University of Sheffield

Assignment 2 System Security Report

Jagpreet

Department of Computer Science

January 20, 2023

Contents

List of Figures

Chapter 1

Continuous Authentication at the University of Sheffield

1.1 Outline a continuous biometric authentication system implementing the above. Provide a diagram indicating major components of the system and explain what they do.

A continuous biometric authentication system using keyboard and mouse can be composed of the following major components:

- Keyboard and Mouse Sensors: These are the physical components that detect the user's typing and mouse movements. They are connected to the computer and send data to the system for analysis. Webcams may also be used to create a Biometric Profile which provide no interference [1].
- Data Collection and Preprocessing: This component collects the data from the sensors and preprocesses it for analysis. It may include filtering out noise or irrelevant data, and transforming the data into a format that is suitable for the machine learning algorithm.
- Biometric Profile Creation: This component uses machine learning algorithms to analyze the user's typing and mouse movements, creating a biometric profile that represents the user's unique patterns. The profile is then stored in a database for later use.
- Authentication: This component compares the current typing and mouse movements to the stored biometric profile in order to confirm the user's identity. If the user's typing and mouse movements match their stored profile, the system grants access.
- Database: This component stores the biometric profiles of all users. The profiles can be updated if there is consistent change in behaviour of user with time.
- A simplified diagram of the system might look like this:

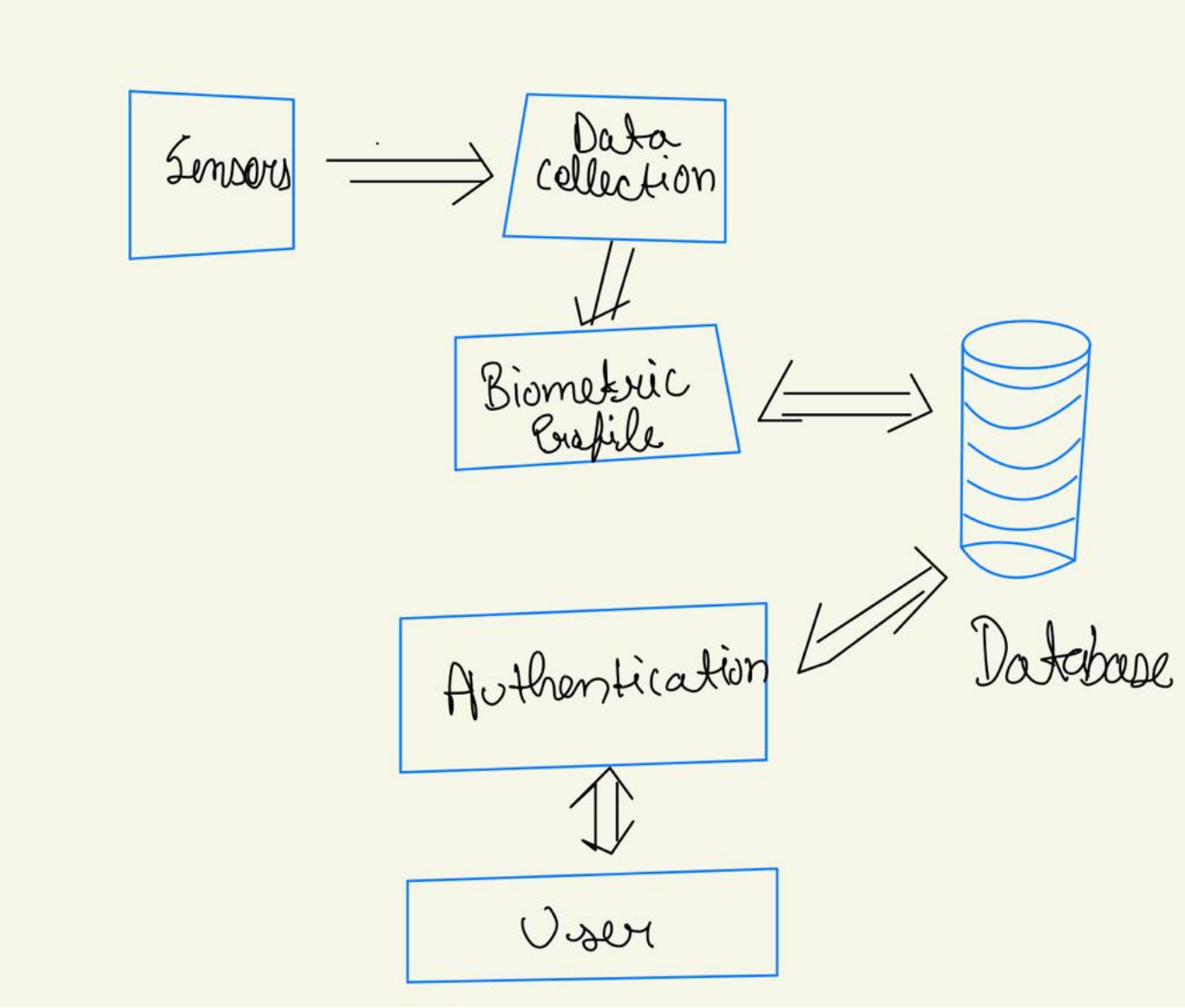

Figure 1.1: *Cntinuous biometric authentication*

1.2 Identify suitable behavioural features and why you believe they are suitable.

Suitable behavioral features for a continuous biometric authentication system using keyboard and mouse might include:

- Typing rhythm[2]: This refers to the timing between key presses and releases. Each person has a unique rhythm of typing, which can be captured and analyzed by the system.
- Keystroke dynamics[2]: This refers to the way a person types, including the duration of key presses, the pressure applied to keys, and the flight time between key presses. Keystroke dynamics can also be used to create a unique biometric profile.
- Mouse movement patterns[3]: This refers to the way a person moves their mouse, including the speed and direction of movement, the duration of clicks, and the number of clicks.
- Mouse dynamics[3]: This refers to the way a person presses and releases the mouse buttons, including the duration of button presses and the pressure applied.
- Mouse cursor movement: This refers to the way a person moves the cursor, including the speed and direction of movement, and the duration of cursor stops.

  These features are suitable because they are unique to each individual and can be captured and analyzed by the system. They are also relatively difficult to imitate or replicate, which makes them more secure than other types of biometric features. Additionally, these features are not static and can be continuously captured providing a continuous authentication.

1.3 Recommend an ML classification technique that could form the basis of the authentication system.

One ML classification technique that could form the basis of a continuous biometric authentication system using keyboard and mouse is Support Vector Machines (SVMs).

SVMs are a powerful supervised learning algorithm that can be used for classification and regression tasks. They work by finding the hyperplane in a high-dimensional space that maximally separates different classes. In the case of biometric authentication, the different classes would be the different users, and the features used for classification would be the behavioral features such as typing rhythm, keystroke dynamics, and mouse movement patterns.

SVMs have several advantages that make them well-suited for biometric authentication systems:

- High accuracy: SVMs have been shown to have high classification accuracy in a variety of applications, including biometric authentication.
- Robustness to noise: SVMs are relatively robust to noise in the data, which can be an issue in biometric authentication systems where there may be variations in the way users type or move their mouse.

- Handling of non-linearly separable data: SVMs can handle non-linearly separable data by using kernel functions, which can transform the data into a higher-dimensional space where it is linearly separable.
- Handling of high-dimensional data: SVMs can handle high-dimensional data, which is important in biometric authentication systems where there may be many different features used for classification.
- SVM can be continuously retrained with the new data, providing the flexibility to adapt to the changes in the user typing and mouse movement patterns.

1.4 Indicate whether you consider the use of the proposed approach to be ethically justified, and why (or why not)

The use of a continuous biometric authentication system using keyboard and mouse can not be ethically justified for conducting practical exams due to the following factors:

- This system may increase the stress on students affecting their performance due to possibility of a false negative(i.e. rejection of a legitimate student).
- The potential privacy implications of using biometric data for authentication. For example, users may be concerned about the collection, storage, and use of their typing and mouse movement data.
- Additionally, it is important to consider the possible biases that may be introduced in the system. For example, typing rhythms, keystroke dynamics, and mouse movement patterns may vary based on factors such as age, gender, or physical abilities. These variations could lead to false negatives (rejecting legitimate users) or false positives (granting access to imposters) for certain groups of users.

However, the proposed approach can provide an additional layer of security beyond traditional username and password authentication. The use of a continuous biometric authentication may be ethically justified if it is implemented in a way that respects the privacy and security of users, with transparency, control over their data, and bias minimization.

1.5 Present a reasoned view as to whether continuous biometric authentication provides the most appropriate solution to the problem

Using Continuous Biometric Authentication may provide the following benefits :

- It can provide a high level of security, as it is difficult for students to imitate or replicate another person's typing and mouse movements. This can help prevent students from cheating by using devices or resources not allowed during the exam.
- Additionally, it can provide a way to detect if someone else is trying to take the exam for the student, by analyzing the patterns of typing and mouse movement.

However, there are also some limitations and challenges to consider when using continuous biometric authentication as a solution:

- Continuous biometric authentication may not be able to detect traditional forms of cheating such as collaboration and sharing.
- This approach might not work well for students with disabilities, as their typing and mouse movement patterns may be different from those of other students.
- The cost and complexity of implementing such a system, a significant investment in terms of hardware and software, as well as training and maintenance.
- The Biometric profiles created during Lab sessions may not be a good indicator as during exam conditions Students may get stressed which can impact their typing and keyboard behaviour[4].
- Keystroke Authentication can be forged [5].

In conclusion, Continuous Biometric Authentication may not be the best idea for the problem, instead Proctoring would be more suitable.

Chapter 2

Covert Channel

A covert channel is a communication method that allows two processes to exchange information in a way that is not detectable by the operating system or other monitoring tools. Most Covert Channels either use storage resource or timing as a means to communicate.

2.1 Give a high-level overview explaining how your covert channel works.

The Covert Channel implemented is a storage-based Covert Channel that uses a file's metadata to send and receive data. It uses the NTFS file system file time [6], namely access-time and modified time to store data, up to 64 bits at a time. The access time and modified time can be changed by the user which allows us to establish a covert channel, A file is created whose access time and modified times are changed on fixed time intervals (for synchronization) , which can then be read by another process.
The secret message to be sent is set as access-time of the dummy file created by the sender process which can then be read by the receiver process.The channel can also be used for longer transmissions of data by repeatedly setting the access and modified time of the file. The file times are changed 200 times every second (intervals of 5 milliseconds) to achieve a good balance between reliability and bandwidth.

2.2 Develop code to implement your channel and run it to communicate the indicated message. Report your results.

The covert channel starts by initiating the receiver.py process which waits for a file to be created to start observing. The sender.py process creates and modifies the time stamps of the file in fixed intervals, which are then observed by the process in receiver.py, which can interpret the data. Upon Completion of the transfer of data, the file is deleted by the sender process.
The code is divided into two files :

- Sender.py :

```
1 import os,time  # os library for manipulating file time and time library
      to calculate bandwidth
```

```
2
3 channel_file = '/file.txt'
4
5 interval = 0.005 # for synchronization
6
7 message = 0b10101001110110011010
8
9 msg_bin = bin(message) #converting to a binary string
10
11 msg = msg_bin[2:] #removing 0b from start
12
13 msg = msg.zfill(32) # padding to make it 32 bit long so as to fill one
       field of file time
14
15 print('creating File\n')
16 cc = open(channel_file,'w')
17 cc.close()
18
19 print('\nSending message')
20
21 os.utime(channel_file,(int(msg, 2),0)) # setting file access time to the
       secret message padded with zeroes
22
23 time.sleep(1) # to allow receiver to catch up
24
25 os.remove(channel_file) # removing the channel file
```

Code 2.1: Sender.py

- Receiver.py:

```
1 import os,time
2
3 channel_file = '/file.txt'
4 band_file = '/band.txt' # to calculate bandwidth
5 interval = 0.005
6
7 print("....Waiting For Message:\n")
8
9 while os.path.isfile(channel_file) == False: # receiver process waits
      till file is created
10   pass
11
12 time.sleep(interval) # allows sender.py to set file time
13 received = ''
14
15 atime = bin(int(os.path.getatime(channel_file))) #accesses file access
      time which is a float value and converts it to binary string
16
17 atime = atime[2:] #removes 0b from start
18 received = received +atime
19 time.sleep(interval)
20 msg_int = int(received) #converts str to int
21 print("Received message is : ",msg_int)
```

Code 2.2: Receiver.py

- The results are as shown:

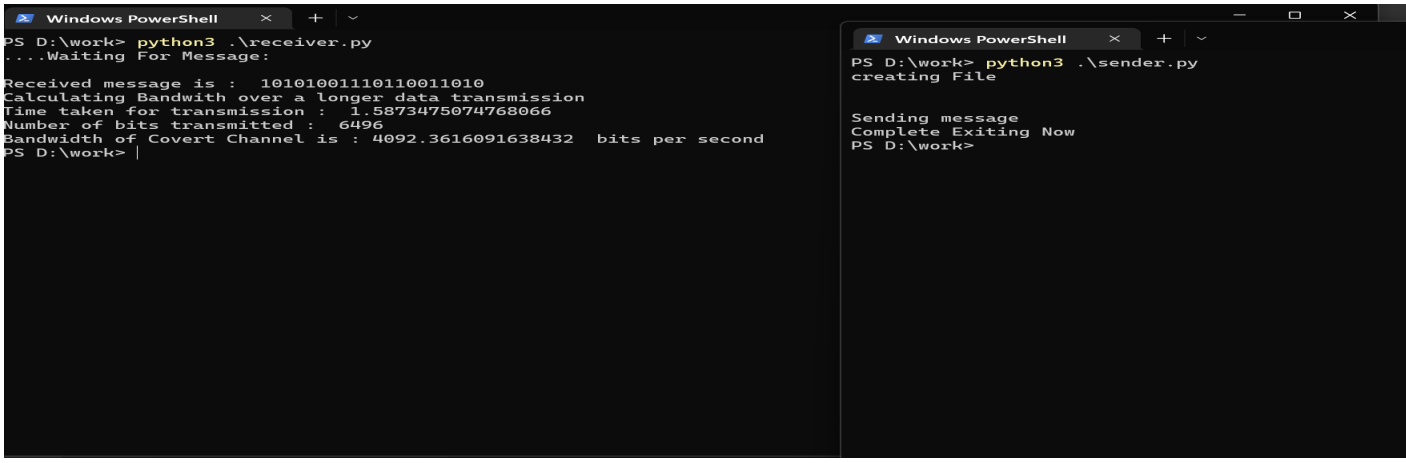

Figure 2.1: *Covert Channel*

2.3 Explain how you would assess the maximum bandwidth of your covert channel.

To assess bandwidth we need to calculate number of bits received by receiver.py per second, To achieve this we generate two 32-bit random integers and set them as access time and modified time for a file, and repeat the process to assess the maximum bandwidth.
Sender.py :

```
# For Bandwidth Calculation
band_file = '/band.txt'
bf = open(band_file,'w')
bf.close()

ctr = 0

while ctr < 100 :
    r1 = 4294967295 # 32 -bit integer
    r2 = 4294967295 # 32 -bit integer
    os.utime(band_file,(r1, r2))
    time.sleep(interval)
    ctr += 1
time.sleep(interval) # allow receiver to catch up
os.remove(band_file)
print('Complete Exiting Now')
```

Code 2.3: Bandwith Sender

We calculate the time taken to receive all 20 messages and calculate Bandwidth as :

$$\frac{number\ of\ bits}{time\ taken}$$

Receiver.py :

```
#Calculating Bandwidth over a longer message :
print('Calculating Bandwith over a longer data transmission')
while os.path.isfile(band_file) == False:
  pass

bnd_start = time.time()
msg = ''
try:
    while os.path.isfile(band_file) == True:
        access_time = bin(int(os.path.getatime(band_file)))
        ac = access_time[2:]
        msg = msg + ac


        modified_time = bin(int(os.path.getmtime(band_file)))
        mt = modified_time[2:]

        msg = msg + mt


        time.sleep(interval)
except FileNotFoundError:
    pass
bnd_end = time.time()
time_bnd = bnd_end - bnd_start
datalen = len(msg)
print("Time taken for transmission : ",time_bnd)
print("Number of bits transmitted : ",datalen)
print('Bandwidth of Covert Channel is : ' + str(datalen/ time_bnd) + ' '+'
    bits per second')
```

Code 2.4: Bandwith Receiver

The Bandwidth of the covert channel is approximately 4092 bits per second. Theoretically this bandwidth can be increased further by reducing time intervals to update the access times and modified times, but it may decrease reliability of the channel.

Bibliography

[1] Joseph Roth, Xiaoming Liu, and Dimitris Metaxas. On continuous user authentication via typing behavior. *IEEE transactions on image processing*, 23(10):4611–4624, 2014. ISSN 1057-7149.

[2] Xiaofeng Lu, Shengfei Zhang, Pan Hui, and Pietro Lio. Continuous authentication by free-text keystroke based on cnn and rnn. *Computers Security*, 96:101861, 2020. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2020.101861. URL `https://www.sciencedirect.com/science/article/pii/S0167404820301334`.

[3] Nan Zheng, Aaron Paloski, and Haining Wang. An efficient user verification system using angle-based mouse movement biometrics. *ACM transactions on information and system security*, 18(3):1–27, 2016. ISSN 1094-9224.

[4] Javier Hernandez, Pablo Paredes, Asta Roseway, and Mary Czerwinski. Under pressure: Sensing stress of computer users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 51–60, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450324731. doi: 10.1145/2556288.2557165. URL `https://doi.org/10.1145/2556288.2557165`.

[5] Yan Sun and Shambhu Upadhyaya. Synthetic forgery attack against continuous keystroke authentication systems. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, volume 2018-, pages 1–7. IEEE, 2018. ISBN 9781538651568.

[6] MS Windows NTFS file system times. `https://learn.microsoft.com/en-us/windows/win32/sysinfo/file-times`.

Appendices