This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCE.2020.3038040, IEEE Consumer Electronics Magazine

BioSec: A Biometric Authentication Framework for Secure and Private Communication among Edge Devices in IoT and Industry 4.0

Muhammed Golec[1], Sukhpal Singh Gill[1], Rami Bahsoon[2] and Omer Rana[3]
[1]School of Electronic Engineering and Computer Science, Queen Mary University of London, UK
[2]School of Computer Science, University of Birmingham, Birmingham, UK
[3]School of Computer Science and Informatics, Cardiff University, Cardiff, UK

***Abstract*— With the rapid increase in the usage areas of Internet of Things (IoT) devices, it brings challenges such as security and privacy. One way to ensure these in IoT-based systems is user authentication. Until today, user authentication is provided by traditional methods such as pin and token based. But traditional methods have challenges such as forgotten, stolen, and shared with another user who is unauthorized. To address these challenges, we proposed a biometric method called BioSec to provide authentication in IoT integrated with edge consumer electronics using fingerprint authentication. Further, we ensured the security of biometric data both in the transmission channel and database with the standard encryption method. BioSec ensures secure and private communication among edge devices in IoT and Industry 4.0. Finally, we have compared three encryption methods used to protect biometric templates in terms of processing times and identified that AES-128-bit key encryption method outperforms others.**

INTRODUCTION

Internet of Things (IoT) technology has a wide range of uses, from consumer applications such as smart homes to infrastructure applications such as energy management[1]. With the growing use of IoT devices, security has become a significant issue in IoT environment. Security in IoT can be examined under the following three main headings: ensuring the security of the collected data, using an encrypted communication channel, and using user authentication[2]. User authentication is essential to protect the privacy of personal data. Traditionally, user authentication in IoT was done with pin-based password systems[1]. However, biometric systems have started to be used due to the weaknesses of the pin-based password, such as being forgotten, stolen, and shared. Because biometric descriptors are inherent in an individual, it is more difficult to manipulate, share, or forget these characteristics[3]. At the same time, Biometric data cannot be changed in case of stolen because it has unique characteristics of the person, and it creates serious privacy problems in case of leakage[4]. Therefore, we can use encryption methods to secure biometric data.

Motivation and Our Contributions

The aim of this study is to provide security in IoT systems with fingerprint authentication to prevent unauthorized access to the system, and ensure the privacy of the IoT system. If a person's fingerprint is stolen, the security of IoT and the privacy of personal data in IoT will also be compromised. Since fingerprints are also unique, they bring a second privacy concern with them in case they are stolen. Therefore, we propose ***BioSec***, a framework for private use of Edge Consumer Electronics (ECE) and secured communication among IoT devices. In our system, the security of biometric data in both transmission channels and databases is ensured by using the Standard Encryption Method, which has not been used in the literature before. In this way, we aim to secure both biometric data and IoT.

BACKGROUND

The literature review shows that there are not many studies on biometric authentication in IoT. Wencheng et al. proposed a biometric system that takes into account the increased energy consumption and processing load in IoT devices and also uses fingerprint authentication for the security issue in IoT[2]. This work provides a significant performance increase in terms of computational cost, the fingerprint templates are vulnerable to attacks as they are not sent encrypted over the communication channel between the IoT device and the server. Nemanja et al. proposed a system in which two different biometric methods are used for authentication[5]. According to this system, using a single camera to reduce different sensor cost, face and iris images will be taken from the user at the same time. In this work, it is considering the recognition success rate and the cost of the sensor used, but the safety of biometric data is not taken into account. Dirk et al. proposed a multimodal biometric authentication that combines Hand Geometry and Gesture via a hardware device Leap Motion Controller to provide authentication[6]. The system is still in the development phase, and no prototype has been made. In addition, aging in the hands of the users and the damages that may occur due to work accidents will negatively affect the recognition performance.

Table I. Comparison of BIOSEC with Related Works

Work	**Biometric Authentication Methods**	**Biometric Template Protection**
Yang et al.[2]	Fingerprint	Lightweight Cryptography
Maček et al.[5]	Iris and Face	NA
Shahim et al.[6]	Hand Geometry & Gesture	Stenographical Encryption
BioSec	Fingerprint	Standard Encryption

Table 1 shows the comparison of BioSec with existing works. In our study, the fingerprint is preferred as the biometric authentication method because: 1) fingerprint authentication is still the easiest to use and one of the most successful biometric methods and 2) the sensor used to obtain fingerprints is cheaper than the sensors required to obtain biometric data such as Iris and Hand Geometry. Unlike other studies, standard encryption methods are used in this work to ensure the security of biometric templates in both transmission and database. Template protections made with standard encryption methods do not negatively affect the performance rate, such as feature transformation (stenographical encryption) and biometric cryptosystem (lightweight cryptography) methods used in other studies.

BIOMETRIC SYSTEMS

In biometric systems, authentication is made by using several physical features (face recognition, fingerprint recognition) and behavioral (voice recognition, signature recognition) features that the person has and is thought to be unique to the person[10]. There are three important metrics used to measure system performance in biometric systems:

1) False Acceptance Rate (FAR): It is called False Acceptance when an unauthorized person enters the system as if they were a registered user in the system, even though they are not registered in the system's database.

2) False Rejection Rate (FRR): The rejection of a user registered to the system as if they were unauthorized by the system is called False Rejection.

3) Equal Error Rate (EER): When a FAR-FRR graph is drawn, the point where the two curves intersect is called the EER point. In a biometric system, it is the most basic metric to look at when evaluating the performance of the system. The lower the EER ratio of the system, the higher the performance of the system is considered. Biometric security and privacy is one of the most known security vulnerabilities in biometric systems is template leakage[7]. If a hacker can hack a biometric system's database, they can learn biographical information (such as name, surname, address) along with biometric information. This leads to serious privacy issues[3]. Therefore, the leakage attack is not only a security threat but also jeopardizes the user's privacy. In our system, the standard encryption methods AES, DES, and 3DES will be compared in terms of processing time.

BIOSEC SYSTEM

The BioSec System Design

We designed the BioSec system to ensure security and privacy with biometric authentication, which consists of two layers: client and server. Raspberry Pi-4, as an IoT device, is used in the client part. The server part of the system is installed on a PC. In the BioSec system, first, the fingerprints of the user or users to be registered to the system are taken with the help of a sensor, and these fingerprint images are sent to the Raspberry Pi device. After fingerprint images pass through the necessary image processes in Raspberry Pi, a biometric template consisting of minutiae points to be used in fingerprint authentication, is obtained. Minutiae points are friction ridge skin impressions believed to be unique on each fingerprint. The biometric template is sent in encrypted form as a precaution against hacking attacks on the communication channel between the server and the Raspberry Pi. The encrypted biometric templates taken in the server section are stored in the database as encrypted as a precaution against biometric template leakages. In this way, security is provided against any hacking in the database.

Figure 1 shows the path followed while creating the database of our BioSec system. The BioSec system is ready for use after the fingerprints of authorized users are registered in the database. The user sends his fingerprint to the system through the fingerprint sensor. The fingerprint taken from the sensor is sent to the Raspberry Pi, and the biometric template is extracted after the necessary image processing operations. The extracted biometric template is sent in encrypted form as a precaution against hacking attacks on the channel between Raspberry Pi – Server. The encrypted biometric templates previously saved in the database with the fingerprint template coming from the user are decrypted and compared on the server part one by one. If the user's fingerprint matches one of the previously registered fingerprints in the system, they are permitted to access the system. If this fingerprint does not match the registered fingerprints in the system, they are not permitted to access the system. Figure 2 presents the working scheme of BioSec.

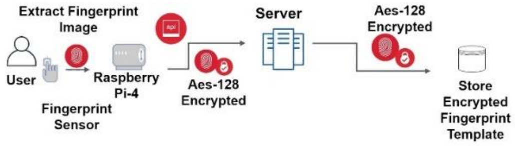


Fig. 1. Fingerprint Enrollment Stage

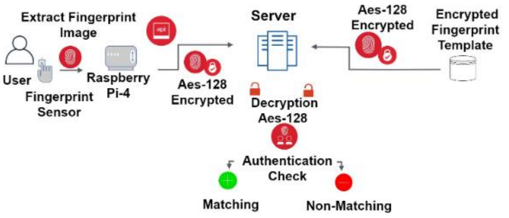


Fig. 2. Operation scheme of the BioSec system

Raspberry Pi + Buttons: A fingerprint sensor is not used in our system for financial reasons. However, a mechanism has been established that sends fingerprint images taken from the fingerprint sensor through buttons. *Button 1* sends the fingerprint of a user registered in our system, while *Button 2* sends the fingerprint of a user who is not registered in our system. The flashing of the LED light differs according to the matching status of the sent fingerprints. If there is a correct match, the LED light turns on intermittently three times in a row. In case of mismatch and no match, the LED light only once. Moreover, in all three cases, the system returns matching, mismatch or no match on the command screen. Figure 3 shows the client part of our system.

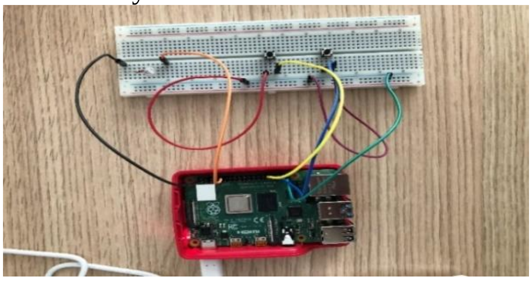


Fig. 3. In the client section, buttons that send Raspberry Pi and fingerprint images are seen.

Configuration Settings: The server part of the system is installed on a PC with 12 GB RAM and Intel (R) Core (TM) i5-7300HQ 2.50 GHz clock frequency CPU. The operating system is Windows 10 Pro-64 Bit.

API and Database: In our system, an Application Programming Interface (API) was written using the Python language and flask framework. API is used to reach applications on a server from different devices and get a response. After sending a fingerprint template from Raspberry Pi to the server in our system, API is used to reach the match results on the server. FVC-2004 DB3 fingerprint database was used, which is available for use on the Internet[8]. The first fingerprint images from 8 images for each user is used for testing. The remaining seven fingerprints were stored in the database in an encrypted form after the image enhancement. We use an open-source fingerprint matching algorithm[9].

The BioSec System Implementation

Fingerprint Enrollment Stage: First of all, fingerprint images of the users who want to be registered in the system are passed through the necessary pre-processing, and biometric templates are created with feature extraction. In our system, biometric templates are shown with the Des variable. Later, these Des data are recorded in the database in an encrypted form. Aes-128 Bit method has been chosen as the Encrypted method. FVC2004 DB-3 DATABASE folder contains 80 fingerprint images of 10 users. Each user has eight different fingerprint images of the same Fingerprint. From these eight fingerprint images, the first images are reserved for later use for system performance and matching testing. The remaining seven images were saved in the system database. As seen in Figure 4, the fingerprint database of our system has been created to show the name, Des, and encryption time for each image. Here *Des* are biometric templates containing minutiae extracted from fingerprint images. Des data containing minutiae points extracted from fingerprint images should not be confused with DES used in the encryption method. The result is returned according to the threshold value. With the des_from_csv () method, all encrypted Des data in the fingerprint database that we created earlier are returned as a dictionary. Figure 5 shows the utils.py file. There are four functions here. These include removedot (), which reduces the noise by improving the given image, get_descriptors (), which returns minutiae points of the given image, des_from_csv (), which we explained earlier, and finally, calculate () function which calculates the similarity of the two minutiae according to the threshold value.

Client Part of the BioSec System: In the client part of our system, the fingerprint images obtained from the fingerprint sensor are sent to the raspberry pi through the buttons. In Raspberry Pi, Des data containing fingerprint minutiae points are obtained with the removedot() and get_descriptors() methods that we have explained before. This Des data is sent to the server in an AES-128 bit encrypted form over the communication channel.

Pseudo Code for Creating Fingerprint Database

```
Start
fvc2004_db3 = the path of fingerprint images
insert_data_into_db( fvc2004_db3)
def insert_data_into_db( PATH) :
    names, des_list, times = list() # Create 3 empty lists
    for i in PATH:
        if i != first picture:
            des = get_descriptors(i) # Extract minutiae points of i and assign
            names.append(i) # Add the image's name to the names list
            Time_start = time.time() # Turn on timer for the duration of the encryption process
            Encrypted_des = Encrypt(des) # Encrypt des
            Time_end = time.time()
            time_result = 1000 * (Time_end - Time_start) # To calculate as second
             times.append(time_result) # Append encryption time to the times list
            des_list.append(Encrypted_des) # Add encrypted des to des_list
      else: # Creates a test folder to test system accuracy
        test_files = f'{db_name}_test'   # Create a folder named test_files and copy the i image there
     # end for loop
  df = pd.DataFrame({"Name": names, "Des": des_list, "Encrypt_Time": times}) # Save names, des_list and times information as an excel file
End
```

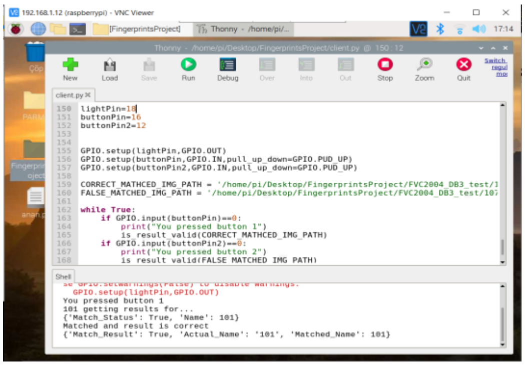
Fig. 4. Creating the system fingerprint database

Pseudo Code for Utils.py

```
Start
def removedot (invertThin) :
    #Define a function to make image enhancement for the image given as argument and return the enhanced image as a return.
def get_descriptors (img) :
    # Define a function to send the image which is given as an argument to the function defined above for image enhancement. Extract minutiae points from the enhanced image and return.
def des_from_csv () :
    # Define a function to return fingerprint minutiae points in the created database as dictionary forma
def calculate (one, two) :
    # Define a function to compare the two minutiae points it received and return True or False according to the given threshold value.
End
```

Fig. 5. Methods in Utils.py file

With the Flask API, the matching results on the server part are returned to the Raspberry Pi through the communication channel. Button1 sends the fingerprint image not found in the database of a user previously registered to the system database, to the Raspberry Pi. Button 2 sends the fingerprint image of a user who is not in the system to the Raspberry Pi.



```
lightPin=18
buttonPin=16
buttonPin2=12


GPIO.setup(lightPin,GPIO.OUT)
GPIO.setup(buttonPin,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(buttonPin2,GPIO.IN,pull_up_down=GPIO.PUD_UP)

CORRECT_MATHCED_IMG_PATH = '/home/pi/Desktop/FingerprintsProject/FVC2004_DB3_test/1
FALSE_MATCHED_IMG_PATH = '/home/pi/Desktop/FingerprintsProject/FVC2004_DB3_test/107

while True:
    if GPIO.input(buttonPin)==0:
        print("You pressed button 1")
        is_result_valid(CORRECT_MATHCED_IMG_PATH)
    if GPIO.input(buttonPin2)==0:
        print("You pressed button 2")
        is_result_valid(FALSE_MATCHED_IMG_PATH)
```

```
se GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(lightPin,GPIO.OUT)
You pressed button 1
101 getting results for...
{'Match_Status': True, 'Name': 101}
Matched and result is correct
{'Match_Result': True, 'Actual_Name': '101', 'Matched_Name': 101}
```

Fig. 6. Raspberry Pi shows the state of accurate fingerprint matching

Since the image sent from Button1 matches a user registered in the system and the correct person, the led in our system blinks three times in a row. Figure 6 shows the correct match status. Since the fingerprint image we send from Button 2 is not registered in the system, the LED blinks only once. Figure 7 shows the mismatch status. Also, if a registered fingerprint image is sent from Button 2 to the system and this fingerprint image matches another person registered in the system, in case of false acceptance, the LED blinks only one time. Figure 8 shows the false acceptance situation.

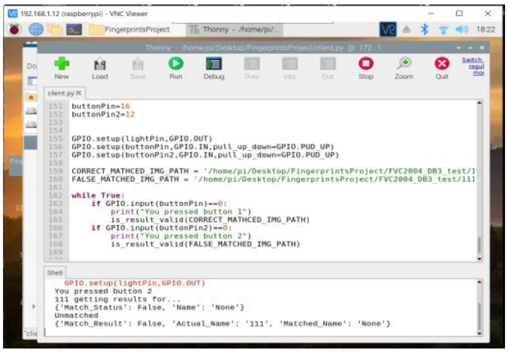
RESULT ANALYSIS

Performance of Fingerprint Authentication System

In biometric systems, the accuracy rate of the system is calculated by drawing a FAR - FRR graph. Along with the changing threshold value, FAR and FRR amounts also change by a trade-off with each other.

Table 2 Transaction Times of Standard Encryption Methods According to Key Lengths

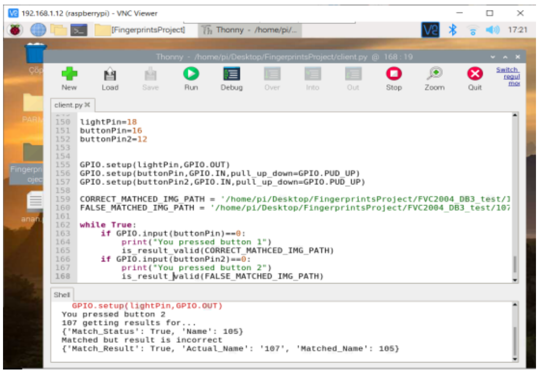
Encryption Method	Key length (bit)	Encryption Time(s)	Decryption Time(s)
AES	**128**	**1,016209**	**0,695434**
AES	**192**	**1,018884**	**0,757293**
AES	**256**	**1,033279**	**0,698176**
DES	**64**	**2,019291**	**1,740159**
3DES	**128**	**25,57337**	**26,09123**
3DES	**192**	**25,58849**	**26,08399**


192.168.1.12 (raspberrypi) - VNC Viewer
FingerprintsProject
Thonny - /home/pi/...
18:22
Thonny - /home/pi/Desktop/FingerprintsProject/client.py @ 172 : 1
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
client.py
Shell

```
buttonPin=16
buttonPin2=12


GPIO.setup(lightPin,GPIO.OUT)
GPIO.setup(buttonPin,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(buttonPin2,GPIO.IN,pull_up_down=GPIO.PUD_UP)

CORRECT_MATHCED_IMG_PATH = '/home/pi/Desktop/FingerprintsProject/FVC2004_DB3_test/1
FALSE_MATCHED_IMG_PATH = '/home/pi/Desktop/FingerprintsProject/FVC2004_DB3_test/111

while True:
    if GPIO.input(buttonPin)==0:
        print("You pressed button 1")
        is_result_valid(CORRECT_MATHCED_IMG_PATH)
    if GPIO.input(buttonPin2)==0:
        print("You pressed button 2")
        is_result_valid(FALSE_MATCHED_IMG_PATH)
```

```
  GPIO.setup(lightPin,GPIO.OUT)
You pressed button 2
111 getting results for...
{'Match_Status': False, 'Name': 'None'}
Unmatched
{'Match_Result': False, 'Actual_Name': '111', 'Matched_Name': 'None'}
```

Fig. 7. The case of sending a fingerprint that does not exist in the database from Button 2


192.168.1.12 (raspberrypi) - VNC Viewer
[FingerprintsProject]
Thonny - /home/pi/...
17:21
Thonny - /home/pi/Desktop/FingerprintsProject/client.py @ 168 : 19
New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
client.py
Shell

```
lightPin=18
buttonPin=16
buttonPin2=12


GPIO.setup(lightPin,GPIO.OUT)
GPIO.setup(buttonPin,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(buttonPin2,GPIO.IN,pull_up_down=GPIO.PUD_UP)

CORRECT_MATHCED_IMG_PATH = '/home/pi/Desktop/FingerprintsProject/FVC2004_DB3_test/1
FALSE_MATCHED_IMG_PATH = '/home/pi/Desktop/FingerprintsProject/FVC2004_DB3_test/107

while True:
    if GPIO.input(buttonPin)==0:
        print("You pressed button 1")
        is_result_valid(CORRECT_MATHCED_IMG_PATH)
    if GPIO.input(buttonPin2)==0:
        print("You pressed button 2")
        is_result_valid(FALSE_MATCHED_IMG_PATH)
```

```
  GPIO.setup(lightPin,GPIO.OUT)
You pressed button 2
107 getting results for...
{'Match_Status': True, 'Name': 105}
Matched but result is incorrect
{'Match_Result': True, 'Actual_Name': '107', 'Matched_Name': 105}
```

Fig. 8. False acceptance status with one of the fingerprints in the database

Figure 9 shows the FAR - FRR graphic of our Fingerprint biometric authentication system that we use in our system. Accordingly, our EER rate was found to be 30%. While our threshold value was 34, our system gave the most optimum performance. The overall performance of our system has been calculated as 70%.

Performance Evaluation of Encryption Algorithms

In the BioSec system, biometric data must be encrypted both in the transmission channel and in the database to ensure the security of biometric data. By testing three different Standard Encryption methods, encryption-decryption performance times were calculated. Table 2 shows the performance times of the three encryption methods with different key lengths. It was seen that the algorithm with the best performance was AES-128 bit with an encryption time of 1.016209s and a decryption time of 0.695434s. In today's world where processing speed is essential, AES-128 bit is still considered a secure algorithm, and therefore AES 128-bit encryption method is used in our system.

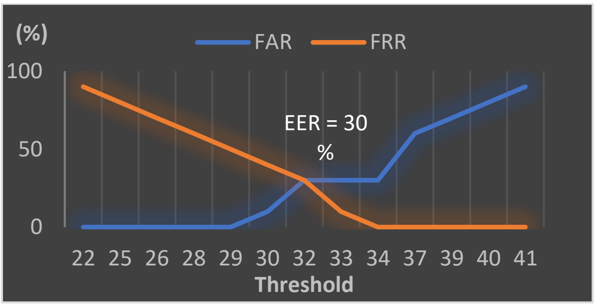


Fig. 9. FAR-FRR graph of the fingerprint authentication algorithm used in the BioSec system

THE EFFECT OF BIOSEC ON CONSUMER EXPERIENCE

In our study, we designed a system with biometric authentication to protect privacy in IoT, which is one of the trending topics of today. According to this system, users who want to log into IoT will be able to enter the system with fingerprint authentication, thus ensuring the security and privacy of IoT[10]. Biometric data used to provide security in IoT raises a second privacy concern in case of stolen because they are information that identifies individuals. Therefore, in the system we proposed a system called BioSec to ensure the privacy of biometric data of users. The availability of the system will increase using biometric authentication because users will not have to carry a token with them or a password they have to remember. In addition, the trust in the system will increase even more because biometric data is more difficult to breaches compare to traditional methods.

CONCLUSIONS

In this article, a BioSec framework is proposed for biometric authentication for secure and private communication among edge devices in IoT and Industry 4.0. The security level of IoT has been further increased by protecting the biometric data used with encryption methods. Biometric data sent and stored in the data transmission channels and database of the system are securely protected by using the encryption method. In our system, processing times are compared using three different encryption methods, and the fastest algorithm is used for encryption. The system can be made much more secure by improving the BioSec work. Since symmetric encryption methods are used in the system, the security of biometric data is also compromised in case the key used in encryption is stolen. Therefore, the whole system can be endangered as security.

FUTURE WORK

The system can be made more useful in terms of both security and processing speed by solving the key distribution problem or finding algorithms that work much faster. In future studies, the performance rate of the system can be increased by using fingerprint authentication algorithms with much higher performance rates or by choosing other biometric methods with high-performance rates such as iris.

ACKNOWLEDGEMENTS

Muhammed Golec would express his thanks to the Ministry of Education of the Turkish Republic, for their support and funding.

REFERENCES

[1] S. S. Gill and A. Shaghaghi. Security-Aware Autonomic Allocation of Cloud Resources: A Model, Research Trends, and Future Directions. Journal of Organizational and End User Computing, 32(3), pp.15-22, 2020.

[2] W. Yang et al., “A privacy-preserving lightweight biometric system for internet of things security,” IEEE Commun. Mag., vol. 57, no. 3, pp. 84–89, 2019.

[3] S.F. Aghili et al, LACO: Lightweight three-factor authentication, access control and ownership transfer scheme for e-health systems in IoT. Future Generation Computer Systems, 96, pp.410-424, 2019

[4] M. Ghahramani et al., RSS: An energy-efficient approach for securing IoT service protocols against the DoS attack. IEEE Internet of Things Journal, 2020.

[5] N. Maček, I. Franc, M. Bogdanoski, and A. Mirković, “Multimodal Biometric Authentication in IoT: Single Camera Case Study,” 2016.

[6] L.-P. Shahim, D. Snyman, T. Toit, and H. A. Kruger, “Cost-Effective Biometric Authentication using Leap Motion and IoT Devices,” 2016.

[7] S. S. Gill and R. Buyya. SECURE: Self-protection approach in cloud resource management. IEEE Cloud Computing, 5(1), 60-72, 2018

[8] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, “FVC2004: Third fingerprint verification competition,” in Biometric Authentication, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–7, 2004.

[9] K. Jankoski, python-fingerprint-recognition. https://github.com/kjanko/python-fingerprint-recognition

[10] C. Liu et al., "Finger-Vein as a Biometric-Based Authentication," in IEEE Consumer Electronics Magazine, vol. 8, no. 6, pp. 29-34, 1 Nov. 2019

Muhammed Golec is a MSc student with Queen Mary University of London, UK. Contact him at m.golec@hss18.qmul.ac.uk

Sukhpal Singh Gill is a Lecturer with Queen Mary University of London, UK. Contact him at s.s.gill@qmul.ac.uk

Rami Bahsoon is a Senior Lecturer with University of Birmingham, Birmingham, UK. Contact him at r.bahsoon@cs.bham.ac.uk

Omer Rana is a Professor with Cardiff University, Cardiff, UK. Contact him at ranaof@cardiff.ac.uk