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. Quantum Computing and Quantum Homomorphic Encryption

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

1.1 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:

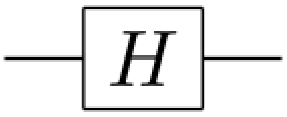

Figure 1.1: Hadamard gate

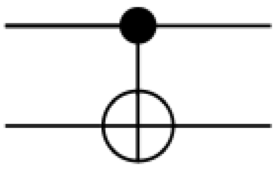

Figure 1.2: CNOT gate

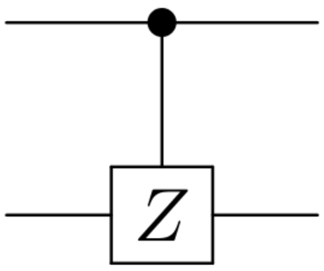

Figure 1.3: Z gate

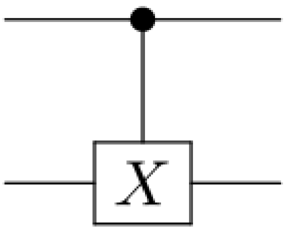

Figure 1.4: X gate

(Nielsen and Chuang, 2000)

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 you have to refer to the other. To illustrate an entanglement let's use a Hadamard and a CNOT gate. A CNOT gate is a Controlled NOT gate. If the control is qubit of one then we apply a NOT to the bit in the target, otherwise nothing happens. This is shown in Figure 1.5 below.

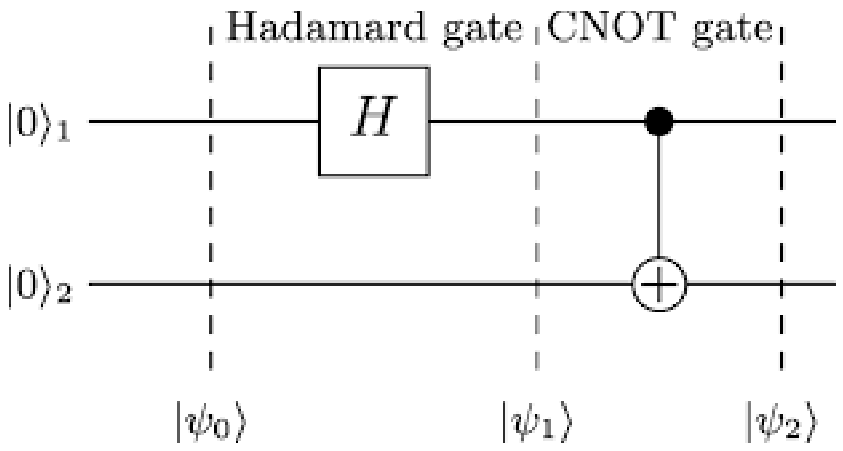

Figure 1.5: An entangled circuit

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).

$$H|0\rangle = |\bar{0}\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}.$$

$|0\rangle_2$ does not change.

The output at $|\psi_1\rangle$ 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 the 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 at $|\psi_2\rangle$ 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.

1.2 Encryption

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 server performs computations they do not know 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.

1.2.1 Differences between Blind Quantum Computing and Quantum Homomorphic Encryption. (Broadbent et al., 2009), (Fitzsimons, 2017),Ouyang et al. (2015),(Fitzsimons, 2012) Table 1.1

	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 1.1: Differences between Blind Quantum Computing and Quantum Homomorphic Encryption

1.2.2 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). This implies that 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).

1.3 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.

1.3.1 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.

1.3.2 How Quantum Homomorphic Encryption works. There are four stages involved namely (Broadbent and Jeffery, 2015) :Key generation,Encryption,Evaluation,Decryption

All these stages are performed by the classical client.

1.3.3 Key generation and Encryption. Key generation is taking any input maybe a number or an alphabet and generating a pair of public and private key. When the client generates the key, it is used to encrypt information. Encrypted data can be referred to as a cipher text while unencrypted data is called plain text. A plain text is transformed into a cipher text using the public key.

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 also to guess.

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 serve 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).

1.3.4 Evaluation and Decryption. After key generation and encryption, we evaluate. This is checking the correctness of the algorithm. Use the public key and the cipher text to produce another cipher text. For example let us encrypt g in mod a prime number p. Then we encrypt it again in mod 2. Our cipher text reads c $= (g \bmod p) \bmod 2$. We first decrypt c in mod 2 to using the public key to check whether will get $g \bmod p$. 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. 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).

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.

Michael A Nielsen and IL Chuang. Quantum computation. *Quantum Information. Cambridge University Press, Cambridge*, 2000.

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.