Q1 Authenticate Using Iris Code

20 Points

Write a program (in any language you like) for a system that uses **Iris code** to authenticate people.

It should have 2 functionalities (for 2 phases!):

1. **Enrollment**: Allow the user to record some iris data (iris code **in hex** with the corresponding person's name).
2. **Recognition**: Use the database stored to authenticate people.
   For example, given an iris code X (in hex) that is claimed to be Alice's, your program can...
   - get Alice's Iris code that's stored in the database,
   - then compute the Hamming distance between X & Alice's Iris code,
   - and finally, use the distance to decide whether to give access or not (i.e. whether X is indeed Alice's Iris or not) based on the accepted-match requirement stated on L14 P13.

Note that the Hamming distance is between 2 binary data. So remember to **convert hex to binary** first.
The program's workflow is up to you. But these 2 functionalities both require taking user input. Either use the I/O mechanism (like a scanner) or use command line arguments.

Q1.1 Iris Code Code
17 Points

Upload the code of your authentication program.

iris.py Download

```
"""

Problem 1: Authenticate Using Iris Code (hamming distance)

Sources:

Pandas for database: https://pandas.pydata.org/
Checking if Iris data csv file exists: https://www.geeksforgeeks.org/python/python-os-path-exists-method/

"""

import pandas as pd
import os

IRIS_DATA_FILE = "iris_data.csv"

if os.path.exists(IRIS_DATA_FILE):
    df = pd.read_csv(IRIS_DATA_FILE)
    iris_data = dict(zip(df['Name'], df['Iris Code']))
else:
    iris_data = {}
    df = pd.DataFrame(columns=['Name', 'Iris Code'])

def save_iris_data():
    df = pd.DataFrame(list(iris_data.items()), columns=['Name', 'Iris Code'])
    df.to_csv(IRIS_DATA_FILE, index=False)

option = int(input("Enter 1 to enroll, 2 to authenticate, 3 to delete a user's iris data: "))

match option:
  case 1:
    while True:
      name = input("Enter your name: ")
      iris_input = input("Enter your iris code in hex: ")
      if name in iris_data.keys():
        print("You are already enrolled")
        break
      elif iris_input in iris_data.values():
        print("Matching iris data already in database, two people cannot have the same iris data.")
        break

      iris_data[name] = iris_input
      save_iris_data()
      print(f"{name}'s iris code has been saved to database.")
      choice = input("Enroll another person? (yes/no): ")
      if choice == 'no':
        break
  case 2:
    while True:
        name = input("Enter your name: ")
        iris_input = input("Enter your iris code you are trying to authenicate in hex: ")
        if name not in iris_data.keys():
            print("You are not enrolled yet, please do that before entering recognition phase")
            break
        given_iris = iris_data[name]

        bit_len = len(iris_input)*4
        input_bin = bin(int(iris_input,16))[2:].zfill(bit_len)

        given_bit_len = len(given_iris)*4
        given_bin = bin(int(given_iris,16))[2:].zfill(given_bit_len)

        if (bit_len == given_bit_len):
            count = 0
            comp_list = list(zip(input_bin, given_bin))
            ham = 0
            for x in comp_list:
                if (x[0] != x[1]):
                    count = count + 1
                    print(input_bin)
                    print(given_bin)

                    ham = (count/bit_len)
                    print(ham)

            if ham < 0.32:
                print("Recognition successful")
            else:
                print("Recognition failed")

            choice = input("Would you like to do another recognition? (yes/no): ")
            if choice == 'no':
                break
        else:
            print("ERROR: wrong sized input given")
            break
  case 3:
    name = input("Enter the name of the user you want to delete iris data of: ")
    if name in iris_data:
        del iris_data[name]
        save_iris_data()
        print(f"{name}'s iris data has been deleted.")
    else:
        print("User not found.")
  case _:
    print("Invalid input")


```

Q1.2 Hamming Distance 1
1 Point

Use your program to record the following iris codes from Alice & Bob (in hex).

- **Alice: 9CF8CD32**
- **Bob: E99D7B76**

Then, try to authenticate...
Iris code X that claim to be **Alice**s': **8CD9F911**

Hamming distance (2 decimal places please):

0.28

Give access?

Yes

No

Q1.3 Hamming Distance 2
1 Point

Using the same database to authenticate...
Iris code Y that claim to be **Bob**s': **D2DE6B62**

Hamming distance (2 decimal places please):

0.34

Give access?

Yes

No

Q1.4 Hamming Distance 3
1 Point

Using the same database to authenticate...
Iris code Z that claim to be **Bob**s': **FDBC6954**

Hamming distance (2 decimal places please):

0.25

Give access?

Yes

No

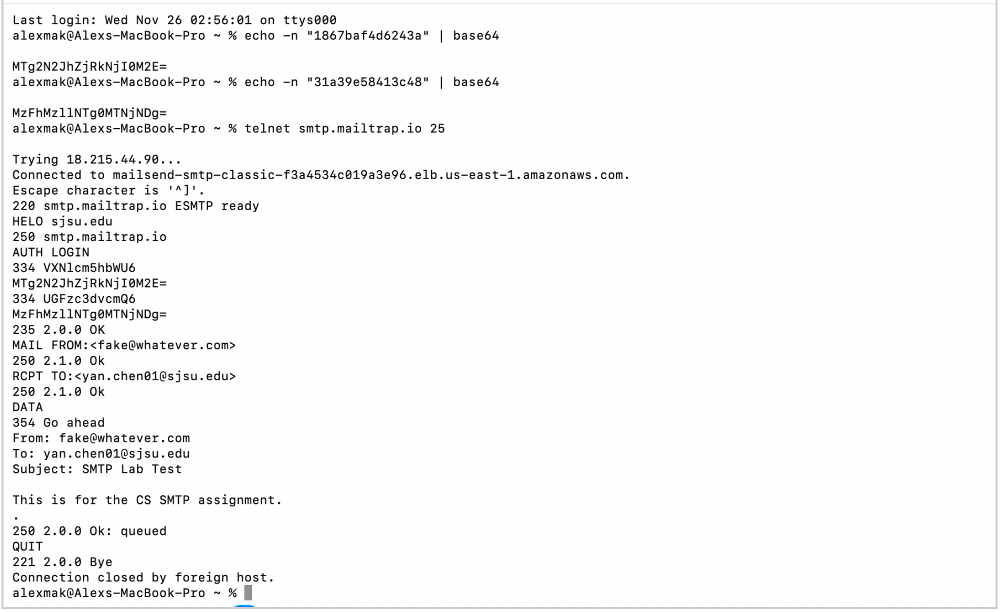
Q2 SMTP
15 Points

In your **terminal/command line**, open a **telnet session** on port 25 on your **SMTP server** and send your instructor (yan.chen01@sjsu.edu) a forged/spoofed e-mail with a fictitious sender. You will use the SMTP commands such as `MAIL FROM:`, `RCPT TO:`, `DATA`, `QUIT`, etc.

Hint: you can use any free SMTP email service such as SendGrid, Amazon SES, etc. And it may not be too easy to send an email from a fictitious sender. So as long as it can hide your actual name, it's fine (you can register another email, use a temporary email address (which may not support sending emails though), etc.)

Please upload screenshots of the code you used to send the email on your **terminal**. Remember to include the code from opening the telnet session until you get the 250 OK message (you can cover/crop your password though). If you can't finish the whole process, submit the screenshot(s) of what you've done.

Programing assignment2.png Download



```
Last login: Wed Nov 26 02:56:01 on ttys000
alexmak@Alexs-MacBook-Pro ~ % echo -n "1867baf4d6243a" | base64

MTg2N2JhZjRkNjI0M2E=
alexmak@Alexs-MacBook-Pro ~ % echo -n "31a39e58413c48" | base64

MzFhMzllNTg0MTNjNDg=
alexmak@Alexs-MacBook-Pro ~ % telnet smtp.mailtrap.io 25

Trying 18.215.44.90...
Connected to mailsend-smtp-classic-f3a4534c019a3e96.elb.us-east-1.amazonaws.com.
Escape character is '^]'.
220 smtp.mailtrap.io ESMTP ready
HELO sjsu.edu
250 smtp.mailtrap.io
AUTH LOGIN
334 VXNlcm5hbWU6
MTg2N2JhZjRkNjI0M2E=
334 UGFzc3dvcmQ6
MzFhMzllNTg0MTNjNDg=
235 2.0.0 OK
MAIL FROM:<fake@whatever.com>
250 2.1.0 Ok
RCPT TO:<yan.chen01@sjsu.edu>
250 2.1.0 Ok
DATA
354 Go ahead
From: fake@whatever.com
To: yan.chen01@sjsu.edu
Subject: SMTP Lab Test

This is for the CS SMTP assignment.
.
250 2.0.0 Ok: queued
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
alexmak@Alexs-MacBook-Pro ~ %
```

Programming Assignment 2

● Graded

Select each question to review feedback and grading details.

Group

Brendan Ly
Tiernan Johnson
Rongjie Mai
View or edit group

Total Points

34 / 35 pts

Question 1

	Authenticate Using Iris Code	**20** / 20 pts
1.1	Iris Code Code	**17** / 17 pts
1.2	Hamming Distance 1	**1** / 1 pt
1.3	Hamming Distance 2	**1** / 1 pt
1.4	Hamming Distance 3	**1** / 1 pt

Question 2

SMTP	**14** / 15 pts