CMP408: Sytem Internals & Cybersecurity
Smart Door using Facial Recognition

Thomas MacKinnon
School of Design and Informatics
Abertay University
DUNDEE, DD1 1HG, UK
Word Count: 1350

Contents

1 Introduction

Traditional door locks are quickly becoming a thing of the past, physical key unlocks seem to be irrelevant compared to modern day technologies. Already it is becoming a standard to use RFID tags to open communal doors and hotel room. This advancement in security doesn't look to be slowing down anytime soon, and as cutting smart systems become cheaper and more efficient it is only inevitable that more and more people start incorporating them into their homes.

Amazon's Ring doorbell system is an excellent example of this, providing customers with a low cost Smart doorbell that doubles as a surveillance camera for your front door (Molla, 2020). Amazon Ring has a huge customer base, growing everyday, figure 1 show the drastic increase in users. This market has huge potential, and is only going to keep increasing.

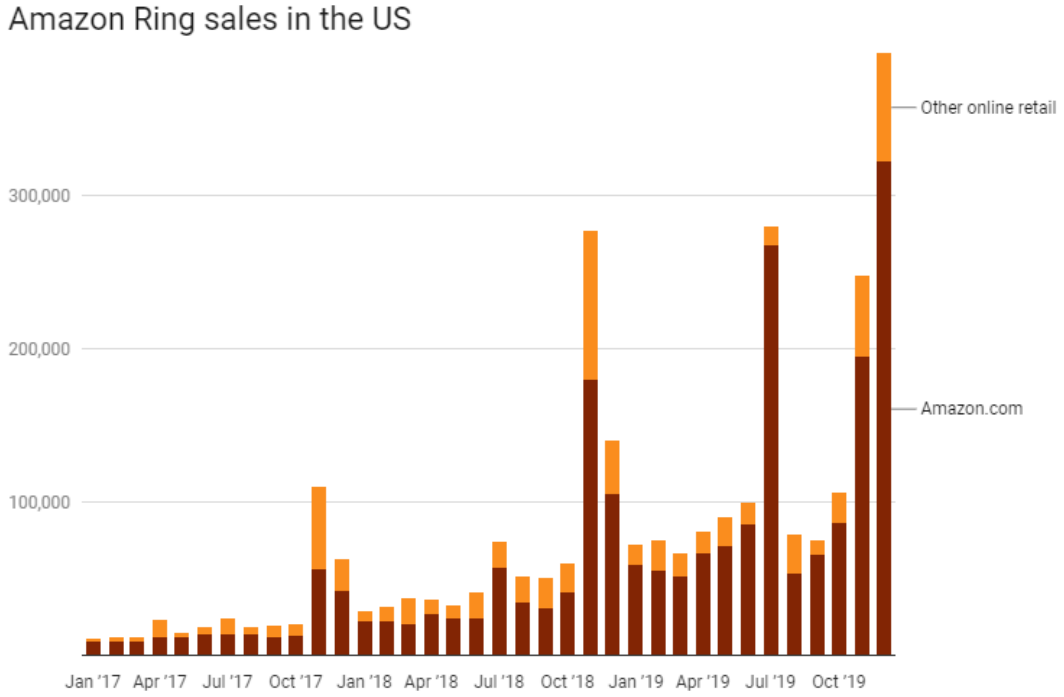

Figure 1: Amazon Ring sales (Molla, 2020)

Facial recognition is clearly the next step in Smart doors, it is already a standard on your phone, soon it will become a standard on your own home. Amazon Web Service (AWS) already provides a facial recognition software (Rekognition) which can accurately tell what an image contains and if it matches a stored face (Amazon, 2020). It seem obvious that this technology will be implemented into Ring devices sometime in the future. This leads to the basis of this project, to create a Smart door using a Raspberry pi and AWS Rekognition, that is cable of not only authenticating users but also physically unlocking the door.

2 Methodology

2.1 Hardware

Several pieces of hardware were used in the making of the Smart lock, each used to the overall effectiveness of the project. As the core concept of the Smart lock was facial recognition a Pi Camera was attached to board using ribbon cables and mounted to the side of the door. This camera required a more powerful board, so a Raspberry pi 4 model B was used for the rest of the project. Figure 2 shows the physical appearance of the door from the back.

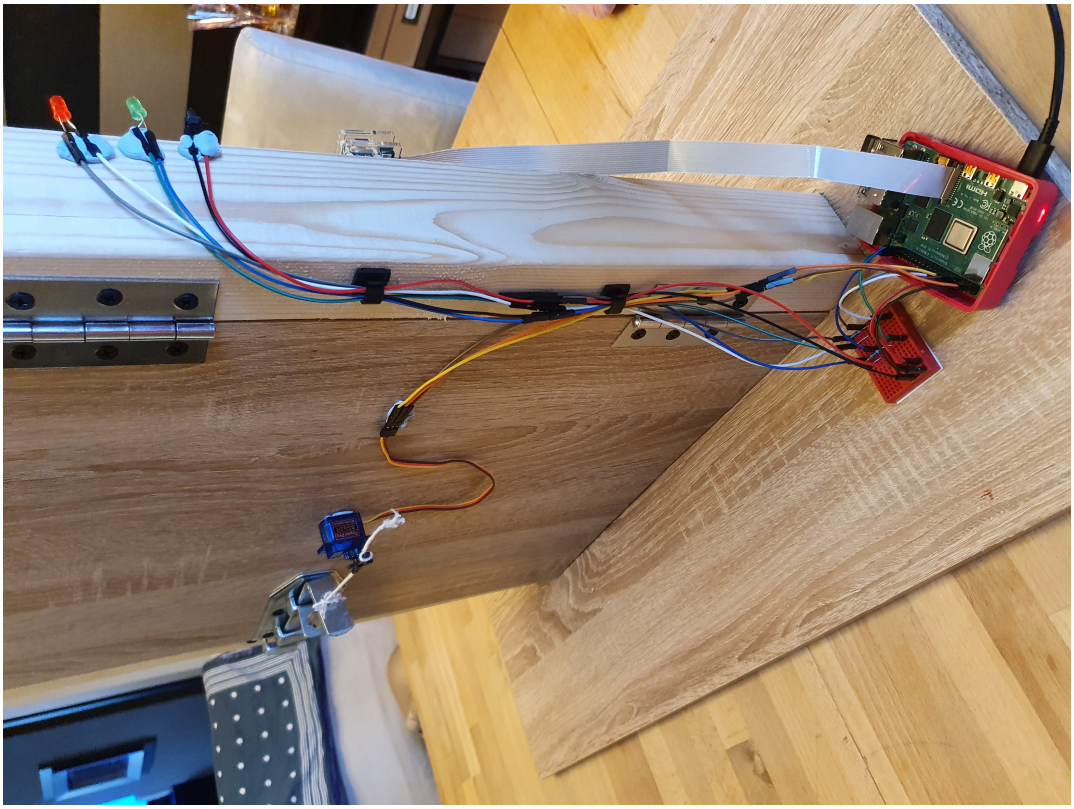

Figure 2: Door with Pi and hardware

A two foot tall physical door had been built from wood, with a slide lock attached, so that the functionally of Smart lock could be shown. The slide lock was attached to a micro servo that could pull it open after a successful unlock attempt. A button was implemented as to begin the operation of the Smart lock by starting the camera, and placed where a door bell might. Two LEDs were used to display the status of the unlock attempt, with green being successful and red being unsuccessful, and a breadboard was used to add resistance. The physical appearance of the door can be seen in figure

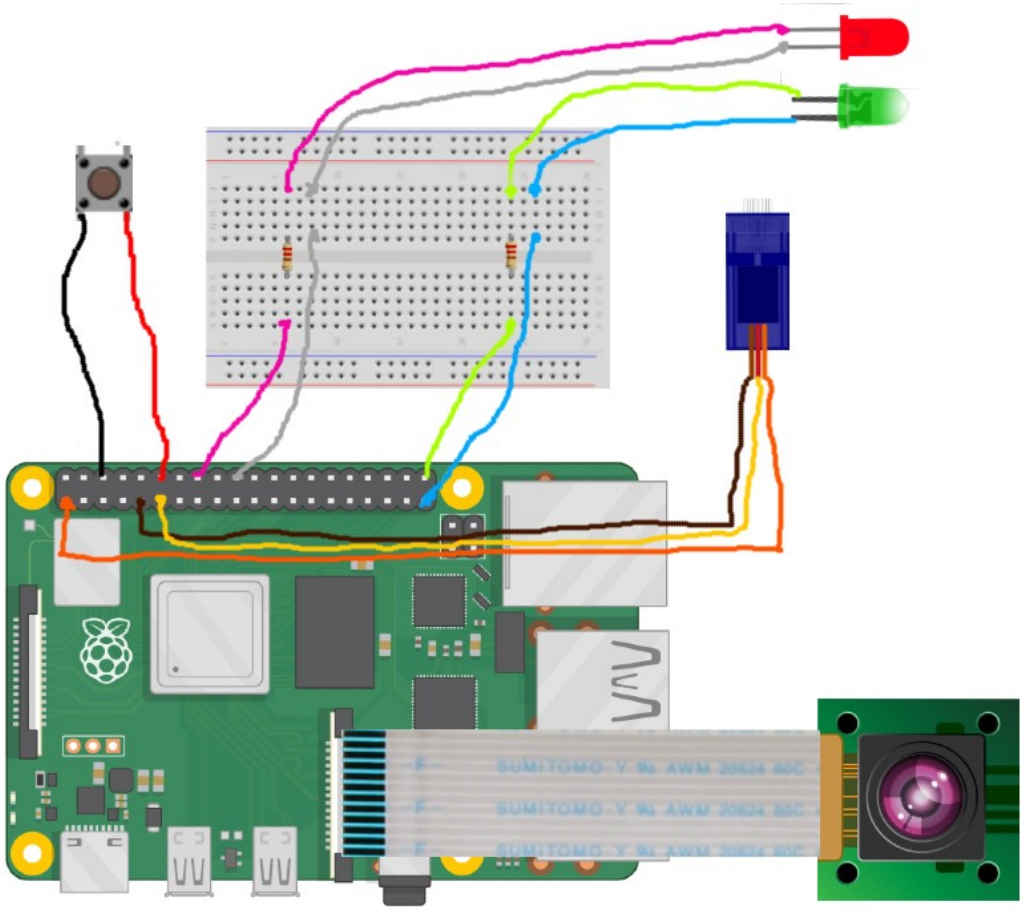

Figure 3: Hardware Layout

Figure 3 shows the setup for GPIO pins on the Raspberry Pi board and the external pieces of hardware, and here is a list of each connection:

- **Button:** Attached to pin 12(GPIO18) and pin 6(GND) with long wires.
- **Red LED:** Attached to pin 16(GPIO23) and pin 20(GND), with a resistor and long wires.
- **Green LED:** Attached to pin 40(GPIO21) and pin 39(GND) with a resistor and long wires.
- **Servo:** Attached to pin 11(GPIO17), pin 9(GND) and pin 1(3V3 power).
- **Pi Camera:** Attached to CSI Connector with a Ribbon cable.

2.2 Software

The software side of the project needed to be able to take a photo of the user, process whether they were authenticated for entry and react accordingly. Once the Python program is running it will print "off" every second until it receives a button press, which will activate the camera. The Pi camera used included a basic command to take and save a picture after a short delay, which was implemented using the OS python module. Boto3, an AWS SDK for python, was used to submit the AWS Rekognition query with the users AWS credentials. The original image has to be sent as Base64-encoded bytes for the AWS to recognise it, and will be compared to a collection of faces already set (See section "2.3 Cloud" for details). AWS Rekognition will compare the image to all faces in the collection, returning an array of matches of 95% or higher.

```
while True:
    if button.is_pressed:
        os.system("raspistill -o image.jpg")
        image = "image.jpg"
        client = boto3.client(service_name="rekognition",
                              aws_access_key_id=accessid, aws_secret_access_key=accesskey,
                              region_name="us-east-1", aws_session_token=sessiontoken)
        with open(image, "rb") as sourceImage:
            sourceBytes = sourceImage.read()

        response = client.search_faces_by_image(Image={'Bytes': sourceBytes}, CollectionId='faces', FaceMatchThreshold=95)
        matches = response['FaceMatches']
```

Figure 4: Rekognition Request

Since the repsonse returns an array of matches it is easy to tell if the user is allowed to enter. Simply checking the size of the array to see if it has one or more entries shows if that the user is already in the stored collection of faces. This will result in the data from the response being printed to terminal, showing the Similarity and Confidence AWS has in the Face match. Hardware elements will then activate, the servo will pull the physical lock open, followed by the green LED flashing. If the user isn't permitted for entry the Red LED will flash three times before returning to normal operation.

```
print("--------------------------------")
if len(matches) > 0:
    face = matches[0]
    specificface = face['Face']
    print("Matched with: {}""%"" Similarity.".format(face['Similarity']))
    print("With Confidence of: {}".format(specificface['Confidence']))
    servo.max()
    greenled.on()
    sleep(3)
    greenled.off()
else:
    redled.on()
    sleep(1)
    redled.off()
    sleep(1)
    redled.on()
    sleep(1)
    redled.off()
```

Figure 5: If there is/isn't a Face Match

Figure 6 shows the back end display of the program, printing “off” until the button is pressed and showing the match data if there is any. The full script can be found in the appendix.

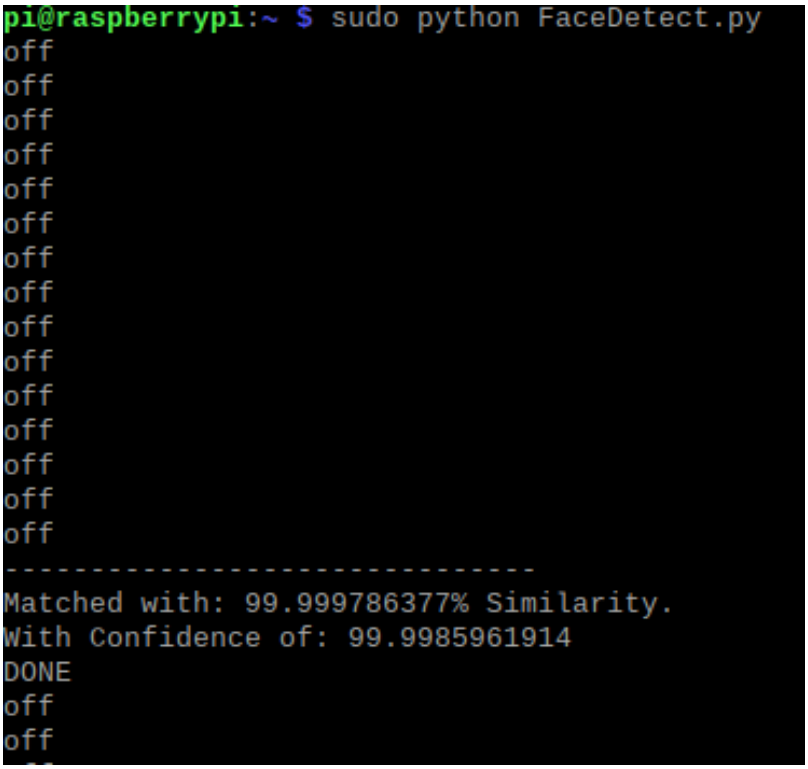

Figure 6: Back end terminal Display

2.3 Cloud

Amazon Web Services was used to provide storage and processing for the Smart lock, specifically using Rekognition and Simple Storage Solution (S3). Rekognition is a face searching service, allowing for identification of people, moods, animals and much more from a photo upload. An S3 bucket can be used to search through a collection of faces and return matches, which would serve as the core feature of this project. The simple flow diagram presented in figure 7 shows how the program operates.

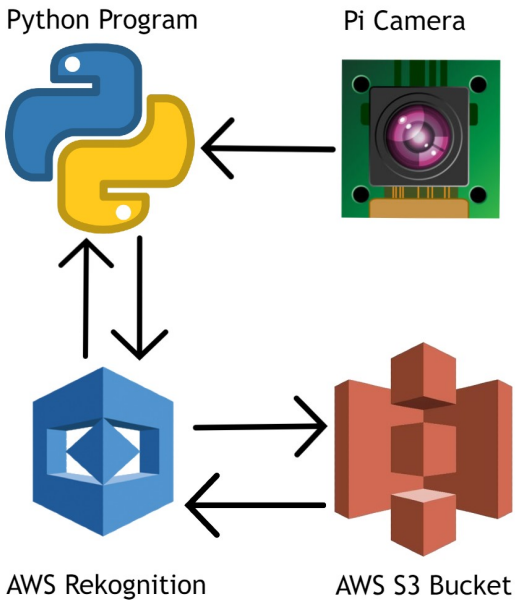

Figure 7: Flow of AWS services

AWS Rekognition Face search feature needs a collection of faces to operate properly, each face needs to come from an S3 bucket. Several different users faces were added to the bucket and then added to the collection using the command seen in figure 8, including two of the same tester as to check for error with multiple matches.

```
C:\Users\Tom>aws rekognition index-faces ^
More? --image S3Object={Bucket=pistorecmp408,Name=tom.jpg} ^
More? --collection-id "faces" ^
```

Figure 8: Add image to collection for Rekognition

Once the picture of the user has been taken, it will be sent to AWS Rekognition with the "Bytes" flag, since it is stored locally. Rekognition will compare the image to a collection of faces, supplied by the S3 bucket, and send back any matches of 95% or higher. This number is deliberately high, as whilst testing the door unlocked for an unauthenticated user. The cause of the error was that the threshold for matches was set to 80%, and all the testers were members of the same family with similar facial features, leading to Rekognition thinking they were a user in the collection. The Matches are printed to the terminal running the program, with the similarity and confidence in the match. A more indepth flow diagram, showing the entire Smart locks process can be seen in figure 9.

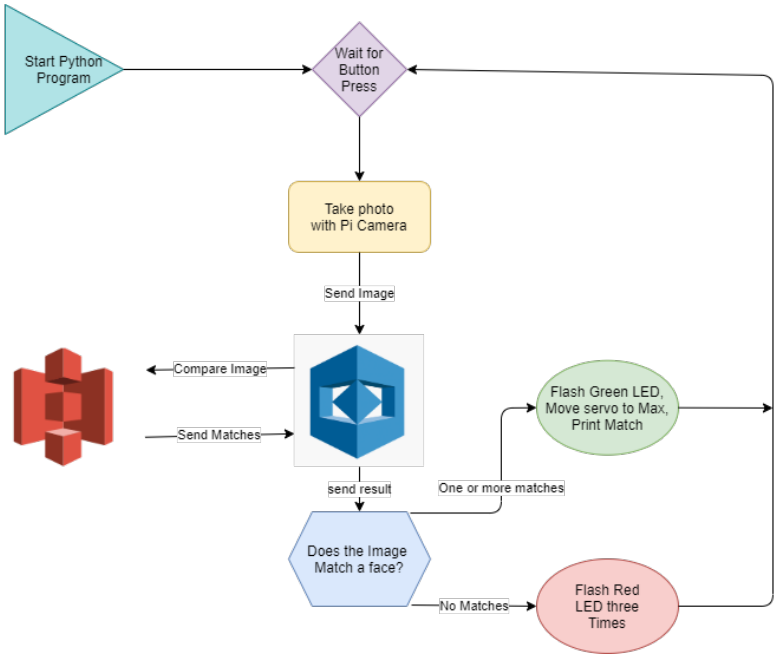

Figure 9: Flow of Smart Lock

3 Conclusion

The Smart door lock using Facial recognition worked exactly as intended, allowing a secure way to unlock a door without the need for keys. The door provides excellent recognition of faces and effective use of hardware for the user to interact with. The hardware can physically unlock the door with the force of the servo, and indicate the result of the unlock attempt to the user. The Smart door met all of the core objectives for this project, making it a success.

3.1 Future Work

Certain features were never added to the Smart lock, if given more time and resources these features would have been developed:

- **Loadable Kernel Module (LKM):** A long period of development was dedicated to making an LKM that would start the program on a button press. However, these attempts were not successful, leading to a lot of wasted time that could have been used for other features. If given more time, the button LKM would have been finished, with the LEDs also being controlled at Kernel level.
- **Database of Logs:** There were plans for each attempted entry to be logged in a database, showing time, image and name of user trying to enter. Due to development issue this was never completed, but given more time an effective logging system would have been added.
- **Website:** A basic website was planned to be developed, showing logs in a neat format and allowing the user to unlock the door remotely.

4 References

Amazon, 2020. Amazon Rekognition. [Online] AWS. Available at: https://aws.amazon.com/rekognition/ [Accessed 14/01/2021]

Anton, 2017. Raspberry Pi Facial Recognition. [Online] Hackster. Available at: https://www.hackster.io/gr1m/raspberry-pi-facial-recognition-16e34e [Accessed 14/01/2021]

Han, H, Jain, A.K & Patel K. 2020. Secure Face Unlock: Spoof Detection on Smartphones. *IEEE Transactions on Information Forensics and Security.* pp. 2268-2283. Molla, R. 2020. Amazon

Ring sales nearly tripled in December despite hacks. [Online] Vox. Available at: https://www.vox.com/recode/2020/1/21/21070402/amazon-ring-sales-jumpshot-data [Accessed 14/01/2021]

5 Appendix

```
from gpiozero import LED, Button, Servo
from time import sleep
import os
import boto3



button = Button(18)
redled = LED(23)
greenled = LED(21)
servo = Servo(17)
servo.min()
redled.off()
greenled.off()

accessid = "your_access_ID"
accesskey = "your_access_key"
sessiontoken = "your_session_token"

while True:
    if button.is_pressed:
        os.system("raspistill -o image.jpg")
        image = "image.jpg"
        client = boto3.client(service_name="rekognition",
                              aws_access_key_id=accessid, aws_secret_access_key=accessk
                              region_name="us-east-1", aws_session_token=sessiontoken)
        with open(image, "rb") as sourceImage:
            sourceBytes = sourceImage.read()

        response = client.search_faces_by_image(Image={'Bytes': sourceBytes}, Collectio
        matches = response['FaceMatches']

        print("--------------------------------------")
        if len(matches) > 0:
            face = matches[0]
            specificface = face['Face']
            print("Matched with: {}""%"" Similarity.".format(face['Similarity']))
            print("With Confidence of: {}".format(specificface['Confidence']))
            servo.max()
            greenled.on()
            sleep(3)
            greenled.off()
        else:
            redled.on()
            sleep(1)
            redled.off()
```

```
            sleep(1)
            redled.on()
            sleep(1)
            redled.off()

        print("DONE")

else:
        print("off")
sleep(0.5)
```