UNIVERSITY OF ALBERTA

PROJECT REPORT ON

DDOS ELIMINATION PROOF OF CONCEPT

SUBMITTED BY

STUDENT NAME:GANESH RAMASAMY

MENTOR NAME:LEONARD ROGERS

ACKNOWLEDGEMENT

A great deal of time and effort has been spent in completing this project. Several special people have guided me and have contributed significantly to this effort and so this becomes obligatory to record my thanks to them.

I thank Prof.Leonard Rogers, my mentor for constantly guiding me in completing this project. Starting as a novice programmer, he helped me to transform into an expert in C programming by guiding me to use Mysql api in C and other encryption libraries. He also helped me to gain in-depth knowledge about Linux operating system. Now I can boast about myself that I am an expert in both Debian and Fedora Linux distributions.

I solemnly express my heartiest gratitude to our Program director Dr.Mike MacGregor for giving me this wonderful opportunity to experiment this project.

Table of Contents

1. ABSTRACT:

The proposed project is mainly based on DDOS (Distributed Denial of services) attack which was performed by hacker group lizard-squad in December 2014 Christmas season. Hacker group were successfully able to perform DDOS attack on XBox, Playstation servers such that gamers were unable to enjoy their Christmas break.

The proposed system for eliminating DDOS uses three host Initiating Host (IH) client, Communiation Authentication Host (CAH) which sits in the middle between server and client acting as authentication medium and Communication Host (CH) which is the server.

The systems divides the TCP/IP 3 way handshake into two systems- one performing the authentication and the other performing connection-oriented communication. Usually in a client-server model, client and socket creates a socket to communicate with each other. But in the proposed project, there is an authenticating server sitting in the middle between client and server. The main purpose of CAH is to authenticate the client at the first place and eliminate DDOS at its end such that the server sitting at the other end is not vulnerable to DDOS attack.

CAH uses two parameters systemID and Initiating Key (IK) to authenticate the client or attacker. If the CAH finds that the client is not authentic or spoofed one, it closes its socket with the client and will not share the details of original server with the client. In this way client will never know who is the server and its TCP syn flood attack will be in vain.

The CAH can find the IH's authenticity by the SYN packet IH is sending. Usually in a TCP/IP 3 way handshake SYN packets are without payload and data transfer starts only after ACK from client is received. But in the proposed system, client or IH is programmed in such a way that it has to include its systemID in SYN packet it is sending to CAH.

If the attacker sends TCP SYN flood without system ID or improper system ID, CAH finds it at the first place and closes it listening socket and does not share any details about the server to the client. In this way DDOS is eliminated by CAH at the first place.

After the client sends proper SYN packet with registered systemID, CAH will ask for the initiating key IH has registered with it. If the client sends Initiating key and System ID, CAH will perform database lookup and decides whether to listen or drop the connection.

Once when CAH finds the IH is authentic, CAH will generate a new Initiating key with IH such that IH will use it next time for authenticating with CAH. In this way CAH performs another authentication to eliminate DDOS.

After IK generation, CAH will get connection details from CH and it will send it to IH. It will also send the IK it has generated to CH such that it will use it to authenticate IH once again. IH after receiving connection details from CAH will create one more socket to listen to CH while CH will create a socket based on the connection details it has sent to communicate with IH.

IH performs normal TCP/IP 3 way handshake to connect to CH. After it reaches CH, IH will send its IK to CH. CH will perform a look-up whether the IK sent by CAH and IH matches with each other or not. If it matches, it will start communicating with IH. If not, it will close the socket for IH abruptly ending the connection even before data transfer. In this way, the proposed system performs DDOS elimination.

In order for the host to have multiple sockets in a single program fork() process was used. It is evident that web servers like Apache uses fork() to handle multiple clients. Similarly it is used in this project to handle multiple sockets for multiple connections. Fork() usually creates one parent process and multiple child process which is multiprocessing capability of operating system.

2. INTRODUCTION

2.1 OBJECTIVE OF THE PROJECT

The main aim of this project is to eliminate Distributed denial of services on console based system such as SONY, Xbox, PlayStation etc. Though there is possibility to expand the project to other types of continuous on-line communication systems, it is primarily designed for the console based systems.

2.2 PROJECT INTRODUCTION

The proposed project uses TCP/IP version 4 and splits the TCP/IP 3 way handshake into two systems. The first systems performs modified three way handshake and acts as authentication medium by generating session key for the second system and the second system acts as connection oriented communication medium.

Though there is supposed to be only server and client in every server-client based communication systems, the above project has three hosts namely Initiating host (IH) (XBox based console client), Communication authentication host (CAH) (authenticating server) which performs authentication and handover to original communicating server and Communication host (CH) (original communicating Xbox server).

The above project uses TCP/IP based communication sockets with TCP Fast Open enabled on Communication authentication host and Initiating host. For database entries and lookups, both Communication authentication host and Initiating host uses mysql database.

3. DOS AND DDOS ATTACK ON TCP/IP BASED SYSTEMS

3.1 DOS ATTACK

For the basic Denial of service attack on a server, client (attacker) sends SYN flood to the server thereby utilizing the server resources fully such that server cannot address clients anymore. This can be explained in TCP/IP 3 way handshake as follows, server usually opens its listening socket on a particular IP address and port number for the client to listen on such that for http service for www.google.com can be as 23.34.56.67 (IP) 80 (port number) and for https service for www.google.com can be as 23.34.56.67 (IP) 443 (port number).

When server receives number SYN packet on its listening socket, it has to acknowledge all the SYN packets before processing the upcoming packets leaving to half open connections due to the crash of server. In this way an attacker by sending TCP SYN flood brings down the services of server.

3.2 DDOS ATTACK

Usually DOS attacks originates from a single attacker while DDOS attacks originates from multiple computers sitting on the internet. Rather than a single system sending SYN flood attack to the server, multiple systems sends the SYN flood attack to the server thereby utilizing the server resources and stopping the services from server. These multiple attackers may be compromised computer systems like botnets which is running one or more bots to perform DDOS attack.

An attacker who is the controller might send malware or any infectious program to multiple users and introduce bots to their systems. In this way many computers have bots installed and at a particular time, the attacker might stimulate all the system as a network(botnet) to perform DDOS attack. In this way it is understandable that DDOS attacks are more powerful than DOS attacks.

3.3 TYPES OF DDOS and DOS ATTACKS

There are 3 types of DDOS and DOS attacks. The first one is volume based attacks which involves UDP, ICMP and other spoofed packet floods. The main aim of this attack is to saturate the bandwidth of server. Magnitude of attack is measured in bits per second (bps).

The second one is Application layer based attacks which includes the attack based on vulnerabilities present in Windows, OpenBSD and Apache. Magnitude of attack is measured in requests per second (rps). The third type is protocol based attacks which includes SYN floods attack. Magnitude of attack is measured in packets per second (pps). The proposed project majorly involves the prevention of protocol based DDOS attacks.

3.4 PlayStation AND XBox CHRISTMAS SEASON DDOS ATTACK

In December 2014, hacker group named **lizard-squad** did a DDOS attack on Xbox live and Playstation network such that gamers were unable to use the gaming console during the peak Christmas season. The proposed project presents a solution to eliminate DDOS attacks on Console based systems such as XBox, PlayStation etc.

3.5 PROPOSED SOLUTION FOR THE DDOS ATTACK ON CONSOLE BASED SYSTEMS

The solution uses a unique system ID for each console users and a unique session key or Initiating key for each users. For each time a user tries to access the gaming server has to provide its unique system ID which was provided when they buy the gaming console box. Though the unique SYSTEM ID remains same for each login requests but the Initiating key varies all the time for login requests.

Initially the console client (IH), sends a SYN packet with its unique system ID to CAH thinking it is sending to original server (CH) but IH is unaware of CAH. CAH holds the database with System ID and Initiating key of all users. CAH uses TCP Fast open TFO cookies to check whether the client with system ID was already registered with it or not. If not registered, it will send a RST packet.

If registered, it will ask for the Initiating key from client. TFO based cookies exchange between client (IH) and server (CAH) can detect IP spoofing as well thereby preventing DDOS attack from IP spoofed attacker. Then the console client will send its Initiating key which was given to it initially while registration.

If both the system ID and Initiating key matches with CAH's database, CAH will start communicating with IH. If not, RST packet will be sent by CAH to IH and it will close its listening socket and will not provide any details of original communicating host(CH) to client (IH).

In this way DDOS attack is eliminated from original server (CH). After successful authentication, both the CAH and IH will enter into IK generation phase. During this phase, IH will first send its randomly generated encrypted Initiating key to CAH.

Once when CAH receives it, it generates its own Initiating key and concatenates its IK with IH's IK and sends the newly generated encrypted IK to IH and both IH and CAH registers the newly generated IK to its newIK field in database while the current IK used for authentication will be moved to currentIK field.

The newly generated IK is used as authentication key for next session and it is used for verification of client (IH) by server communication host (CH). Both the new IK and current IK are stored in database of IH and CAH such that it can be used by system administrators for constantly monitoring the unusual activities.

Once when the IK is generated, CAH sends the session IK to the original server Communicating Host (CH) sitting at its back. The CH, after receiving the Initiating Key, will send its IP address and port number it is willing to offer its service on to CAH, which the CAH will send to IH.

IH after receiving the connection oriented details creates a client socket and starts communicating with CH. Before accessing the services, CH requests the IH for the IK which it has registered with the CAH. IH will send the Initiating key for session if CH find it matches with the one sent by CAH, it starts providing the services to IH. If not, it will close its listening socket and asks IH to re-register with CAH.

4. TCP/IP 3 WAY HANDSHAKE AND TCP FAST OPEN

4.1 TCP/IP 3 WAY HANDSHAKE

In a client-server based communication systems, TCP/IP 3 way handshake works as follows. Client sends a SYN packet to the server, server then acknowledges the SYN packet with SYN/ACK and then client acknowledges the server by sending ACK packet back to the server and the communication socket is established then.

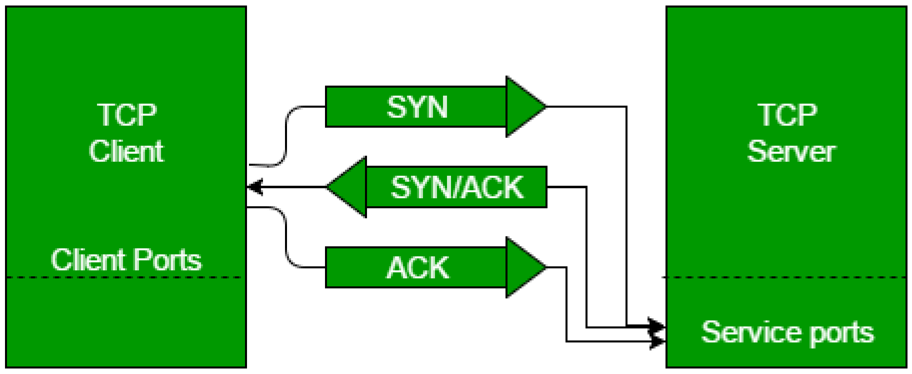


Figure 1: TCP/IP 3-Way Handshake

4.2 TCP FAST OPEN AND APPLICATION IN THE PROJECT

In the proposed project, when IH requests for accessing the services from CH, sends a SYN packet. Original users might send SYN packet but the attackers might send SYN flood and in order to avoid this, the proposed project makes IH programmed in such a way that it includes its system ID in its SYN packet.

In normal TCP/IP sockets it is impossible to send data with SYN packet because windows/linux kernel allows data transfer between sockets only after ACK from client. In order to intelligently send TCP SYN packet with payload in it a new RFC 7413 called TCP Fast open is used.

TCP fast open allows to use payload in SYN packet and it uses TCP based fast open cookies(TFO) exchanges between client and server. When a new user is buying a XBox console, his XBox console would have already exchanged encrypted TFO cookies with CAH and when an attacker tries to spoof the CAH with system ID, CAH will find it at the first place, though it will ask for Initiating key from the attacker, it will close its listening socket after receiving it.

Even after TFO cookie exchange, if system ID and Initiating key of IH doesn’t match, CAH will close its listening socket preventing access to original Communicating host(CH) thereby eliminating DDOS on server such that authentic users can still access thereby there are no disruption of services.

Initially when the new xbox console client registers with CAH the flow diagram is seen below. It is usually done before user buys the XBox console such that attacker without registered XBox console client kit cannot perform DDOS attack.

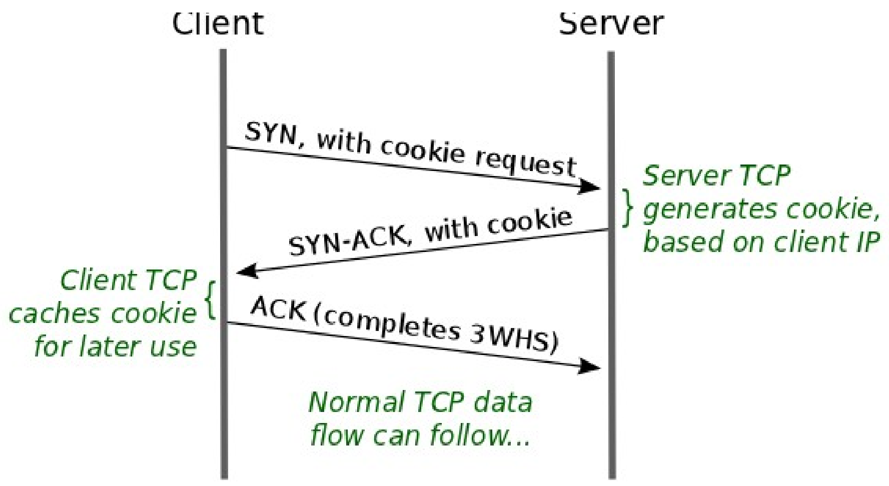

Client
Server
SYN, with cookie request
Server TCP generates cookie, based on client IP
SYN-ACK, with cookie
Client TCP caches cookie for later use
ACK (completes 3WHS)
Normal TCP data flow can follow...

Figure 2: TFO Cookie exchange

After the TFO cookie is cached on XBox client (IH) from CAH, XBox console will be ready for sale and unique system ID and initial IK will be provided by System administrators from XBox and PlayStation. Now after system ID is allocated, IH is programmed in such a way that SYN packet it is sending will have its system ID as payload.

Hacker sending SYN floods without system ID or spoofed SYN floods with system ID will make the CAH to close its listening socket, and prevents the hacker from getting server’s (CH) IP address and port number to perform DDOS on original server.

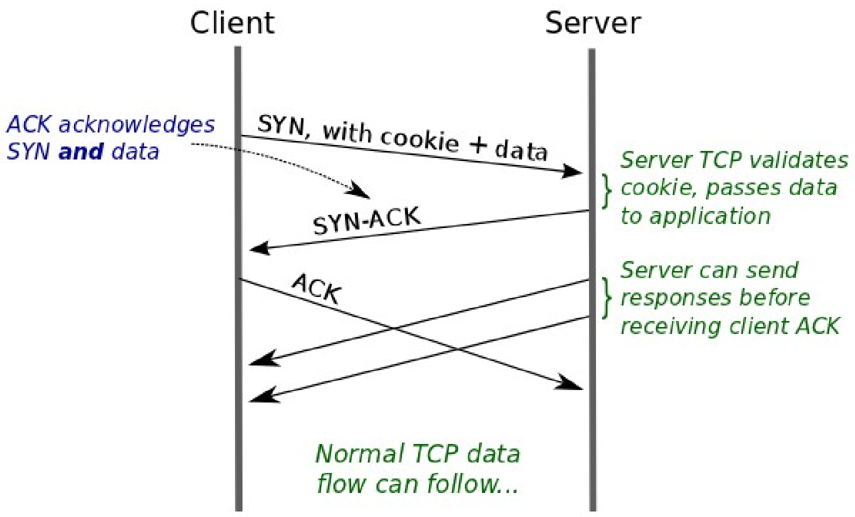

Client
Server
ACK acknowledges SYN **and** data
SYN, with cookie + data
Server TCP validates cookie, passes data to application
SYN-ACK
ACK
Server can send responses before receiving client ACK
Normal TCP data flow can follow...

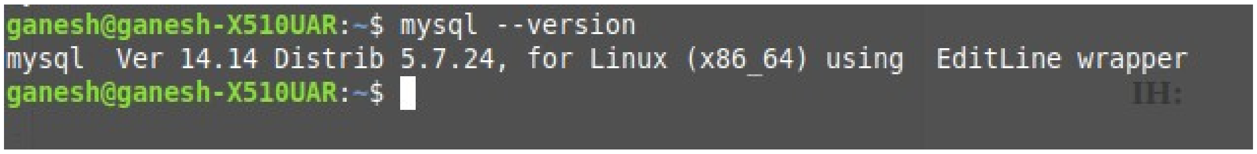
Figure 3: TCP Fast Open 3-Way Handshake

5.MYSQL DATABASE

5.1 APPLICATION IN PROJECT

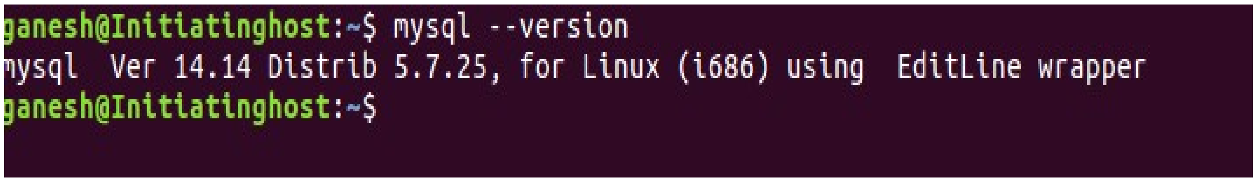
Both Initiating host and Communication Authentication host uses mysql database for storing its initiating key or session key.

The below screenshots shows the version of mysql database in CAH and IH



```
ganesh@ganesh-X510UAR:~$ mysql --version
mysql  Ver 14.14 Distrib 5.7.24, for Linux (x86_64) using  EditLine wrapper
ganesh@ganesh-X510UAR:~$                                          IH:
```

Figure 4: CAH Database Version

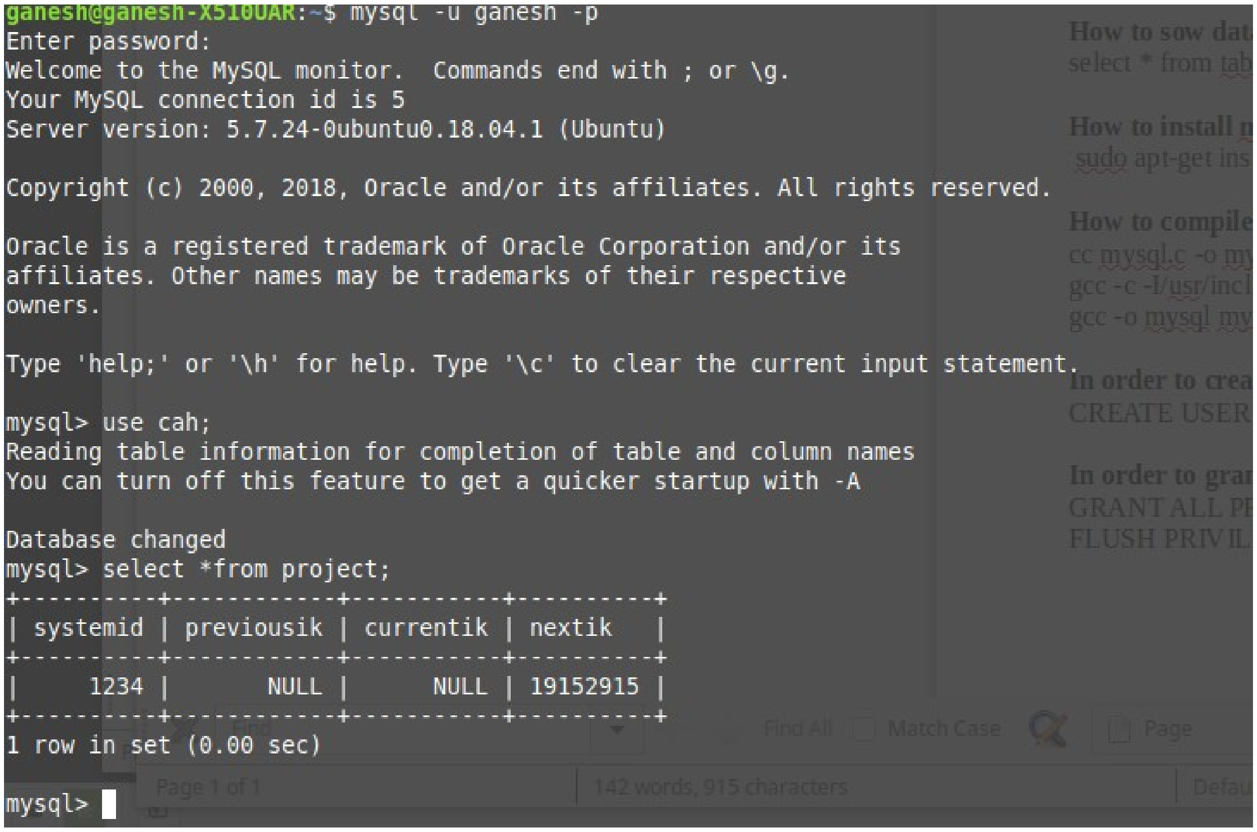


```
ganesh@Initiatinghost:~$ mysql --version
mysql  Ver 14.14 Distrib 5.7.25, for Linux (i686) using  EditLine wrapper
ganesh@Initiatinghost:~$
```

Figure 5: IH Database Version

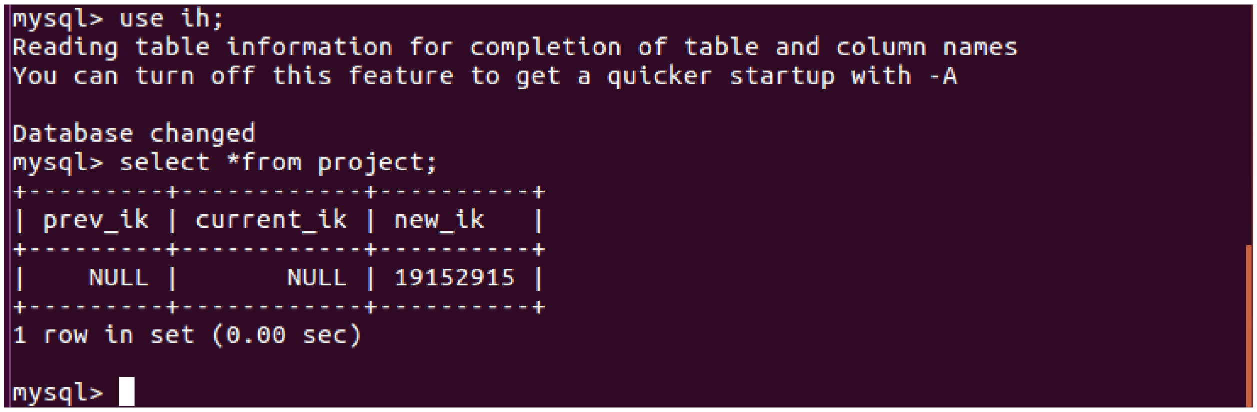
Initially when XBox or PlayStation console is brought, the CAH has the system ID and Initiating key of the console box. CAH has four fields in its database- systemID,nextIK, currentIK, previousIK. During the sale stage, CAH has the nextik and SystemID field initialised while currentik and previousik fields will be filled up everytime when the system is generating a new IK.

For IH, there is three fields, nextik, currentik, previousik. The systemID has to be manually entered by every client (IH) since it has to be sent over SYN packet and since it is uniqueID, it is vulnerable when present over database. Below screenshots show how the database looks at both CAH and IH during initial stage.



```
ganesh@ganesh-X510UAR:~$ mysql -u ganesh -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.24-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use cah;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select *from project;
+----------+-----------+----------+----------+
| systemid | previousik | currentik | nextik   |
+----------+-----------+----------+----------+
|     1234 |       NULL |      NULL | 19152915 |
+----------+-----------+----------+----------+
1 row in set (0.00 sec)

mysql> 
```

Figure 6: CAH's Database



```
mysql> use ih;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select *from project;
+---------+------------+----------+
| prev_ik | current_ik | new_ik   |
+---------+------------+----------+
|    NULL |       NULL | 19152915 |
+---------+------------+----------+
1 row in set (0.00 sec)

mysql> 
```

Figure 7: IH's Database

The above screenshot shows how the database at both IH and CAH looks during sales stage of gaming console box. Once when the client starts using the product, when newik is generated, nextIK field value goes to currentik and the newIK generated will go to nextIK field while the IK in currentIK will go to previousIK field. The values in database are used as references by database administrator to check for any abnormalities.

5.2 MYSQL API in CAH

Code Snippets

```
 static char host = "localhost"; \*Connection parameters for connecting to mysql. Localhost CAH. */
static char *user = "ganesh"; \*Connection parameters for connecting to mysql. Username*/
static char *pass ="1234"; \*Connection parameters for connecting to mysql. Password*/
static char *dbname = "cah";\*Connection parameters for connecting to mysql. Tablename*/
unsigned int port = 3306;\*Connection parameters for connecting to mysql. Portnumber*/
static char *unix_socket = NULL;\*Connection parameters for connecting to mysql.*/
unsigned int flag = 0;\*Connection parameters for connecting to mysql.*/
MYSQL *conn; \*Mysql Connection variable*/
        MYSQL_RES *res; \*Mysql Connection result variable*/
        MYSQL_ROW row;\*Mysql Connection row variable*/
        conn =mysql_init(NULL); \*Initializing the mysql connection*/
              if(!(mysql_real_connect(conn,host,user,pass,dbname,port,unix_socket,flag)))  \*Conditional
statement for mysql Connection*/
        {
                       fprintf(stderr,"\n  error:  %s[%d]\n",mysql_error(conn),mysql_errno(conn));  \*  If
connection  is  not  possible  after  using  all  the  connection  parameters,  print  couldn't  connect  error
message */
        exit(1);
        }
```

5.3 MYSQL API in IH

Code Snippets

```
MYSQL *conn= mysql_init(NULL); \*Initializing the mysql connection*/
if(mysql_real_connect(conn,"localhost","root","1234","ih",3306,NULL,0)==NULL)  \*Connection  parameters
for connecting to mysql. */
{
fprintf(stderr,"%s\n",mysql_error(conn));  \*  If  connection  is  not  possible  after  using  all  the
connection parameters, print couldn't connect error message */
exit(1);
}
```

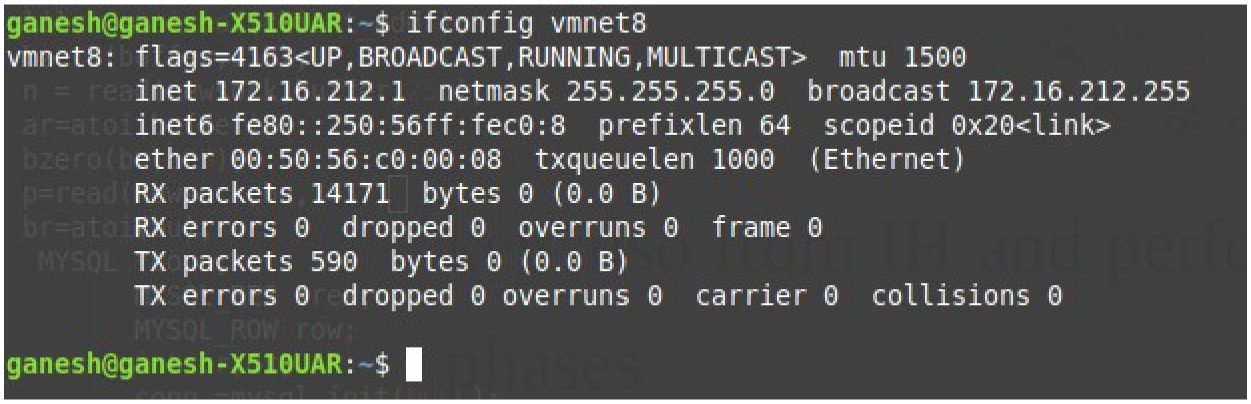
6. PROJECT EXPLANATION

6.1 AUTHENTICATION PHASE

This is the initial phase of the project. During this phase, the console box client Initiating host (IH) will initiate a connection to Communication host with an IP address and port number . IH will be thinking it is communicating with original server but the SYN request of IH with its systemID will go to the Communication Authentication Host (CAH).

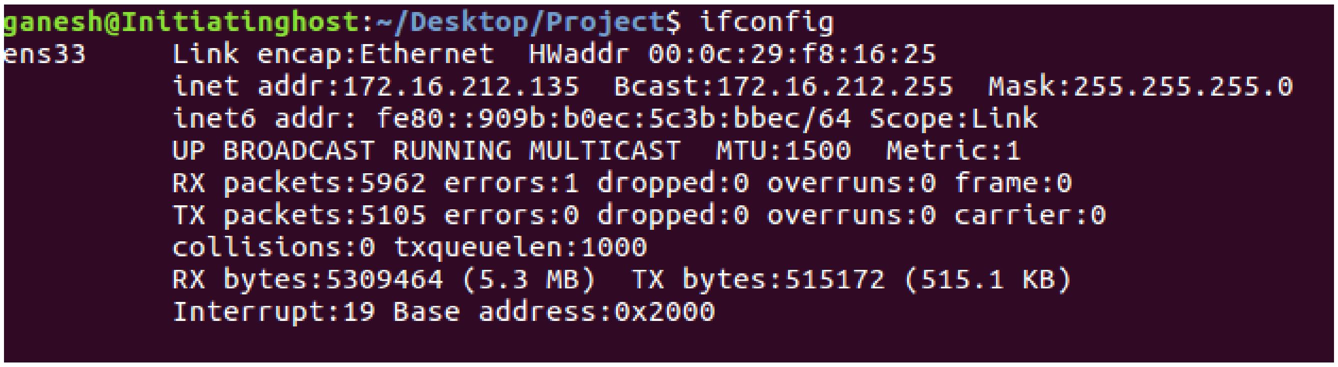
CAH after receiving the systemID of IH gets the Initiating key also from IH and performs a database lookup to authenticate the IH and moving forward to further phases.

Below are the code snippets and screenshots for this phase. Before proceeding to each phase explanation screenshots, below are the IP configuration for all three hosts (IH,CAH,CH).



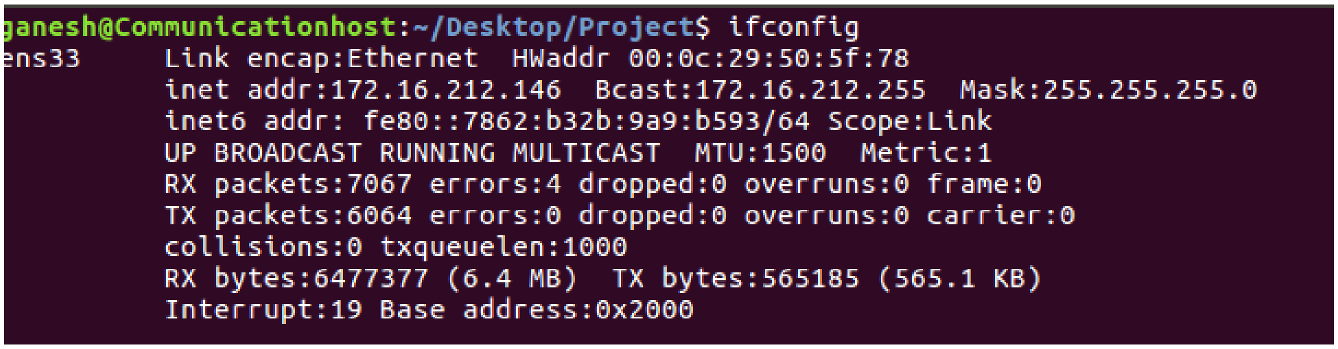
```
ganesh@ganesh-X510UAR:~$ ifconfig vmnet8
vmnet8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.212.1  netmask 255.255.255.0  broadcast 172.16.212.255
        inet6 fe80::250:56ff:fec0:8  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:c0:00:08  txqueuelen 1000  (Ethernet)
        RX packets 14171  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 590  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ganesh@ganesh-X510UAR:~$
```

Figure 8: CAH ifconfig



```
ganesh@Initiatinghost:~/Desktop/Project$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:f8:16:25
          inet addr:172.16.212.135  Bcast:172.16.212.255  Mask:255.255.255.0
          inet6 addr: fe80::909b:b0ec:5c3b:bbec/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5962 errors:1 dropped:0 overruns:0 frame:0
          TX packets:5105 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5309464 (5.3 MB)  TX bytes:515172 (515.1 KB)
          Interrupt:19 Base address:0x2000
```

Figure 9: IH ifconfig

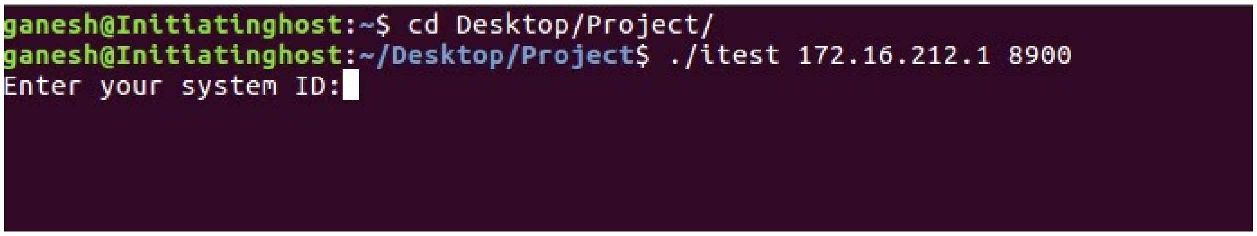

```
ganesh@Communicationhost:~/Desktop/Project$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:50:5f:78
          inet addr:172.16.212.146  Bcast:172.16.212.255  Mask:255.255.255.0
          inet6 addr: fe80::7862:b32b:9a9:b593/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7067 errors:4 dropped:0 overruns:0 frame:0
          TX packets:6064 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6477377 (6.4 MB)  TX bytes:565185 (565.1 KB)
          Interrupt:19 Base address:0x2000
```
Figure 10: CH ifconfig

Code Snippets for CAH

```
int main(int argc, char argv[]) \*Main program with argument passing enabled at terminal*/
{
struct sockaddr_in server_addr,client_addr,cli_addr; \* Structure initialisation for server address and
client address*/
   if(argc<0) \* if the program is executed without passing any argument at terminal */
   {
            fprintf(stderr,"ERROR,no port provided \n");\* Since only port number is passed at CAH end,
if it is not passed at terminal print the error no port number is provided */
           exit(1);\* Exit if true */
   }
   sock= socket(AF_INET,SOCK_STREAM,0); \* Initialising the TCP socket to a variable sock */
   bzero((char*) &server_addr, sizeof(server_addr)); \* Clearing the memory of variable server address
*/
   portnumber = atoi(argv[1]); \* passing the port number as variable */
   server_addr.sin_family=AF_INET; \* Initializing the server address as INET family */
   server_addr.sin_addr.s_addr= INADDR_ANY; \* Accepts any address from INET family as server address
*/
   server_addr.sin_port=htons(portnumber); \* Accepts any portnumber for server and converting it from
unsigned short integer to network  format */
   int qlen=5; \* Buffer length for TCP Fast open connection */
setsockopt(sock,SOL_TCP,TCP_FASTOPEN,&qlen,sizeof(qlen));\* setting the socket as TCP Fast open socket
with protocol TCP */
     if (bind(sock,(struct sockaddr *)&server_addr,sizeof(server_addr))<0) \*Binding the socket with
server address and port number */
   {
         error("Binding ERROR"); \* If there is problem in binding, print binding error */
   }
listen(sock,5); \* Specify the number of clients can listen at a time.
 newsock=accept(sock,(struct sockaddr *)&client_addr,&clilen); \*Initialize the listening socket to the
variable newsock */
clilen = sizeof(client_addr); \* Client address is initialized to variable clilen */
```

Code Snippets for IH

```
struct sockaddr_in server_addr; \* Initializing the structure for server address */
struct hostent *serverd; \*Initialising the structure to a variable for entering the server address */
sock=socket(AF_INET,SOCK_STREAM,0); \* Create TCP socket and initializing to a variable sock */
serverd = gethostbyname(argv[1]);\* getting the server address from client as command line argument */
portnumber = atoi(argv[2]);\* getting the server portnumber to connect to as command line argument */
bzero((char*)&server_addr,sizeof(server_addr));\* clearing the memory of server address variable */
server_addr.sin_family=AF_INET;\* specifying the server address as INET family */
bcopy((char *)serverd->h_addr,(char *)&server_addr.sin_addr.s_addr,serverd ->h_length);\* copying the
server parameters  */
server_addr.sin_port=htons(portnumber);\* specifying the server portnumber and converting it from
unsigned short integer to network  format */
```

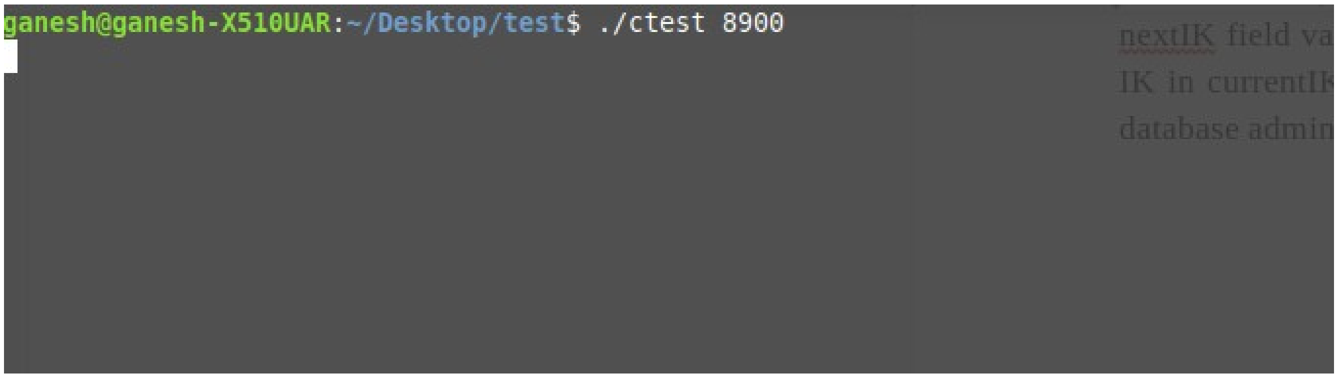


```
ganesh@Initiatinghost:~$ cd Desktop/Project/
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:
```

Figure 11: IH system ID request

Code Snippets

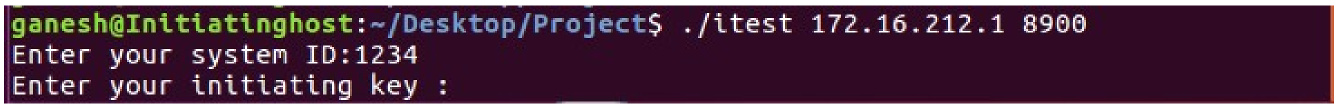
```
printf("Enter your system ID:");\* Print statement for getting client’s system ID */
bzero(buffer,255); \* Clears the memory of buffer */
fgets(buffer,255,stdin);\* gets the system ID as standard input */
sendto(sock,buffer,sizeof(buffer),MSG_FASTOPEN,(struct sockaddr_in *)&server_addr,sizeof(server_addr));
\*Sends the system ID over SYN packet using TCP FAST OPEN to CAH. */
```



```
ganesh@ganesh-X510UAR:~/Desktop/test$ ./ctest 8900
```

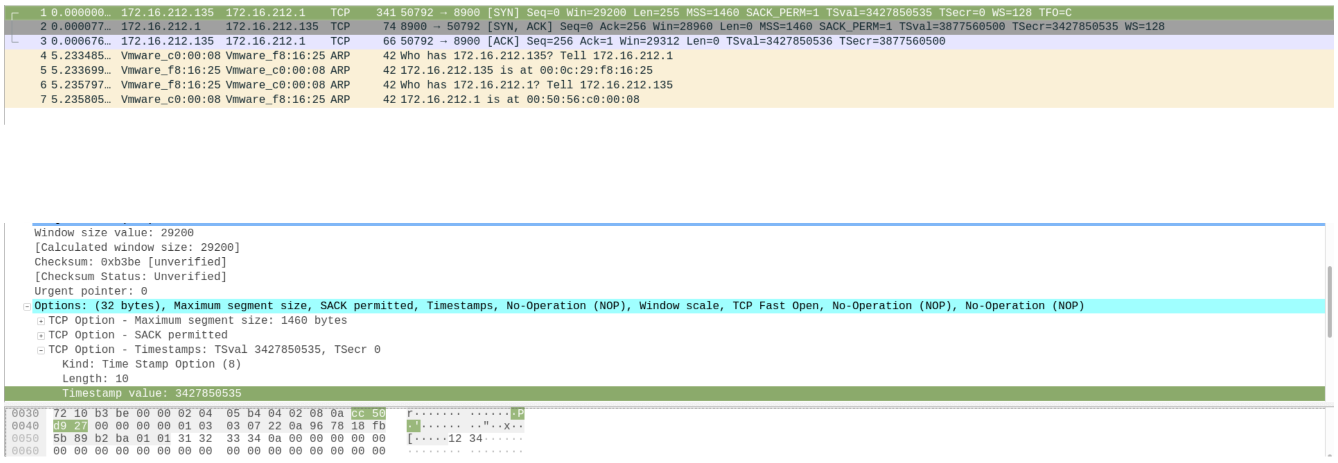
nextIK field va
IK in currentIK
database admin

Figure 12: CAH opening port for authentication



```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :
```

Figure 13: IH sending SYN packet with SystemID

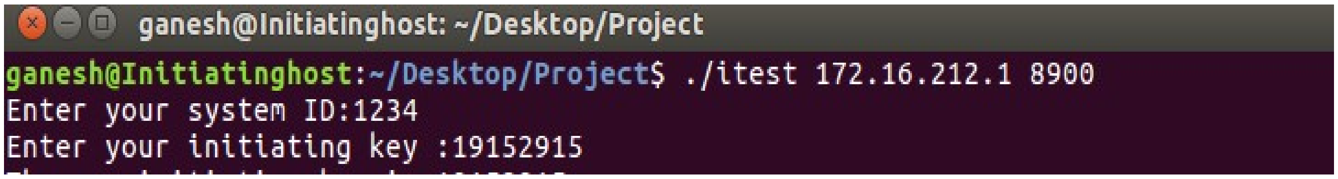


```
1 0.000000… 172.16.212.135  172.16.212.1    TCP   341 50792 → 8900 [SYN] Seq=0 Win=29200 Len=255 MSS=1460 SACK_PERM=1 TSval=3427850535 TSecr=0 WS=128 TFO=C
2 0.000077… 172.16.212.1    172.16.212.135  TCP    74 8900 → 50792 [SYN, ACK] Seq=0 Ack=256 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3877560500 TSecr=3427850535 WS=128
3 0.000676… 172.16.212.135  172.16.212.1    TCP    66 50792 → 8900 [ACK] Seq=256 Ack=1 Win=29312 Len=0 TSval=3427850536 TSecr=3877560500
4 5.233485… Vmware_c0:00:08 Vmware_f8:16:25 ARP    42 Who has 172.16.212.135? Tell 172.16.212.1
5 5.233699… Vmware_f8:16:25 Vmware_c0:00:08 ARP    42 172.16.212.135 is at 00:0c:29:f8:16:25
6 5.235797… Vmware_f8:16:25 Vmware_c0:00:08 ARP    42 Who has 172.16.212.1? Tell 172.16.212.135
7 5.235805… Vmware_c0:00:08 Vmware_f8:16:25 ARP    42 172.16.212.1 is at 00:50:56:c0:00:08
```

```
  Window size value: 29200
  [Calculated window size: 29200]
  Checksum: 0xb3be [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
⊟ Options: (32 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale, TCP Fast Open, No-Operation (NOP), No-Operation (NOP)
  ⊞ TCP Option - Maximum segment size: 1460 bytes
  ⊞ TCP Option - SACK permitted
  ⊟ TCP Option - Timestamps: TSval 3427850535, TSecr 0
      Kind: Time Stamp Option (8)
      Length: 10
      Timestamp value: 3427850535

0030  72 10 b3 be 00 00 02 04  05 b4 04 02 08 0a cc 50   r······· ······P
0040  d9 27 00 00 00 00 01 03  03 07 22 0a 96 78 18 fb   ·'······ ··"··x··
0050  5b 89 b2 ba 01 01 31 32  33 34 0a 00 00 00 00 00   [·····12 34······
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
```

Figure 14: Wireshark packet capture of SYN packet with payload

The above screenshot shows the TCP syn packet sent from IH to CAH with systemID 1234 displayed in the payload section.

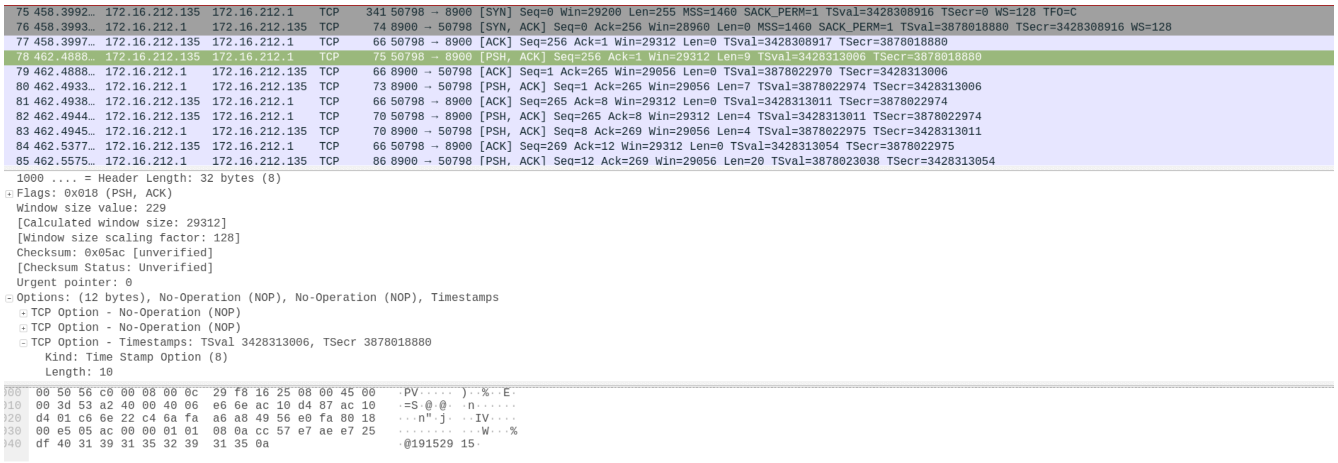


```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :19152915
```

Figure 15: IH sending its IK to CAH

Code Snippets

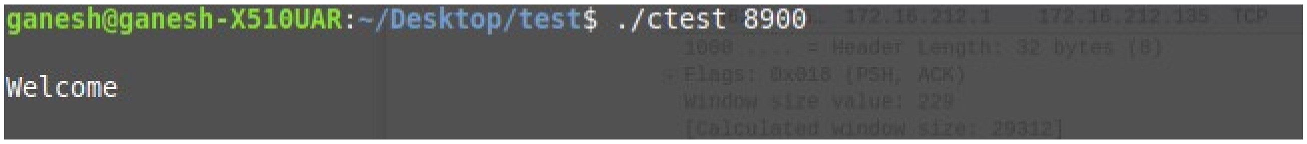
```
printf("Enter your initiating key :");\* Print statement for getting client’s Initiating key */
bzero(ba,255);\* Clears the memory of variable ba */
fgets(ba,255,stdin);\* gets the Initiating key as standard input */
n=write(sock,ba,strlen(ba));\* send the initiating key entered by client to CAH over its socket */
```



```
75 458.3992…  172.16.212.135  172.16.212.1    TCP   341 50798 → 8900 [SYN] Seq=0 Win=29200 Len=255 MSS=1460 SACK_PERM=1 TSval=3428308916 TSecr=0 WS=128 TFO=C
76 458.3993…  172.16.212.1    172.16.212.135  TCP    74 8900 → 50798 [SYN, ACK] Seq=0 Ack=256 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3878018880 TSecr=3428308916 WS=128
77 458.3997…  172.16.212.135  172.16.212.1    TCP    66 50798 → 8900 [ACK] Seq=256 Ack=1 Win=29312 Len=0 TSval=3428308917 TSecr=3878018880
78 462.4888…  172.16.212.135  172.16.212.1    TCP    75 50798 → 8900 [PSH, ACK] Seq=256 Ack=1 Win=29312 Len=9 TSval=3428313006 TSecr=3878018880
79 462.4888…  172.16.212.1    172.16.212.135  TCP    66 8900 → 50798 [ACK] Seq=1 Ack=265 Win=29056 Len=0 TSval=3878022970 TSecr=3428313006
80 462.4933…  172.16.212.1    172.16.212.135  TCP    73 8900 → 50798 [PSH, ACK] Seq=1 Ack=265 Win=29056 Len=7 TSval=3878022974 TSecr=3428313006
81 462.4938…  172.16.212.135  172.16.212.1    TCP    66 50798 → 8900 [ACK] Seq=265 Ack=8 Win=29312 Len=0 TSval=3428313011 TSecr=3878022974
82 462.4944…  172.16.212.135  172.16.212.1    TCP    70 50798 → 8900 [PSH, ACK] Seq=265 Ack=8 Win=29312 Len=4 TSval=3428313011 TSecr=3878022974
83 462.4945…  172.16.212.1    172.16.212.135  TCP    70 8900 → 50798 [PSH, ACK] Seq=8 Ack=269 Win=29056 Len=4 TSval=3878022975 TSecr=3428313011
84 462.5377…  172.16.212.135  172.16.212.1    TCP    66 50798 → 8900 [ACK] Seq=269 Ack=12 Win=29312 Len=0 TSval=3428313054 TSecr=3878022975
85 462.5575…  172.16.212.1    172.16.212.135  TCP    86 8900 → 50798 [PSH, ACK] Seq=12 Ack=269 Win=29056 Len=20 TSval=3878023038 TSecr=3428313054

  1000 .... = Header Length: 32 bytes (8)
+ Flags: 0x018 (PSH, ACK)
  Window size value: 229
  [Calculated window size: 29312]
  [Window size scaling factor: 128]
  Checksum: 0x05ac [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
- Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  + TCP Option - No-Operation (NOP)
  + TCP Option - No-Operation (NOP)
  - TCP Option - Timestamps: TSval 3428313006, TSecr 3878018880
      Kind: Time Stamp Option (8)
      Length: 10

000  00 50 56 c0 00 08 00 0c  29 f8 16 25 08 00 45 00   ·PV·····  )··%··E·
010  00 3d 53 a2 40 00 40 06  e6 6e ac 10 d4 87 ac 10   ·=S·@·@·  ·n······
020  d4 01 c6 6e 22 c4 6a fa  a6 a8 49 56 e0 fa 80 18   ···n"·j·  ··IV····
030  00 e5 05 ac 00 00 01 01  08 0a cc 57 e7 ae e7 25   ········  ···W···%
040  df 40 31 39 31 35 32 39  31 35 0a                  ·@191529 15·
```

Figure 16: Wireshark capture of IH’s IK

After receiving IK if it matches with its database, CAH will send welcome packet to IH. The screenshot showing CAH generating welcome packet.

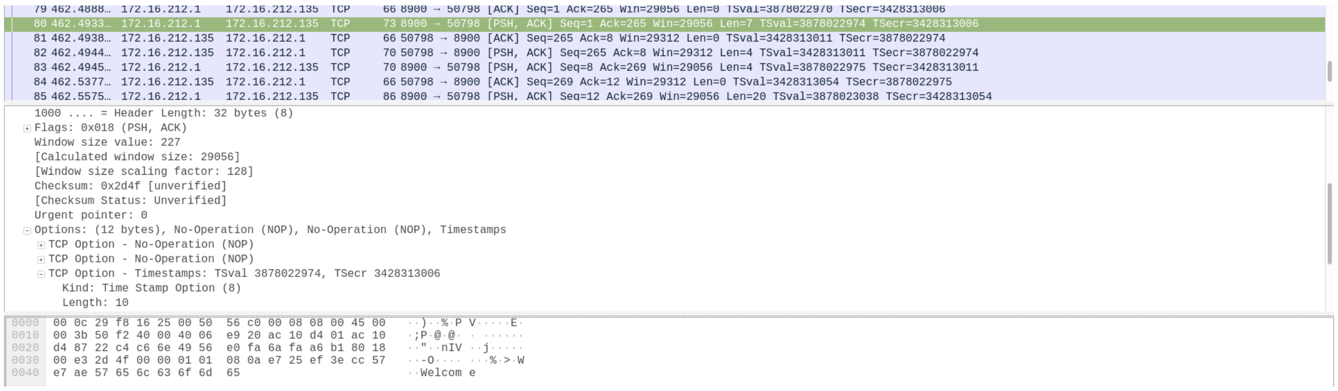


```
ganesh@ganesh-X510UAR:~/Desktop/test$ ./ctest 8900

Welcome
```

Figure 17: CAH after receiving IK

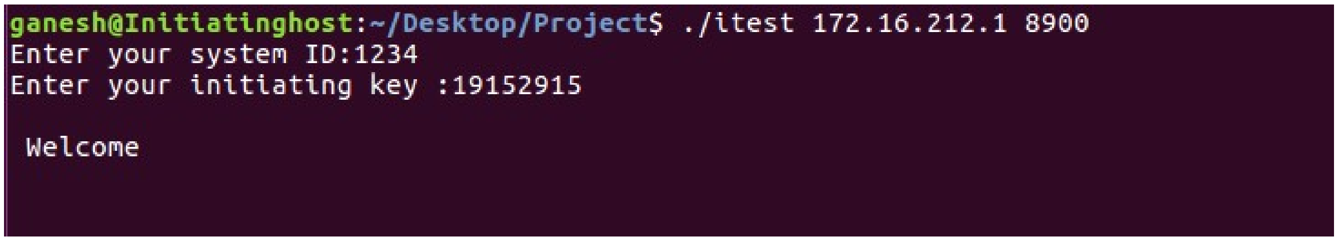
Code Snippets

```
bzero(buffer,255);\* Clear the memory of variable buffer */

n = read(newsock,buffer,255);\* Read the system ID sent by IH using CAH’s listening socket newsock */

ar=atoi(buffer);\* Since the system ID from IH is obtained as string standard input, it is converted to
integer using atoi conversion */

bzero(bu,255);\* Clear the memory of variable bu */

p=read(newsock,bu,255);\* Read the Initiating key sent by IH using CAH’s listening socket newsock */

br=atoi(bu);\* Since the Initiating key from IH is obtained as string standard input, it is converted
to integer using atoi conversion */
cr = atoi(row[0]); \*Convert the string in mysql database at row 0 to integer using standard atoi
conversion */

        dr=atoi(row[3]);\* Convert the string in mysql database at row 3 to integer using standard atoi
conversion */

if(ar == cr && br == dr) \* Condition if the input from IH and the data in CAH database matches each
other*/

{

        er=write(newsock,message,strlen(message));\* Write the message “WELCOME” to IH */
        printf("\n%s\n\n",message);\* Print the message to be written. */

}

else \* Condition if the input from IH and the data in CAH database does not match with each other*/

{

        fr=write(newsock,mas,strlen(mas));\* Write the message “RESET” to IH */

        printf("\n %s\n",mas);\* Print the message to be written. */

}
```



```
79 462.4888…  172.16.212.1    172.16.212.135  TCP     66 8900 → 50798 [ACK] Seq=1 Ack=265 Win=29056 Len=0 TSval=3878022970 TSecr=3428313006
80 462.4933…  172.16.212.1    172.16.212.135  TCP     73 8900 → 50798 [PSH, ACK] Seq=1 Ack=265 Win=29056 Len=7 TSval=3878022974 TSecr=3428313006
81 462.4938…  172.16.212.135  172.16.212.1    TCP     66 50798 → 8900 [ACK] Seq=265 Ack=8 Win=29312 Len=0 TSval=3428313011 TSecr=3878022974
82 462.4944…  172.16.212.135  172.16.212.1    TCP     70 50798 → 8900 [PSH, ACK] Seq=265 Ack=8 Win=29312 Len=4 TSval=3428313011 TSecr=3878022974
83 462.4945…  172.16.212.1    172.16.212.135  TCP     70 8900 → 50798 [PSH, ACK] Seq=8 Ack=269 Win=29056 Len=4 TSval=3878022975 TSecr=3428313011
84 462.5377…  172.16.212.135  172.16.212.1    TCP     66 50798 → 8900 [ACK] Seq=269 Ack=12 Win=29312 Len=0 TSval=3428313054 TSecr=3878022975
85 462.5575…  172.16.212.1    172.16.212.135  TCP     86 8900 → 50798 [PSH, ACK] Seq=12 Ack=269 Win=29056 Len=20 TSval=3878023038 TSecr=3428313054

  1000 .... = Header Length: 32 bytes (8)
+ Flags: 0x018 (PSH, ACK)
  Window size value: 227
  [Calculated window size: 29056]
  [Window size scaling factor: 128]
  Checksum: 0x2d4f [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
- Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  + TCP Option - No-Operation (NOP)
  + TCP Option - No-Operation (NOP)
  - TCP Option - Timestamps: TSval 3878022974, TSecr 3428313006
      Kind: Time Stamp Option (8)
      Length: 10

0000  00 0c 29 f8 16 25 00 50  56 c0 00 08 08 00 45 00   ··)··%·P V·····E·
0010  00 3b 50 f2 40 00 40 06  e9 20 ac 10 d4 01 ac 10   ·;P·@·@·  ······
0020  d4 87 22 c4 c6 6e 49 56  e0 fa 6a fa a6 b1 80 18   ··"··nIV ··j·····
0030  00 e3 2d 4f 00 00 01 01  08 0a e7 25 ef 3e cc 57   ···-O····  ···%·>·W
0040  e7 ae 57 65 6c 63 6f 6d  65                        ··Welcom e
```

Figure 18: Screenshot capture of CAH sending welcome packet to IH



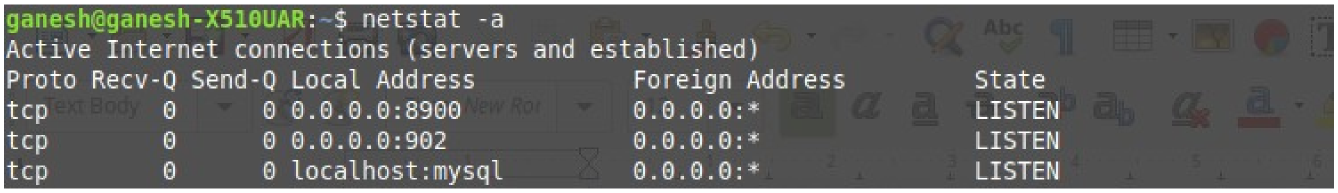
```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :19152915

 Welcome
```

Figure 19: IH receiving the welcome message after successful authentication

Code Snippets

```
n=read(sock,bc,255);\* Read function for reading the message sent by CAH */
printf("\n %s\n\n\n\n",bc);\* Print the message */
```

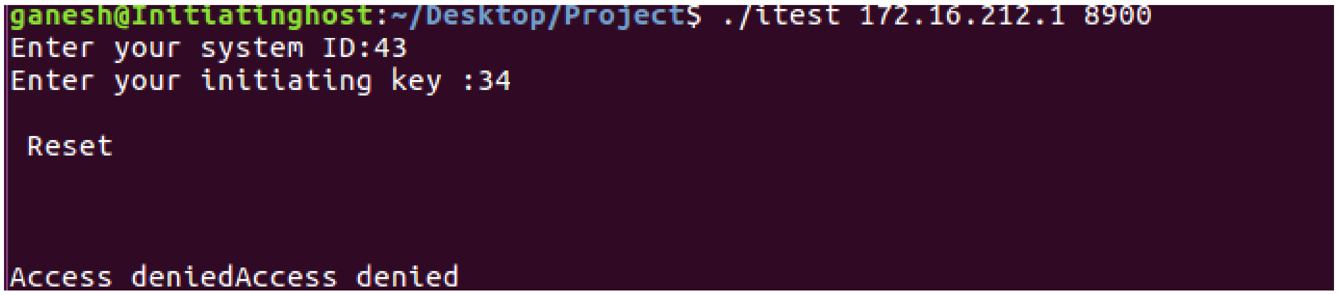
Port number 8900 on listening state at CAH end when authentication is success.



```
ganesh@ganesh-X510UAR:~$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:8900            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:902             0.0.0.0:*               LISTEN
tcp        0      0 localhost:mysql         0.0.0.0:*               LISTEN
```

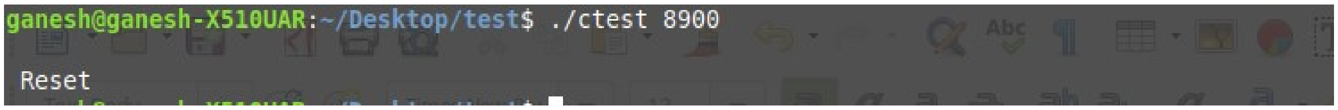
Figure 20: Port status of CAH when there is successful authentication

We can see that CAH is sending reset message and access denied message after unsuccessful authentication.



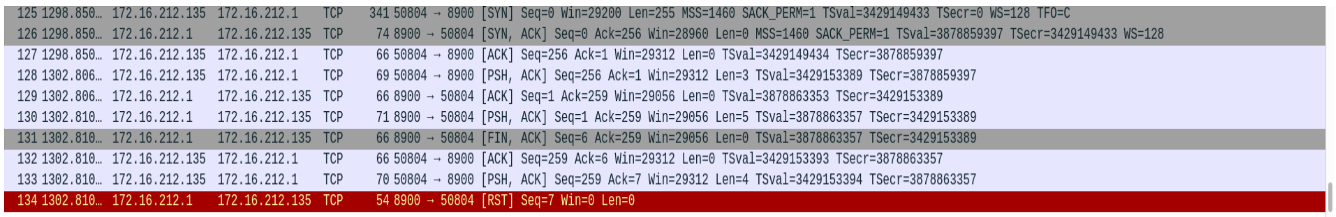
```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:43
Enter your initiating key :34

 Reset



Access deniedAccess denied
```

Figure 21: Unsuccessful authentication at IH end



```
ganesh@ganesh-X510UAR:~/Desktop/test$ ./ctest 8900

 Reset
```

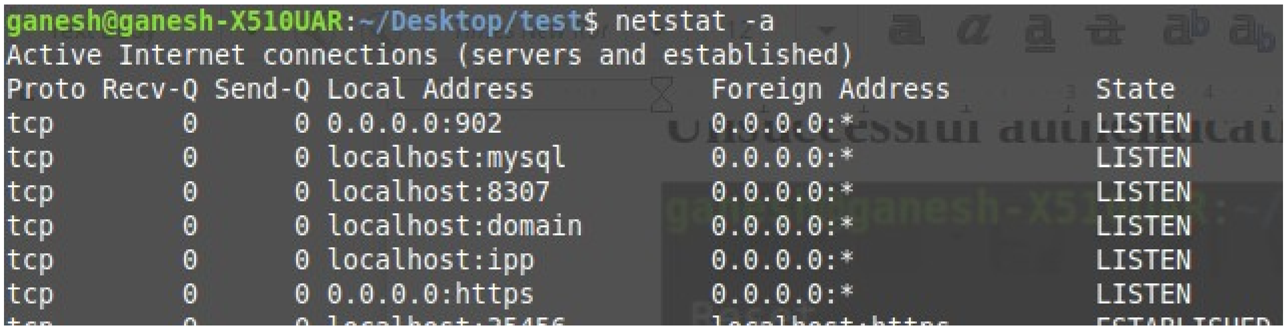
Figure 22: Unsuccessful authentication at CAH end



```
125 1298.850… 172.16.212.135 172.16.212.1   TCP   341 50804 → 8900 [SYN] Seq=0 Win=29200 Len=255 MSS=1460 SACK_PERM=1 TSval=3429149433 TSecr=0 WS=128 TFO=C
126 1298.850… 172.16.212.1   172.16.212.135 TCP    74 8900 → 50804 [SYN, ACK] Seq=0 Ack=256 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3878859397 TSecr=3429149433 WS=128
127 1298.850… 172.16.212.135 172.16.212.1   TCP    66 50804 → 8900 [ACK] Seq=256 Ack=1 Win=29312 Len=0 TSval=3429149434 TSecr=3878859397
128 1302.806… 172.16.212.135 172.16.212.1   TCP    69 50804 → 8900 [PSH, ACK] Seq=256 Ack=1 Win=29312 Len=3 TSval=3429153389 TSecr=3878859397
129 1302.806… 172.16.212.1   172.16.212.135 TCP    66 8900 → 50804 [ACK] Seq=1 Ack=259 Win=29056 Len=0 TSval=3878863353 TSecr=3429153389
130 1302.810… 172.16.212.1   172.16.212.135 TCP    71 8900 → 50804 [PSH, ACK] Seq=1 Ack=259 Win=29056 Len=5 TSval=3878863357 TSecr=3429153389
131 1302.810… 172.16.212.1   172.16.212.135 TCP    66 8900 → 50804 [FIN, ACK] Seq=6 Ack=259 Win=29056 Len=0 TSval=3878863357 TSecr=3429153389
132 1302.810… 172.16.212.135 172.16.212.1   TCP    66 50804 → 8900 [ACK] Seq=259 Ack=6 Win=29312 Len=0 TSval=3429153393 TSecr=3878863357
133 1302.810… 172.16.212.135 172.16.212.1   TCP    70 50804 → 8900 [PSH, ACK] Seq=259 Ack=7 Win=29312 Len=4 TSval=3429153394 TSecr=3878863357
134 1302.810… 172.16.212.1   172.16.212.135 TCP    54 8900 → 50804 [RST] Seq=7 Win=0 Len=0
```

Figure 23: Packet capture of the RST packet sent by CAH to IH

Once when CAH identifies that an attacker or unregistered user is trying to access the services from Communication host, it closes the listening socket and it will not send IP address and port number details of the original server (CH) to the client or attacker such that it can never reach the server to disrupt the services.

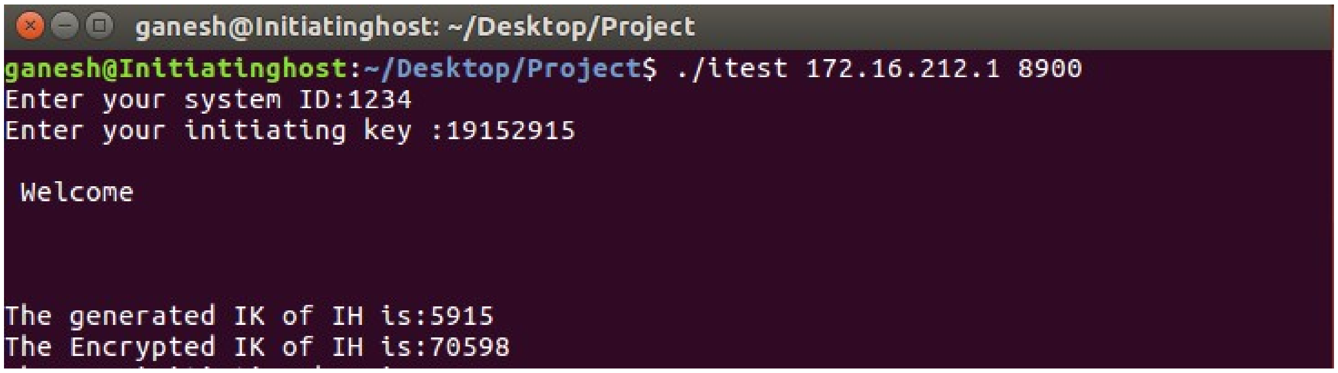


```
ganesh@ganesh-X510UAR:~/Desktop/test$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:902             0.0.0.0:*               LISTEN
tcp        0      0 localhost:mysql         0.0.0.0:*               LISTEN
tcp        0      0 localhost:8307          0.0.0.0:*               LISTEN
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN
tcp        0      0 localhost:ipp           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:https           0.0.0.0:*               LISTEN
```

Figure 24: Port status of CAH after unsuccessful authentication

6.2 KEY GENERATION PHASE

After receiving the welcome packet, IH will generate its Initiating key and sending it to CAH in the encrypted form. The screenshot is seen below



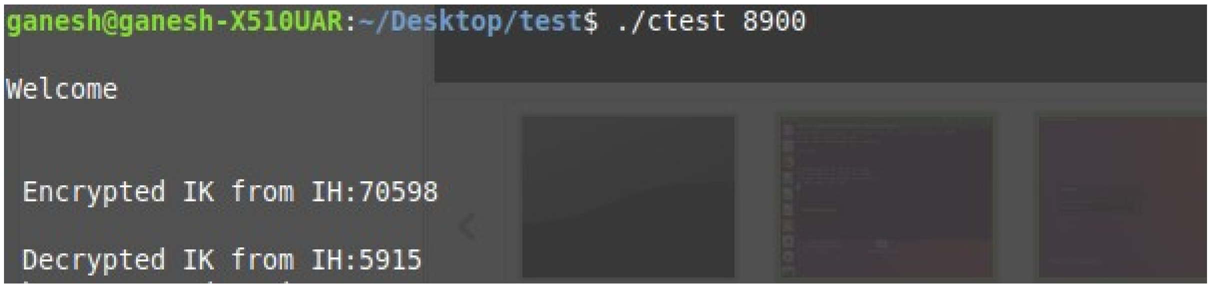
ganesh@Initiatinghost: ~/Desktop/Project

```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :19152915

 Welcome



The generated IK of IH is:5915
The Encrypted IK of IH is:70598
```

Figure 25: IH generating its IK and sending the Encrypted IK to CAH

Code Snippets

```
if(strcmp(bc,"Welcome") == 0) \* Based on the message sent by CAH, IH will read the message and if the message says welcome, it will proceed with the upcoming code */
{
int key=0xFCAB;\* Key used for performing private key symmetric encryption and decryption shared only by CAH and IH and not shared over network*/
for(i=1;i<5;i++)\* Loop for generating random sequence */
{
e=1000+(rand()%5000);\* Random function for generating a random sequence*/
}
printf("The generated IK of IH is:%d\n",e);\* Printing the generated random sequence by IH */
f=e+key;\* Performing encryption function*/
printf("The Encrypted IK of IH is:%d\n",f);\* Printing the encrypted IK */
n=write(sock,&f,sizeof(f));\* Writing the IK over socket to CAH*/
}
else \* If the message received is not "WELCOME"*/
{
printf("Access denied"); \* It will print access denied*/
}
```



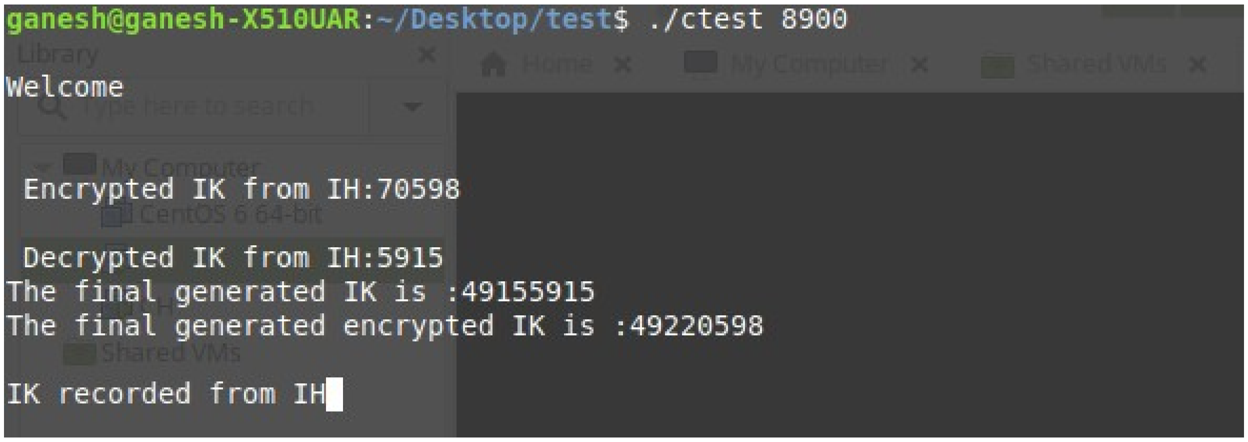
```
ganesh@ganesh-X510UAR:~/Desktop/test$ ./ctest 8900

Welcome


 Encrypted IK from IH:70598

 Decrypted IK from IH:5915
```

Figure 26: CAH receiving the encrypted IK and decrypting it

Code Snippets

```
int key=0xFCAB;\* Key used for performing private key symmetric encryption and decryption */
    hr=read(newsock,&gr,sizeof(gr));\* Read function for reading the encrypted IK sent by IH*/
        printf("\n Encrypted IK from IH:%d\n",gr);\* Printing the encrypted IK*/
        ir = gr - key;\* Generating the decrypted text using the private key symmetric decryption */
        printf("\n Decrypted IK from IH:%d\n",ir);\* Printing the decrypted IK*/
```

CAH receives the IK from IH and it generates its own IK and it concatenates both the IK to generate the final session key or new initiating key. After sending the final IK to IH once when IH records the IK to its database, CAH receives IK recorded message from IH.

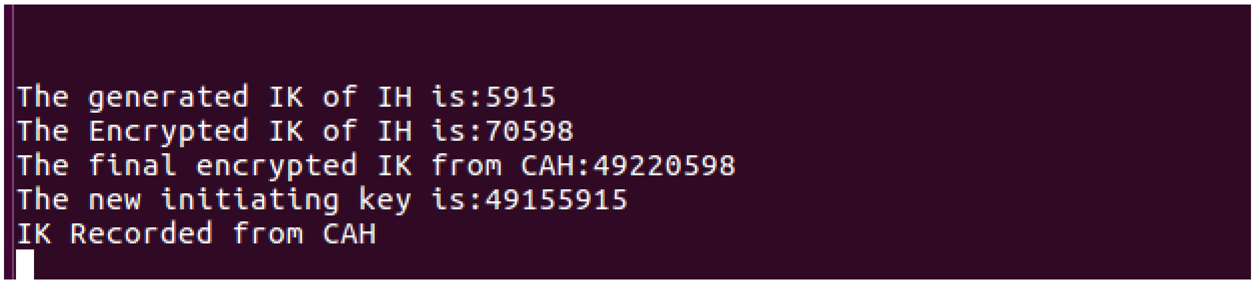


```
ganesh@ganesh-X510UAR:~/Desktop/test$ ./ctest 8900

Welcome


 Encrypted IK from IH:70598

 Decrypted IK from IH:5915
The final generated IK is :49155915
The final generated encrypted IK is :49220598

IK recorded from IH
```

Figure 27: CAH generating its IK and sending the Encrypted IK to IH

Code Snippets

```
int i=0;\* Initializing i to 0*/
        for(i=1;i<5;i++)\* Loop for generating random sequence for IK generated by CAH*/
        {
                e=1000+(rand()%9000); \* Generating random sequence using rand() function*/
        }
        int count=0; \* Initialising count to 0*/
        a=e;\* Initialising a as e*/
         while (e != 0)   \* Function for concatenating CAH’s random sequence and IH’s random sequence
to generate final IK*/
        {
                e /= 10;  \* Calculating the number of digits in e*/
                ++ count;\* Increment e each time when e as not reached its end*/
        }
        b=pow(10,count); \* Performing power operation with count of number of digits of e*/
          c= a * b; \* If e is for digit, calculate the power function first and multiply it to the
original generated sequence by CAH*/
         de= c + ir; \* Then add the multiplied value to IK generated by IH. In this way concatenation
of IK of CAH and IH is performed*/
        printf("The final generated IK is :%d\n",de); \* Printing the final generated IK */
        fe = de+key; \* Encrypting the IK*/
        printf("The final generated encrypted IK is :%d\n",fe); \* Printing the final encrypted IK*/
        er=write(newsock,&fe,sizeof(fe)); \* Writing the final encrypted IK over socket to IH*/
```



```
The generated IK of IH is:5915
The Encrypted IK of IH is:70598
The final encrypted IK from CAH:49220598
The new initiating key is:49155915
IK Recorded from CAH
```

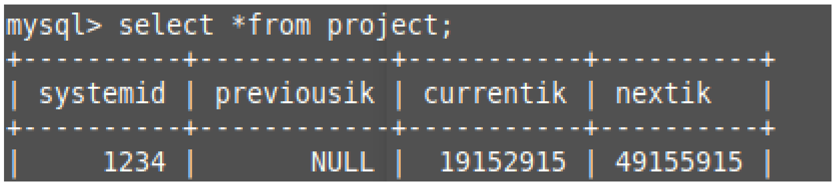
Figure 28: IH receiving the encrypted IK and decrypting it

IH after receiving the encrypted final IK from CAH, decrypts it and stores the IK to its database and CAH sends it has recorded the IK in its database message to IH.

Code Snippets

```
n=read(sock,&g,sizeof(g)); \* Reading the encrypted IK over socket sent by CAH*/
printf("The final encrypted IK from CAH:%d\n",g);\* Printing the final encrypted IK*/
h=g-key; \* Performing decryption for IK*/
printf("The new initiating key is:%d\n",h); \* Printing the new initiating key*/
```

CAH after generating the new IK will store the IK for next session in nextik field while the current IK in next IK field will move to current IK field. Since after initial registration this is the first time the client has requested for a new IK, previousIK field is empty. But when the next time, client requests for newIK, value from currentIK will move to previousIK field. The screenshot can be seen below



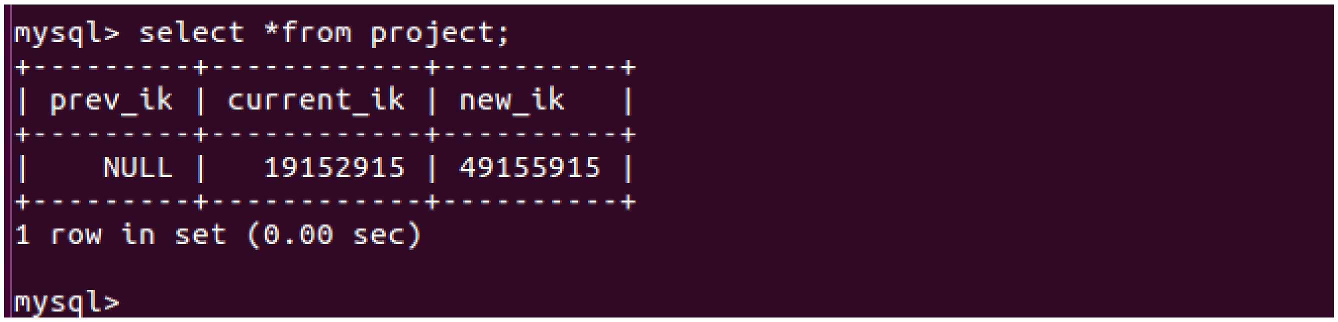
```
mysql> select *from project;
+----------+------------+-----------+----------+
| systemid | previousik | currentik | nextik   |
+----------+------------+-----------+----------+
|     1234 |       NULL |  19152915 | 49155915 |
```

Figure 29: CAH updating its database after NEW IK is generated

Code Snippets

```
char query[2000]; \* Initializing the query */
sprintf(query,"UPDATE project SET nextik = %d WHERE systemid =1234",de);\* Printing the IK as query into CAH’s mysql database*/
mysql_query(conn,query); \* Connection function for query*/
```

IH after getting the new IK will store the IK for next session in nextik field while the current IK in next IK field will move to current IK field. Since after initial registration this is the first time the client has requested for a new IK, previousIK field is empty. But when the next time, client requests for newIK, value from currentIK will move to previousIK field. The screenshot can be seen below



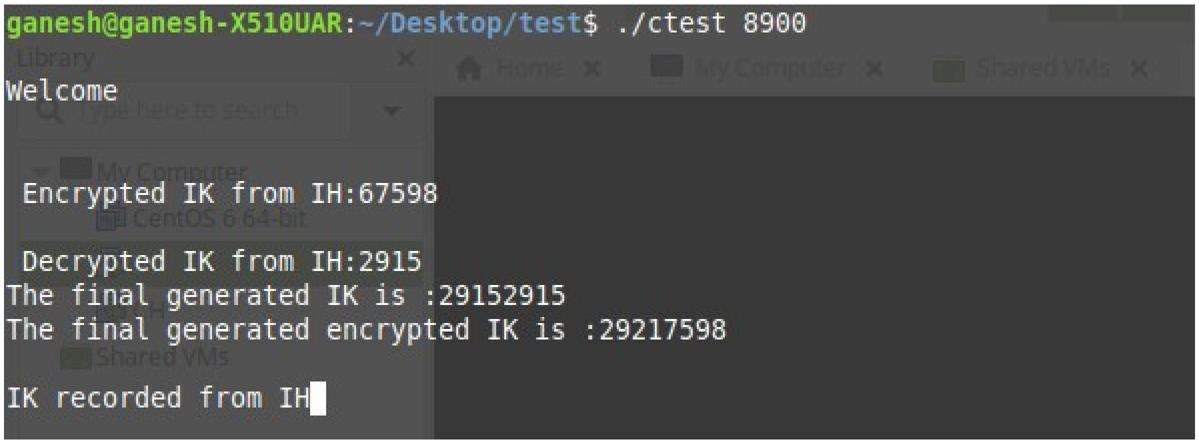
```
mysql> select *from project;
+---------+------------+----------+
| prev_ik | current_ik | new_ik   |
+---------+------------+----------+
|    NULL |   19152915 | 49155915 |
+---------+------------+----------+
1 row in set (0.00 sec)

mysql>
```

Figure 30: IH updating its database after NEW IK is generated

Code Snippets

```
char query[2000]; \* Initializing the query */
```

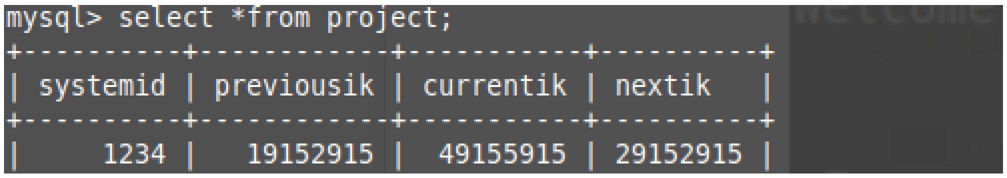
```
sprintf(query,"UPDATE project SET new_ik= %d",h); \* Printing the query value into IH’s database*/
mysql_query(conn,query); \* Connection function for query*/
```



```
ganesh@ganesh-X510UAR:~/Desktop/test$ ./ctest 8900
Welcome


 Encrypted IK from IH:67598

 Decrypted IK from IH:2915
The final generated IK is :29152915
The final generated encrypted IK is :29217598

IK recorded from IH
```

Figure 31: CAH end when one more newik is generated

Update in CAH database can be seen that the previousik field has been updated. And the newik will be used for next session. The main purpose of having three fields for initiating key is to monitor the database by database administrator for any abnormalities and logging.



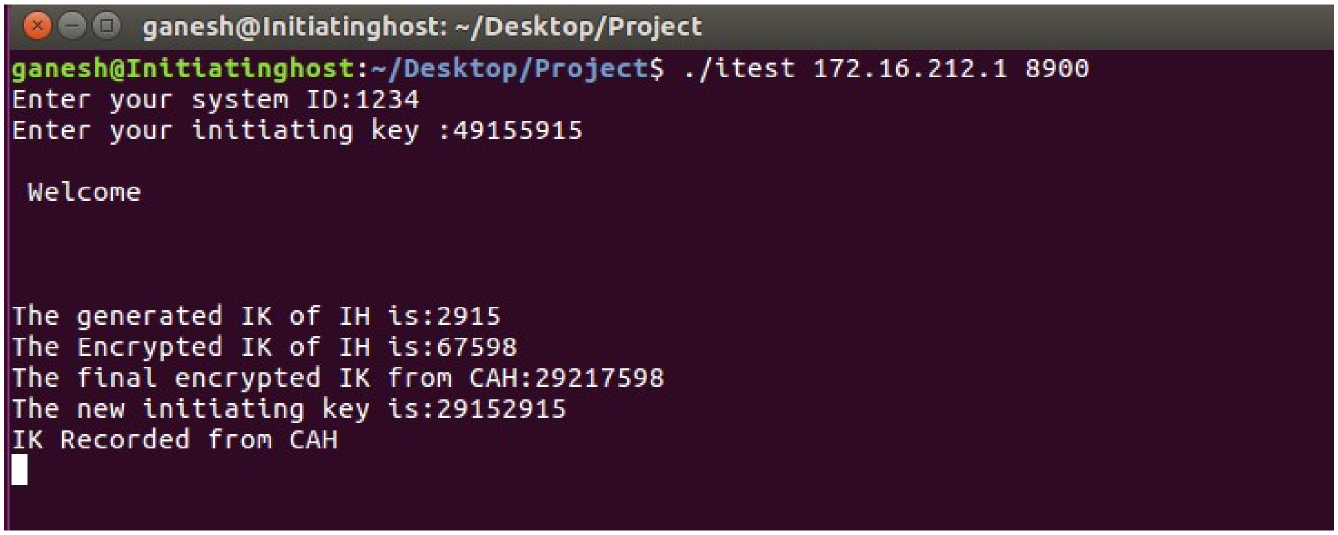
```
mysql> select *from project;
+----------+------------+-----------+----------+
| systemid | previousik | currentik | nextik   |
+----------+------------+-----------+----------+
|     1234 |   19152915 |  49155915 | 29152915 |
```

Figure 32: Changes in CAH database when IH requests for one new IK

Code Snippets

```
mysql_query(conn,"UPDATE project SET previousik = currentik WHERE systemid=1234"); \* Update function in mysql api updating the database field previouik at CAH after new IK is generated*/
mysql_query(conn,"UPDATE project SET currentik = nextik WHERE systemid=1234");\* Update function in mysql api updating the database field currentik at CAH after new IK is generated*/
```

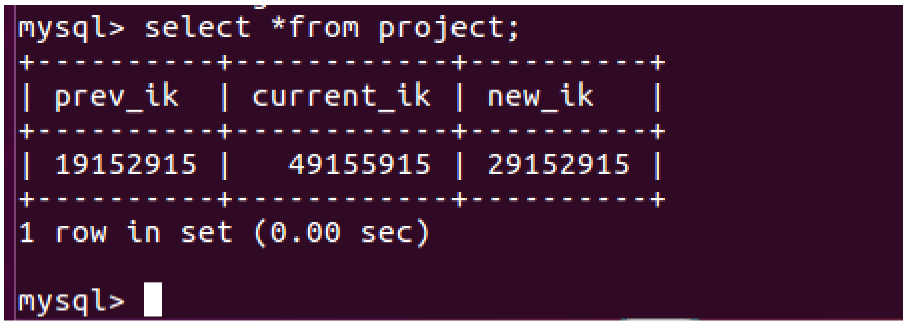
We can see that the client has to use the IK generated from last session to login to next session and generating new IK.



```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :49155915

 Welcome



The generated IK of IH is:2915
The Encrypted IK of IH is:67598
The final encrypted IK from CAH:29217598
The new initiating key is:29152915
IK Recorded from CAH
```

Figure 33: IH end when one more newik is generated

IH database can be seen as



```
mysql> select *from project;
+----------+------------+----------+
| prev_ik  | current_ik | new_ik   |
+----------+------------+----------+
| 19152915 |   49155915 | 29152915 |
+----------+------------+----------+
1 row in set (0.00 sec)

mysql>
```

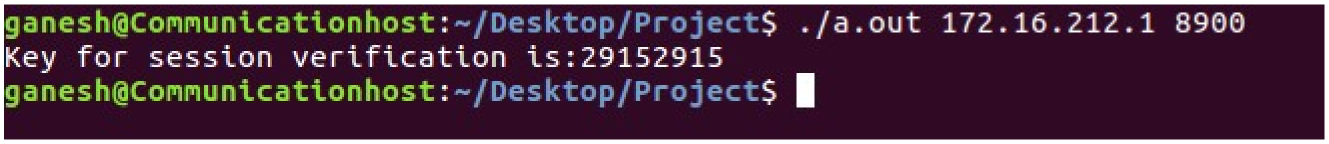
Figure 34: Changes in IH database when IH requests for one new IK

Code Snippets

```
mysql_query(conn,"UPDATE project SET prev_ik = current_ik"); \* Update function in mysql api updating the database field previousik at IH after new IK is generated*/
mysql_query(conn,"UPDATE project SET current_ik = new_ik");\* Update function in mysql api updating the database field currentik at IH after new IK is generated*/
```

6.3 HANDOVER PHASE:

CH will always be connected to CAH. Once when the CAH performs IK generation, CAH will send the created session key or IK to CH. CH after receiving the session key, will send its socket details like IP address and port number it has created its socket on to listen IH or xbox console client.



```
ganesh@Communicationhost:~/Desktop/Project$ ./a.out 172.16.212.1 8900
Key for session verification is:29152915
ganesh@Communicationhost:~/Desktop/Project$
```

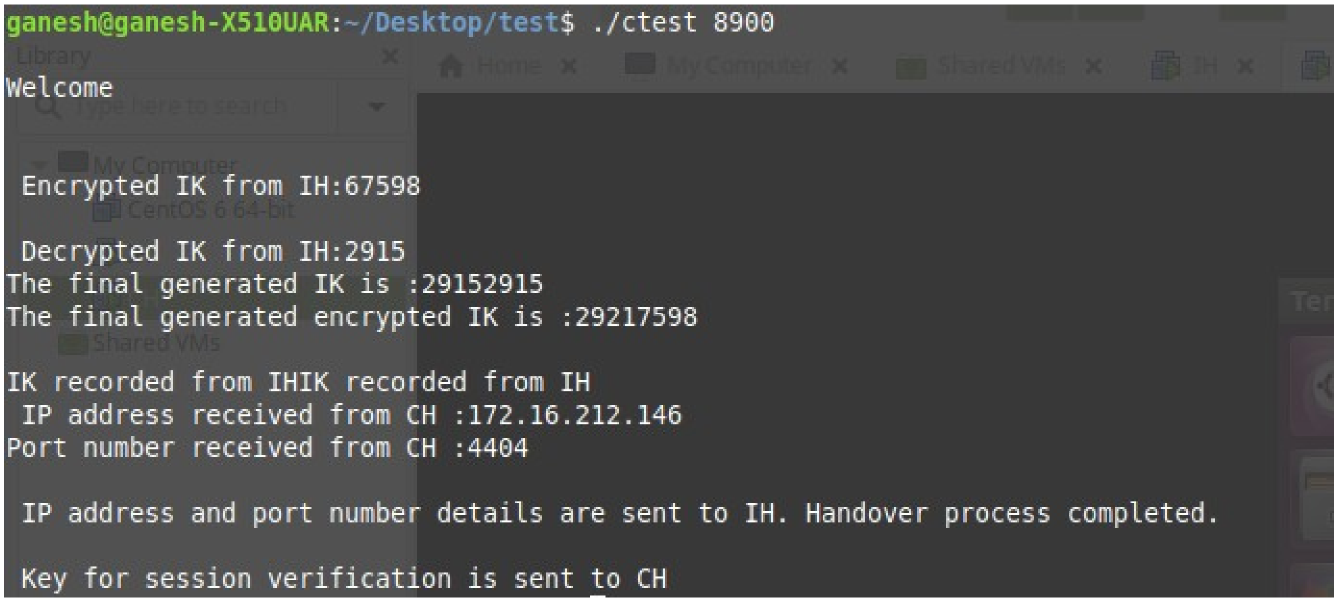
Figure 35: CH getting connected to CAH and getting the Initiating key for verification of IH

Code Snippets

```
struct hostent *serverd;\*Initialising the structure to a variable for entering the server address */
sock=socket(AF_INET,SOCK_STREAM,0);\*TCP socket creation for CH to connect to CAH*/
serverd = gethostbyname(argv[1]);\*Getting server address as input in command line */
portnumber = atoi(argv[2]); \*Getting server portnumber as input in command line */
bzero((char*)&server_addr,sizeof(server_addr)); \*Clearing the memory of server address*/
server_addr.sin_family=AF_INET; \*Specifying server address is AF_INET family */
bcopy((char *)serverd->h_addr,(char *)&server_addr.sin_addr.s_addr,serverd ->h_length);\* copying the
server parameters  */
server_addr.sin_port=htons(portnumber); \* specifying the server portnumber and converting it from
unsigned short integer to network  format */
if(connect(sock,&server_addr,sizeof(server_addr))<0) \*Connecting to server (CAH) socket using connect
function */
{
printf("Error to connect"); \*If couldn’t connect print the error statement */
}
else \* IF connection is success */
{
n=write(sock,bf,255);\* Writing the IP address of CH to CAH over socket*/
n=write(sock,&portno,sizeof(portno)); \* Writing port number of CH to CAH over socket*/
n=read(sock,&a,sizeof(a)); \* Reading IK sent by CAH over socket*/
printf("Key for session verification is:%d\n",a);\* Print the IK*/
}
```

CAH after sending the generated IK to CH receives the IP address and port number details of CH that it would like to listen on to communicate with IH. After it receives the details, it will send it to IH. Then it will display the message that IP address and port number details of CH are sent to IH, handover process is completed.

The main purpose CAH performs here is elimination of DDOS by authentication and introducing the CH to IH such that IH can create a socket to listen to CH.

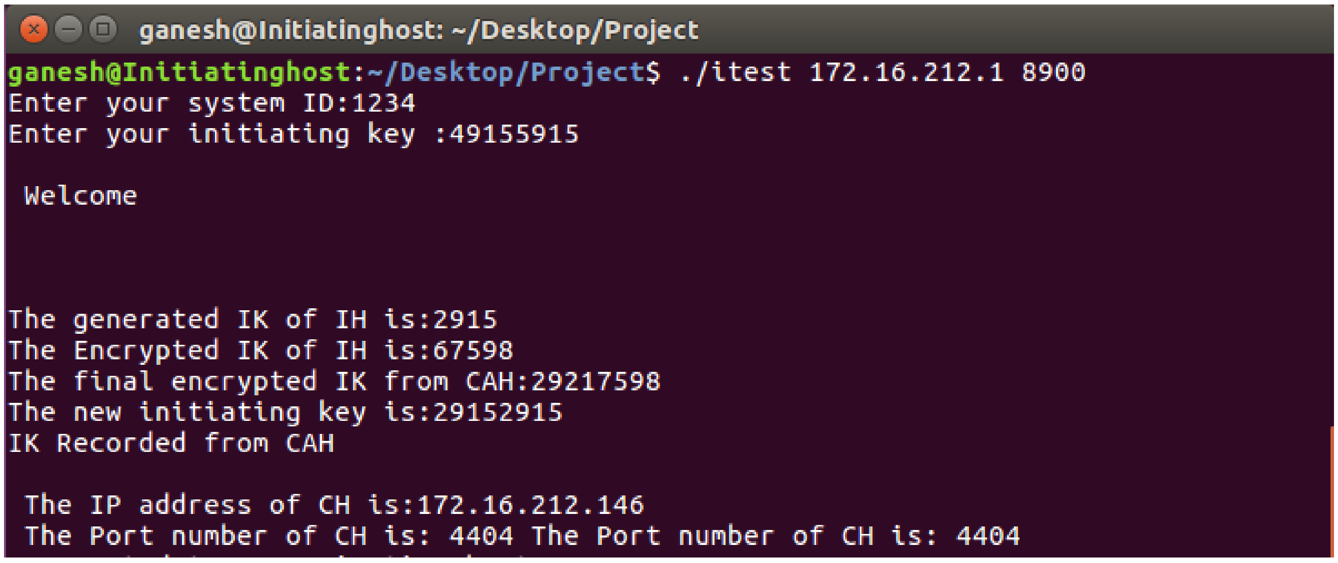


```
ganesh@ganesh-X510UAR:~/Desktop/test$ ./ctest 8900
Welcome

 Encrypted IK from IH:67598

 Decrypted IK from IH:2915
The final generated IK is :29152915
The final generated encrypted IK is :29217598

IK recorded from IHIK recorded from IH
 IP address received from CH :172.16.212.146
Port number received from CH :4404

 IP address and port number details are sent to IH. Handover process completed.

 Key for session verification is sent to CH
```

Figure 36: CAH after sending the IK to CH

Code Snippets

```
if((childpid=fork())==0) \*Using fork system call to create a new process for handling CH since CAH
already has parent process running for IH */
        {
        close(sock); \* If the process is not created, close the socket */
        }
        else \* If not perform the following*/
         {
           newsork=accept(sock,(struct sockaddr *)&cli_addr,&clileng);\* Create a new listening socket
for CH */
         clileng = sizeof(cli_addr); \* Client address of CH*/
         bzero(bf,255); \* Clear the memory for bf */
         a=read(newsork,bf,255); \* Reading IP address sent by CH */
         a=read(newsork,&b,sizeof(b)); \* Reading port number sent by CH */
         printf("\n IP address received from CH :%s\n",bf);\* Printing IP address sent by CH */
       printf("Port number received from CH :%d\n",b);\* Printing IP Port number sent by CH */
         z=write(newsork,&de,sizeof(de)); \* Writing IK over socket to CH */
         printf("\n Key for session verification is sent to CH \n"); \* Printing it has sent the IK to
CH */
         c=write(newsock,bf,255); \* Writing IP address of CH to IH. newsock is for IH, newsork is for
CH */
         d=write(newsock,&b,sizeof(b)); \* Writing port number of CH to IH. */
         printf("\n IP address and port number details are sent to IH. Handover process completed. \
n"); \* Printing the handover processis completed */
}
```



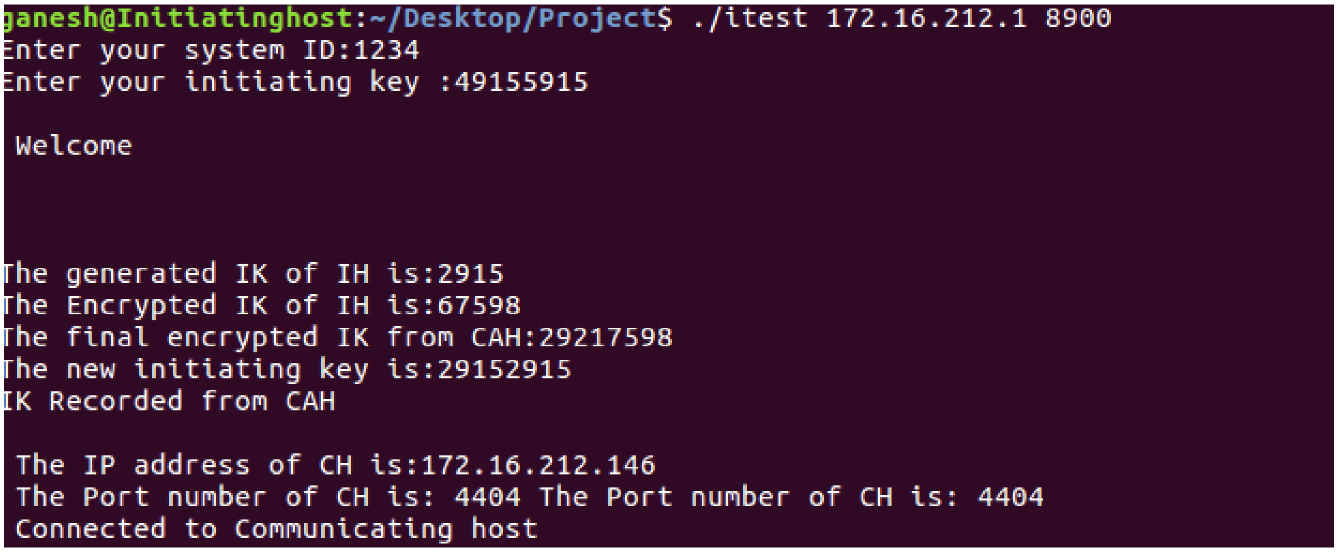
```
ganesh@Initiatinghost: ~/Desktop/Project
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :49155915

 Welcome



The generated IK of IH is:2915
The Encrypted IK of IH is:67598
The final encrypted IK from CAH:29217598
The new initiating key is:29152915
IK Recorded from CAH

 The IP address of CH is:172.16.212.146
 The Port number of CH is: 4404 The Port number of CH is: 4404
```

Figure 37: IH receiving the IP address and port number details of CH

Code Snippets

```
n=read(sock,bf,255);\* Read IP address sent by CAH and store it in varaiable bf */
printf("\n The IP address of CH is:%s",bf);\* Print IP address of CH sent by CAH*/
n=read(sock,&j,sizeof(j)); \* Read port number sent by CAH and store it in varaiable j */
n=read(sock,bf,255);\* Read IP address sent by CAH and store it in varaiable bf */
printf("\n The IP address of CH is:%s",bf);\* Print IP address of CH sent by CAH*/
printf("\n The Port number of CH is: %d",j); \* Print port number of CH sent by CAH*/
```

IH after receiving the connection details from CAH, uses fork() process to create a new client socket to listen CH while it uses parent process to communicate with CAH. After successfully creating the socket, if IH reaches CH, it has the following display message in console client.



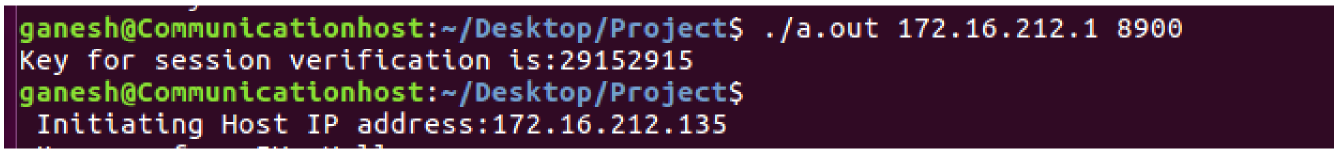
```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :49155915

 Welcome



The generated IK of IH is:2915
The Encrypted IK of IH is:67598
The final encrypted IK from CAH:29217598
The new initiating key is:29152915
IK Recorded from CAH

 The IP address of CH is:172.16.212.146
 The Port number of CH is: 4404 The Port number of CH is: 4404
 Connected to Communicating host
```

Figure 38: IH after reaching CH

Code Snippets

```
if((childpid = fork()) == 0) \* Creating child process for communicating with CH while parent process is for communication with CAH*/
{
close(sock); \* If process is not created, close the parent socket and program. */
}
else \* If created successfully */
{
struct sockaddr_in servere_addr; \* Structure for server address */
sork=socket(AF_INET,SOCK_STREAM,0); \* Creating TCP socket*/
bzero((char*)&servere_addr,sizeof(servere_addr)); \* Clearing the memory for server address*/
servere_addr.sin_family=AF_INET; \* Specifying server address family as INET or IP family */
servere_addr.sin_addr.s_addr=inet_addr(bf); \* Server address as the variable got from CAH */
servere_addr.sin_port=htons(j); \* Server port number as the variable got from CAH. It uses short integer to network converter for the port number */
if(connect(sork,&servere_addr,sizeof(servere_addr))<0) \* Connect function to connect to the server address and port number */
{
printf("\nError to connect\n");\* If couldn’t connect */
}
else
{
printf("\n Connected to Communicating host");\* If connection is success, Print the connected statement */
}
```

CH will also use fork() function to create a new process to create a listening socket for IH. While it has another parent process running to listen CAH.



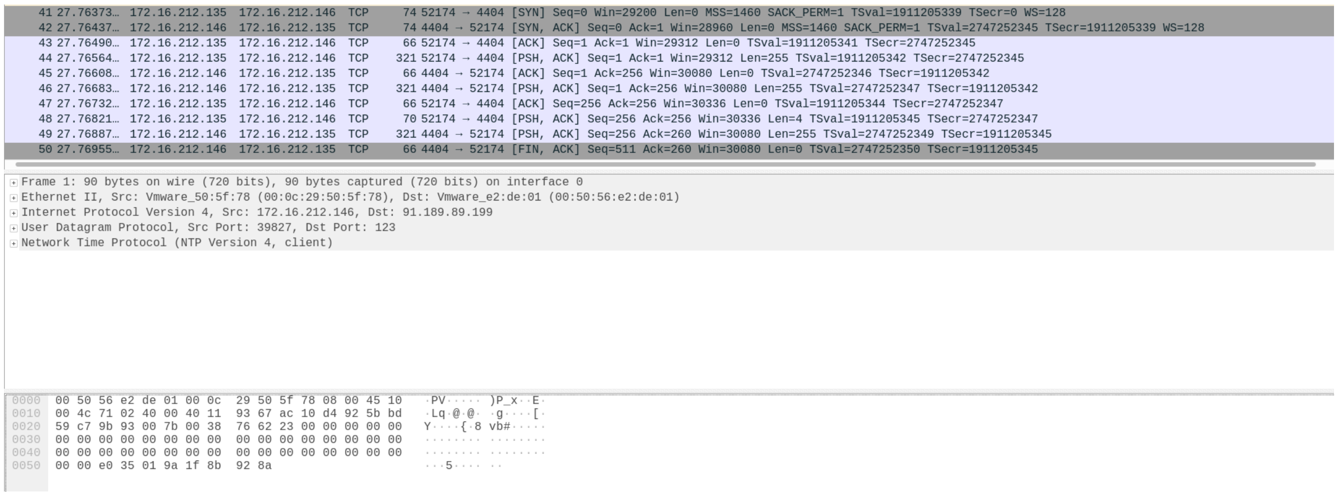
```
ganesh@Communicationhost:~/Desktop/Project$ ./a.out 172.16.212.1 8900
Key for session verification is:29152915
ganesh@Communicationhost:~/Desktop/Project$
 Initiating Host IP address:172.16.212.135
```

Figure 39: CH after reaching IH

We can see that CH displaying the IP address of IH after IH gets connected to CH.

Code Snippets

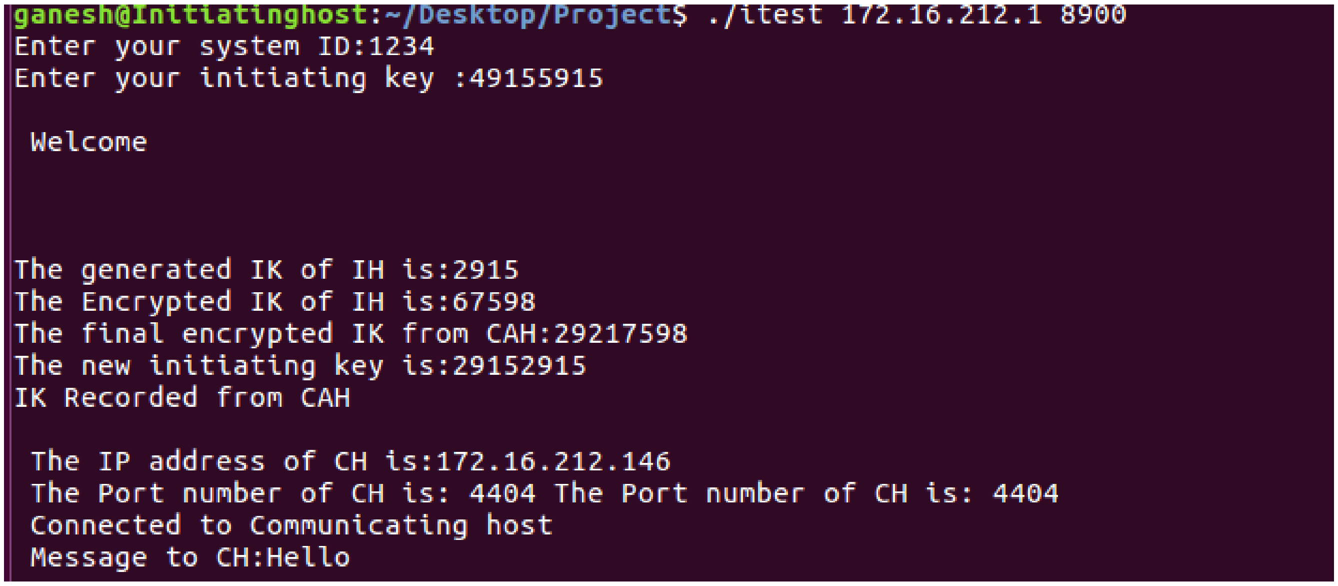
```
if((childpid = fork()) == 0) \* Creating a child process at CH to make a listening socket to listen to IH */
{
close(sock);\* If not created properly, close the parent socket and program*/
{
int portno=4404; \* Variable for port number */
struct sockaddr_in servere_addr,client_addr; \* Structure for server address */
sork=socket(AF_INET,SOCK_STREAM,0); \* Creating a new socket under child process at CH to communicate with IH */
bzero ((char*) &servere_addr,sizeof(servere_addr)); \* Clearing the memory space of server address */
servere_addr.sin_family=AF_INET; \* Specifying the address family of server */
servere_addr.sin_addr.s_addr=inet_addr("172.16.212.146"); \* Specifying the address of server. Usually it will be a static public IP sitting over internet */
servere_addr.sin_port=htons(portno); \* Specifying the port number for CH. Usually it is declared by CH itself after getting IK from CAH */
if (bind(sork,(struct sockaddr *)&servere_addr,sizeof(servere_addr))<0) \*Binding the address and port of server CH socket */
{
error("Binding error"); \* Print error if there is any */
}
else
{
listen(sork,5); \* Mention the child socket to be listening socket */
newsork=accept(sork,(struct sockaddr *)&client_addr,&clilen); \* Create the listening socket for IH */
clilen=sizeof(client_addr); \* Client address length */
char *bt; \*Initialize character bt */
bt=inet_ntoa(client_addr.sin_addr); \* IH’s IP address is stored over bt. Ntoa function converts network address to string format */
printf("\n Initiating Host IP address:%s",bt); \* Print the IP address of client IH */
}
```



```
41 27.76373… 172.16.212.135  172.16.212.146  TCP     74 52174 → 4404 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1911205339 TSecr=0 WS=128
42 27.76437… 172.16.212.146  172.16.212.135  TCP     74 4404 → 52174 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=2747252345 TSecr=1911205339 WS=128
43 27.76490… 172.16.212.135  172.16.212.146  TCP     66 52174 → 4404 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1911205341 TSecr=2747252345
44 27.76564… 172.16.212.135  172.16.212.146  TCP    321 52174 → 4404 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=255 TSval=1911205342 TSecr=2747252345
45 27.76608… 172.16.212.146  172.16.212.135  TCP     66 4404 → 52174 [ACK] Seq=1 Ack=256 Win=30080 Len=0 TSval=2747252346 TSecr=1911205342
46 27.76683… 172.16.212.146  172.16.212.135  TCP    321 4404 → 52174 [PSH, ACK] Seq=1 Ack=256 Win=30080 Len=255 TSval=2747252347 TSecr=1911205342
47 27.76732… 172.16.212.135  172.16.212.146  TCP     66 52174 → 4404 [ACK] Seq=256 Ack=256 Win=30336 Len=0 TSval=1911205344 TSecr=2747252347
48 27.76821… 172.16.212.135  172.16.212.146  TCP     70 52174 → 4404 [PSH, ACK] Seq=256 Ack=256 Win=30336 Len=4 TSval=1911205345 TSecr=2747252347
49 27.76887… 172.16.212.146  172.16.212.135  TCP    321 4404 → 52174 [PSH, ACK] Seq=256 Ack=260 Win=30080 Len=255 TSval=2747252349 TSecr=1911205345
50 27.76955… 172.16.212.146  172.16.212.135  TCP     66 4404 → 52174 [FIN, ACK] Seq=511 Ack=260 Win=30080 Len=0 TSval=2747252350 TSecr=1911205345

Frame 1: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0
Ethernet II, Src: Vmware_50:5f:78 (00:0c:29:50:5f:78), Dst: Vmware_e2:de:01 (00:50:56:e2:de:01)
Internet Protocol Version 4, Src: 172.16.212.146, Dst: 91.189.89.199
User Datagram Protocol, Src Port: 39827, Dst Port: 123
Network Time Protocol (NTP Version 4, client)

0000  00 50 56 e2 de 01 00 0c  29 50 5f 78 08 00 45 10   ·PV····· )P_x··E·
0010  00 4c 71 02 40 00 40 11  93 67 ac 10 d4 92 5b bd   ·Lq·@·@·  ·g····[·
0020  59 c7 9b 93 00 7b 00 38  76 62 23 00 00 00 00 00   Y····{·8 vb#·····
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 e0 35 01 9a 1f 8b  92 8a                     ···5···· ··
```

Figure 40: Wireshark capture of TCP/IP 3 way handshake and data transfer between CH and IH

6.4 COMMUNICATION PHASE

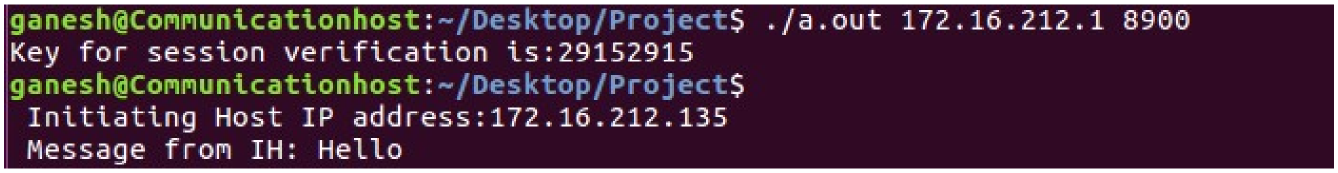


```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :49155915

 Welcome



The generated IK of IH is:2915
The Encrypted IK of IH is:67598
The final encrypted IK from CAH:29217598
The new initiating key is:29152915
IK Recorded from CAH

 The IP address of CH is:172.16.212.146
 The Port number of CH is: 4404 The Port number of CH is: 4404
 Connected to Communicating host
 Message to CH:Hello
```

Figure 41: IH sending hello packet to CH

Code Snippets

```
char vs[255]="Hello"; \* Initializing the message to be sent */
o=write(sork,vs,255); \* Writing the message over socket sork. IH uses sock for CAH and sork for CH */
printf("\n Message to CH:%s\n",vs); \*Print the message */
```

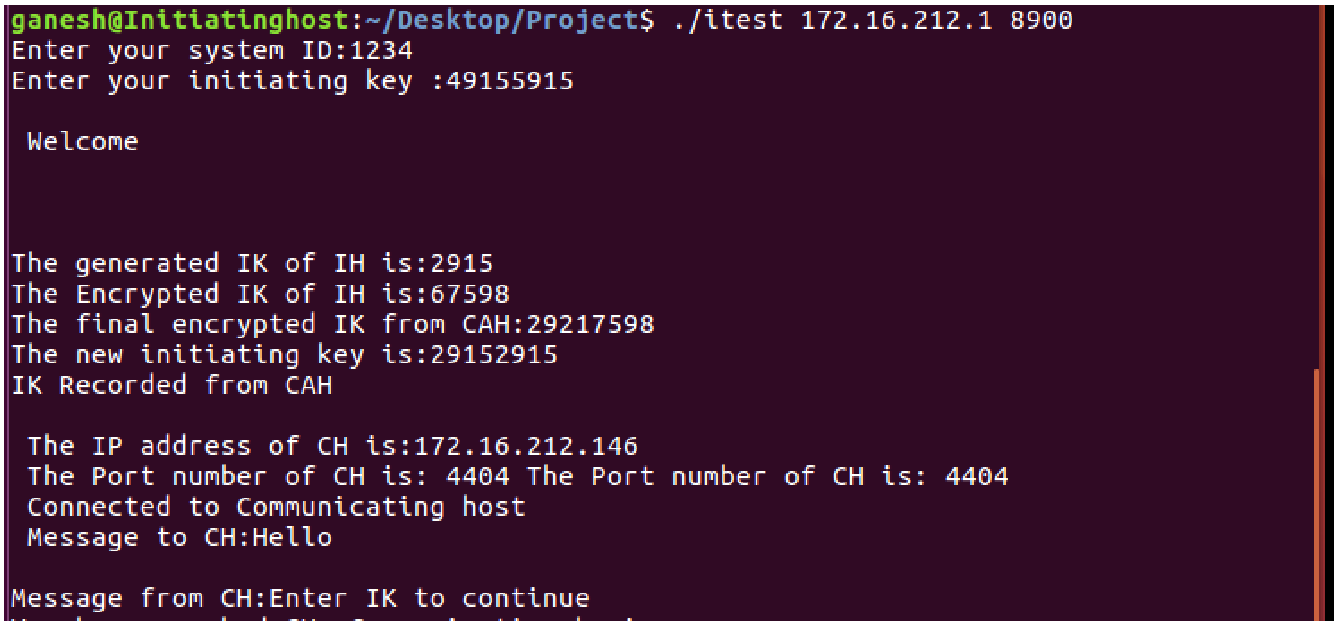


```
ganesh@Communicationhost:~/Desktop/Project$ ./a.out 172.16.212.1 8900
Key for session verification is:29152915
ganesh@Communicationhost:~/Desktop/Project$
 Initiating Host IP address:172.16.212.135
 Message from IH: Hello
```

Figure 42: CH receiving hello message from IH

Code Snippets

```
char bo[255]; \* Initializing the character bo */
bzero(bo,255); \* Clearing the memory for bo */
r=read(newsork,bo,255); \* Reading the message sent by IH. CH uses sock for communicating with CAH and
newsork for listening to IH */
printf("\n Message from IH: %s",bo); \* Print the message */
```



```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :49155915

 Welcome



The generated IK of IH is:2915
The Encrypted IK of IH is:67598
The final encrypted IK from CAH:29217598
The new initiating key is:29152915
IK Recorded from CAH

 The IP address of CH is:172.16.212.146
 The Port number of CH is: 4404 The Port number of CH is: 4404
 Connected to Communicating host
 Message to CH:Hello

Message from CH:Enter IK to continue
```

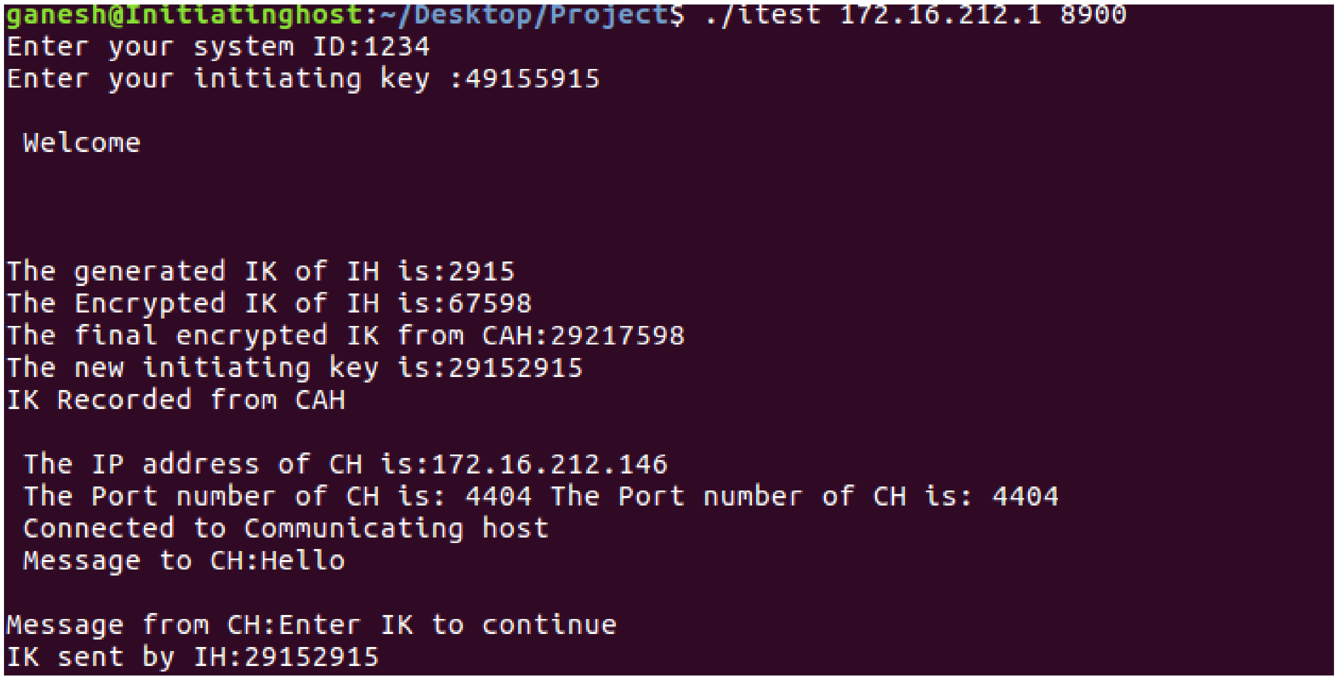
Figure 43: CH requesting IH to send its IK to continue

IH is programmed in such a way that it will automatically send the IH it has generated to CH.

Code Snippets

```
p=read(sork,qz,255); \* IH reading the message sent by CH */
printf("\nMessage from CH:%s\n",qz); \* IH printing the message sent by CH */
```

This is an additional level of security performed at this step. CH will receive the IK fro CAH and it will ask the IH to send its IK such that CH can verify the authenticity of IH.

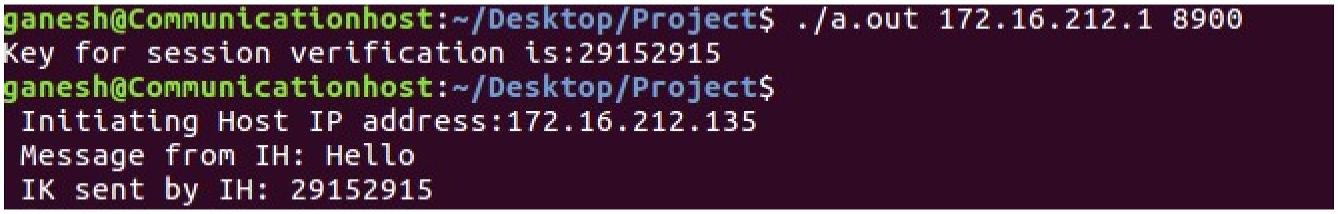


```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :49155915

 Welcome



The generated IK of IH is:2915
The Encrypted IK of IH is:67598
The final encrypted IK from CAH:29217598
The new initiating key is:29152915
IK Recorded from CAH

 The IP address of CH is:172.16.212.146
 The Port number of CH is: 4404 The Port number of CH is: 4404
 Connected to Communicating host
 Message to CH:Hello

Message from CH:Enter IK to continue
IK sent by IH:29152915
```

Figure 44: IH sending its session key or IK to CH for authentication

Code Snippets

```
o=write(sork,&j,sizeof(h)); \* IH writing the IK it has generated with CAH to CH. h was the variable IH used for storing final IK with CAH */
printf("IK sent by IH:%d\n",h);\* IH printing the IK it has sent to IH*/
```



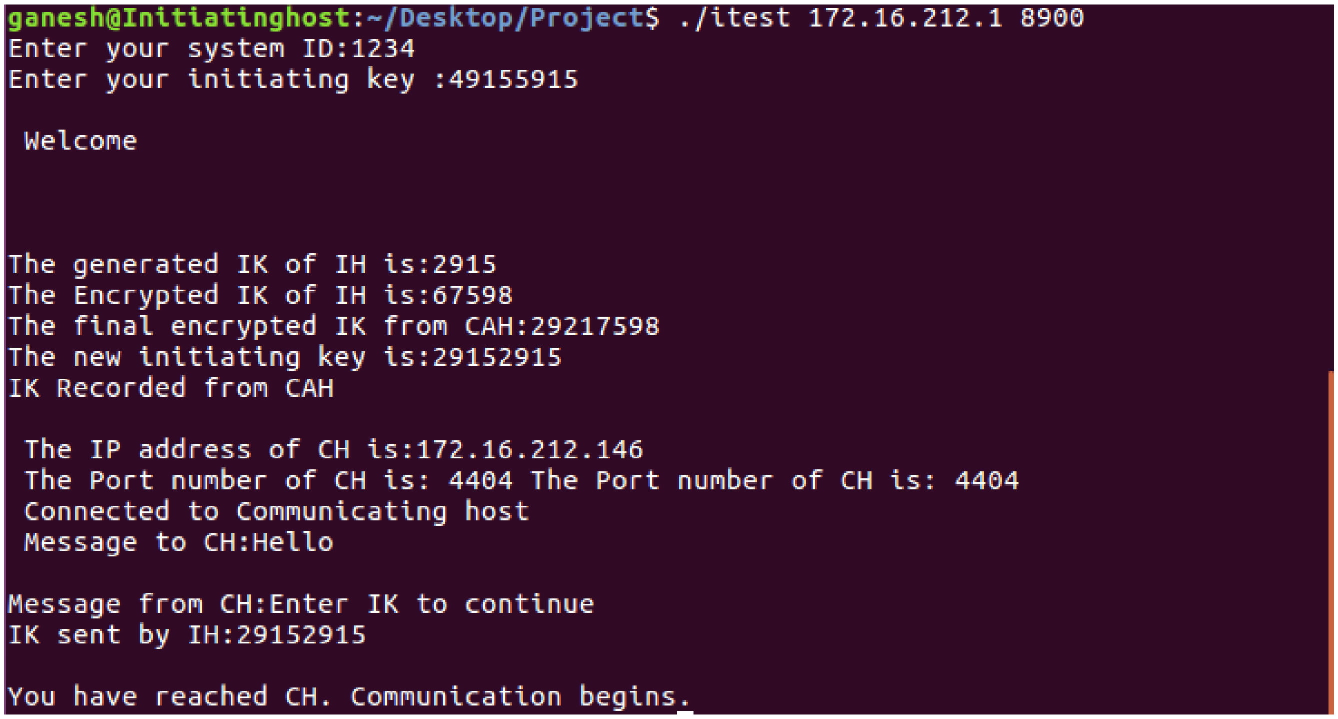
```
ganesh@Communicationhost:~/Desktop/Project$ ./a.out 172.16.212.1 8900
Key for session verification is:29152915
ganesh@Communicationhost:~/Desktop/Project$
 Initiating Host IP address:172.16.212.135
 Message from IH: Hello
 IK sent by IH: 29152915
```

Figure 45: CH displaying IK received from IH

Code Snippets

```
r=read(newsork,&dh,sizeof(dh)); \* CH reading IK sent by IH */
printf("\n IK sent by IH: %d",dh); \* CH printing IK sent by IH */
```

If the IK sent by IH matches with IK sent by CAH, CH and IH starts communication process.

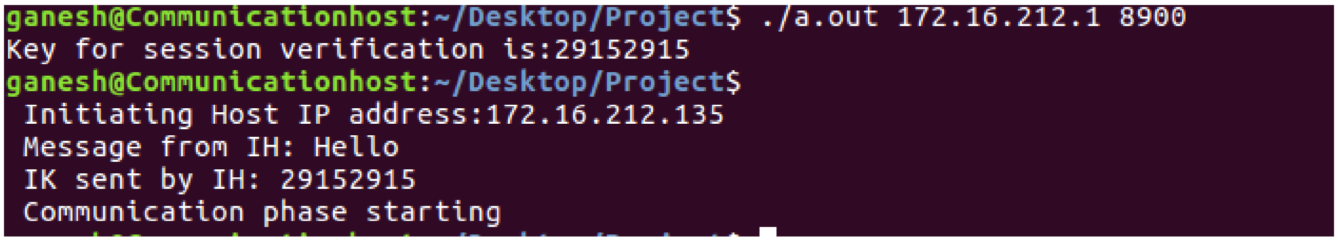


```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :49155915

 Welcome



The generated IK of IH is:2915
The Encrypted IK of IH is:67598
The final encrypted IK from CAH:29217598
The new initiating key is:29152915
IK Recorded from CAH

 The IP address of CH is:172.16.212.146
 The Port number of CH is: 4404 The Port number of CH is: 4404
 Connected to Communicating host
 Message to CH:Hello

Message from CH:Enter IK to continue
IK sent by IH:29152915

You have reached CH. Communication begins.
```

Figure 46: IH entering communication phase

Code Snippets

```
char rz[255];\* Initialising the variable rz */
p=read(sork,rz,255); \* Reading the message sent by CH whetherit can continue communicating or have to
re-register */
printf("\n%s\n",rz);\* Printing the message */
```

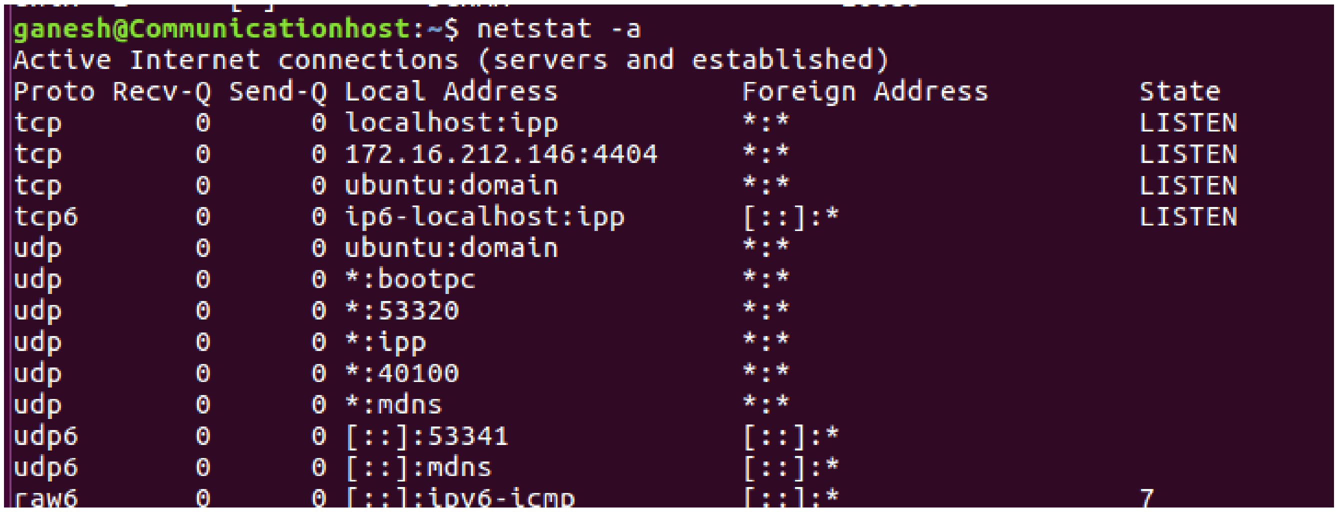


```
ganesh@Communicationhost:~/Desktop/Project$ ./a.out 172.16.212.1 8900
Key for session verification is:29152915
ganesh@Communicationhost:~/Desktop/Project$
 Initiating Host IP address:172.16.212.135
 Message from IH: Hello
 IK sent by IH: 29152915
 Communication phase starting
```

Figure 47: CH entering communication phase

Code Snippets

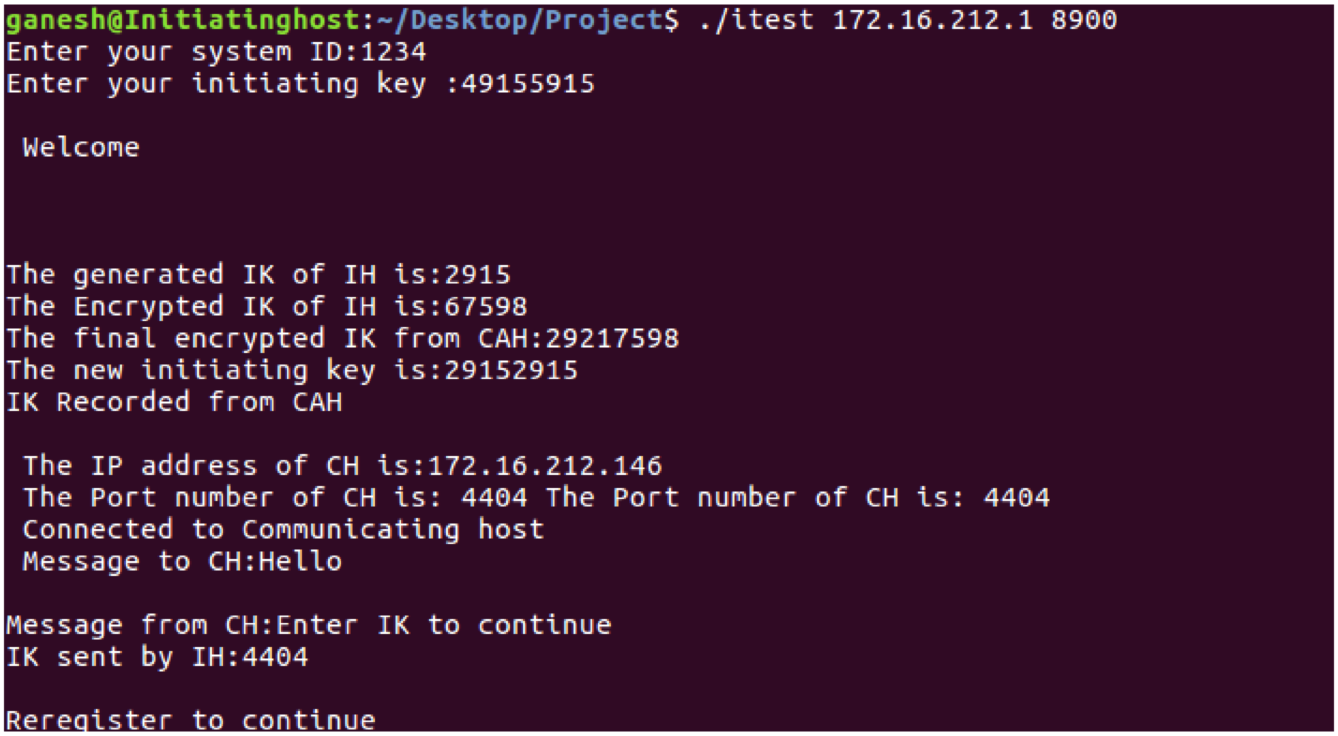
```
if(a==dh) \* a was the variable ch used for storing IK sent by CAH and dh was used by ch for storing IK
sent by IH. If both matches */
{
char cz[255]="You have reached CH. Communication begins."; \* Initializing the cz */
r=write(newsork,cz,255); \* Write message we are communicating to IH */
printf("\n Communication phase starting"); \* Printing the message and still listening on socket */
}
else \* If IK does not match */
{
char by[255]="Reregister to continue"; \* Initialise re-register message*/
r=write(newsork,by,255); \* Write message re-register to IH */
close(sork); \* Close the socket avoiding further communication */
}
```



```
ganesh@Communicationhost:~$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 172.16.212.146:4404     *:*                     LISTEN
tcp        0      0 ubuntu:domain           *:*                     LISTEN
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
udp        0      0 ubuntu:domain           *:*
udp        0      0 *:bootpc                *:*
udp        0      0 *:53320                 *:*
udp        0      0 *:ipp                   *:*
udp        0      0 *:40100                 *:*
udp        0      0 *:mdns                  *:*
udp6       0      0 [::]:53341              [::]:*
udp6       0      0 [::]:mdns               [::]:*
raw6       0      0 [::]:ipv6-icmp          [::]:*                  7
```

Figure 48: Port status of CH after successful IK verification

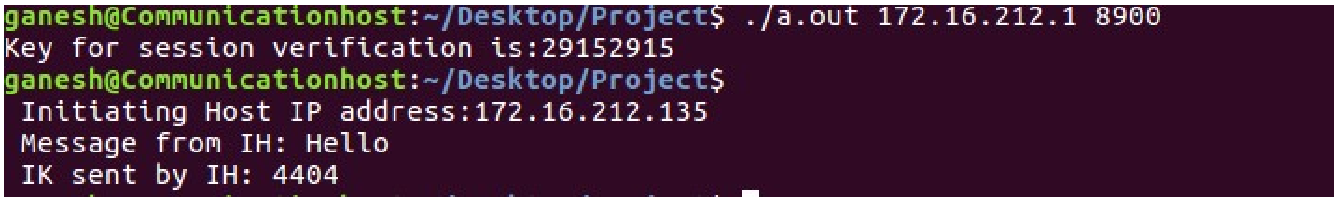
Suppose if there is a bug in IH program and if IH sends a different IK compared to what CAH has sent to CH, then CH will send re-register IK message to IH to enter communication process. If IH receives this message, CH will close its listening socket, IH has to restart the whole process again to start communicating with CH.



```
ganesh@Initiatinghost:~/Desktop/Project$ ./itest 172.16.212.1 8900
Enter your system ID:1234
Enter your initiating key :49155915

 Welcome



The generated IK of IH is:2915
The Encrypted IK of IH is:67598
The final encrypted IK from CAH:29217598
The new initiating key is:29152915
IK Recorded from CAH

 The IP address of CH is:172.16.212.146
 The Port number of CH is: 4404 The Port number of CH is: 4404
 Connected to Communicating host
 Message to CH:Hello

Message from CH:Enter IK to continue
IK sent by IH:4404

Reregister to continue
```

Figure 49: IH after unsuccessful IK verification

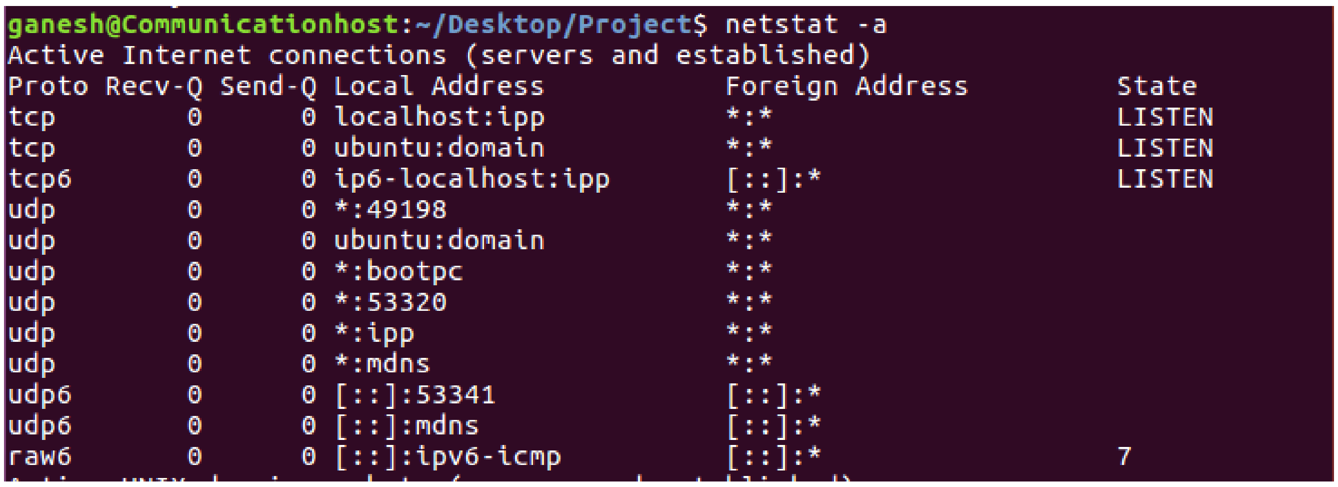
CH displaying the IK sent by IH.



```
ganesh@Communicationhost:~/Desktop/Project$ ./a.out 172.16.212.1 8900
Key for session verification is:29152915
ganesh@Communicationhost:~/Desktop/Project$
 Initiating Host IP address:172.16.212.135
 Message from IH: Hello
 IK sent by IH: 4404
```

Figure 50: CAH after unsuccessful IK verification:

We can see that CH closing its listening socket abruptly after IK sends a different IK. In this way, CH also eliminates unauthorized entry for hackers. Though CAH stops attacker at the first place, CH also has additional security functionality to stop the intruders from performing DDOS on it.

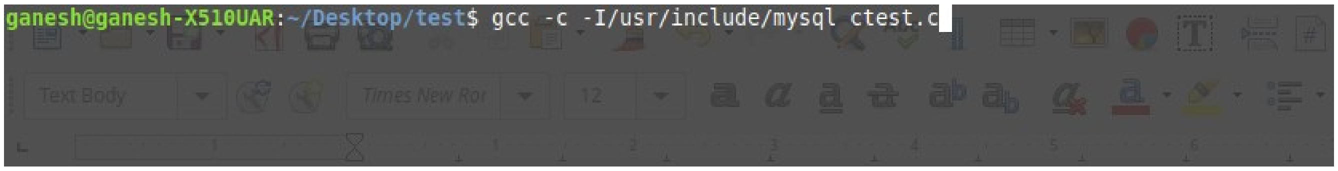


```
ganesh@Communicationhost:~/Desktop/Project$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 ubuntu:domain           *:*                     LISTEN
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
udp        0      0 *:49198                 *:*
udp        0      0 ubuntu:domain           *:*
udp        0      0 *:bootpc                *:*
udp        0      0 *:53320                 *:*
udp        0      0 *:ipp                   *:*
udp        0      0 *:mdns                  *:*
udp6       0      0 [::]:53341              [::]:*
udp6       0      0 [::]:mdns               [::]:*
raw6       0      0 [::]:ipv6-icmp          [::]:*                  7
```

Figure 51: Port status of CH after unsuccessful IK verification

7 APPENDIX

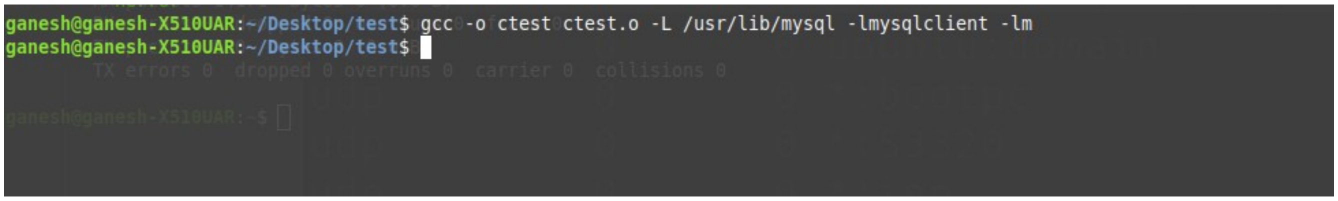
The screenshot for compiling cah program by including mysql library



```
ganesh@ganesh-X510UAR:~/Desktop/test$ gcc -c -I/usr/include/mysql ctest.c
```

Figure 52: Compiling CAH program

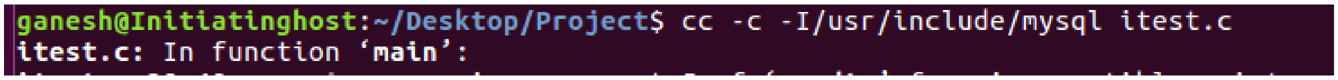
The screenshot for compiling cah program by including mysql libraries. Lm library is for math function.



```
ganesh@ganesh-X510UAR:~/Desktop/test$ gcc -o ctest ctest.o -L /usr/lib/mysql -lmysqlclient -lm
ganesh@ganesh-X510UAR:~/Desktop/test$ 
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ganesh@ganesh-X510UAR:~$ 
```

Figure 53: Compiling CAH program

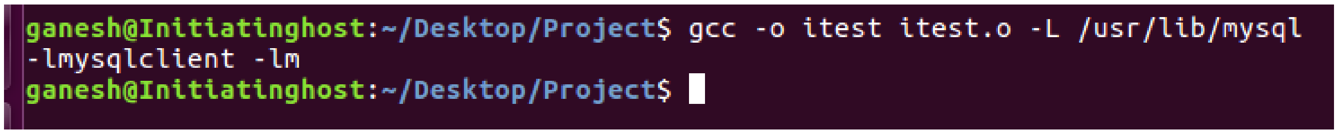
The screenshot for compiling IH program by including mysql library



```
ganesh@Initiatinghost:~/Desktop/Project$ cc -c -I/usr/include/mysql itest.c
itest.c: In function ‘main’:
```

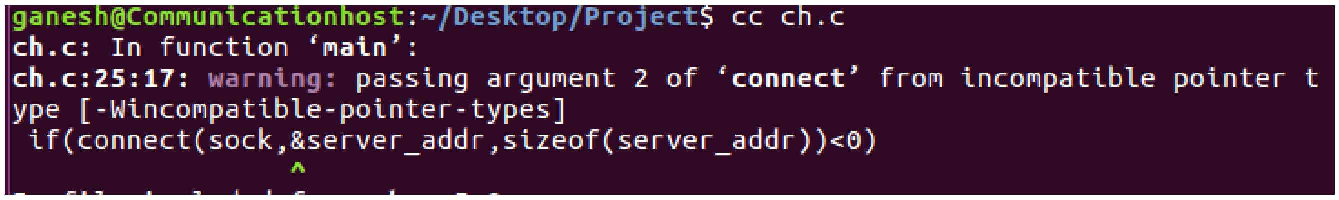
Figure 54: Compiling IH program

The screenshot for compiling IH program by including mysql and lm library. Lm library is used for math function



```
ganesh@Initiatinghost:~/Desktop/Project$ gcc -o itest itest.o -L /usr/lib/mysql
-lmysqlclient -lm
ganesh@Initiatinghost:~/Desktop/Project$ 
```

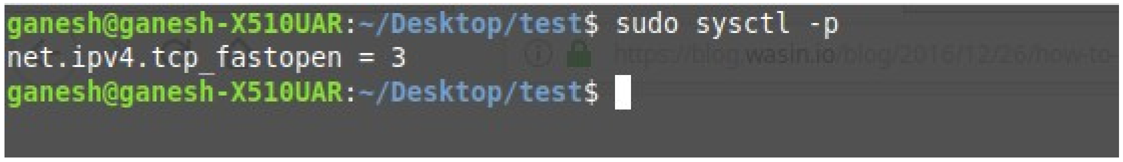
Figure 55: Compiling IH program



```
ganesh@Communicationhost:~/Desktop/Project$ cc ch.c
ch.c: In function ‘main’:
ch.c:25:17: warning: passing argument 2 of ‘connect’ from incompatible pointer t
ype [-Wincompatible-pointer-types]
 if(connect(sock,&server_addr,sizeof(server_addr))<0)
                 ^
```

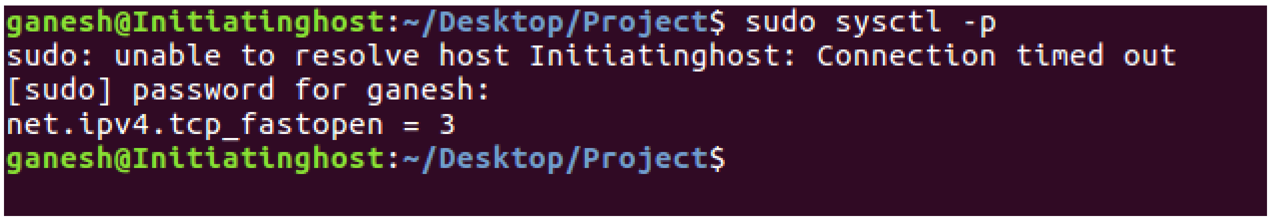
Figure 56: Compiling CH program

The TCP fast open value as 3 means CAH can act both as server and client. TCP Fast open is enabled only for linux kernels 3.7 or above



```
ganesh@ganesh-X510UAR:~/Desktop/test$ sudo sysctl -p
net.ipv4.tcp_fastopen = 3
ganesh@ganesh-X510UAR:~/Desktop/test$
```

Figure 57: How to enable TCP fast open in CAH



```
ganesh@Initiatinghost:~/Desktop/Project$ sudo sysctl -p
sudo: unable to resolve host Initiatinghost: Connection timed out
[sudo] password for ganesh:
net.ipv4.tcp_fastopen = 3
ganesh@Initiatinghost:~/Desktop/Project$
```

Figure 58: How to enable TCP fast open in IH

8. FUTURE IMPROVEMENTS

In the handover phase, CAH receives the IP address and port number details from CH and sends it over its listening socket to IH such that IH uses them to create a connection to CH. When the connection details are shared over internet, it is vulnerable to hacker.

The attacker might craft TCP syn flood after hacking the connection detailed packets and perform DDOS attack directly on CH. In order to avoid this, SSH(Secured shell) can be implemented on all three hosts and end-to-end data encryption can be implemented for all data transfers such that all the data these three hosts are communicating over internet are encrypted and the hackers might take years to decrypt them to perform DDOS attack.

9. CONCLUSION

In this way a system which performs DDOS elimination using four phases is designed. Though there are many systems present currently which can eliminate DDOS attack from the internet, the proposed system is a standalone approach because it embeds payload in SYN packet using TCP fast open and eliminates DDOS at the first place by avoiding TCP SYN flood attack by sending the RST packet when it receives SYN packet without systemID or unregistered systemID.

The system also adds additional security by placing Communication authentication host (CAH) at the middle by moving point of attack to CAH. Though it might look CAH is vulnerable to SYN flood attack, CAH uses intelligent decision strategies at the first place to avoid SYN flood attack. After performing authentication CAH performs a clean handover to CH for IH and creates proper environment for CH and IH to communicate with each other.

Though the proposed system is implemented only for console based gaming systems like Xbox, PlayStation etc, its functionality can be later implemented in real-time web applications and other systems too.

10. LIST OF FIGURES

Illustration Index

11. REFERENCES

https://www.youtube.com/watch?v=CMDBF84vSRk&index=4&list=PLPyaR5G9aNDvs6TtdpLcVO43_jvxp4emIhttps://www.youtube.com/watch?v=Ts8eXOkx8TE&list=PLPyaR5G9aNDvs6TtdpLcVO43_jvxp4emI&index=5

https://www.youtube.com/watch?v=DboEGcU6rLI&index=6&list=PLPyaR5G9aNDvs6TtdpLcVO43_jvxp4emI
http://www.kitebird.com/mysql-book/ch06-3ed.pdf
https://www.youtube.com/watch?v=jACHG6tZakw&t=1159s
https://www.youtube.com/watch?v=vgs1eQJ1avs
https://codereview.stackexchange.com/questions/126812/multithreaded-client-socket
https://www.geeksforgeeks.org/computer-network-tcp-3-way-handshake-process/
https://lwn.net/Articles/508865/

https://stackoverflow.com/questions/5791860/beginners-socket-programming-in-c

https://stackoverflow.com/questions/30079248/how-to-activate-the-tcp-fast-open-in-linux

https://blog.wasin.io/blog/2016/12/26/how-to-enable-fast-tcp-open-on-ubuntu.html

https://c-program-example.com/2012/04/c-program-to-encrypt-and-decrypt-a-password.html