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

DECLARATION

This work was carried out at AIMS Rwanda in partial fulfilment of the requirements for a Master of Science Degree.

I hereby declare that except where due acknowledgement is made, this work has never been presented wholly or in part for the award of a degree at AIMS Rwanda or any other University.

Student: Joan Watiri Ngure

Supervisor: Barry C Sanders

ACKNOWLEDGEMENTS

The sun rises from the East and sets in the West. This would not be the case if the earth does not orbit around it. Just as the sun, this work has been made possible by different parties. First of all, I would like to thank the Almighty for giving me the strength and courage to finish the project. African Institute for Mathematical Sciences (AIMS), saw the potential in me, and gave me a platform where I would learn and bring out the best from my potential. I will forever remain grateful for this opportunity. I would also like to pass my special thanks to Barry C Sanders, my supervisor, who supported me very much in this new area of study. His tireless efforts to explain new concepts and give constant feedbacks cannot go unnoticed. The first time I heard the name tutor I did not actually know what it really means. Today it means sacrifice, continued support and service to the people. This was the meaning given by my tutor Edward. I lack words to express my gratitude to him. Lastly I would like to thank my fellow colleagues and everyone who contributed to my success.

DEDICATION

I would like to dedicate this work to my family, always my number one funs.

Abstract

Imagine you have limited computational capabilities and you wish to delegate tasks. You want to be in a position where only you can access, read and interpret the data. Homomorphic Encryption is a scheme that allows computation on encrypted data. The journey to achieve homomorphic encryption and the progress that has been made to date is presented here. We have both Classical Homomorphic Encryption and Quantum Homomorphic Encryption. In classical both the client and server are classical while in quantum the client is classical and the server is quantum. Quantum Homomorphic Encryption has several advantages over Classical Homomorphic Encryption like some tasks that are considered to be hard can easily be performed. A vivid description on how both work is explained in the essay.

Contents

1. Introduction

1.1 Overview

Owning a computer was the new thing in the world. Everyone wanted to have one. The first computers were very large in size but as the years passed by, the size decreased. We now have portable computers. Computers may crash and we lose all our data, or the computation we want to perform, our computers cannot handle. A need arose to increase storage space and processing power. We would not always be purchasing new machines and training staff all the time as the needs changed everyday. Thus we required to store and process our data elsewhere in addition to the local storage. This was characterized by Cloud Computing. Cloud Computing can be described as using the cloud (some server stored somewhere anonymous) to carry out tasks like storage and data management over a network rather than doing it in our own computers.

We enjoyed cloud computing for a while but then the issue of security came along. A user does not have control of whom would access his/her data while in the cloud. Mischievous users may intentionally or accidentally get hold of someone else's data. One can imagine a leakage of a patient's examination results or diagnoses. It started by using for example a symbol to represent a patient. Yes the patient's name may be anonymous, but if someone has access to the hospital's data, they would know how many people are suffering from a certain disease among others. This user can decide to cause unnecessary chaos or theft in case of a bank where a customer's personal information such as credit card pin code is accessed. It became possible to encrypt data and store it but performing computations on this encrypted data was not possible. Encryption is simply using a different representation of data for instance using 123 to represent the word transform. Only someone who knows how to interpret it can use it.

Homomorphic Encryption shows that it is actually possible to perform computations on encrypted data.

1.2 Background and Motivation

In April 2011, a Play Station's network owned by Sony was hacked. There was exposure of customers information like credit card and passwords. The responsibility for this incidence was accepted by Sony. They realized that they had not done enough to ensure security of their customers' data. They would have encrypted it before storage. In this same period researchers also discovered that files in Dropbox were stored unencrypted. This lead to users leaving and closing their accounts with Dropbox since they felt insecure (Ogburn et al., 2013).

According to Pfleeger and Pfleeger (2002), ensuring security would be done at different levels. Securing the hardware (this is the physical system), software (set of instructions executed in the system) and the data itself. In the hardware, locks were put to limit access to unauthorized persons. Systems to detect an intrusion and devices that verify persons identity were put into place. In the software, programs that limit access to database and protect one user from other

users were developed. Also, software programs were written in such a way that their weaknesses could not be easily exploited. Password checkers and virus scanners application programs were also developed. The solution to the data was encryption. Data is only useful to parties if it can be readable and interpreted. Encrypted data is only useful if one can decrypt it. All this assured the user of security. At the instances where the user needed to use the data then there was a problem because the data needed to be decrypted before any form of computations. This raised a security issue again. We needed to develop a system that allows computations on encrypted data. This is known as Homomorphic Encryption. This assured the user of total security.

Previous works have been done trying to explain how Homomorphic Encryption can be implemented on various platforms using different types of data. In 2013 (Ogburn et al., 2013) tried to explain a model which encrypts hospital data. After the computations are done the number of patients suffering from a certain disease is returned in an encrypted format. The hospital's management then decrypts it. Yokoo and Suzuki (2002) showed a system where different people are using it to perform an optimization problem among themselves. An optimization problem is a problem where you want the output to be the maximum based on certain inputs, for example maximum profit. The task is to find the inputs. These people work without revealing any information about their inputs to the server. Using a similar approach to these works, we will discuss possible applications of Quantum Homomorphic Encryption using example use cases.

1.3 Aim and Objectives of the Study

Imagine you own a business and you want to charge different prices to different people based on their financial capabilities and customer loyalty. Your resources are limited thus you have to outsource resources such as storage. You do not want this information to leak to the public since this may make some customers to withdraw. You want to maintain the privacy of this information. The system should be able to determine the financial status and loyalty of a customer then charge accordingly. No information about the customer is revealed or the prices charged to other customers.

The aim of this essay is to show the current state of security while requiring services from an untrusted server. The objectives include explaining vividly the security concerns and showing steps taken to counter these issues. This is done by discussing about classical computing with its limitations compared to quantum computing. We will also see the different ideas quantum computing brings about which makes it attractive compared to classical computing, the process of Quantum Homomorphic Encryption in details and application examples.

1.4 Purpose of the Study

There was a time when bank robbery was the order of the day. Banks solved this issue by putting so many security measures such as several locks and authentication before accessing where the actual money was. Banks also required several people to approve before large transactions were

made. In 2017, three men dug a tunnel for six months into a Kenyan bank and made away with 50 million Kenyan shillings which is approximately 500,000 US dollars. This shows that security issues cannot be solved 100 percent. A determined person will always find a loophole. How can we ensure that the loopholes are not easily discovered and exploited?

The purpose of this essay is to show how computing security has evolved over the years, the steps and measures taken in different settings using the resources available, the plans under way to secure the data and computing in future and the new discoveries especially Quantum Computing and its effect on computing security. We will also see why one would consider a Quantum Homomorphic Encryption over a Classical Homomorphic Encryption.

1.5 Organization of Essay

This essay is subdivided into five different chapters. In Chapter 2 we will discuss about classical and cloud computing with their limitations compared to quantum computing. In Chapter 3, quantum computing is vividly discussed. Different types of encryption in quantum computing and why one is chosen over the other are also discussed. Chapter 4 talks about examples of use cases where Quantum Homomorphic Encryption could be used. Chapter 5 contains the conclusion with recommendations.

2. Classical and Cloud Computing

In this chapter we will discuss about classical computing and its limitations compared to quantum computing.

2.1 Classical Computing

All the computers we use today are classical. This means that information is stored in bits of 0s and 1s. Operations performed by the computers are faster than actually performing it by hand. For example, performing calculations using a calculator as opposed to calculation by hand. The probability of making errors is very high by hand.

A computer is a universal machine. A universal machine can be described as an electronic device with the ability to solve problems. These problems have solutions, that can be represented by a set of instructions. These problems are known as computational problems because the classical computer can solve them.

Computational problems include decision problems, search problems, function problems and optimization problems. Decision problems are similar to making a choice between true or false, where true can be represented by 1 and false 0. Search problems are like looking for a word in a text. Function problems are for example evaluating $\cos(\theta)$. Optimization problems are problems whereby, you want the output to be the maximum possible value. We therefore find the inputs which satisfy this condition. What prices will you charge the customers in order to make maximum profit without exploiting your customers?

Computers can be connected to each other in order to communicate. This is known as distributed computing. Each processor has its own memory but can communicate with others by sharing messages. One memory can also be shared by multiple processors (parallel computing). Parallelism speeds up computations since different tasks can be performed at the same time. The more the number of processors the more expensive a machine is.

To solve problems, a computer requires resources. The more resources required the more complex it is described. Thus computational complexity is defined by the resources used. A client may wish to perform tasks which require computational power beyond their capabilities. They may also have limited resources, therefore, they decide to outsource resources from remote servers. The client therefore delegates tasks. Can the client trust the output from the server? If the client cannot trust the server, then this raises a reliability issue. One of the solution to this issue is that, the client can choose multiple servers provided atleast one of them is reliable (honest) and delegate tasks to them (Canetti et al., 2011a). The problem with this protocol is that, it is difficult to determine which server is honest. This makes verifying correctness to be based on probability. This can be done by a client dividing the task into stages. The client can therefore access the output of each stage. This allows the client to check for inconsistencies between the servers' outputs. In case of inconsistencies and if the stage affected can easily be performed by the client, then determining the dishonest server becomes simple otherwise it would be very

difficult. There are times when a client requires urgent results. During these times, the client may have no time to compare outputs from different servers. Requiring services from different servers is expensive and time consuming.

Servers could decide to give false results (dishonest) based on a couple of reasons. The resources required to perform the computation may be a lot to the server. This makes the profit made less than the estimated one. The server could also benefit from some outputs for example poll results. An employee could just decide to interfere with the computations being executed (Canetti et al., 2011a).

Cloud computing is an example of a delegated computing system.

2.2 Cloud Computing

2.2.1 What is Cloud Computing?. Cloud Computing is the process whereby, computation is done somewhere else (in the cloud), and is similar to working on your personal computers (Hayes, 2008). Cloud is some server stored somewhere anonymous only the service providers knows about its whereabouts. Clients for example companies or individuals are able to request services such as storage, manipulation of data and computations among others. Cloud services are offered as demanded by the clients. Clients do not have to train their staff, worry about data loss in case of system crashes, buy new infrastructure or get licenses for softwares.

The three major types of cloud computing services are: Platform as a Service (PaaS) such as Google App Engine (which is mainly used by developers), Software as a Service (SaaS) for example email and Facebook and Infrastructure as a Service (IaaS) like storage facilities and servers.

Examples of companies offering cloud services include Amazon, Google and IBM.

The main disadvantage of cloud computing is security on (Subashini and Kavitha, 2011) and (Miller, 2008). Security is one of the issues that is making some companies to shy away from cloud computing. Most governments have rules that their sensitive information should not leave their respective countries. The cloud servers may be located anywhere in the world. In cloud computing data may be changed and this interferes with its integrity. Integrity can be defined as the quality of wholeness. Data which has been changed is not whole. Data could be changed to manipulate intended results. Duplication of data could also occur. This amounts to information theft. In cloud computing making an exact copy of data is possible and the client may not learn about it in good time. Data can be interfered with. Since many clients are using the same server, the possibility of their data mixing up is very high. Some clients could also be malicious and interfere with other client's data. Leakage may also occur. Sensitive information may be exposed to unintended audience. For example, if a user receives emails from different people but always performs a search on a specific sender, this could reveal to the server that this person has a level of importance to the client. The server may exploit this loophole to gather information on why this person interest the client.

The bottom line is, we do not trust the server. Our privacy, confidentiality and integrity is put at risk. The data may end up falling into the wrong hands with unauthorized personnel which

would lead to severe consequences. Organizations whose data is mainly sensitive information, would find it hard to adopt cloud computing. A user can encrypt data before storing it in the cloud and when he/she needs it they can download and decrypt it. Thus, this user cannot use other services apart from the storage. Other costs of hardware and software, which will be used to process the data are incurred (Yang et al., 2013).

A solution to this is homomorphic encryption, whereby a server performs computations on encrypted data.

2.3 Classical Homomorphic Encryption and its Limitations

Homomorphic Encryption is also an example of a delegated computing system. This system ensures secure delivery of information, storage and computations. Homomorphic encryption is the ability of a server to work on encrypted data. According to Ogburn et al. (2013), homomorphic encryption can either be partial, somewhat or fully. In Partial Homomorphic Encryption, you can either perform addition or multiplication on encrypted data but not both. A Somewhat Homomorphic Encryption technique has the ability to perform both multiplication and addition on limited numbers. A Fully Homomorphic Encryption can support both multiplication and addition and is not limited to any numbers. Most of us would choose a Fully Homomorphic Encryption system. Its only disadvantage is that, it is less efficient as compared to Partial Homomorphic Encryption and Somewhat Homomorphic Encryption.

Figure 2.1 is an example of a Homomorphic Encryption. The words are concatenated after encryption. The result obtained after decryption, is the same with the result if the concatenation was performed without the encryption.

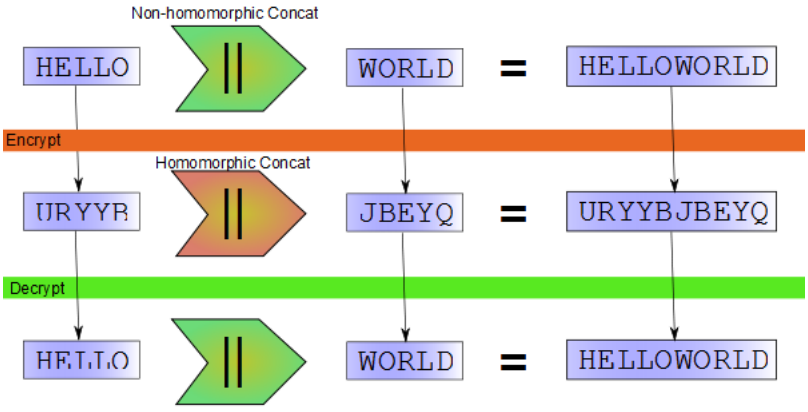

Figure 2.1: Encrypted concatenation
(Stuntz, 2010)

2.3.1 Why is Homomorphic Encryption considered secure?. As it can be seen from Fig 2.1, no information is revealed to the server. The server can learn about the information only if it can decrypt it. The server can only decrypt the information if they know how you as the

client encrypted it and they possess the decryption key. This may be termed as negligence from the client's side. In Homomorphic Encryption the integrity of the data is maintained. How will the server change something like 'JBEYQ' which they can't even interpret? The client may get some results and tell that something is wrong, in case the server changes something. It is very easy to manipulate something that you can understand but difficult otherwise. Data cannot be duplicated. The encrypted data can be duplicated but the real unencrypted data cannot. Sensitive information cannot leak too. Privacy and secrecy are also maintained.

2.3.2 Limitations of Classical Homomorphic Encryption as compared to Quantum Homomorphic Encryption. We always seek to improve the existing things. The big question is why Quantum and not Classical? A classical computer is limited in terms of computation power as compared to a quantum computer. Some tasks like factorization that can be performed easily on a quantum computer are hard and require a lot of time on a classical computer (Unruh, 1995). The main idea is to improve speed and efficiency. This is especially when dealing with large chunks of data like performing a search operation in a large database and factorization of large numbers.

This makes Quantum Homomorphic Encryption attractive.

3. Quantum Computing and Quantum Homomorphic Encryption

In this chapter we discuss what quantum computing and Quantum Homomorphic Encryption entails.

3.1 Quantum Computing

In Quantum Computing, we have quantum bits (qubits) represented as $|0\rangle$ and $|1\rangle$ as opposed to the classical bits (0 or 1) in classical computing. We also have quantum logic gates namely Hadamard (H), Z, X and CNOT which form the backbone of quantum computing. The gates can be represented as follows using circuit diagrams:

Figure 3.1: Hadamard gate

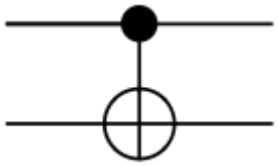

Figure 3.2: CNOT gate

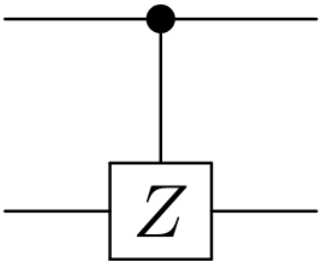

Figure 3.3: Z gate

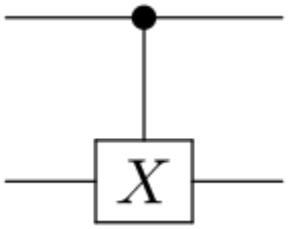

Figure 3.4: X gate

The two main ideas brought about by quantum computing are superposition and entanglement which are adopted from quantum mechanics. If any computation uses these two mechanisms it is known as quantum computing. In quantum computing, a zero and a one can be represented simultaneously. This is known as superposition. It saves on processing time. For example, you have a box full of white chalks, but, you have one green chalk that you want to obtain from the box. The box is not transparent hence you cannot see what is inside. To solve the problem classically, you have to pull one chalk at a time until you get the green one. Quantum permits you to hold all the chalks at the same time and use probability to pull out the green chalk. This is the idea brought about by superposition.

Entanglement is when two particles interact and acquire states of each other, thus the output will be a mixture of the interacting particles, hence to describe one particle one has to refer to

the other. To illustrate an entanglement, let us use a Hadamard and a CNOT gate. A CNOT gate is a Controlled NOT gate. If the control is a qubit of one then we apply a NOT to the bit in the target, otherwise nothing happens. This is shown in Figure 3.5 below.

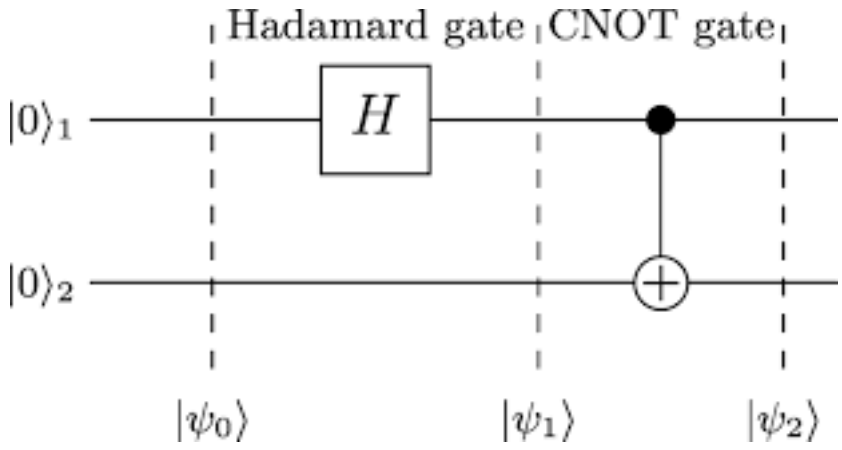

Figure 3.5: An entangled circuit

At $|\psi_0\rangle$ we have the output as $|0\rangle_1|0\rangle_2$. It can also be written as $|0_1 0_2\rangle$. We have $|0\rangle_1$ qubit passing through the Hadamard gate (H). By definition $H|0\rangle = |\bar{0}\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$. $|0\rangle_2$ does not change. The output $|\psi_1\rangle$ is $\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)|0\rangle = \frac{|0\rangle_1|0\rangle_2 + |1\rangle_1|0\rangle_2}{\sqrt{2}}$. We then pass through the CNOT. Since $|0\rangle_1$ is 0, $|0\rangle_2$ remains unchanged, $|1\rangle_1$ is 1 hence $|0\rangle_2$ flips to $|1\rangle_2$. Thus the output $|\psi_2\rangle$ is $\frac{|0\rangle_1|0\rangle_2 + |1\rangle_1|1\rangle_2}{\sqrt{2}} = \frac{|0_1 0_2\rangle + |1_1 1_2\rangle}{\sqrt{2}} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$.

This is a maximally entangled state.

Quantum computing also supports parallelism. As opposed to classical computing where different processors are required, quantum computing parallelism is done on the same chip. As long as the tasks are related, they are carried out at the same time.

3.2 Approach Techniques

The properties of quantum computing enable computations to be performed on encrypted data (hidden). This is the ability for a server to process data delegated to it, without accessing it. The two types of techniques used are Blind Quantum Computing and Quantum Homomorphic Encryption. Blind actually means you cannot see. This encryption uses the same application. The server is blinded from input, output and computation. This is a technique where the server performs computations they do not know of on encrypted data and produces results which they do not also know from an encrypted input. Sounds hilarious right? Quantum Homomorphic Encryption is a technique where the server performs computations they actually know of on encrypted data, that is, the input and output is encrypted but the computation is known.

3.2.1 Differences between Blind Quantum Computing and Quantum Homomorphic Encryption. (Broadbent et al., 2009), (Fitzsimons, 2017),Ouyang et al. (2015),(Fitzsimons, 2012)

	Blind Quantum Computing	Quantum Homomorphic Encryption
1	Everything is encrypted (input, output and computation) (Huang et al., 2017) (abstract).	Only the input and output are encrypted. (Broadbent and Jeffery, 2015) (Introduction).
2	Uses information theoretical security protocol (can not be broken in) (Fitzsimons, 2017) (pg1).	Uses computational security protocol (it is hard to break in).
3	It is possible to expand the system based on the client's needs (Fitzsimons, 2012).	The system is fixed (Fitzsimons, 2012).
4	An interference with the protocol is usually detected because measures are put into place (Broadbent et al., 2009)(abstract).	No measures are put into place hence there is a probability that an interference can go undetected.

Table 3.1: Differences between Blind Quantum Computing and Quantum Homomorphic Encryption

3.2.2 Why Choose Quantum Homomorphic Encryption Over Blind Quantum Computing?. Information theoretical security used in Blind Quantum Computing is very hard to implement on a classical client hence its implementation is very expensive (Ouyang et al., 2015). You cannot break into a system using information theoretical security even if your computational power is greater than the system. Due to the high cost of implementation, very few clients will afford it. In most Blind Quantum Computing protocols, the client has some quantum computation requirements, though protocols that do not have this requirement exist (Broadbent et al., 2009). This makes us choose Quantum Homomorphic Encryption which uses computational security which is hard to break in. Thus the risk of an adversary attacking the system is low.

3.3 Quantum Homomorphic Encryption

As seen earlier, it is possible to delegate tasks (data processing) to a server while the access to the data is not given away. The ability of a server to perform computations on encrypted data solves our security issue.

3.3.1 How the security issue is solved in Quantum Homomorphic Encryption?. Since the server does not know the input or output it is impossible to change data to manipulate results, also, the integrity of the information is upheld. Data cannot be duplicated because, in quantum, duplication involves measurement and this changes the state thus you will never get the exact copy (Kassal et al., 2011). Quantum Homomorphic Encryption as we saw earlier, uses computational security. It is very hard to break or attack the encrypted data. To break in you require a lot of time. This makes it almost impossible for sensitive information to be leaked.

3.3.2 How does Quantum Homomorphic Encryption work?. There are four stages involved namely: Key generation, Encryption, Evaluation and Decryption (Broadbent and Jeffery, 2015).

All these stages are performed by the classical client .

3.3.3 Key generation and Encryption. Key generation produces a pair of public and private key. The public key is used to encrypt data while the private decrypts it. Key generation can be done using the RSA method. While using a computer program such as Python, an already existing RSA module exists. This implies that the client inputs a number and a pair of public and private key is generated.

RSA can also be done using a mathematical approach. Two random large prime numbers say p and q are chosen. Let us take $p = 17$ and $q = 11$. We then calculate $N = p \times q$, $17 \times 11 = 187$. N is the system modulus. The factorial of p and q is then calculated to find the public key which is used for encryption. In our example, we get the factorial as follows $(17-1)(11-1) = 16 \times 10 = 160 = 2^5 \times 5$. We now choose a number e that does not divide any factor of $(p-1)(q-1)$. We can take our $e = 3$ as 3 does not divide either 10 or 16. Thus 3 is our public key. To get the private key we multiply our public key with a number say d such that $1 = e \times d \mod (p-1)(q-1)$. We therefore calculate d as follows,

$$\begin{aligned} e \times d &\equiv 1 \mod (p-1)(q-1) \\ 3d &\equiv 1 \mod 160 \\ 3d &= 1 + k \cdot 160 \end{aligned}$$

k is any random number that satisfies the equation. If we let $k = 2$, then $d = 107$. Thus our private key is 107 (Meissen, 2012). Encrypted data can be referred to as a cipher text while unencrypted data is called a plain text. A plain text is transformed into a cipher text using the public key. Cipher text $=$ (plain text)$^e \mod N$ while the plain text $=$ (cipher text)$^d \mod N$.

The client can generate as many key pairs as they want to. This means that different public keys can encrypt the data producing different forms of encrypted data. The client is in a position to choose randomly which encryption to send. Since in homomorphic encryption the computation is known to the server, if you always send a certain encryption to perform the same computation, it would be easy for the server to start guessing. Also, if the server knows that different encryptions represent the same data, then it would be easy to guess too.

For example, you delegate the server to perform factorization. Factorization problems are easy to verify when given the results but hard to solve. It is very difficult for the server to know exactly which number you are factorizing since the input and output are encrypted. The computation does not reveal sufficient information about the input and output which would make an attack easy, apart from the input is a number and the outputs are prime numbers.

How many numbers and prime numbers exist? Will the server start guessing one by one? How long will it take for the server to actually guess the correct number? It is possible for the server to guess the correct number but the probability is very low maybe 1 out of a million trials (Yang et al., 2013).

3.3.4 Evaluation and Decryption. After key generation and encryption, we then evaluate. This is checking the correctness of the algorithm. The public key encrypts the already existing cipher text to produce another cipher text. For example let us encrypt g in mod of a prime number say p. Then we encrypt it again in $\mod 2$. Our cipher text reads c $=$ (g mod p) mod 2. We first decrypt c in $\mod 2$ using the public key to check whether we will get g mod p. If we get the desired result then the algorithm is correct. However the time taken to evaluate and the overall cost should be less than the individual cost of key generation, encryption and decryption.

We have our cipher text and private key, we combine them to produce the plain text. Plain text $=$ (cipher text)$^d \mod N$. This is our decryption process. The decryption is done by the client after they receive the results of the delegated task (Ouyang et al., 2015).

4. Applications

In this chapter we will discuss examples of places where Homomorphic Encryption would be used. This section seeks to review already existing proposals and use cases.

4.1 School Example

Figure 4.1: Students in the library.
(Archer et al., 2017)

We are familiar with students dropping out of schools based on different reasons. For example not everyone who joins graduate school makes it for graduation. The reasons for dropping out vary from one student to another. In some areas, most girls drop out in high school due to early pregnancies. What factors lead to early pregnancies? Peer influence, lack of proper guidance or adventure could be some of the reasons. Other reasons that could lead to school dropout may include illnesses, lack of school fees and psychological issues.

We aim at reducing the number of school dropouts. We want to make predictions of a possible dropout before it actually happens. This will enable us to find a way to best assist the student as the Ministry of Education. Predictions are made based on statistics which means that we have to access the students data. We are dealing with all the schools for example high schools. This implies that the data should be stored in a central place.

We do not fully trust the server. We also do not trust the other parties who may be using the server. No school is given access to another school's data. In some instances, the school may not

have sufficient information to make a prediction. These are cases where for example, a sickness may be the cause of a dropout. This means that we require information from a hospital. We may also require information from different hospitals if the student has sought medication from more than one hospital.

Another scenario is when a student drops out due to lack of school fees. We may need to check the financial status of the parents or guardian. This implies that banks and welfare services would be of great help. Different institutions therefore need to be integrated into the system. This makes it even more difficult. The schools may be reluctant to provide private information about the students due to laws that govern them. The central place where information is stored could be a potential target for attackers (Archer et al., 2017).

Homomorphic Encryption provides a solution to most of these problems. Data is encrypted before storage and computations are performed on the encrypted data. This ensures data sharing without breaking the laws. It also reduces the risk of an attack on the database. Data is encrypted in storage and also in transit. Different schools and sectors encrypt their data differently. They all use for instance, the RSA method, but have different public and private keys, therefore, no institution can decrypt another institution's data. As a result no single institution can access the whole data. We may also use secret sharing as an additional security measure to allow any computation. Secret Sharing as the name suggests, is the process whereby, authorization requires different pieces to be put together. The pieces are held by different people such that all combined can access the database, but an incomplete number of persons cannot. A good example is in a bank where a large transaction must be authorized by different people before it is processed. This builds the confidence and a sense of ownership to the parties involved. We also ensure that the minimum number of persons who can authorize involve persons who cannot collude.

The first step is that different parties should approve access of a party to the database. After access is approved, a query is made to the server. The query is encrypted but the server knows the computation it is going to perform for example search function. Data about a specific student is accessed from different access points like a hospital and school. The data from these two places is encrypted differently and may be stored in different databases. If for example we get data from the school which shows that a student has been constantly sent home as a result of failure to pay school fees, we can query about the bank details. After the data is retrieved, it is then merged. A prediction is therefore made from the merged data based on the characteristics or the model used. Batch requests can also be made such that the dropout risk of students from an institution is performed simultaneously.

This could be slow using a classical server and can even take days for a query to be processed. We therefore recommend a quantum computer as a server as it is fast and efficient.

4.2 Hospital Example

Information about a person's health is usually sensitive and medical practitioners swear an oath of secrecy. This information should be protected at all costs to avoid leakage which may cause damage both to an individual and the institution itself. Researchers or the medical practitioners

would want to use this data to conduct a research about a certain drug and its effect to the patients. They may also want to use the data for better planning or to predict an outcome. All these should be done while preserving the privacy of the patient.

Archer et al. (2017) describes a use case where the doctors record patients' details like blood group, genotype, signs and symptoms, lab test results, drug prescription and the efficiency of the medication administered. This is done over a period of time. The data is then encrypted and stored in the database where the server is not fully trusted as shown in figure 4.2.

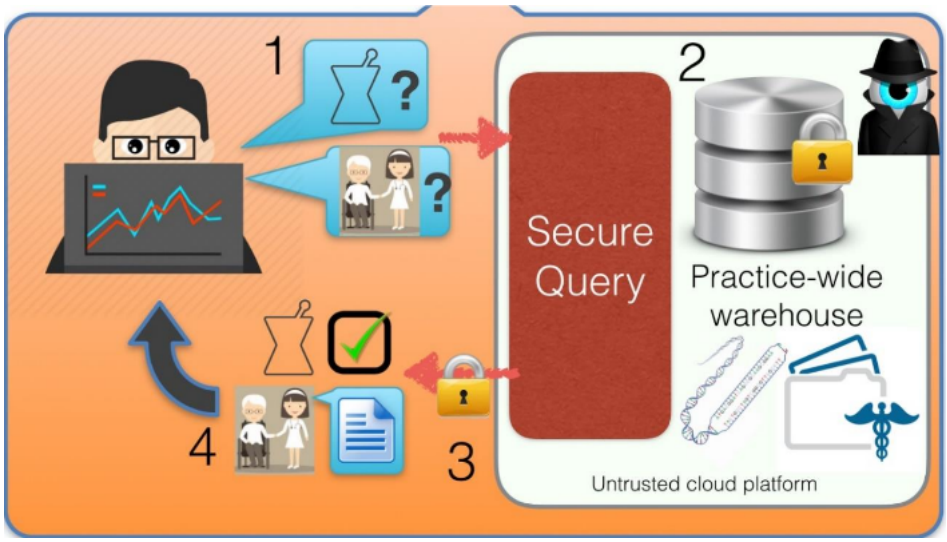

Figure 4.2: Hospital Example
(Archer et al., 2017)

For cancer patients, the tumors differ from one patient to another. This implies that the treatments also differ and patients may also react differently to the medication. The reaction may depend on things like the size of the tumor, allergic reactions and genes just to name a few. The details of different patients are already in the database. If a new patient visits the hospital the doctor queries the database based on the patient's details that are similar to those stored in the database, which may include signs and symptoms among others. The query is encrypted but the computation is known to the server. After getting the results, the doctor decrypts and knows the best treatment to administer to the current patient based on different past results. This saves on time required to test the medication that would work on the patient. The patient requires to pay the bill after treatment. To pay the bill things like tests performed, diagnosis and prescription are exposed to the cashier. This could lead to leakage. Using homomorphic encryption, the amount could just be calculated and the payment made without exposing any additional details.

According to (Naehrig et al., 2011), the data stored about the patients can also be useful in several ways. Based on the patients data, functions would be computed to determine certain risks such as possibility of a re-infection and attacks like hypertension. Alerts are then sent to the doctor and the patient. The patient is hence in a position to act accordingly before the risk becomes a reality and safety precautions taken. This improves the efficiency of the services provided by the hospital.

This argument is based on a classical client and a classical server. It can be implemented using a classical client and a quantum server as we have earlier seen the advantages of quantum computing over classical computing for instance speed and efficiency. This use case would therefore work best using Quantum Homomorphic Encryption.

The school and hospital use cases can also be implemented using Blind Quantum Computing. As we saw earlier, its main disadvantage is cost of implementation. This makes Quantum Homomorphic Encryption our most preferred option.

5. Conclusion and Recommendations

Security and privacy are the two most valued priorities among the many things we regard as important. When selecting a house location, we often put security as the major factor of consideration. You do not want someone to intrude your privacy or walk away with your hard earned cash or favorite items. Before telling our friends and family members something about ourselves we always consider whether they are in a position to keep it to themselves or not. We apply for jobs where we expose a lot of our details. We remain hopeful that none of this information will fall into the wrong hands. For instance, most ladies want to hide their age. After providing their information, they trust that the institution will handle it in a very secure and private manner.

There is a time people did not believe that we would actually have computers. Look at the way technology has change the world! Personal computers could no longer handle our problems. We needed to replace them with better machines which was expensive for large organizations. We therefore required to outsource resources. Cloud computing came along and we enjoyed it for a while until the issue of security popped up. There have been numerous trials to solve this issue like having several authentications before being granted access. The most interesting solution so far is Homomorphic Encryption. One can store, process and transmit encrypted data while still maintaining its confidentiality, privacy and security.

In this essay, a solution to the security issue while outsourcing and requiring services of untrusted server has been presented. Homomorphic Encryption is the process where by, computations are performed on encrypted data. The different solutions tried before Homomorphic Encryption such as multiple authentications, have their own limitations. All these solutions have loopholes like bypassing passwords which put security of the information at a risk. Homomorphic Encryption has proved to be a solution with close to no loopholes, unless someone steals your private key. A private key as we can recall is the key used for decryption. Data is encrypted in storage, transit and during computations.

Two examples, a school and a hospital are given. These are places where Quantum Homomorphic Encryption (QHE) could be useful. QHE has several advantages over Classical Homomorphic Encryption (CHE) like better computational power, speed and efficiency. Quantum states cannot also be duplicated due to their non-cloning property. Cloning involves measurement and measurement changes the state as illustrated in the essay. These advantages of QHE over CHE make QHE more attractive. A comparison between Blind Quantum Computing (BQC) and QHE is clearly outlined. The main disadvantage of BQC over QHE is cost of implementation. We want more clients to be involved thus QHE is the most attractive solution to our problem.

Clearly QHE is possible. We have seen it can be implemented using a classical client and a quantum server. The problem is, there are very few quantum computers in the world hence testing and implementation is difficult. The number of people with quantum knowledge are also few in the world. This is a major setback in this area of study. This work is mainly theoretical. We need more people to enroll in quantum courses and help develop more quantum computers. With an increase in the number of quantum computers, research and experiments will be easy to perform. Different people come up with different ideas, thus, an increase to the number of people with quantum knowledge will lead to great improvements and new innovations. This will

make cloud computing environments more reliable and popular among institutions dealing with very delicate data. Major improvements can also be made in the examples given and spread in other areas not mentioned.

References

Archer, D., Chen, L., Cheon, J. H., Gilad-Bachrach, R., Hallman, R. A., Huang, Z., Jiang, X., Kumaresan, R., Malin, B. A., Sofia, H., et al. (2017). Applications of homomorphic encryption.

Barz, S., Kashefi, E., Broadbent, A., Fitzsimons, J. F., Zeilinger, A., and Walther, P. (2012). Demonstration of blind quantum computing. *Science*, 335(6066):303–308.

Bremner, M. J., Jozsa, R., and Shepherd, D. J. (2010). 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.

Broadbent, A., Fitzsimons, J., and Kashefi, E. (2009). Universal blind quantum computation. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 517–526. IEEE.

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

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

Canetti, R., Riva, B., and Rothblum, G. N. (2011b). Two 1-round protocols for delegation of computation. *IACR Cryptology ePrint Archive*, 2011.

Fitzsimons, J. (2012). Blind quantum computing and fully homomorphic encryption. Stackexchange.

Fitzsimons, J., Xiao, L., Benjamin, S. C., and Jones, J. A. (2007). Quantum information processing with delocalized qubits under global control. *Physical review letters*, 99(3).

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

Gentry, C. (2009). *A fully homomorphic encryption scheme*. Stanford University.

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

Huang, H.-L., Zhao, Q., Ma, X., Liu, C., Su, Z.-E., Wang, X.-L., Li, L., Liu, N.-L., Sanders, B. C., Lu, C.-Y., et al. (2017). Experimental blind quantum computing for a classical client. *Physical review letters*, 119(5).

Kassal, I., Whitfield, J. D., Perdomo-Ortiz, A., Yung, M.-H., and Aspuru-Guzik, A. (2011). Simulating chemistry using quantum computers. *Annual review of physical chemistry*, 62:185–207.

Meissen, R. (2012). A mathematical approach to fully homomorphic encryption. *A Major Qualifying Project Report Submitted to The Faculty of the Worcester Polytechnic Institute*.

Metzgar, C. N. (2017). *RSA Cryptosystem: An Analysis and Python Simulator*. PhD thesis, Appalachian State University.

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

Naehrig, M., Lauter, K., and Vaikuntanathan, V. (2011). Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM.

Nielsen, M. A. and Chuang, I. (2000). Quantum computation. *Quantum Information. Cambridge University Press, Cambridge*.

Ogburn, M., Turner, C., and Dahal, P. (2013). Homomorphic encryption. *Procedia Computer Science*, 20:502–509.

Ouyang, Y., Tan, S.-H., and Fitzsimons, J. (2015). Quantum homomorphic encryption from quantum codes. *arXiv preprint arXiv:1508.00938*.

Pfleeger, C. P. and Pfleeger, S. L. (2002). *Security in computing*. Prentice Hall Professional Technical Reference.

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

Stuntz, C. (2010). What is homomorphic encryption, and why should I care? Blog.

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

Tan, S.-H., Kettlewell, J. A., Ouyang, Y., Chen, L., and Fitzsimons, J. F. (2016). A quantum approach to homomorphic encryption. *Scientific reports*, 6.

Tillmann, M., Tan, S.-H., Stoeckl, S. E., Sanders, B. C., de Guise, H., Heilmann, R., Nolte, S., Szameit, A., and Walther, P. (2015). Generalized multiphoton quantum interference. *Physical Review X*, 5(4).

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

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

Yokoo, M. and Suzuki, K. (2002). 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.