is Hello!
```

Group Project- 1

Write a socket program both for client and server. The server will be able to send the pattern of 0's and 1's only. The client will count the number of 0's and 1's sent by the server.

Features Implemented (changes recommended were also included)–

- Multiple clients can connect to the server using selectors
- Encryption of the messages sent
- Time stamp of the message
- Bit-wise checking of message

Client –

```
import sys
import socket
import selectors
import types
import datetime
from cryptography.fernet import Fernet

sel = selectors.DefaultSelector()

def get_server_ip():
    server_host = input("Enter the server's hostname or IP address: ")
    try:
        server_ip = socket.gethostbyname(server_host)
        return server_ip
    except socket.gaierror:
        timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        print("[{timestamp}] Could not resolve the host. Please check the
hostname or IP address.")
        sys.exit(1)

def start_connection(server_ip, port):
    server_addr = (server_ip, port)
    connid = 1
    timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    print(f"[{timestamp}] Starting connection to {server_addr}")
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.setblocking(False)
    sock.connect_ex(server_addr)
    events = selectors.EVENT_READ | selectors.EVENT_WRITE
    data = types.SimpleNamespace(
        connid=connid,
        recv_total=0,
        zero_count=0,
```

```
        one_count=0,
        outb=b"",
        key=None,  # Add a key field to store the received key
    )
    sel.register(sock, events, data=data)

def service_connection(key, mask):
    sock = key.fileobj
    data = key.data

    if mask & selectors.EVENT_READ:
        if data.key is None:  # If the key is not received yet
            recv_key = sock.recv(1024)
            if recv_key:
                data.key = recv_key
                data.cipher_suite = Fernet(data.key)
                timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
                print(f"[{timestamp}] Received and set the key")

        else:  # If the key is received, proceed with data decryption
            recv_data = sock.recv(1024)
            if recv_data:
                print(f"Received data from connection {data.connid}:
{recv_data}")
                decrypted_data = data.cipher_suite.decrypt(recv_data)  # Decrypt
received data
                print(f"Received (decrypted) from connection {data.connid}:
{decrypted_data.decode()}")
                data.recv_total += len(decrypted_data)
                data.zero_count += decrypted_data.count(b'0')
                data.one_count += decrypted_data.count(b'1')
                if decrypted_data.strip() == b"end":
                    timestamp = datetime.datetime.now().strftime('%Y-%m-%d
%H:%M:%S')
                    print(f"[{timestamp}] Closing connection {data.connid}")
                    print(f"Total 0s received: {data.zero_count}")
                    print(f"Total 1s received: {data.one_count}")
                    sel.unregister(sock)
                    sock.close()

    if mask & selectors.EVENT_WRITE:
        pass  # No data to send in this scenario

if __name__ == "__main__":
    server_ip = get_server_ip()
```

```
    port = 12346

    start_connection(server_ip, port)

    while True:
        events = sel.select(timeout=None)
        for key, mask in events:
            service_connection(key, mask)
```

Server –

```
import sys
import socket
import selectors
import types
import datetime
from cryptography.fernet import Fernet

sel = selectors.DefaultSelector()

host, port = '127.0.0.1', 12346
lsock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
lsock.bind((host, port))
lsock.listen()
print(f"Listening on {(host, port)}")
lsock.setblocking(False)
sel.register(lsock, selectors.EVENT_READ, data=None)

# Define a set to keep track of connected clients
connected_clients = set()
max_clients = 5

def accept_wrapper(sock):
    conn, addr = sock.accept()
    if len(connected_clients) >= max_clients:
        timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        print(f"[{timestamp}] Rejecting connection from {addr} - Maximum clients
reached")
        conn.close()
        return

    if addr[0] in connected_clients:
        timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        print(f"[{timestamp}] Rejecting connection from {addr} - Client already
connected from this IP")
```

```
        conn.close()
        return
    timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    print(f"[{timestamp}] Accepted connection from {addr}")
    key = Fernet.generate_key()
    cipher_suite = Fernet(key)
    print("URL-safe base64-encoded key:", key.decode())
    conn.send(key)
    conn.setblocking(False)
    data = types.SimpleNamespace(addr=addr, inb=b"", outb=b"")
    events = selectors.EVENT_READ | selectors.EVENT_WRITE
    sel.register(conn, events, data=data)
    connected_clients.add(addr[0])
    handle_user_input(conn, addr,cipher_suite)

def handle_user_input(conn, addr, cipher_suite):
    while True:
        user_input = input("Enter 0 or 1 or 'end' to terminate: ")
        if user_input not in ('0', '1', 'end'):
            print("Invalid input. Please enter '0', '1', or 'end'.")
            continue

        if user_input == 'end':
            encrypted_data = cipher_suite.encrypt(user_input.encode())
            conn.send(encrypted_data)
            timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
            print(f"[{timestamp}] Closing connection to {addr}")
            sel.unregister(conn)
            conn.close()
            connected_clients.remove(addr[0])
            break
        else:
            encrypted_data = cipher_suite.encrypt(user_input.encode())
            conn.send(encrypted_data)

def service_connection(key, mask):
    cl_socket = key.fileobj
    data = key.data

    if mask & selectors.EVENT_READ:
        recv_data = cl_socket.recv(1024)
        if recv_data:
            decrypted_data = cipher_suite.decrypt(recv_data)  # Decrypt received
data
            decrypted_data = decrypted_data.decode()
```

```
            if decrypted_data == 'end':
                timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
                print(f"[{timestamp}] Closing connection to {data.addr}")
                sel.unregister(cl_socket)
                cl_socket.close()
                connected_clients.remove(data.addr[0])
            elif decrypted_data in ('0', '1'):
                print(f"Received (decrypted) from {data.addr}: {decrypted_data}")
            else:
                print("Invalid data received. Discarding the packet.")
                # Discard the invalid data and don't transmit to the client

    if mask & selectors.EVENT_WRITE:
        if data.outb:
            next_char = data.outb[:1]
            timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
            print(f"[{timestamp}] Sending {next_char} to {data.addr}")
            encrypted_data = cipher_suite.encrypt(next_char)  # Encrypt data
before sending
            sent = cl_socket.send(encrypted_data)
            data.outb = data.outb[1:]

try:
    while True:
        events = sel.select(timeout=None)
        for key, mask in events:
            if key.data is None:
                accept_wrapper(key.fileobj)
            else:
                service_connection(key, mask)
except KeyboardInterrupt:
    print("Caught keyboard interrupt, exiting")
finally:
    sel.close()
```

```
PS C:\Users\DELL\OneDrive\Documents\Ramsha's Work\semester 5\comp_network> python3 server3.py
Listening on ('127.0.0.1', 12346)
[2023-11-22 10:00:43] Accepted connection from ('127.0.0.1', 53970)
URL-safe base64-encoded key: ZFwfESY1KBwPcWQEIIi4SJVSdrzxdFVrXk0ZRdUpcAo=
Enter 0 or 1 or 'end' to terminate: 0
Enter 0 or 1 or 'end' to terminate: 1
Enter 0 or 1 or 'end' to terminate: 0
Enter 0 or 1 or 'end' to terminate: 0
Enter 0 or 1 or 'end' to terminate: 1
Enter 0 or 1 or 'end' to terminate: 1
Enter 0 or 1 or 'end' to terminate: end
[2023-11-22 10:01:06] Closing connection to ('127.0.0.1', 53970)
```

```
PS C:\Users\DELL\OneDrive\Documents\Ramsha's Work\semester 5\comp_network> python3 client3.py
Enter the server's hostname or IP address: 127.0.0.1
[2023-11-22 10:00:43] Starting connection to ('127.0.0.1', 12346)
[2023-11-22 10:00:43] Received and set the key
Received data from connection 1: b'gAAAAABlXYQBTa_Y1FAqrEH5VkBaLT2aoHkJMgtOWnwUY7dKEm065qkPvH-1A6FV6kgaOMwHf-gEqddP0jyWe_13
ueNb5pEEBA=='
Received (decrypted) from connection 1: 0
Received data from connection 1: b'gAAAAABlXYQD2rfD9T5tvBw-1YcUP4cWSr6d1svIf7-ue3vJBuQrmlEzQ0ZrvQBTMXkN3lC-O7DGozv0qkiI9nPk
fWWiHmLEuQ=='
Received (decrypted) from connection 1: 1
Received data from connection 1: b'gAAAAABlXYQFwiLjo3awQ96UpOBAdA2hOyQciPLNtJnlpodJsw8rR46brYuR2owk85u7H5fF5-HWSPqsM1cEP03b
FK9j3C4QMA=='
Received (decrypted) from connection 1: 0
Received data from connection 1: b'gAAAAABlXYQHFO2zCmEvonr88bnrq8ZHypL8VuC0quCoiy-UieEhelZSy8Pey3JkugQPI50uYc_SIgVQVvTif55P
mj0ioBLxzg=='
Received (decrypted) from connection 1: 0
Received data from connection 1: b'gAAAAABlXYQHohjNA2EpGvpV_yGe4IPPxVxBjnkuNvaO1qrb7f96x6Ie1f8wiJuP9zYCXdq9vvmL4rrBZCUYNkx-
LVWVgrS3bA=='
Received (decrypted) from connection 1: 1
Received data from connection 1: b'gAAAAABlXYQICBoeRu_6nslL3QHrskTM1c5N5lEEIgxCQ1nJVtv_L2KCA-EVnvUyHZnI7uAYyM8r7O1DMJ_2smMU
7de9qKqm5w=='
Received (decrypted) from connection 1: 1
Received data from connection 1: b'gAAAAABlXYQKr5iDiy8EnxYSfvppCg3dttyL5BzEs7Vf25-9XjJPD4s5mBOhFjmwH50__TvdL7OYzBRnaOjVR1vi
X-lDIJoPLw=='
Received (decrypted) from connection 1: end
[2023-11-22 10:01:06] Closing connection 1
Total 0s received: 3
Total 1s received: 3
```

Word Replacement

Features Implemented –

- Multi-clients are connecting to server and can send messages to each other through multi-threading
- Abusive words checked in message
- Log file is being maintained at the server's end
- Search for a keyword logged in the log file
- Replace any word with a new word in the log file

Server-

```
import threading
import socket
import re

host = '127.0.0.1'
port = 59001
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((host, port))
server.listen()

clients = []
aliases = []
client_details = {}

# File for storing chats
chat_log_file = "chat_log.txt"
abusive_words = ["abuse1", "abuse2"]  # Add your abusive words to this list


# Function to handle clients' connections
def handle_client(client, alias):
    while True:
        try:
            message = client.recv(1024).decode('utf-8')

            # Check for abusive words
            if any(word in message for word in abusive_words):
                client.send("Warning: Your message contains abusive
words.".encode('utf-8'))
                continue

            # Log the chat to the file with client details
            with open(chat_log_file, "a") as log:
                log.write(f"{alias} ({client_details[alias]}): {message}\n")
```

```
            # Implement direct message functionality
            if message.startswith("DM"):
                parts = message.split(" ", 2)
                if len(parts) == 3:
                    recipient_alias = parts[1]
                    dm_message = parts[2]
                    send_direct_message(alias, recipient_alias, dm_message)
            elif message.startswith("RequestAliases"):
                send_aliases(client)
            elif message.startswith("Search"):
                keyword = message.split(" ", 1)[1]
                search_and_send_results(client, keyword)
            elif message.startswith("Replace"):
                parts = message.split(" ", 2)
                if len(parts) == 3:
                    old_word, new_word = parts[1], parts[2]
                    replace_and_broadcast(alias, old_word, new_word)


        except Exception as e:
            print(f'Error: {e}')
            index = clients.index(client)
            clients.remove(client)
            client.close()
            alias = aliases[index]
            aliases.remove(alias)
            del client_details[alias]
            break

# Function to send a direct message to a specific client
def send_direct_message(sender_alias, recipient_alias, message):
    for client, client_alias in zip(clients, aliases):
        if client_alias == recipient_alias:
            try:
                client.send(f'DM from {sender_alias}: {message}'.encode('utf-8'))
            except:
                print(f"Failed to send a direct message to {recipient_alias}")

# Function to handle requests for aliases
def send_aliases(client):
    aliases_msg = ", ".join(aliases)
    client.send(f'Connected aliases: {aliases_msg}'.encode('utf-8'))

# Function to handle search requests
```

```
def search_and_send_results(client, keyword):
    matches = []
    with open(chat_log_file, "r") as log:
        for line in log:
            if keyword in line:
                matches.append(line)
    if matches:
        client.send(f'Search Results:\n{" ".join(matches)}'.encode('utf-8'))
    else:
        client.send('No matches found.'.encode('utf-8'))

# Function to handle replace requests
def replace_and_broadcast(sender_alias, old_word, new_word):
    with open(chat_log_file, "r") as log:
        lines = log.readlines()

    with open(chat_log_file, "w") as log:
        for line in lines:
            new_line = re.sub(r'\b%s\b' % old_word, new_word, line)
            log.write(new_line)

    # # Broadcast the replacement to all clients
    # broadcast(f'{sender_alias} replaced "{old_word}" with
"{new_word}"'.encode('utf-8'))

# Main function to receive the clients' connection
def receive():
    while True:
        print('Server is running and listening ...')
        client, address = server.accept()
        print(f'Connection is established with {str(address)}')
        client.send('alias?'.encode('utf-8'))
        alias = client.recv(1024).decode('utf-8')
        aliases.append(alias)
        clients.append(client)
        client_details[alias] = address  # Store client IP and port
        print(f'The alias of this client is {alias} ({address})'.encode('utf-8'))
        client.send('You are now connected!'.encode('utf-8'))
        thread = threading.Thread(target=handle_client, args=(client, alias))
        thread.start()

if __name__ == "__main__":
    receive()
```

Client –

```
import threading
import socket

alias = input('Choose an alias >>> ')
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('127.0.0.1', 59001))
print("Commands:\n1. DM <recipient_alias> <message>\n2. RequestAliases\n3. Search
<keyword>\n4. Replace <old_word> <new_word>\n")

def client_receive():
    while True:
        try:
            message = client.recv(1024).decode('utf-8')
            print(message)
        except:
            print('Error!')
            client.close()
            break

def client_send():
    while True:
        message = input("")
        client.send(message.encode('utf-8'))

receive_thread = threading.Thread(target=client_receive)
receive_thread.start()

send_thread = threading.Thread(target=client_send)
send_thread.start()
```

Client's end –

(ramsha)

```
PS C:\Users\DELL\OneDrive\Documents\Ramsha's Work\semester 5\comp_network> python -u "c:\Users\DELL\Downloads\word_replace_clt.py"
Choose an alias >>> ramsha
Commands:
1. DM <recipient_alias> <message>
2. RequestAliases
3. Search <keyword>
4. Replace <old_word> <new_word>

alias?
ramsha
You are now connected!
RequestAliases
Connected aliases: ramsha, rayyan
RequestAliases
Connected aliases: ramsha, rayyan, kashif
DM kashif hi
DM rayyan hello
DM from kashif: hello
DM from rayyan: hey
Search hey
Search Results:
rayyan (('192.168.120.27', 54431)): DMDDM ramsha hey
 ramsha (('192.168.120.1', 55266)): Search hey

Replace hey whatsup
DM kashif abuse1
Warning: Your message contains abusive words.
```

(rayyan)

(kashif)

```
PS C:\Users\HP\OneDrive\Desktop\network_project> python -u "c:\Users\HP\OneDrive\Desktop\network_project\
Choose an alias >>> kashif
Commands:
1. DM <recipient_alias> <message>
2. RequestAliases
3. Search <keyword>
4. Replace <old_word> <new_word>

alias?
kashif
You are now connected!
DM from ramsha: hi
DM ramsha hello
DM rayyan Hey
RequestAliases
Connected aliases: ramsha, rayyan, kashif
Error!
```

Server's end –

```
PS F:\Whioo\Sem V\CN Lab\Word_replacement> python3 server.py
Server is running and listening ...
Connection is established with ('192.168.120.1', 55266)
b"The alias of this client is ramsha (('192.168.120.1', 55266))"
Server is running and listening ...
Connection is established with ('192.168.120.27', 54431)
b"The alias of this client is rayyan (('192.168.120.27', 54431))"
Server is running and listening ...
Connection is established with ('192.168.120.10', 52916)
b"The alias of this client is kashif (('192.168.120.10', 52916))"
Server is running and listening ...
```

Log file at server's end –

```
ramsha (('192.168.120.1', 55266)): RequestAliases
ramsha (('192.168.120.1', 55266)): RequestAliases
ramsha (('192.168.120.1', 55266)): DM kashif hi
ramsha (('192.168.120.1', 55266)): DM rayyan hello
kashif (('192.168.120.10', 52916)): DM ramsha hello
kashif (('192.168.120.10', 52916)): DM rayyan Hey
rayyan (('192.168.120.27', 54431)): DMDDM ramsha whatsup
ramsha (('192.168.120.1', 55266)): Search whatsup
ramsha (('192.168.120.1', 55266)): Replace whatsup whatsup
kashif (('192.168.120.10', 52916)): RequestAliases
```