Quantum Homomorphic Encryption

By

Joan Watiri Ngure (njoan@aims.ac.rw)
African Institute for Mathematical Sciences (AIMS), Rwanda
Supervised by: Professor Barry C Sanders
University of Calgary, Canada

June 2018

AN ESSAY PRESENTED TO AIMS RWANDA IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF SCIENCE IN MATHEMATICAL SCIENCES

Contents

1. Applications

In this chapter we will discuss an example of a place where Quantum Homomorphic Encryption can be used.

1.1 The Airport Example

1.1.1 Problem Definition. We want to detect unauthorized personnel accessing distinguished places in the airport. We do not trust the server since it can be interfered with and information changed hence we are unable to detect the unauthorized personnel. The server may also decide to be malicious and relay the wrong information.

1.1.2 Requirements. We will require Closed Circuit Television (CCTV) cameras, a client who is classical and requires to request services of a quantum computer.

1.1.3 How it works. The CCTV cameras relay images to the client on real time. The client encrypts the face images received. The encrypted images are sent to the server. The server has the face images of authorized persons which are also encrypted. The server performs face recognition on the encrypted images. The server knows the computation it is performing since in Homomorphic Encryption the computation is not encrypted. If an image does not match the images they have, then an encrypted intrusion message is produced. The server does not know what the output is. The message is sent to the doors and the security. The doors are supposed to automatically close and the security is supposed to immediately appear on the scene.

If a message is only sent when an intruder is detected then this may reveal some information to the server. A message has to be sent to the doors and security all the time. In case no intruder is detected then the doors should remain open and the security should not appear.

The CCTV should also be placed in locations where a face can be clearly captured and is unknown to the public since some clever people may decide to interfere with it. The output of the server should not also be a single quantum bit for example $|0\rangle$ when an intruder is detected and $|1\rangle$ for an authorized personnel. This same clever person could flip the bit when it is being relayed. As we saw earlier, a quantum state cannot be cloned because cloning involves measurement and measurement changes the state.

1.1.4 Why does the client require services of the server in this example. An airport is automatically large and the number of authorized personnel in different departments and sections are also many. The client requires storage facilities. The client does not have the capability of performing face recognition on a very large number of images. This computation requires high processing power and speed which the client does not possess.

Fig1.1 shows the model

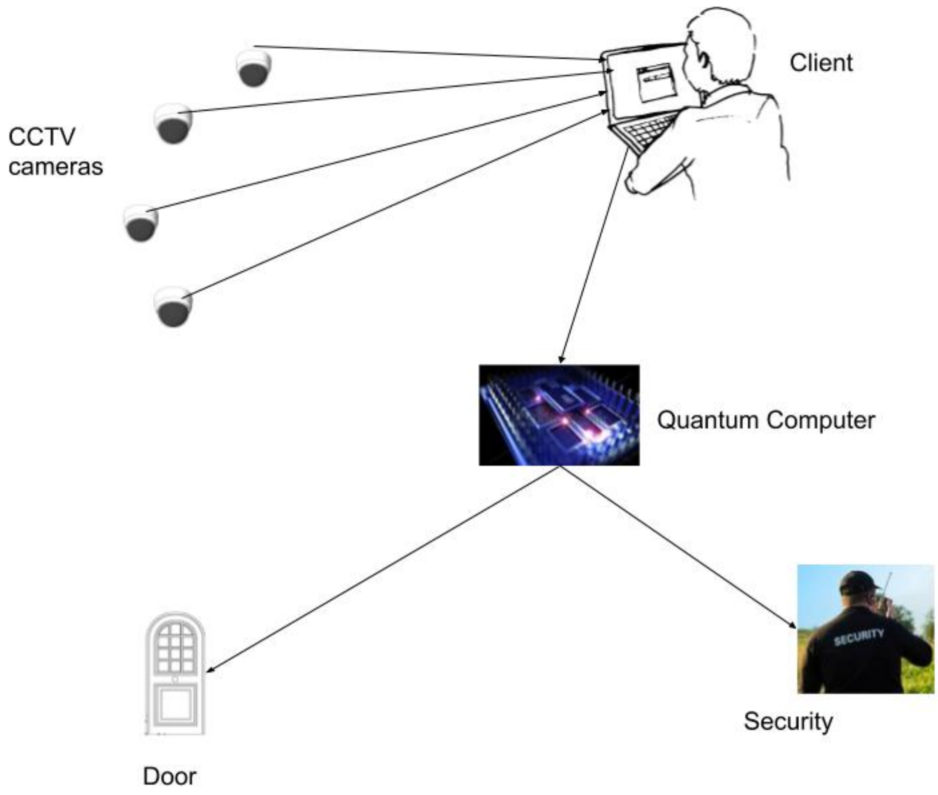

Figure 1.1: The Airport Model

1.1.5 Assumptions made in this model. There are few assumptions made while building this model. The server has the ability to perform face recognition on encrypted face images. Since the output options are two, whether an authorized person or an intruder then different encryptions to represent each of them are produced at different times.

1.1.6 Algorithm. Figure 1.2 is an algorithm of how the model works. The client generates the keys both public and private using (Rivest-Shamir-Adleman)RSA method. The public key is used to encrypt the images before they are sent to the quantum server for storage. The client then then receives images relayed by the camera and encrypts them one by one. The encrypted images relayed are then sent to the quantum server to perform face recognition. The quantum server performs the face recognition on the encrypted images. If the face image matches any of the encrypted images stored, then a message is sent to the doors to remain open and to the security not to act. If the image does not match any of the images stored then a message that doors should close and security should go to the scene is sent.

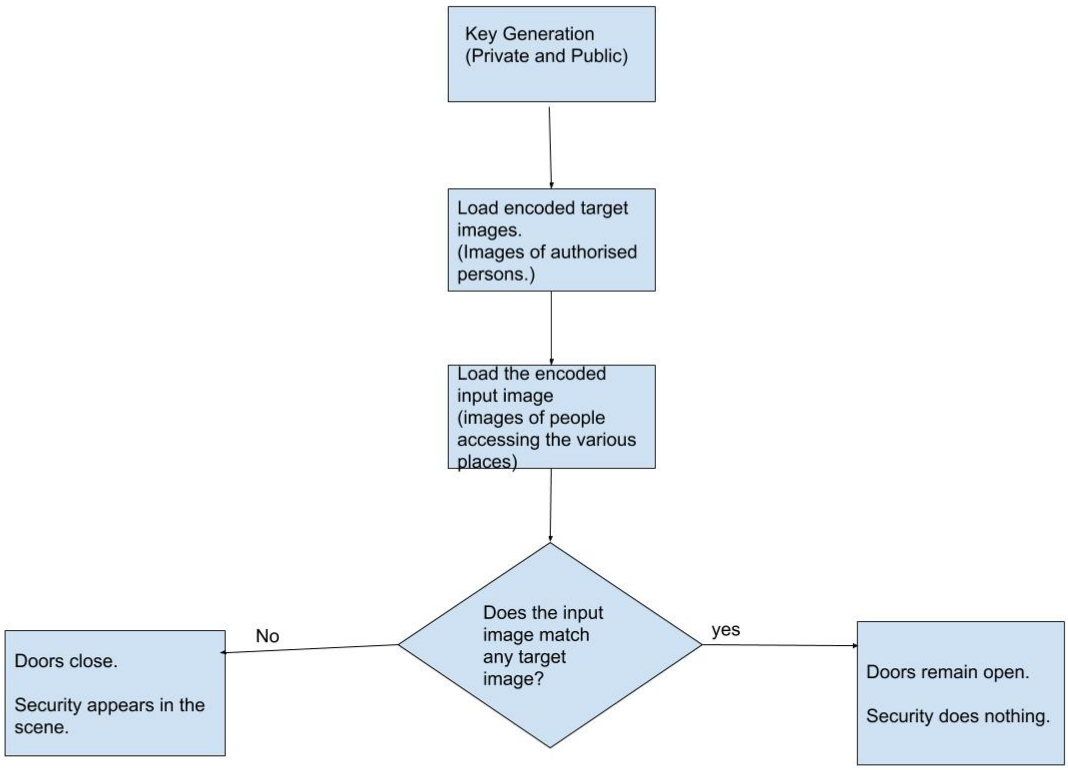

Figure 1.2: Algorithm

1.1.7 How the python code for the model works. The following is a python code using face recognition module in Python (Python Code in Appendix). The code uses two images. One is the image of the authorized personnel while the other is relayed image. In the code the images are not encrypted but we assume in the quantum server they are encrypted. The program uses RSA a module that produces a private key and public key. The public key encrypts the message and the private key decrypts it. In the code the output is encrypted after performing face recognition. The encrypted code is the sent to the classical client operating the doors and to the security. They are prompted to enter their private codes known only to them. The message is then decrypted and they can act accordingly. In case a wrong code is entered then two more chances are given after which the system automatically blocks.

1.1.8 Sample outputs using the python code. For the case where the input face image matches with the target face image. Correct authentication implies that doors should remain open and the security should not appear anyway Fig 1.3.

For the case where the input face does not match with the target face image. Intruder detected implies that doors should close and the security should appear in the scene Fig 1.4.

Figure 1.3: Face match

Figure 1.4: No Match

1.1.9 Limitation of this model. This model poses one major limitation which is very vital. In case a wrong code is entered three times the system automatically blocks. This implies that the required action is not taken for example doors closing. If does are not closed the intruder may get away before the security get to the specific location.

2. Appendix

2.1 Python Code

```
#This code is written in python3

import dlib
import face_recognition
from Crypto.PublicKey import RSA
from Crypto import Random
#This is a face recognition model.

#Key generation.
rand = Random.new().read
private_key = RSA.generate(1280, rand)
public_key = private_key.publickey()

def face():

        #The inputFace is compared with the target face to see whether
        #they images represent same or different persons.

        targetFace = face_recognition.load_image_file
                        ('/home/joan/joan/pythoncode/jj.jpg')

        encodedFace = face_recognition.face_encodings(targetFace)[0]

        inputFace = face_recognition.load_image_file
                        ("/home/joan/joan/pythoncode/DigitalPhoto.jpg")

        encodedInputFace = face_recognition.face_encodings(inputFace)[0]

        outPut = face_recognition.compare_faces([encodedFace ], encodedInputFace)

        u = "The face matches!"
        l = "No match!"

        if outPut [0] == True:
                a = u
        else:
                a = l
        return a

```

```
def encrypt():

        #Here the message encryption is performed based on the results of face()

        a = face()
        k = public_key.encrypt('Intruder detected'.encode('utf-8'),10)
        p = public_key.encrypt('Correct authentication'.encode('utf-8'),10)

        if a == "The face matches":

                enc_data = k

        else:
                enc_data = p

        return enc_data


def decrypt():

        #Decryption of the results from encrypt() are done.

        b = encrypt()
        privateKey = int(input('Please enter your password and press enter'))
        attempts = 1
        chances = 3

        if privateKey == 3045:

                raw_text = private_key.decrypt(b)
                print (raw_text)
        else:

                print('Wrong password')

                while attempts < 3 :
                        chances -=1
                        attempts +=1
                        att = int(input('Attempt %d, you have %d chances left : '
                                                                %(attempts, chances)))
                        if att == 3045:

                                raw_text = private_key.decrypt(b)
                                print (raw_text)
                        else :
```

```
                                print ('Wrong password')

                print ('You have been blocked from the system!')

if __name__ == '__main__':

        print('Recognizing face .......')
        print (face())

        print('Encrypting ..........')
        print(encrypt())

        print('Decrypting ..........')
        print(decrypt())
```

References

Stefanie Barz, Elham Kashefi, Anne Broadbent, Joseph F Fitzsimons, Anton Zeilinger, and Philip Walther. Demonstration of blind quantum computing. *Science*, 335(6066):303–308, 2012.

Michael J Bremner, Richard Jozsa, and Dan J Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*. The Royal Society, 2010.

Anne Broadbent and Stacey Jeffery. Quantum homomorphic encryption for circuits of low T-gate complexity. In *Annual Cryptology Conference*, pages 609–629. Springer, 2015.

Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 517–526. IEEE, 2009.

Ran Canetti, Ben Riva, and Guy N Rothblum. Practical delegation of computation using multiple servers. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 445–454. ACM, 2011a.

Ran Canetti, Ben Riva, and Guy N Rothblum. Two 1-round protocols for delegation of computation. *IACR Cryptology ePrint Archive*, 2011, 2011b.

Joe Fitzsimons. Blind quantum computing and fully homomorphic encryption, 2012. Stackexchange.

Joseph Fitzsimons, Li Xiao, Simon C Benjamin, and Jonathan A Jones. Quantum information processing with delocalized qubits under global control. *Physical review letters*, 99(3), 2007.

Joseph F Fitzsimons. Private quantum computation: An introduction to blind quantum computing and related protocols. *npj Quantum Information*, 3(1):23, 2017.

Craig Gentry. *A fully homomorphic encryption scheme*. Stanford University, 2009.

Brian Hayes. Cloud computing. *Communications of the ACM*, 51(7):9–11, 2008.

He-Liang Huang, Qi Zhao, Xiongfeng Ma, Chang Liu, Zu-En Su, Xi-Lin Wang, Li Li, Nai-Le Liu, Barry C Sanders, Chao-Yang Lu, et al. Experimental blind quantum computing for a classical client. *Physical review letters*, 119(5), 2017.

Ivan Kassal, James D Whitfield, Alejandro Perdomo-Ortiz, Man-Hong Yung, and Alán Aspuru-Guzik. Simulating chemistry using quantum computers. *Annual review of physical chemistry*, 62:185–207, 2011.

Cescily Nicole Metzgar. *RSA Cryptosystem: An Analysis and Python Simulator*. PhD thesis, Appalachian State University, 2017.

Michael Miller. *Cloud computing: Web-based applications that change the way you work and collaborate online*. Que publishing, 2008.

Monique Ogburn, Claude Turner, and Pushkar Dahal. Homomorphic encryption. *Procedia Computer Science*, 20:502–509, 2013.

Yingkai Ouyang, Si-Hui Tan, and Joseph Fitzsimons. Quantum homomorphic encryption from quantum codes. *arXiv preprint arXiv:1508.00938*, 2015.

Charles P Pfleeger and Shari Lawrence Pfleeger. *Security in computing*. Prentice Hall Professional Technical Reference, 2002.

Andrew Steane. Quantum computing. *Reports on Progress in Physics*, 61(2):117, 1998.

Craig Stuntz. What is homomorphic encryption, and why should I care?, 2010. Blog.

Subashini Subashini and Veeraruna Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1):1–11, 2011.

Si-Hui Tan, Joshua A Kettlewell, Yingkai Ouyang, Lin Chen, and Joseph F Fitzsimons. A quantum approach to homomorphic encryption. *Scientific reports*, 6, 2016.

Max Tillmann, Si-Hui Tan, Sarah E Stoeckl, Barry C Sanders, Hubert de Guise, René Heilmann, Stefan Nolte, Alexander Szameit, and Philip Walther. Generalized multiphoton quantum interference. *Physical Review X*, 5(4), 2015.

William G Unruh. Maintaining coherence in quantum computers. *Physical Review A*, 51(2):992, 1995.

Yang Yang et al. *Evaluation of somewhat homomorphic encryption schemes*. PhD thesis, Massachusetts Institute of Technology, 2013.

Makoto Yokoo and Koutarou Suzuki. Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 112–119. ACM, 2002.