# 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. Classical and Cloud Computing

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

## 1.1 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 describe 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 that satisfies 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, 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 the inconsistencies between the servers' outputs. In case of inconsistencies and the stage affected can easily be performed by the client, then determining the dishonest server is 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 be give false results (dishonest) based on a couple of reasons. The resources required to perform the computation maybe a lot to the server. This makes the profit made less than the estimated one. The server could also be a beneficially of 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.

## 1.2 Cloud Computing

**1.2.1 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. This is mainly used by developers. Software as a Service (SaaS) for example email and Facebook. 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. This argument is based 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 maybe located anywhere in the world. In cloud computing data maybe changed which 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, possibility of 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 maybe exposed to unintended audience. For example, if a user receives emails from different people but always performs a search from 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 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 in 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.

## 1.3 Classical Homomorphic Encryption and its limitations

Homormophic 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, you can either perform addition or multiplication on encrypted data but not both. A somewhat technique has the ability to perform both multiplication and addition on limited numbers. Fully can support both multiplication and addition and is not limited to any numbers. Most of us would choose a Fully Homomorphic Encryption system. It only disadvantage is that it is less efficient as compared to partial and somewhat.

Figure1.1 is an example of an 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.

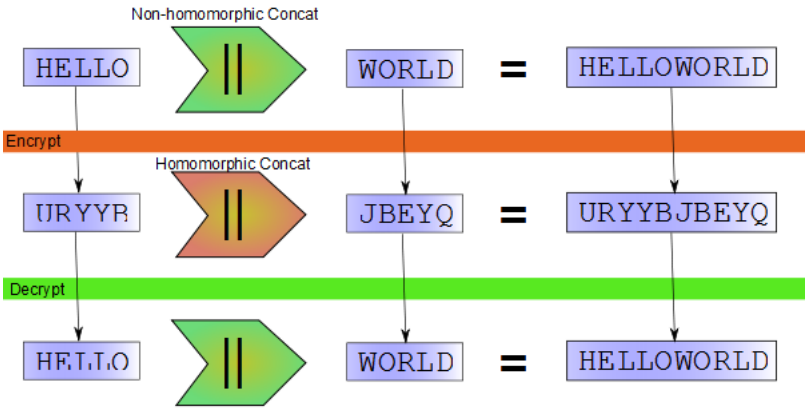


Figure 1.1: Encrypted concatenation
(Stuntz, 2010)

**1.3.1 Why is Homormophic Encryption considered secure?.** As can be seen from fig1.1, 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 they 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 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 is also maintained.

### 1.3.2 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.

# 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.

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

161 Monique Ogburn, Claude Turner, and Pushkar Dahal. Homomorphic encryption. *Procedia Com-*
162 *puter Science*, 20:502–509, 2013.

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

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

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

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

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

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

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

176 William G Unruh. Maintaining coherence in quantum computers. *Physical Review A*, 51(2):992,
177 1995.

178 Yang Yang et al. *Evaluation of somewhat homomorphic encryption schemes*. PhD thesis, Mas-
179 sachusetts Institute of Technology, 2013.

180 Makoto Yokoo and Koutarou Suzuki. Secure multi-agent dynamic programming based on homo-
181 morphic encryption and its application to combinatorial auctions. In *Proceedings of the first*
182 *international joint conference on Autonomous agents and multiagent systems: part 1*, pages
183 112–119. ACM, 2002.