

COMPUTER NETWORKS

MINIPROJECT

TOPIC :- Text message encrypting and decrypting in Morse code

BY:

DHANUVARSHA S S - PES2UG22CS130

BHAVIKA PRADEEP - PES2UG22CS130

**ABOUT THE PROJECT**

The project consists of two parts: a server script and a graphical user interface (GUI) client script.

Server Script:

The server script creates a socket and listens for client connections on localhost(as well as ip address) at port 12345.

It defines a dictionary for Morse code translation and provides functions for encryption (text to Morse) and decryption (Morse to text).

The server handles incoming client connections, wraps the socket with SSL for secure communication, and uses a separate thread to handle each client.

The server receives Morse code from clients, translates it using the defined dictionary, and sends the translated text back to the clients.

Client GUI Script:

The client script uses Tkinter to create a simple GUI for Morse code translation.

It includes a text entry for users to input text to be encoded or decoded.

Upon clicking the "Translate" button, the client connects to the server using sockets and SSL encryption.

It sends the entered Morse code to the server, receives the translated text, and displays it in the GUI. The server uses multithreading to handle multiple clients concurrently. The client-server communication is secured using SSL/TLS encryption

CODE SNIPPETS:

client.py

from tkinter import \*

from socket import \*

import ssl

def send\_receive\_morse():

    morse\_text = morse\_entry.get()

    # Connect to the server

    clientSocket = socket(AF\_INET, SOCK\_STREAM)

    clientSocket.connect(('192.168.66.167', 12345))

    # Wrap the socket with SSL

    context = ssl.create\_default\_context(ssl.Purpose.SERVER\_AUTH)

    context.options |= ssl.OP\_NO\_TLSv1 | ssl.OP\_NO\_TLSv1\_1  # Disable older TLS versions

    clientSocket = context.wrap\_socket(clientSocket, server\_hostname='muthu')

    # Send input

    clientSocket.send(morse\_text.encode())

    # Receive output

    translated = clientSocket.recv(2000)

    # Display result in the GUI

    result\_label.config(text='Translated from server: ' + translated.decode())

    # Close the connection

    clientSocket.close()

# Create the main window

window = Tk()

window.title("Morse Code Translator")

window.geometry("500x500")

# Create GUI components

morse\_label = Label(window, text="Input text to be encoded/decoded:",font=('Arial', 14))

morse\_entry = Entry(window,font=('Arial', 14))

translate\_button = Button(window, text="Translate", command=send\_receive\_morse,font=('Arial', 14))

result\_label = Label(window, text="Translated from server:",font=('Arial', 14))

# Place components in the window

morse\_label.pack(pady=10)

morse\_entry.pack(pady=10)

translate\_button.pack(pady=10)

result\_label.pack(pady=10)

# Start the GUI main loop

window.mainloop()

server.py

from socket import \*

import socket

from threading import Thread

import ssl

# morse code dictionary

Dictionary = {

    'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.', 'F': '..-.', 'G': '--.',

    'H': '....', 'I': '..', 'J': '.---', 'K': '-.-', 'L': '.-..', 'M': '--',

    'N': '-.', 'O': '---', 'P': '.--.', 'Q': '--.-', 'R': '.-.', 'S': '...',

    'T': '-', 'U': '..-', 'V': '...-', 'W': '.--', 'X': '-..-', 'Y': '-.--',

    'Z': '--..', '1': '.----', '2': '..---', '3': '...--', '4': '....-',

    '5': '.....', '6': '-....', '7': '--...', '8': '---..', '9': '----.', '0': '-----',

    ', ': '--..--', '.': '.-.-.-', '?': '..--..', '/': '-..-.', '-': '-....-',

    '(': '.-.-.', ')': '-.--.'

}

def handle\_client(connectionSocket):

    while True:

        inp = connectionSocket.recv(2000).decode()

        if not inp:

            break

        else:

            print("Client is requesting:",inp)

            if all(char in {'.', '-', ' '} for char in inp):

                translated = decrypt(inp.upper())

            else:

                translated = encrypt(inp.upper())

            connectionSocket.send(translated.encode())

    connectionSocket.close()

def encrypt(msg):

    cipher = ''

    for letter in msg:

        if letter != ' ':

            cipher += Dictionary[letter] + ' '

        else:

            cipher += ' '

    return cipher

def decrypt(msg):

    msg += ' '

    decipher = ''

    citext = ''

    for letter in msg:

        if letter != ' ':

            i = 0

            citext += letter

        else:

            i += 1

            if i == 2:

                decipher += ' '

            else:

                decipher += list(Dictionary.keys())[list(Dictionary.values()).index(citext)]

                citext = ''

    return decipher

def main():

    server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

    server\_socket.bind(('localhost', 12345))

    server\_socket.listen()

    print("Server is listening for connections...")

    while True:

        client\_socket, \_ = server\_socket.accept()

        print("Client connected.")

        # Wrap the client socket with SSL

        context = ssl.create\_default\_context(ssl.Purpose.CLIENT\_AUTH)

        context.load\_cert\_chain(certfile='server.crt', keyfile='private.key')

        context.options |= ssl.OP\_NO\_TLSv1 | ssl.OP\_NO\_TLSv1\_1  # Disable older TLS versions

        client\_socket = context.wrap\_socket(client\_socket, server\_side=True)

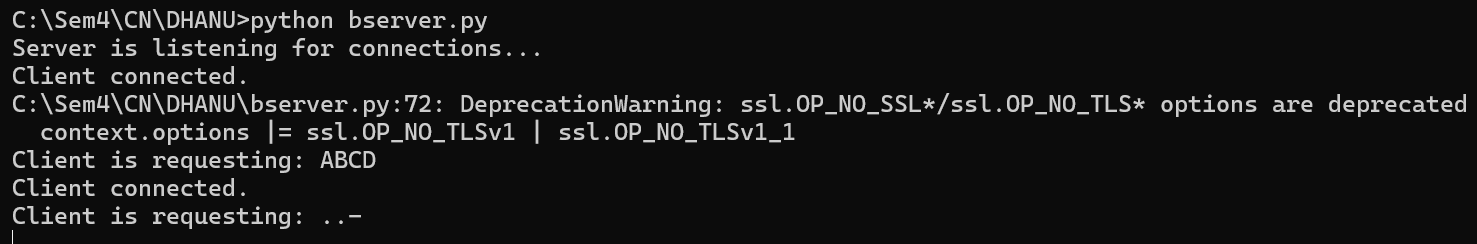
        handle\_client(client\_socket)

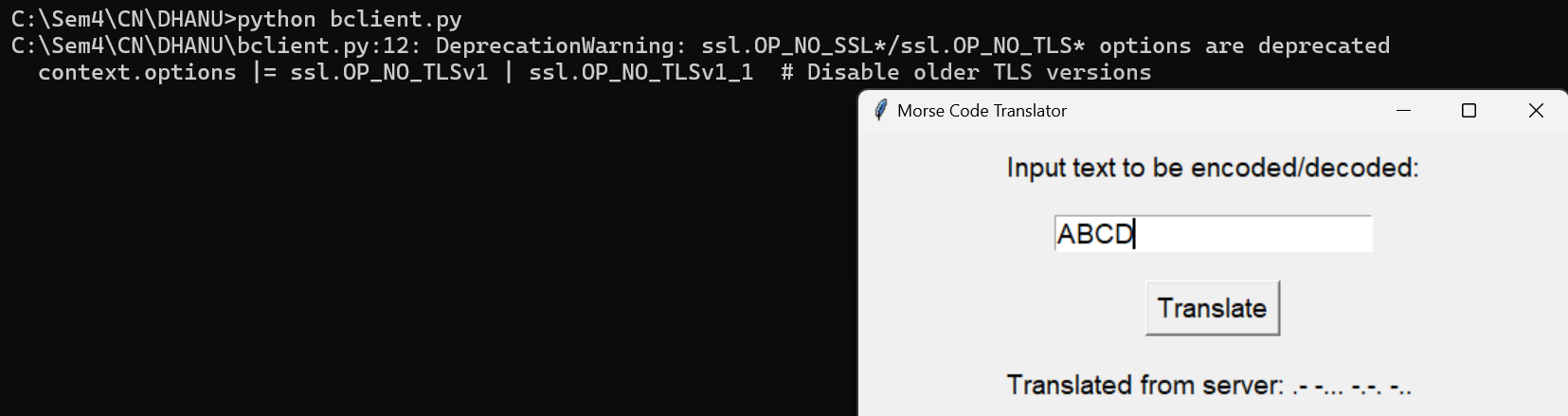
        client\_socket.close()

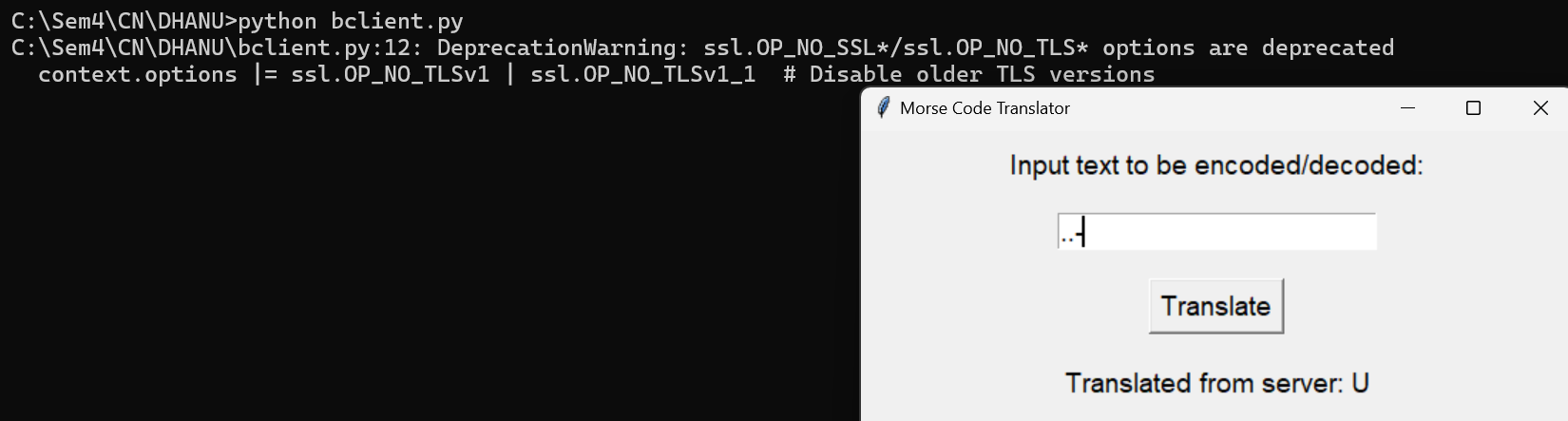
if \_\_name\_\_ == "\_\_main\_\_":

    main()

OUTPUT:







THANK YOU.