Homomorphic Encryption: From Scratch To Quantum

C. Brunetta[1]

Chalmers University of Technology, Gothenburg, Sweden

1 Introduction

During the centuries, the human needs to have and maintain secrets developed in different ways giving birth to Cryptography: *the art of writing or solving codes*. This science wants to solve a communication problem between Alice and Bob: *Can Alice speak with Bob using an* ***encryption scheme****, without* ***anyone*** *eavesdrop their communication?*

For centuries, mathematicians developed different encryption scheme in order to protect data. They are composed by three different algorithm:

- A **key generation** algorithm G that return two different key: a private key sk and a public key pk
- An **encryption** algorithm E that takes the public key and a message m and outputs a ciphertext c
- A **decryption** algorithm D that takes the private key and a ciphertext c and outputs a message m

One requirement needed in order to use the scheme is **correctness**: for a given pair of keys sk, pk generated by G, the decryption with sk of the encryption with pk of a message m, is the message m. Formally $D(\mathrm{sk}, E(\mathrm{pk}, m)) = m$.

In this scenario, Alice can give Bob the public key and Bob will send different ciphertext $c_1, c_2, \cdots, c_n$ to Alice ans she will be able to decrypt them and compute all the function that she wants, like the sum of all the messages. What if Alice wants to compute the sum but she doesn't have the computational power or the memory space in order to save all the ciphertext? Can we build a new algorithm for evaluation Eval that takes as input a function f and some ciphertexts $c_1, \cdots, c_n$ and outputs the encryption of the function evaluated in the messages $E(\mathrm{pk}, f(m_1, \cdots, m_n))$?

This is the idea and requirement that we add to an encryption scheme in order to define a **homomorphic encryption scheme**: the ability to compute functions on encrypted data.

The use cases are quite different: consider Alice sending a lot of encrypted health-information to a cloud storage. One day, Alice wants to get some statistics in order to check if she's healthy. Alice out-source the computation to the cloud and she just receive the result that only she, as the private key owner, can decrypt and check.

2 Gentry's Idea: Bootstrapping [FDAT:Intro]

In 1978, Rivest, Adleman and Dertouzos [7] suggested that it might be possible to modify the RSA scheme in order to obtain a fully homomorphic encryption scheme, without succeeding. The word **fully** describe the possibility to compute **all** the possible functions. For 20 years, a concrete fully homomorphic scheme was missing in the cryptography scenario.

Until in 2009, Gentry's PhD thesis [4] explains how to build such an encryption scheme and only one year later, the first implementations started to appear [5]. Gentry's idea focuses on **lattices** which can be defined as the integer linear combinations of a vector basis of a vector space. In fact, it is linear algebra.

We randomly pick an odd integer p and then we encrypt a bit-message m as $c = m + 2r + qp$ where q is a random integer and r is a *small* noise. The decryption is made by computing different modulo $(c \pmod p) \pmod 2$. In this way, we retrieve the correct message **only if** the particular noise was *small*[1].

Since the scheme is based on linear algebra, we can easily compute sum and products. The only problem is the noise that will grow up every time we compute a single operation and if the noise is too big, the decryption algorithm will return the wrong message.

How can we avoid this problem? **Bootstrapping**.

Imagine that in our homomorphic scheme, we can evaluate the decryption function without having the noise problem: we fix the ciphertext c and we define the decryption function $D(\cdot, c)$ that takes as input a secret key and always decrypt c. For this reason, we homomorphically evaluate $\text{Eval}(D(\cdot, c), \cdot)$ and in order to maintain correctness, we have to input the *encryption of the secret key*. This is the main concept behind fully homomorphic encryption.
We decrypt a ciphertext with a lot of noise in a secure/encrypted way and we obtain a new ciphertext with *small* noise and so a ciphertext that we can re-use in our computation. Bootstrapping is a function that **re-fresh** the ciphertexts and allows us to compute arbitrary polynomial functions.

3 Security and Improvements [FDAT:Adv1]

Encryption scheme must to be secure, but defining security is not an easy task. There are different definitions of security and they are sometimes used in specific cases. Either the case, proving security has always the same construction: **assuming** the complexity-hardness of a particular problem, **fixing a security model** and so what an adversary can or cannot do, **reducing** the security of the encryption scheme to finding the solution of the hard problem.
In this way, security is based on the **computational model** and the **complexity/mathematical assumptions**.

[1] In fact, $|r| < \frac{p}{2}$

3.1 Assumptions

Almost all the fully homomorphic encryption scheme, that can be found in literature, are based on solving lattice problems that are well know hard problem in mathematics [1]. Usually, the problem used[2] are:

- **Learning With Error (LWE):** it is the *decision* problem of distinguishing between a *noisy* inner-product and uniformly random samples. The *searching* problem is defined as: choose $a \in \mathbb{Z}_q^n$ uniformly at random, r according to some distribution χ. Given polynomial pair $(a, \langle a, s \rangle / q + r)$, find $s \in \mathbb{Z}_q^n$.
- **Shortest Vector Problem (SVP):** it is the problem of finding the shortest (or closest) vector in a lattice with respect to a norm $|\cdot|$ (informally, a distance): given $\{b_1, \cdots, b_m\}$ vectors in $\mathbb{R}^n$, find the shortest non-zero vector v such that for all non-zero vectors x in the lattice, it holds $|v| \leq |x|$.

3.2 Reductions, Post-Quantum Crypto and Efficiency

A complexity reduction is an algorithm that transform a problem A into another B. It can be denoted with $A \leq_{Comp} B$ where $Comp$ will represent the computational model that we are using. A computational model is just the properties and the machine that it is used for computation. For example it can be a Turing Machine, a Circuit but even the more rare *"Paper, Pen and a Human"*.
The reduction gives us a order between problems and even more: if $A \leq_{Comp} B$ and we are able to solve B, it means that we are able to solve A. Let's thing about some daily problem using *ourself* as a computational model. Imagine $A =$ *"find the right key for a specific door"* and $B =$ *"opening a closed door"* are two particular problems. It is easy to reduct A to B just considering that *"wen you open a door, you have to search for a specific key"*. So we have that $A \leq_{Comp} B$. If we now imagine that we can easily open all the closed door, we are trivially able to find the key of a specific door.

The security of lattice based homomorphic encryption reduces to lattice's problems and they are considered hard in the classical computational model which is the probabilistic polynomial time Turing Machine, in fact our computers. On the other hand, these problems are considered hard even using a quantum-computational model that uses a quantum-computer.
This means that if one day we will be able to use a quantum-computer, the lattice based encryption schemes will still be secure while different encryption scheme, such as RSA or Diffie Hellman, will be broken since they reduct to different problem like *integer factorization* or *discrete logarithm in a finite field* that can be easily be solved using the Shor's algorithm, a quantum algorithm that is really fast in solving the discrete logarithm.

On the other hand, fully homomorphic encryption scheme are not completely implementable or efficient. The main reason is the trade-off that we have to consider when we define the *security* of a scheme. It is considered secure[3] if the

[2] Or with small variation.
[3] NIST recommendation give standard security parameter that we can use in daily implementation.

computational cost to break the scheme is more than 2^λ operation. This is defined as λ-bit of security.

In the original Gentry's paper [**?**], from λ-bit of security, a ciphertext will approximately have λ^5 bits. This means that in the standard case of $\lambda = 80$, we will have $\sim 3,03$ gigabit for a single ciphertext. Even the bootstrapping is a really slow algorithm due to the really big ciphertext. This is clearly unfeasible to use in a real-world scenario.
On the other hand, different papers tried to improve this gap by fining new construction and assumptions like Ring-LWE or Ideal-LWE that are build over a algebra's rings; or by *"moving"* the problem to another part of the encryption scheme like *"bigger public key, but small ciphertext"*. As an example, in Gentry et al.'s [6], they simplify the encryption scheme construction in order to achieve a faster bootstrapping. Or Chillotti et al.'s homomorphic encryption scheme [2] that is even faster!

4 Post-Quantum Quantum Encryption [FDAT:Adv2]

Quantum computer is a new technology that slowly get its position in the computer science world. A quantum computer is nothing more than a standard computer with a fundamental property: its register can be in a *superposition*. Superposition is a special spin-state of a particle. Generally, the spin of a particle can be encoded with *0* **xor** *1*. In quantum physics, a particle can have an additional spin which is *0* ***and*** *1*. A quantum computer register, in fact, can have this special superposition state that allows computation in a completely different way.

It is almost of common knowledge that quantum computer can break classical public key encryption since, in 1997, Shor [8] published a polynomial time quantum-algorithm that can factorize and solve the discrete logarithm in finite fields. This are the main hard-assumption at the base of RSA, Diffie-Hellman and Elliptic Curves Cryptography.

While waiting for a quantum computer to destroy almost all the public-key cryptography, we can prepare ourself with **quantum** cryptography: using a quantum computer in cryptography. The problem of developing quantum-algorithm is far more spread in different research area such as information theory, since quantum-information is not usual Shannon's theory, electronics and physics, since building quantum-circuits is not a well known technology like the transistor, and mathematics/computer science, since algorithms for this machine compute in a different way and have different properties with respect to a more *deterministic* computation.

Dulek et al.[3] quantum-homomorphic encryption scheme are just one example of the study of different research area in order to build an encryption scheme that can be used by a quantum computer.

Bibliography

[1] Arora, S., Babai, L., Stern, J., Sweedyk, Z.: The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations. Journal of Computer and System Sciences 54(2), 317–331 (Apr 1997), http://www.sciencedirect.com/science/article/pii/S0022000097914720

[2] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 Seconds. Tech. Rep. 870 (2016), https://eprint.iacr.org/2016/870

[3] Dulek, Y., Schaffner, C., Speelman, F.: Quantum homomorphic encryption for polynomial-sized circuits. arXiv:1603.09717 [quant-ph] 9816, 3–32 (2016), http://arxiv.org/abs/1603.09717, arXiv: 1603.09717

[4] Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009)

[5] Gentry, C., Halevi, S.: Implementing Gentry's Fully-Homomorphic Encryption Scheme. Tech. Rep. 520 (2010), https://eprint.iacr.org/2010/520

[6] Gentry, C., Sahai, A., Waters, B.: Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In: Advances in Cryptology – CRYPTO 2013, pp. 75–92. Springer, Berlin, Heidelberg (2013), https://link.springer.com/chapter/10.1007/978-3-642-40041-4_5, dOI: 10.1007/978-3-642-40041-4_5

[7] Rivest, R.L., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-key Cryptosystems. Commun. ACM 21(2), 120–126 (Feb 1978), http://doi.acm.org/10.1145/359340.359342

[8] Shor, P.W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing 26(5), 1484–1509 (Oct 1997), http://arxiv.org/abs/quant-ph/9508027, arXiv: quant-ph/9508027