PROJECT REPORT

Group Members:

Levent Sarıoğlu 2102183
Barış Balkaya 2105157
Kerem Düşünmez 2104282

Levent had been running Linux Mint 21.1, Barış and Kerem were on Windows10 during the project.

We met online using Discord and did pair programming to complete the project. We used ZeroTier to create a virtual network among group members, so we could run the program over the Internet as it was running on a virtual LAN. We have met several times and coded together in several hour-long sessions, the workload was divided equally.

Faced Challenges:

Challenge #1 locked_input:

Since Windows and Linux have different ways to deal with the threading, getting user input in different threads created a challenge for us. Linux had no problems while getting input from threads, but it did not work for windows threads; windows would exit the program when the **input()** function was being called from different threads. We overcame this challenge by using a **locked_input** function instead of the regular input. This is due to the nature of input function, which is created to run from the main thread normally.

Challenge #2 Finding the user IP address:

We had to write a linear search function **get_ip_adress** and use it in chat functions of Chat_Initiator so that we can find the user IP addresses by using the usernames. Since in our dictionary; IP addresses are keys and usernames are values. We had to implement it this way to find IP addresses.

Challenge #3 inputflag and simulating Enter:

Since in our program's structure, Chat_Responder is mostly a listener and Chat_Initiator is mostly a sender, we had a hard time implementing a public key exchange function due to nature of the functions. We had to figure out a way to stop Chat_Initiator from running and asking for “Enter an action” input when someone initiated secure chat by sending a public key. To solve this problem, we used a variable called inputflag to stop the Chat_initiator from running and initiating the secure chat instead. We still needed to press enter to get to the “Enter a secret key” prompt, we figured out this problem by simulating the "Enter" key with an empty value using the pynput library.

Challenge #4 Extra Challenges:

We faced some minor challenges. One of them was creating an Exit function which would also work properly for Linux. Windows did not have this problem, but Linux had the ports open for several seconds even after closing the program, this prevented us from restarting the program without waiting for a minute. We solved this by adding a **signal_handler()** function and closing the sockets after that exiting the program by using **os.exit()**. We also used **time.sleep()** function to make the program correctly with proper wait times.