Assignment 2

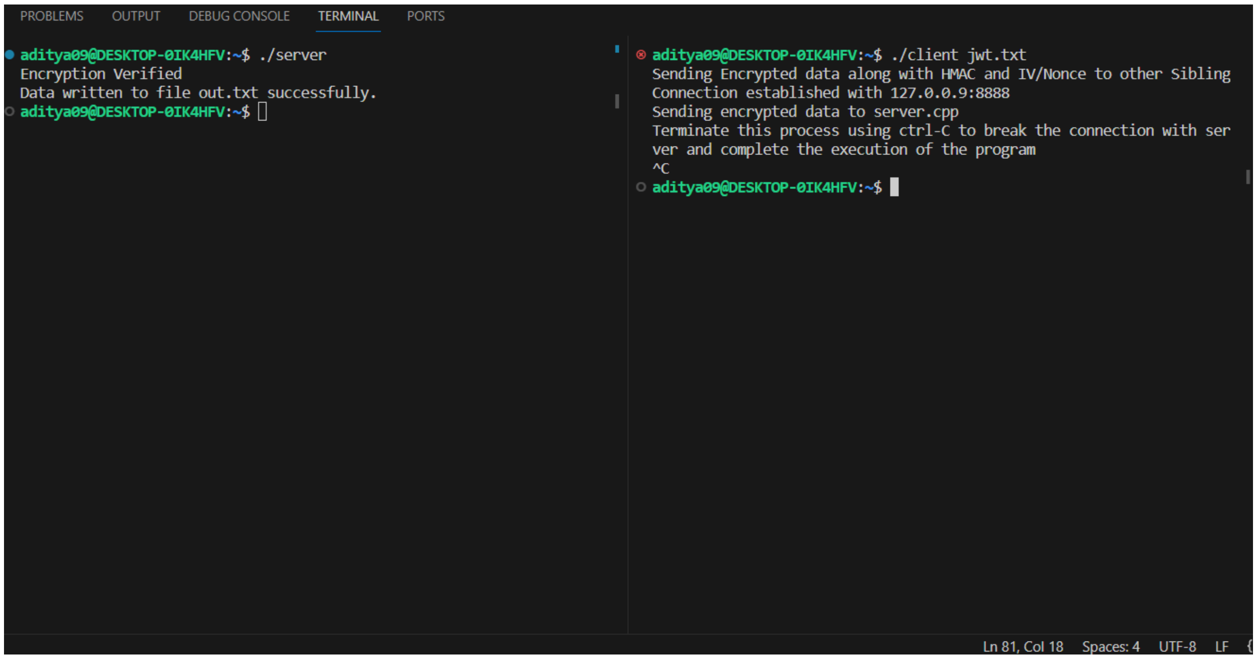
Q-1:

System Design:
Client and server forks two processes handled by two functions(func1, func2).

For server:
func1(): Listens on IP 127.0.0.9 and port 8888 using netcat command, and once it receives data from the client it opens a connection with another child process of the server using sockets.
func2(): Listens on IP 127.0.0.5 and port 8080 using sockets API, and once it receives data from the other child process of the server, it decrypts the data and checks for HMAC validation. Once HMAC is validated, it writes the data in out.txt.

For Client:
func1(): Reads the file(provided as an argument), and then encrypts and generates the HMAC of the unencrypted data. Then, it opens a connection with another child process of the client using sockets API, and sends the encrypted packet to it.
func2(): Listens on IP 127.0.0.1 and port 8080 using sockets, and once it receives data from the other process, it opens a new connection with the server using netcat and sends the encrypted TCP packet payload to the server.

Running of programs:

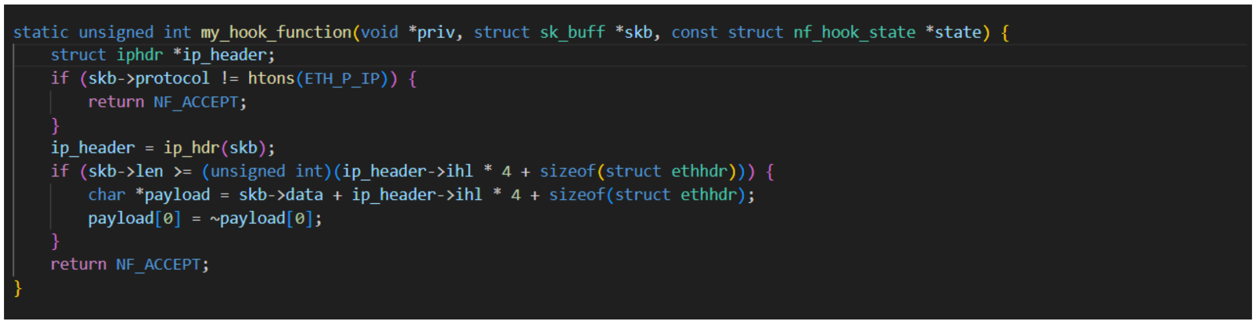


```
aditya09@DESKTOP-0IK4HFV:~$ ./server
Encryption Verified
Data written to file out.txt successfully.
aditya09@DESKTOP-0IK4HFV:~$
```

```
aditya09@DESKTOP-0IK4HFV:~$ ./client jwt.txt
Sending Encrypted data along with HMAC and IV/Nonce to other Sibling
Connection established with 127.0.0.9:8888
Sending encrypted data to server.cpp
Terminate this process using ctrl-C to break the connection with ser
ver and complete the execution of the program
^C
aditya09@DESKTOP-0IK4HFV:~$
```

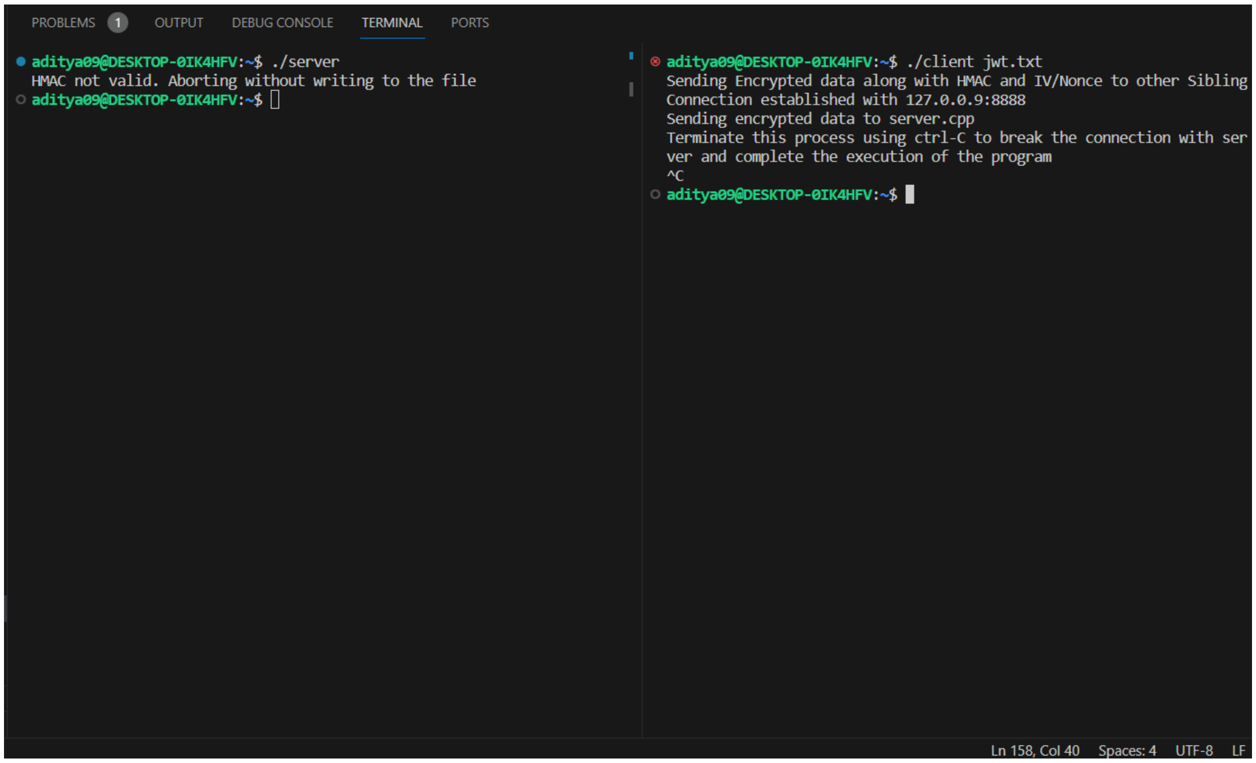
Assumptions:
1) Client process is manually terminated to complete the execution of the program as once netcat connection is opened it has to be closed manually.
2) Name of a file is passed as an argument to the client executable.
3) keyGenerate.cpp is executed before anything to produce symmetric keys for encryption-decryption.

HMAC Correct Validation:
Used netfilter module to write a hook function to intercept and modify the packets.
Hook function:



```c
static unsigned int my_hook_function(void *priv, struct sk_buff *skb, const struct nf_hook_state *state) {
    struct iphdr *ip_header;
    if (skb->protocol != htons(ETH_P_IP)) {
        return NF_ACCEPT;
    }
    ip_header = ip_hdr(skb);
    if (skb->len >= (unsigned int)(ip_header->ihl * 4 + sizeof(struct ethhdr))) {
        char *payload = skb->data + ip_header->ihl * 4 + sizeof(struct ethhdr);
        payload[0] = ~payload[0];
    }
    return NF_ACCEPT;
}
```

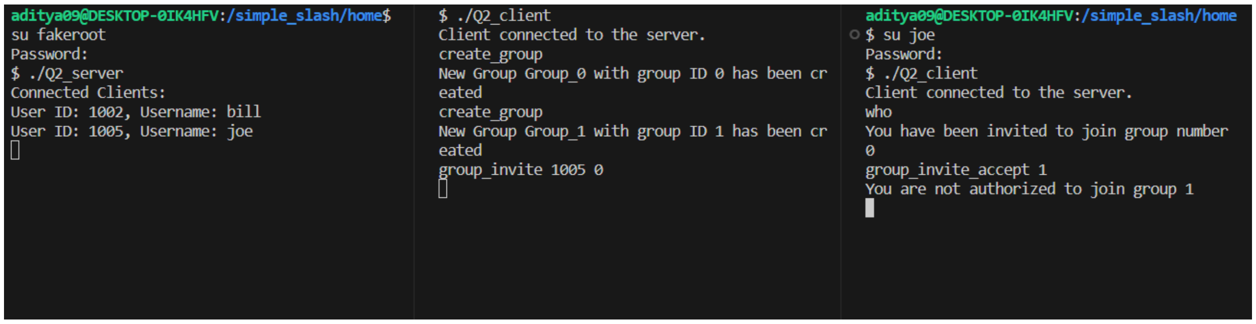
Result of inverted encrypted bit:



```
aditya09@DESKTOP-0IK4HFV:~$ ./server
HMAC not valid. Aborting without writing to the file
aditya09@DESKTOP-0IK4HFV:~$
```

```
aditya09@DESKTOP-0IK4HFV:~$ ./client jwt.txt
Sending Encrypted data along with HMAC and IV/Nonce to other Sibling
Connection established with 127.0.0.9:8888
Sending encrypted data to server.cpp
Terminate this process using ctrl-C to break the connection with ser
ver and complete the execution of the program
^C
aditya09@DESKTOP-0IK4HFV:~$
```

Q-2:
Vulnerabilities Protected:
1) Restricts the user to join groups for which he/she has not received any group invite. Since the Group ID of the groups can be guessed by an adversary, the program takes care that a user only joins a group for which he/she is authorized to join.

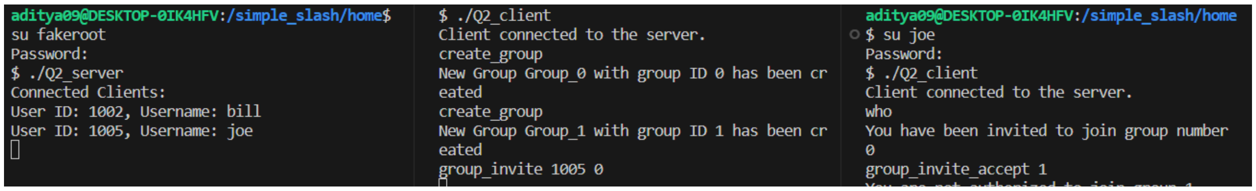


```
aditya09@DESKTOP-0IK4HFV:/simple_slash/home$
su fakeroot
Password:
$ ./Q2_server
Connected Clients:
User ID: 1002, Username: bill
User ID: 1005, Username: joe
```

```
$ ./Q2_client
Client connected to the server.
create_group
New Group Group_0 with group ID 0 has been cr
eated
create_group
New Group Group_1 with group ID 1 has been cr
eated
group_invite 1005 0
```

```
aditya09@DESKTOP-0IK4HFV:/simple_slash/home
$ su joe
Password:
$ ./Q2_client
Client connected to the server.
who
You have been invited to join group number
0
group_invite_accept 1
You are not authorized to join group 1
```

2) Does not give any information about the formed groups to other users who are not part of it. Example- A user "steve" will not know who all have formed groups before he formed any group.



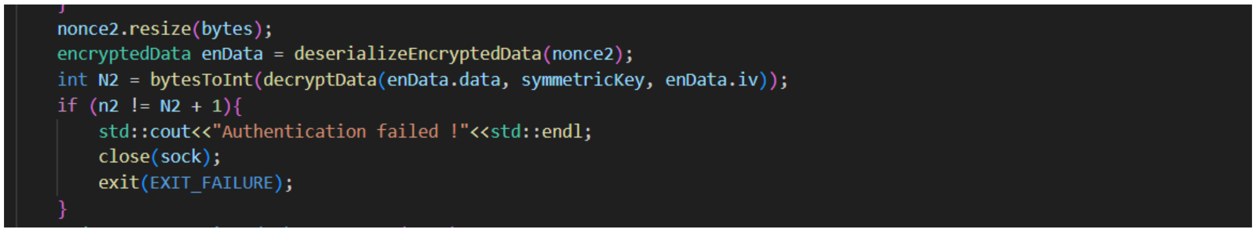
```
aditya09@DESKTOP-0IK4HFV:/simple_slash/home$
su fakeroot
Password:
$ ./Q2_server
Connected Clients:
User ID: 1002, Username: bill
User ID: 1005, Username: joe
```

```
$ ./Q2_client
Client connected to the server.
create_group
New Group Group_0 with group ID 0 has been cr
eated
create_group
New Group Group_1 with group ID 1 has been cr
eated
group_invite 1005 0
```

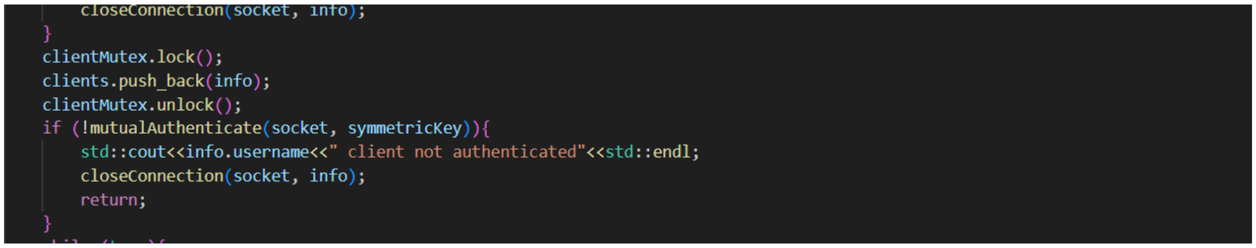
```
aditya09@DESKTOP-0IK4HFV:/simple_slash/home
$ su joe
Password:
$ ./Q2_client
Client connected to the server.
who
You have been invited to join group number
0
group_invite_accept 1
```

In the above example Joe has no information who has made groups before him and who all are part of it.

3) Uses Needham-Schroeder mutual authentication model to authenticate a new client as well as server whenever he tries to open a connection with a server using nonce based schemes.



```
    nonce2.resize(bytes);
    encryptedData enData = deserializeEncryptedData(nonce2);
    int N2 = bytesToInt(decryptData(enData.data, symmetricKey, enData.iv));
    if (n2 != N2 + 1){
        std::cout<<"Authentication failed !"<<std::endl;
        close(sock);
        exit(EXIT_FAILURE);
    }
```

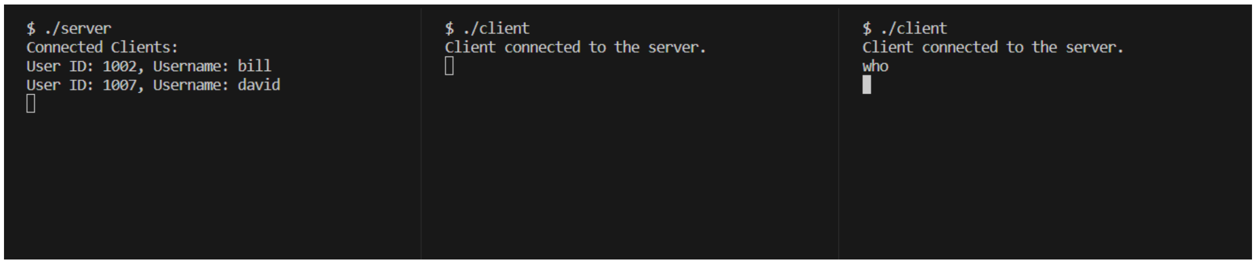


```
        closeConnection(socket, info);
    }
    clientMutex.lock();
    clients.push_back(info);
    clientMutex.unlock();
    if (!mutualAuthenticate(socket, symmetricKey)){
        std::cout<<info.username<<" client not authenticated"<<std::endl;
        closeConnection(socket, info);
        return;
    }
```

System Design:

1) “who” function can let the users see all the users that are currently logged in to the server. It prints all the userIds and usernames of the connected clients.

Syntax of command: **who**



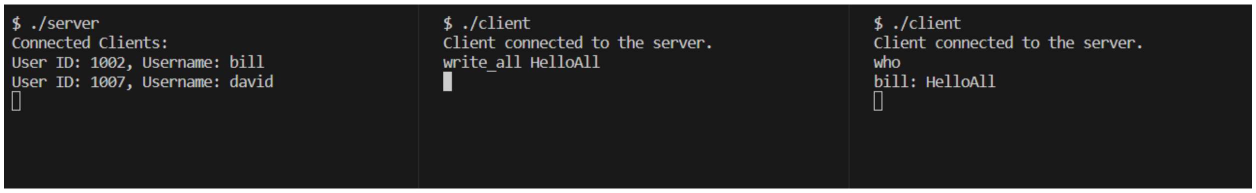
```
$ ./server
Connected Clients:
User ID: 1002, Username: bill
User ID: 1007, Username: david
```

```
$ ./client
Client connected to the server.
```

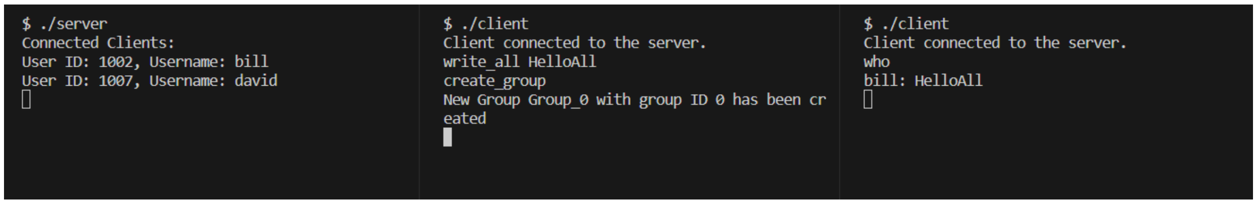
```
$ ./client
Client connected to the server.
who
```

2) “write_all” function can let the users broadcast their messages to all the users that are currently logged in.

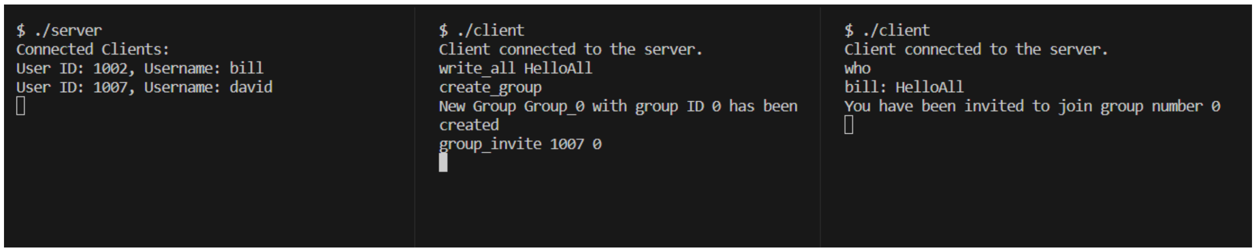
Syntax of the command: **write_all message**


```
$ ./server
Connected Clients:
User ID: 1002, Username: bill
User ID: 1007, Username: david
```
```
$ ./client
Client connected to the server.
write_all HelloAll
```
```
$ ./client
Client connected to the server.
who
bill: HelloAll
```

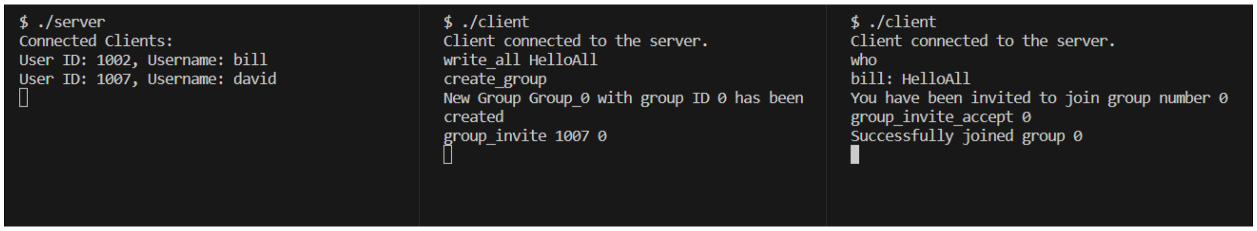
3) "create_group" function can let the users create a new group, the function returns a name and group ID to the user for future reference.
Syntax of the command: **create_group**


```
$ ./server
Connected Clients:
User ID: 1002, Username: bill
User ID: 1007, Username: david
```
```
$ ./client
Client connected to the server.
write_all HelloAll
create_group
New Group Group_0 with group ID 0 has been cr
eated
```
```
$ ./client
Client connected to the server.
who
bill: HelloAll
```

4) "group_invite" function can let the users invite new users to join a created group, the function sends a request to any particular user to join the group.
Syntax of the command: **group_invite userID groupID**
where userID is the uid of a user logged in and groupID is the group ID of the group for which request is sent.


```
$ ./server
Connected Clients:
User ID: 1002, Username: bill
User ID: 1007, Username: david
```
```
$ ./client
Client connected to the server.
write_all HelloAll
create_group
New Group Group_0 with group ID 0 has been
created
group_invite 1007 0
```
```
$ ./client
Client connected to the server.
who
bill: HelloAll
You have been invited to join group number 0
```

5) "group_invite_accept" function lets the user who has been invited to a group join the group of users. This function only allows the users to join groups but does not guarantee initiation of group messages before a shared key is established.
Syntax of the command: **group_invite_accept groupID**
where groupID refers to the ID of the group for which the user has received an invitation.

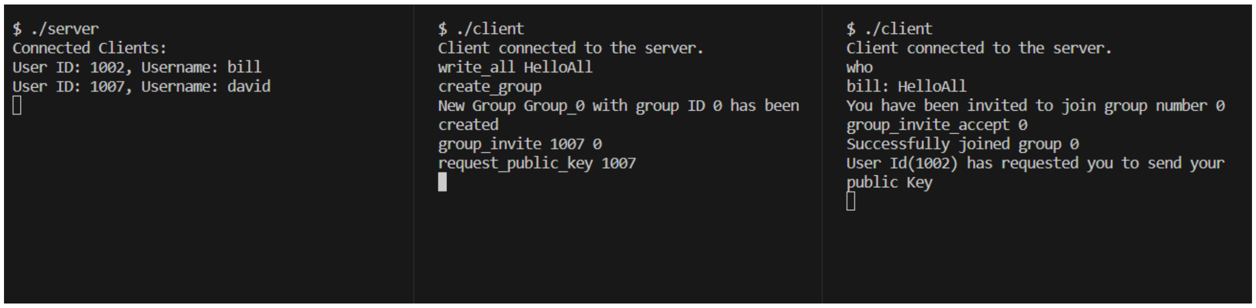


```
$ ./server
Connected Clients:
User ID: 1002, Username: bill
User ID: 1007, Username: david
```

```
$ ./client
Client connected to the server.
write_all HelloAll
create_group
New Group Group_0 with group ID 0 has been
created
group_invite 1007 0
```

```
$ ./client
Client connected to the server.
who
bill: HelloAll
You have been invited to join group number 0
group_invite_accept 0
Successfully joined group 0
```

6) “request_public_key” function allows a user to request another user to send him his public key.
Syntax of the command: **request_public_key userID**
where userID refers to the uid of the user to whom request is to be sent.

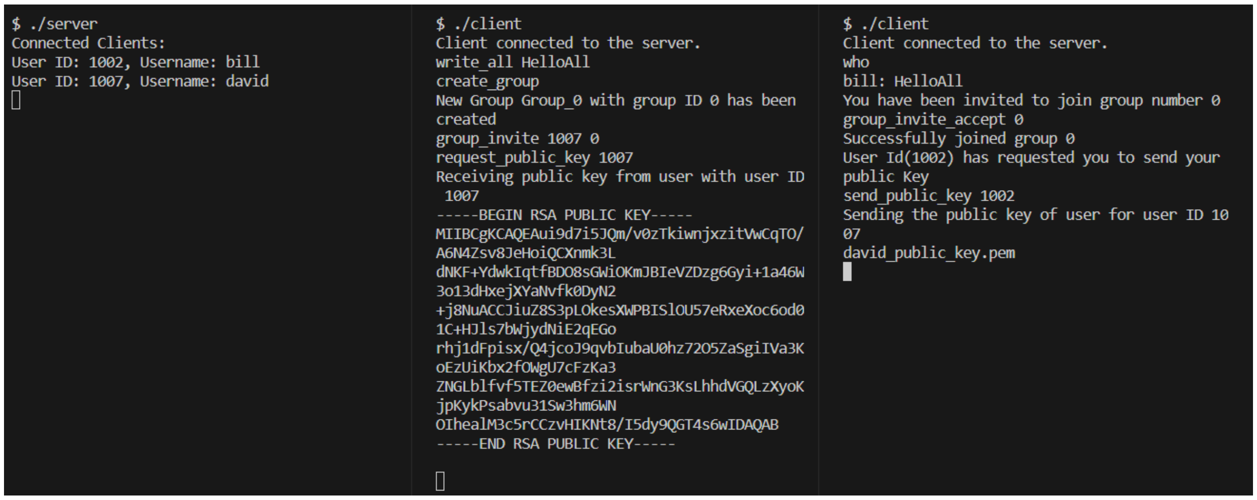


```
$ ./server
Connected Clients:
User ID: 1002, Username: bill
User ID: 1007, Username: david
```

```
$ ./client
Client connected to the server.
write_all HelloAll
create_group
New Group Group_0 with group ID 0 has been
created
group_invite 1007 0
request_public_key 1007
```

```
$ ./client
Client connected to the server.
who
bill: HelloAll
You have been invited to join group number 0
group_invite_accept 0
Successfully joined group 0
User Id(1002) has requested you to send your
public Key
```

7) “send_public_key” function allows the user to send his public key to the requesting user.
Syntax of the command: **send_public_key userID**
where userID refers to the uid of the user to whom public key is to be sent.



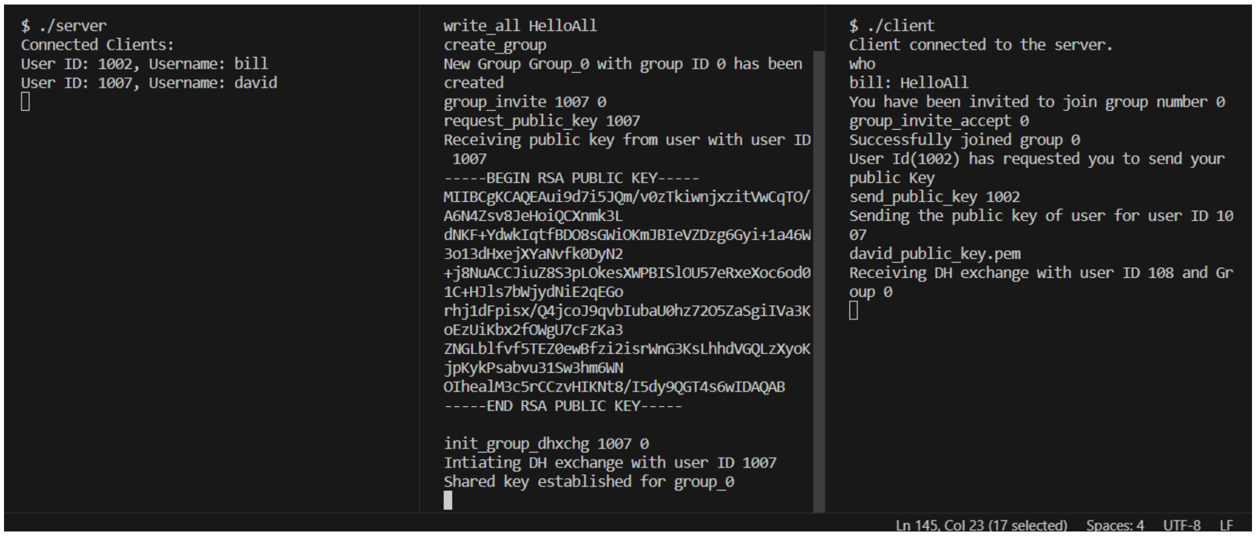
```
$ ./server
Connected Clients:
User ID: 1002, Username: bill
User ID: 1007, Username: david
```

```
$ ./client
Client connected to the server.
write_all HelloAll
create_group
New Group Group_0 with group ID 0 has been
created
group_invite 1007 0
request_public_key 1007
Receiving public key from user with user ID
 1007
-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAui9d7i5JQm/v0zTkiwnjxzitVwCqTO/
A6N4Zsv8JeHoiQCXnmk3L
dNKF+YdwkIqtfBDO8sGWiOKmJBIeVZDzg6Gyi+1a46W
3o13dHxejXYaNvfk0DyN2
+j8NuACCJiuZ8S3pLOkesXWPBISlOU57eRxeXoc6od0
1C+HJls7bWjydNiE2qEGo
rhj1dFpisx/Q4jcoJ9qvbIubaU0hz72O5ZaSgiIVa3K
oEzUiKbx2fOWgU7cFzKa3
ZNGLblfvf5TEZ0ewBfzi2isrWnG3KsLhhdVGQLzXyoK
jpKykPsabvu31Sw3hm6WN
OIhealM3c5rCCzvHIKNt8/I5dy9QGT4s6wIDAQAB
-----END RSA PUBLIC KEY-----
```

```
$ ./client
Client connected to the server.
who
bill: HelloAll
You have been invited to join group number 0
group_invite_accept 0
Successfully joined group 0
User Id(1002) has requested you to send your
public Key
send_public_key 1002
Sending the public key of user for user ID 10
07
david_public_key.pem
```

8)”init_group_dhxchg” function allows two users in a group to perform Diffie Hellman Key exchange to arrive at a shared secret which can be used as key to encrypt and decrypt group messages.

Syntax of the command: **init_group_dhxchg userID groupID**
where userID is the uid of the user with whom DH exchange is to be performed and groupID is group ID for which key is to be derived.



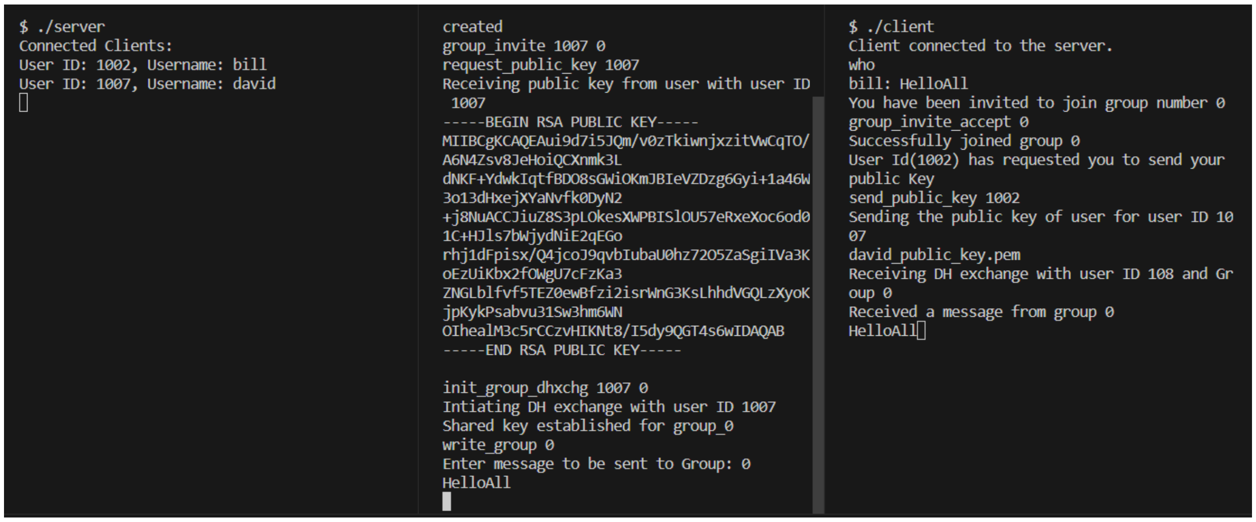
```
$ ./server
Connected Clients:
User ID: 1002, Username: bill
User ID: 1007, Username: david
```

```
write_all HelloAll
create_group
New Group Group_0 with group ID 0 has been
created
group_invite 1007 0
request_public_key 1007
Receiving public key from user with user ID
 1007
-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAui9d7i5JQm/v0zTkiwnjxzitVwCqTO/
A6N4Zsv8JeHoiQCXnmk3L
dNKF+YdwkIqtfBDO8sGWiOKmJBIeVZDzg6Gyi+1a46W
3o13dHxejXYaNvfk0DyN2
+j8NuACCJiuZ8S3pLOkesXWPBISlOU57eRxeXoc6od0
1C+HJls7bWjydNiE2qEGo
rhj1dFpisx/Q4jcoJ9qvbIubaU0hz72O5ZaSgiIVa3K
oEzUiKbx2fOWgU7cFzKa3
ZNGLblfvf5TEZ0ewBfzi2isrWnG3KsLhhdVGQLzXyoK
jpKykPsabvu31Sw3hm6WN
OIhealM3c5rCCzvHIKNt8/I5dy9QGT4s6wIDAQAB
-----END RSA PUBLIC KEY-----

init_group_dhxchg 1007 0
Intiating DH exchange with user ID 1007
Shared key established for group_0
```

```
$ ./client
Client connected to the server.
who
bill: HelloAll
You have been invited to join group number 0
group_invite_accept 0
Successfully joined group 0
User Id(1002) has requested you to send your
public Key
send_public_key 1002
Sending the public key of user for user ID 10
07
david_public_key.pem
Receiving DH exchange with user ID 108 and Gr
oup 0
```

9)"write_group" function allows users to write private messages to groups they are part of.
Syntax of the command: write_group groupID
Where groupID is the ID of the group to which message is to be sent.



```
$ ./server
Connected Clients:
User ID: 1002, Username: bill
User ID: 1007, Username: david
```

```
created
group_invite 1007 0
request_public_key 1007
Receiving public key from user with user ID
 1007
-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAui9d7i5JQm/v0zTkiwnjxzitVwCqTO/
A6N4Zsv8JeHoiQCXnmk3L
dNKF+YdwkIqtfBDO8sGWiOKmJBIeVZDzg6Gyi+1a46W
3o13dHxejXYaNvfk0DyN2
+j8NuACCJiuZ8S3pLOkesXWPBISlOU57eRxeXoc6od0
1C+HJls7bWjydNiE2qEGo
rhj1dFpisx/Q4jcoJ9qvbIubaU0hz72O5ZaSgiIVa3K
oEzUiKbx2fOWgU7cFzKa3
ZNGLblfvf5TEZ0ewBfzi2isrWnG3KsLhhdVGQLzXyoK
jpKykPsabvu31Sw3hm6WN
OIhealM3c5rCCzvHIKNt8/I5dy9QGT4s6wIDAQAB
-----END RSA PUBLIC KEY-----

init_group_dhxchg 1007 0
Intiating DH exchange with user ID 1007
Shared key established for group_0
write_group 0
Enter message to be sent to Group: 0
HelloAll
```

```
$ ./client
Client connected to the server.
who
bill: HelloAll
You have been invited to join group number 0
group_invite_accept 0
Successfully joined group 0
User Id(1002) has requested you to send your
public Key
send_public_key 1002
Sending the public key of user for user ID 10
07
david_public_key.pem
Receiving DH exchange with user ID 108 and Gr
oup 0
Received a message from group 0
HelloAll
```

Server spawns two threads, one which listens for KDC operations and the other acts as an interface for client communication with each other.
KDC=> IP:127.0.0.1:8080

chatServer=> IP:127.0.0.1:8888

Assumptions:

1) Group keys are established before starting group communications.
2) rsaDerive.cpp and passDerive.cpp programs are executed before the execution of the main program.
3) Before DH exchange a user has the public key of the other user.
4) All the users are in the set {“bill”, “david”, “joe”, “travis”, “steve”, “kane”} and “fakeroot” runs the server program.
5) All the commands are run according to their syntax.
6) Any user would not terminate the connection with the server using any signals.
7) Any user would have access to his keys only like user bill will have access to bill_public_key.pem, bill_private_key.pem, and bill.bin only.