Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de las Telecomunicaciones

Mecánica Cuántica y comunicación segura: El protocolo BB84 de Criptografía Cuántica

Autor: Alberto Romero Centeno

Tutor: Alberto Casado Rodríguez

Dpto. Física Aplicada III
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de las Telecomunicaciones

Mecánica Cuántica y comunicación segura: El protocolo BB84 de Criptografía Cuántica

Autor:

Alberto Romero Centeno

Tutor:

Alberto Casado Rodríguez

Profesor Titular

Dpto. Física Aplicada III
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Grado: Mecánica Cuántica y comunicación segura: El protocolo BB84 de Criptografía Cuántica

Autor: Alberto Romero Centeno
Tutor: Alberto Casado Rodríguez

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

A todos los maestros que he tenido desde bien pequeño, tanto la realización de este trabajo como mi desarrollo académico no hubiese sido posible sin ellos. Les agradezco su tiempo dedicado que bien espero que no haya sido en absoluto en balde, pues siento que no sería yo sin lo aprendido en este tiempo. A mis amigos, los cuales siempre en buena sintonía me han permitido y fomentado el poder alternar el estudio con el disfrute junto a ellos. Y por supuesto a mi familia, los que están, los que estuvieron y los que no tengo duda alguna, seguirán estando ahí para lo fundamental. Su cariño y tender siempre la mano cuando se hace necesario me convirtieron en la persona que soy. Ser así por ellos me mantiene muy orgulloso. Mención especial a mis padres, los cuales amo profundamente. No son sino ellos la razón de haber llegado hasta aquí y eso hace que les muestre un agradecimiento ∞.

A todos, muchas gracias

Resumen

Este trabajo incluye la relación entre la Mecánica Cuántica y la posibilidad de realizar comunicación segura, una revisión del protocolo BB84 y finalmente, una simulación en MATLAB® del BB84 en escenarios plausibles.

Abstract

This work includes the relationship between Quantum Mechanics and the possibility of performing secure communication, a review of the BB84 protocol and finally, a BB84 simulation in some possible scenarios using MATLAB® .

Contents

1 Quantum Principles

In this chapter we shall review the postulates of Quantum Mechanics, emphasizing the relationship of the quantum principles and theorems with the possibility of performing secure quantum cryptography. We will follow the *"Principles of Quantum Computation and Information"* [1] and *"A quantum mechanics primer"* [2].

1.1 First Postulate

A quantum system can be described by the state vector $|\psi\rangle$, which resides in the Hilbert Space H_s associated with this system. The time evolution of the state, without performing any measure, is governed by the Schrödinger equation:

$$i\hbar\frac{d|\psi(t)\rangle}{dt} = \hat{H}|\psi(t)\rangle, \tag{1.1}$$

where $\hbar \equiv h/2\pi$, h known as Plank's constant ($h \approx 6.626 \times 10^{-34}\,\mathrm{J\,s}$) and $\hat{H}$ is a self-adjoint operator known as the Hamiltonian of the system.

1.2 Second Postulate

We associate with any observable A a self-adjoint operator $\hat{A}$ on the Hilbert Space H_s. The only possible outcome of a measurement of the observable A is one of the eigenvalues of this operator. If we write the eigenvalue equation for the operator $\hat{A}$,

$$\hat{A}|a_i\rangle = a_i|a_i\rangle, \tag{1.2}$$

where $|a_i\rangle$ is an orthonormal basis of eigenvectors of the operator $\hat{A}$ and we write the state vector over the eigenvectors orthogonal basis:

$$|\psi(t)\rangle = \sum_i c_i(t)|a_i\rangle, \tag{1.3}$$

then the probability that a measurement of the observable A at time t results in outcome a_i is:

$$p_i(t) = p(a = a_i|t) = |\langle i|\psi(t)\rangle|^2 = |c_i(t)|^2. \tag{1.4}$$

From now on we shall use two-dimensional Hilbert spaces, which are necessary for the description of the quantum information theory. For notation simplicity, we shall ignore the

time dependence. The state vector can be written as:

$$|\psi\rangle = c_1 |a_1\rangle + c_2 |a_2\rangle , \tag{1.5}$$

where $|a_1\rangle$ and $|a_2\rangle$ are the normalized eigenvectors of the operator $\hat{A}$. So the probability of obtaining a_1 is $|c_1|^2$, while the probability of obtaining a_2 is $|c_2|^2$. As the addition of all probability values must be 1,

$$|c_1|^2 + |c_2|^2 = 1. \tag{1.6}$$

1.3 Third Postulate (Collapse Postulate)

If a system is described by the state vector $|\psi\rangle$ and the observable A is measured obtaining the eigenvalue a_1, then immediately after the measurement the state of the system will correspond to the eigenvector $|a_1\rangle$.

1.4 The Heisenberg uncertainty principle

From the previous postulates it can be deduced the following result: if $\hat{A}$ and $\hat{B}$ are operators associated with observables of the same quantum system and $|\psi\rangle$ is the actual quantum state, the following relation holds:

$$\Delta A \Delta B \geq \frac{\left| \langle \psi | [\hat{A},\hat{B}] | \psi \rangle \right|}{2}, \tag{1.7}$$

where $[\hat{A},\hat{B}] = \hat{A}\hat{B} - \hat{B}\hat{A}$ is the commutator between these two operators and ΔA (ΔB) the standard deviation corresponding to the observable A (B) in the state $|\psi\rangle$.

The uncertainty principle tells us that if these two operators do not commute we will not be able to predict with certainty the two observables at the same time. This is closely related to the compatibility theorem, that will be stated below.

1.5 Compatibility Theorem

Having the two operators $\hat{A}$ and $\hat{B}$ associated with the observables A and B, the compatibility theorem tells us that *any of the following statemets implies the other two*:

- *$\hat{A}$ and $\hat{B}$ are compatible ($[\hat{A},\hat{B}] = 0$).*
- *$\hat{A}$ and $\hat{B}$ have a common eigenbasis.*
- *If we measure the observable A, immediately after the observable B and immediately next the observable A again, the result of the third measure will necessary coincide with the first.*

Also the following statements are equivalent:

- *$\hat{A}$ and $\hat{B}$ are incompatible ($[\hat{A},\hat{B}] \neq 0$).*
- *$\hat{A}$ and $\hat{B}$ do not have a common eigenbasis.*

- *If we measure the observable A, immediately after the observable B and immediately next the observable A again, the result of the third measure will not necessary coincide with the first.*

1.6 Entanglement

The Hilbert space H_s associated with a composite system is the tensor product of the Hilbert spaces H_i associated with the system components i. In a bipartite system, we have:

$$H = H_1 \otimes H_2 \tag{1.8}$$

A state in H is said to be entangled if it cannot be written as a simple tensor product of a state $|\alpha_1\rangle$, belonging to H_1 and a state $|\beta_1\rangle$, belonging to H_2. If a state is separable (not entangled) then

$$|\psi\rangle = |\alpha_1\rangle \otimes |\beta_1\rangle . \tag{1.9}$$

When two systems are entangled, it is not possible to assign state vector to each system. Here we have an example of two spin-$\frac{1}{2}$ particles with the angular momentums spin up ($\uparrow$) and spin down ($\downarrow$):

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|\uparrow\uparrow\rangle + |\downarrow\downarrow\rangle) \tag{1.10}$$

When we say that two particles are entangled we mean that neither of systems have a definite value.

1.7 The EPR paradox and Bell's inequalities

This section is not necessary to understand the BB84 protocol of quantum cryptography. Nevertheless, the EPR paradox and Bell's inequalitieshave a great importance, not only in the fundamental problems of quantum mechanics but also for the application of Bell's inequalities to quantum cryptography using entanglement [3].

As we know from the entanglement concept, when systems are entangled, we cannot assign them individual state vectors. This concept contradicts the reality and locality principles [4].

Reality principle: If with no perturbation on the system we can predict with certainty the value of a physical magnitude, then there is a physical reality associated to this magnitude.

Locality principle: If two systems are causally disconnected, any measurement performed on one of the two systems cannot influence on the other.

Let us consider a bipartite system in the singlet state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle). \tag{1.11}$$

The two particles have opposite spin, this means that if one particle is "up" (defined by the state vector $|\uparrow\rangle$ with the corresponding eigenvalue +1), the other is "down" (defined by the state vector $|\downarrow\rangle$ with the corresponding eigenvalue -1) and vice versa. In other words, if we

measure one particle and obtain $\sigma_z = +1$ if we immediately measure the other particle we will get with unit probability $\sigma_z = -1$. Let us consider that a source S emits two entangled particles each travelling towards an observer, so that the two observers A and B are widely separated like in the figure 1.1.

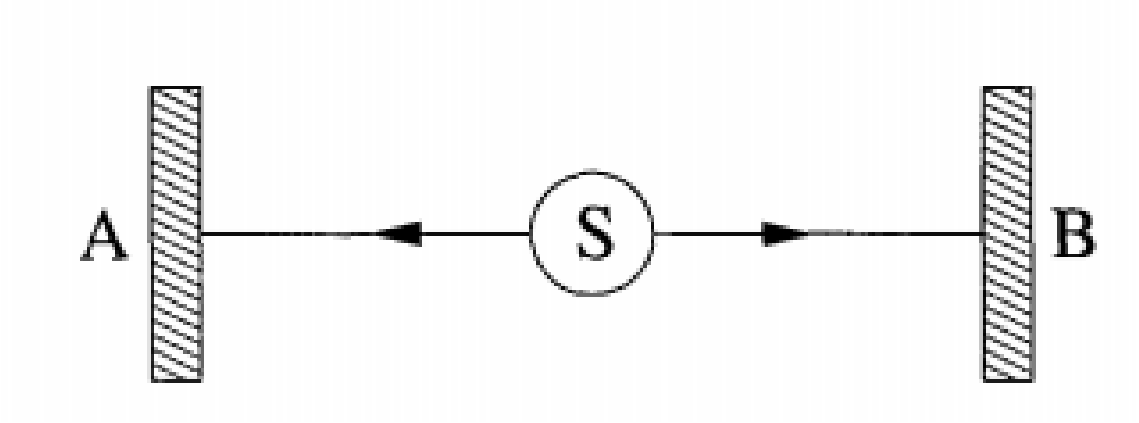

Figure 1.1 Schematic drawing of the EPR experiment from Principles of Quantum Computation and Information.

Knowing that the two observers are far away that we can say that they are causally disconnected then quantum theory leads to a contradiction with locality and reality principles.

Bell's inequalities are obtained assuming the principles of realism and locality [5]. Since it is possible to devise situations in which quantum mechanics predicts a violation of these inequalities, any experimental of such a violation excludes the possibility of a local and realistic description of natural phenomena. In few words, Bell shows that the principles of realism and locality lead to experimentally testable inequality relations in disagreement with the predictions of quantum mechanics. Genuine Bell's inequalities (based on realism and locality principles) have not been violated in experiments yet, but a derivation from these inequalities based on local realism and additional hypothesis, like the CHSH (Clauser, Horne, Shimony and Holt) inequalities [6] which have been violated in laboratory. More recently some other experiments have come closer to the requirements of the ideal EPR scheme and agreement has always been found. Nonetheless, there is no general consensus as to whether or not these experiments may be considered conclusive.

2 Quantum information

In this chapter we shall present the basic concepts of the quantum information theory [3].

2.1 Qubit: the quantum bit

In classical information the fundamental entity is the bit, a bit can carry two possible values. The bit comes from a physical system where the value of the bit depends on being lower than a margin (the bit carries a “0”) or higher than a margin (the bit carries a “1”). In quantum information we have the qubit (quantum bit), a two-state ($|0\rangle$ and $|1\rangle$) quantum system. The main difference is that the qubit does not carry a concret value but the superposition of the two states. So a general way of expressing the qubit will be:

$$|Q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,. \tag{2.1}$$

The computational basis $\{|0\rangle, |1\rangle\}$ is the two basis states composed by (any of) the two distinct quantum states that the qubit can phisically be in. By taking into account the second postulate in the equations (1.4) and (1.6) with $|\alpha|^2 + |\beta|^2 = 1$, these complex numbers will also determine the probabilities of obtaining a "0" or a "1".

We can also write the superposition of two qubits as:

$$|\psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma|10\rangle + \delta\,|11\rangle \tag{2.2}$$

2.2 Qubit sources

There are many different sources which can generate qubits, here we show two of the most common and the one that we will use in the following chapters of this work (polarized photons) [7].

2.2.1 Polarized photons

Photon qubits can be realized using polarization. The two arbitrary states could be the horizontal $|H\rangle$ and vertical $|V\rangle$ polarization. The main advantage is the fact that they can be easily created and manipulated with high precision by optical elements. The inconvenience

is that they are hard affected by environmental noise and need very sensitive devices that may fail to detect them. A generic state of a polarized photon qubit can be:

$$|\psi\rangle = \alpha |H\rangle + \beta |V\rangle . \tag{2.3}$$

2.2.2 Spin-$\frac{1}{2}$ particle

Another quantum system is a spin-$\frac{1}{2}$ particle such as an electron. So the spin angular momentums, spin up ($\uparrow$) or spin down ($\downarrow$) are the two quantum states. The main advantage is the fact that they are very easy to create and manipulate. The inconvenience is that they are very sensitive to environmental fluctuations of the electric potential so they can be most probably affected if the path is too long. A generic state of a spin-$\frac{1}{2}$ particle qubit can be:

$$|\psi\rangle = \alpha |\uparrow\rangle + \beta |\downarrow\rangle . \tag{2.4}$$

2.3 Qubit transformations

As well as the bits can be transformed using logic gates such as the OR, AND or negation gates, qubits can be manipulated using unitary transformations in the Hilbert Space. We will see single-qubit gates and two-qubit gates also known as entanglement generators.

2.3.1 Single-qubit gates. The Hadamard and Phase-shift gates

When a qubit goes through single-qubit gates it can swap the states or introduce a phase change. The fundamental gate is the Hadamard transformation H, this gate turns the $|0\rangle$, $|1\rangle$ basis into a new one, made from the superposition of this. Hadamard transformation is defined as

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{2.5}$$

Given that $H^2 = I$, if a qubit goes through two Hadamard gates, it will remain unchanged. Another common gate is the shifting gate ϕ:

$$\begin{aligned} \phi |0\rangle &= e^{i\delta} |0\rangle \\ \phi |1\rangle &= |1\rangle . \end{aligned} \tag{2.6}$$

Any unitary operation on a single qubit can be constructed using only Hadamard and phase-shifting gates.

2.3.2 Two-qubit gates. The Controlled NOT gate

The controlled NOT gate generates entanglement between two qubits. The first qubit acts as a control and the second will be the target, so if the state of the first qubit is $|1\rangle$ the

second qubit state will be flipped and nothing will happen if the first qubit state is $|0\rangle$:

$$|0\rangle_c |0\rangle_t \rightarrow |0\rangle_c |0\rangle_t ,$$

$$|1\rangle_c |0\rangle_t \rightarrow |1\rangle_c |1\rangle_t ,$$

$$|0\rangle_c |1\rangle_t \rightarrow |0\rangle_c |1\rangle_t ,$$

$$|1\rangle_c |1\rangle_t \rightarrow |1\rangle_c |0\rangle_t . \tag{2.7}$$

2.4 The no-cloning theorem

We know that classical bits can be easily copied. We only have to be able to measure its value and save in a memory. But when we are working with qubits we have a quantum system, in which any measure is not a simple classical measure and makes the state vector collapse on the eigenvector related to the resulting eigenvalue. This property makes us think that copying a qubit will not be something determined nor even possible in most cases.

The no-cloning theorem tells us that *it is impossible to build a machine that operates unitary transformations and is able to clone the generic state of a qubit* [8].

Let us consider the generic quantum bit we want to copy:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle . \tag{2.8}$$

The machine, in the initial state $|A_i\rangle$, should be able to perform a unitary transformation U on the second qubit $|\phi\rangle$ (this qubit is used to save the copied qubit) such us:

$$U(|\psi\rangle |\phi\rangle |A_i\rangle) = |\psi\rangle |\psi\rangle \left|A_{f\psi}\right\rangle = (\alpha |0\rangle + \beta |1\rangle)(\alpha |0\rangle + \beta |1\rangle) \left|A_{f\psi}\right\rangle , \tag{2.9}$$

where the final state of the machine depends on the state $|\psi\rangle$. The action of the machine on $|0\rangle$ and $|1\rangle$ must be:

$$U(|0\rangle |\phi\rangle |A_i\rangle) = |0\rangle |0\rangle \left|A_{f0}\right\rangle$$

$$U(|1\rangle |\phi\rangle |A_i\rangle) = |1\rangle |1\rangle \left|A_{f1}\right\rangle . \tag{2.10}$$

Because of the fact that unitary transformations are linear, then

$$U(|\psi\rangle |\phi\rangle |A_i\rangle) = \alpha |0\rangle |0\rangle \left|A_{f0}\right\rangle + \beta |1\rangle |1\rangle \left|A_{f1}\right\rangle , \tag{2.11}$$

which is clearly different from the desired (2.9). So we can determine that only if we know from the beginning that the state of the qubit is prepared in one out of two states we will be able to prepare copies as desired. Then we can state that if a machine could copy quantum states, based on the third postulate, this machine would destroy any information from the non orthogonal states. So, this machine could perfectly work for two orthogonal states such as the rectilinear pair of polarization states of a photon, but it would not be able to copy any other superposition states. In sum, without having a prior knowledge of the quantum state it is impossible to select a machine that copied that state.

3 Cryptography

Cryptography is the science or study of the techniques of secret writing, especially code and cipher system, methods, and the like [9]. In this chapter we give a brief review of the most important principles and elements involved in cryptography.

3.1 Principles of Cryptography

The objective of cryptography is to keep the message unknown from everybody else but the receiver. This is achieved by the cipher, a rule that tells you how to encrypt so later it can be also decrypted with some extra information. Some simple examples are substitution or transposition [10].

Example 3.1.1 Encryption of the words “BE HUMBLE” using substitution.

Solution. We are going to substitute every letter with its following in the alphabet:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
BCDEFGHIJKLMNOPQRSTUVWXYZA
"BE HUMBLE" $\rightarrow$ "CF IVNCMF"

In a more advanced encryption and decryption, keys are used to do both, they make the system stronger against an attack and are easier to distribute to the channel users. The algorithms used to cipher make the encrypted message almost impossible to decrypt. Only very powerful computers like quantum computers could potentially break the encryption.

3.2 Cryptography elements

3.2.1 Message

The message is the information we want to keep unknown from everybody else but the receptor. For example: a word or a number.

3.2.2 Transmitter (Alice)

The transmitter is the one who has the message and will be able to encode it before sending it to the receptor.

3.2.3 Receptor (Bob)

The receptor has to be able to decode the encrypted message and get the original one.

3.2.4 Keys

The keys are the data that allows the transmitter and receptor being the only ones that can encrypt and decrypt the message. There are symmetric and asymmetric keys. Symmetric keys can be use both to decrypt and encrypt while we need one asymmetric key to encrypt and other to decrypt. Symmetric keys are used in secret-key cryptography, where keeping the privacy of the key is essential. On the other hand, asymmetric keys are the ones needed in public-key cryptography where everybody knows the key to encrypt but only who has the key to decrypt can do so.

3.3 Key Distribution

In order to allow the receptor or receptors to decrypt the message it is important that they are the only ones that have the key. The classical problem is that the channel used to share this key may not be 100% safe. Then a potential eavesdropper could obtain the key to break the encryption. Quantum cryptography brings the certain that our key will not be obtained by the eavesdropper when he attacks the quantum channel.

3.4 Vulnerability

In the actual public key cryptography breaking the encryption is very hard and almost impossible for the most part of the computers, so we could say public key cryptography is not vulnerable against key-break attacks. However, the appearance of quantum computers, computers with an enormous calculus potential based on qubits, would make the key cryptography vulnerable against these attacks. This new advance would be a big problem. Nevertheless the quantum cryptography brings the tools to avoid this potential problem.

4 BB84 Protocol

The BB84 protocol was developed by Charles H. Bennett and Gilles Brassard [11]. They are two scientists who work at quantum information field, particularly they brought the concept of quantum cryptography. Charles H. Bennett is a physicist from the United States and he also developed the Bennett's four laws of quantum information. Gilles Brassard studied Computer Science, but he is known for his work related to quantum cryptography, teleportation, entanglement and pseudo-telepathy. The protocol's main purpose is to distribute a key between two users with the certain that it remains unknown for everybody else. It uses a quantum channel to make the secret communication possible, this channel is based on polarized photons, which are sent and received by the specific apparatus. The protocol also lies on the no-cloning theorem and the third postulate which are essential to discover an eavesdropper action.

4.1 System description

The protocol uses polarized photons to encode the information. Both, the sender (Alice) and receiver (Bob) will use two different polarization bases (rectilinear and diagonal), and two possible values of the information (bits: "0" or "1"). For each photon the polarization base is randomly chosen either for encoding or measuring, with no correlation between Alice and Bob's chosen bases. Alice sends a "0" or a "1" by polarizing the photon in the randomly chosen base. The values of the polarization angles are the following:

Table 4.1 Polarization angles.

Polarization	Bit	Angle
Rectilinear	0	0º
Rectilinear	1	90º
Diagonal	0	45º
Diagonal	1	135º

Alice will save both the information bit and the polarization used to send. At the same time Bob will measure the incoming polarized photons with his chosen base. So, if the base is the same as Alice's the bits will coincide, if not, in half of the cases it will coincide and in the other half it will not. Bob will save the bits received and the polarization used to measure.

4.2 Eavesdropper effect

Let us consider the following situation: an Eavesdropper (Eve) is attacking the communication, in such a wat that she intercepts the photons sent by Alice, and measures them by randomly using one of the two basis. Then, she resends the photons to Bob, in the same base that she has measured them. If Eve measures in a base that is different to Alice's, the polarization state of the photon will change. In this case, the bit value measured by Bob will not coincide, in general, with the one sent by Alice. Even if both used the same basis (compatibility theorem).

4.3 Result comparison and final key

Once all the bits have been transmitted, Alice and Bob publish in a public channel the random bases they chose. They discard the bits in which the bases are different. So, they will approximately keep half of the bit string. Then, Alice and Bob will also publish a part of the remaining bits in order to estimate the error rate. These bits may have been affected by Eve's attack or a system error, and they will be discarded too, due to the fact that they are now public. If the error rate is too high (higher than the threshold, commonly 5%) the communication will be stopped, if the error rate is acceptable, then, Alice and Bob will do some error corrections.

4.3.1 Information reconciliation

In order to make sure the key they share is strictly the same, it is necessary to remove the wrong bits from the key. Alice and Bob divide their strings into subsets of length l with the following condition $Rl << 1$, R being the error rate that they obtained at the previous step. Then Alice and Bob publish the parity of these subsets. In case the parities are different they will divide the subset into two, repeating this process until the parities are the same or discarding the string. Every time they publish the parity they will discard the last bit of the block, this is done to make sure the eavesdropper has no information from the strings.

Example 4.3.1 We have the following keys:
key A : [0 1 1 0 1 0 1 0 1 0 1 1 1 0 0 1 0 1] key B : [0 1 *0* 0 1 0 1 0 1 0 1 1 1 0 0 *0* 0 1]

Solution. If l is 4 (we need to have a correct l to make sure we will not have more than 1 error in each block), we will divide the keys into 4 blocks of 4 bits and 1 of 2 bits. Then we will share the parities of each block and remove the last bit. In case the parities are different we will repeat the process dividing the block into two.
key A_1 : [0 1 1 Ø] (parity = 0) key B_1 : [0 1 *0* Ø] (parity = 1)
key A_2 : [1 0 1 Ø] (parity = 0) key B_2 : [1 0 1 Ø] (parity = 0)
key A_3 : [1 0 1 ✗] (parity = 1) key B_3 : [1 0 1 ✗] (parity = 1)
key A_4 : [1 0 0 ✗] (parity = 0) key B_4 : [1 0 0 Ø] (parity = 1)
key A_5 : [0 ✗] (parity = 1) key B_4 : [0 ✗] (parity = 1)
So, we will continue the process for the blocks 1 and 4, where parities are different.
key $A_{1.1}$: [0 ✗] (parity=1) key $B_{1.1}$: [0 ✗] (parity=1)
key $A_{1.2}$: [✗] (parity=1) key $B_{1.1}$: [Ø] (parity=0)
key $A_{4.1}$: [1 Ø] (parity=1) key $B_{4.1}$: [1 Ø] (parity=1)
key $A_{4.2}$: [Ø] (parity=0) key $B_{4.2}$: [Ø] (parity=0)

Now all the parities are the same, so we rejoin the blocks with the remain bits.

final key A: [0 1 0 1 1 0 1 1 0]

final key B: [0 1 0 1 1 0 1 1 0]

We can see that the remaining key is half of the one that contained errors, if l is correctly chosen, the final key will be aproximately half of the original one.

4.3.2 Privacy amplification

After removing all the errors with the information reconciliation, to make sure the eavesdropper has no information from the key, the raw key is divided into m blocks. Then the parity of each block is calculated. The final key will be the consecutive parities of all the blocks. m is chosen according to the following condition:

$$m = n - k - s \tag{4.1}$$

where n is the number of bits from the key, k depends on the bits the eavesdropper might have and it is calculated from the error rate and s, a security parameter. While k or s are bigger, the shorter and safer the key will be, also the eavesdropper will have less or no information at all.

4.4 Protocol simulations

In order to see how the protocol works, its limits and power we will see some simulations already made by some investigators. This will help us on our own simulations and to compare with our results. These are some of the different simulations:

- *https://www.qkdsimulator.com/* is a web application that let us simulate with different parameters where we can see the resulting number of bits Bob will have or a possible eavesdropper detection.
- *Quantum Key Distribution: Simulation and Characterizations (ICCMIT 2015)* [12]: The authors detail the simulation steps, and then they simulate the whole protocol to see its behaviour with a communication between a master and a slave. They come to the conclusion of the viability with satisfactory results.

4.5 Real implementation

This protocol can be implemented by some specific devices. Here we can see a setup from Matthias Scholz [13]. Where the main parts are a photon source, polarization encoder and reader and the photodetector.

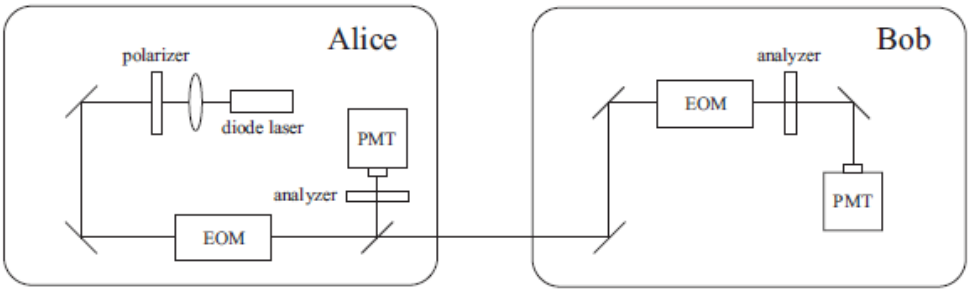

Figure 4.1 System schema.

4.5.1 Photon source

The photon source is a 5mW diode laser emitting at 633 nm [14]. To produce single photons is required a big attenuation.

Figure 4.2 633nm Laser Diode.

4.5.2 Polarization encoder and reader (EOM)

In order to polarize the photons and read their polarization the experiment uses two electro-optic modulators, one for each task. The one used to encode will polarize into four different states (0º, 45º, 90º, 135º) depending which bases (rectilinear or diagonal) and the bit value (‘0’ or ‘1’). While the other will project the sent photon onto the basis state chosen by Bob, so only the bits where the bases coincide will certainly keep their values [15].

Figure 4.3 Electro-Optic Modulator.

4.5.3 Photodetector

A photodetector connected to a computer to convert the information detected into bits and save them. This photodetector must be able to read every single photon [16]. In case the polarization chosen from Alice and Bob is not the same, the computer that saves both, the bit and the polarization chosen by Bob will discard it. The greatest inconvenient of Quantum Cryptography is the deficient work of these devices that introduce a high error probability to the system.

Figure 4.4 Photodetector.

4.6 Practical errors

Although as we have already seen the protocol is theorically 100% secure. When it is implemented with real devices we come across some defaults due to the precision of the elements, the environmental noise or the non correct working of the devices. Some of these can be associated to an error that will be introduced to the system that will only change some bits and can be fixed at information reconciliation. However, some others will be discussed because they can help the eavesdropper to get the key without being discovered [17].

4.6.1 Multi-photons

We know that the photon source used must only send one photon each output. However there is a probability of emitting multi-photons. In case this happens, the eavesdropper can make an attack called PNS (Photon Number Splitting), where he will not disturb the conversation between Alice and Bob, and still get the information of the key. This attack consists in saving one photon at a quatum memory and let the other go to Bob. Once Alice and Bob share their bases, the eavesdropper will measure in their correct base, rectilinear or diagonal, those bits stored.

5 Protocol simulation

The aim of the simulation is to see the global behaviour when we are using the BB84 protocol for key distribution. The simulation will have some variables which include: protocol parameters, eavesdropper actions and possible errors. We will work following the quantum concepts we have already seen such as no cloning theorem or compatibility theorem. As well, we will simulate every step included in the protocol from the first bits and polarization chosen until the final bit string that Alice and Bob will keep. We will use MATLAB® due to the fact that this tool helps us to get some graphics that give us the possibility to well understand the whole information we can get and compare the results among the different scenarios.

5.1 Simulation Parameters

Code 5.1 Function parameters.

```
function y = BB84simulation(simulations, N, error, comp, startEve,
   endEve, threshold, s)
```

5.1.1 Number of simulations

The parameter *simulations* determines the times the experiment will be repeated keeping the same parameters. This is useful to have as results the means of every simulation results, which are much more accurate (particular events have less effect).

5.1.2 Number of bits

N is the number of bits Alice will send. The higher this number is the longer the final key will be. Like the number of simulations, this parameter will make the simulation last longer while we introduce higher numbers.

5.1.3 System error rate

error(%) will represent all the noises or devices failures that make Bob get wrong bits even if there is no polarization conflicts. We have simplified this in just one parameter. So, we will suppose an error rate that comes from all the system defects. We will see how this

error rate works and the maximum the communication will allow before stopping the key distribution.

5.1.4 Bits comparison

comp is the number of bits Alice and Bob will announce, and immediately discard, in the public channel in order to get the bit error rate due to eavesdropper, noises or devices failures.

5.1.5 Eavesdropper actuation

The way the eavesdropper (Eve) will attack the communication will be determined by the start *startEve* and the end *endEve* of the attack. These parameters are referred to the positions of the bits at the Alice's string where the attack takes place (beginning and end).

5.1.6 Threshold error

threshold(%) is the maximum error rate Alice and Bob will allow. If the error rate obtained in the bits comparison is higher than this threshold the communication will be immediately stopped and they will discard the key.

5.1.7 Privacy amplification parameter

The parameter s is the one related to the privacy amplification, as the following equation $subsets = n - k - s$. So we will be able to select the value in order to have a more safety but shorter key.

5.2 Simulation code

We can see the complete code used to simulate in the appendix A.

5.2.1 Random bits and polarization generation

In order to generate the polarization that Alice, Bob and Eve will randomly choose and the bit string that Alice will send, we will use a function that let us have the same probability of obtaining one or another bit or polarization value (randi). We will encode the polarization into an array whose elements could take two possible values "0" (rectilinear) and "1" (diagonal). The bits transmitted (Alice's qubits) will be randomly generated with the same possible values, but also the ones received will be first initiated with the same length but they will be empty.

Code 5.2 Bits and polarization generation.

```
polAli = randi(2,1,N)-1;\%Alice polarization 0 = rectilinear 1 =
    diagonal
polBob = randi(2,1,N)-1;\% Bob polarization 0 = rectilinear 1 =
    diagonal
polEve = randi(2,1,N)-1;\%Eve polarization 0 = rectilinear 1 =
    diagonal
```

```
qbitAli = randi(2,1,N)-1;\%Alice bits
qbitBob = zeros(1,N);\%Bob bits (empty)
qbitEve = zeros(1,N);\%Eve bits (empty)
```

5.2.2 System error

We will use another array, initially an exact copy of Alice's qubits, with the particularity that some bits will change with the system error probability we have selected. From now on, part of this array (based on the polarization coincidence) is the information that Bob and Eve will have from Alice, the qubits affected by the errors.

Code 5.3 System error.

```
qbitAli1 = qbitAli; \%qbits which affected by system error will
    arrive to Eve and Bob
errorSystem = error*10^5;
prob = randi(10000000,1,N);

for R = 1:N
    if prob(R) < errorSystem
        if qbitAli1(R) == 0
            qbitAli1(R) = 1;
        else
            qbitAli1(R) = 0;
        end
    end
end
```

5.2.3 Bits received by Eve and Bob

We will have two different scenarios, one in which we will be inside the range where Eve is attacking and other where there is no attack. We will differentiate each one with the parameters *startEve* and *endEve* in a "for loop". If Eve is acting, she will measure with the polarization she had chosen and she will save the consequent bits, then Bob will measure but according to Eve polarization. He will have with 100% probability the same bit that Eve has if Eve and Bob's polarization is the same, if not, the bits will be chosen randomly. In the case Eve is not attacking, Bob will compare his polarization with Alice's, where if the polarization coincides bits will be the same and randomly generated when polarizations are different.

Code 5.4 Bob and Eve qubits measure.

```
\%bits obtained by Eve
 for R = 1:N
    if R > startEve && R < endEve
        if polAli(R) == polEve(R)
            qbitEve(R) = qbitAli1(R);
```

```
            else
                qbitEve(R) = randi(2)-1;
            end
        end
    end

    \%bits obtained by Bob
    for R = 1:N
        if R > startEve && R < endEve
            if polBob(R) == polEve(R)
                qbitBob(R) = qbitEve(R);
            else
                qbitBob(R) = randi(2)-1;
            end
        else
            if polAli(R) == polBob(R)
                qbitBob(R) = qbitAli1(R);
            else
                qbitBob(R) = randi(2)-1;
            end
        end
    end
```

5.2.4 Polarization publication

We remove from the arrays the bits whose polarization differs. We do this by loading in a new array the bits whose polarization is the same.

Code 5.5 Polarization comparison.

```
    i=0;
    for R= 1:N
        if polAli(R) == polBob(R)
            i= i+1;
            rawAlice(i) = qbitAli(R);
            rawBob(i) = qbitBob(R);
        end
    end
```

5.2.5 Bits comparison and error rate

We select a number of bits determined by the parameter *comp* from the remaining key. We count the errors and get the error rate that will be used in the future. The bits will have the same gap in between, from the first to the last bit of the key in order to have a closer error rate to the entire key error rate. If this error rate is higher than the threshold the communication will be immediately stopped.

Code 5.6 Bits comparison.

```
    longrawAlice = length(rawAlice);
    longrawBob = length(rawBob);
    step = floor(longrawBob/comp);
    if step > 1
        i = 1;
        limit = longrawAlice;
        while i <= limit
            if rawAlice(i) ~= rawBob(i)
                biterrors = biterrors +1;
            end
            rawAlice(i) = [];
            rawBob(i) = [];
            i = i + step - 1;
            limit = limit-1;
        end
    else
        j=0;
        while j < comp
            if rawAlice(1) ~= rawBob(1)
                biterrors = biterrors +1;
            end
            rawAlice(1) = [];
            rawBob(1) = [];
            j= j+1;
        end
    end
```

5.2.6 Information reconciliation

After removing a part of the key, the remaining key may have some errors, so we need to remove them. First we will calculate the parameter l, the length of the subsets in which we will divide the key, with the condition $Rl << 1$. Then we send the bit strings with length l to a function that will do the information reconciliation algorithm as follow:

Code 5.7 Algorithm to remove errors.

```
function y = depuraError2(code1, code2)
L = length(code1);

if L > 1

        L = length(code1);
   \%we compare parities calculaParidad returns the parity of the
      code
        if calculaParidad(code1) == calculaParidad(code2)
            \%if are the same we remove last bit and return the
               codes
        code1(L) = [];
```

```
        code2(L) = [];

            y = code1;
        else if L > 2
    \%if not we divide the code into 2 halves if it is possible

        code1(L) = [];
        code2(L) = [];
        L = length(code1);
        \%we calculate the positions to divide the code
    if mod(L,2) == 0
        half = floor(L/2);
    else
        half = floor(L/2) +1;
    end
    \%we repeat the process for the halves
    y1 = depuraError2(code1(1:half),code2(1:half));
    y2 = depuraError2(code1((half+1):L),code2((half+1):L));
    y = horzcat(y1,y2);
            else
               \%in case the code is 2 or lower, dividing is useless
                   so we
               \%return empty value
                y =[];
            end
        end
else
    \%if L= we return empty value
            y = [];
end
```

Once all the subsets errors have been removed we concatenate them to get the entire key.

Code 5.8 Information reconciliation.

```
    \%if the error rate is lower than the threshold we start the
       information
    \%reconciliation
    if errorRate < threshold && errorRate > 0
        L = floor(0.05/errorRate);
        \%number of subsets
        if rem(longrawAlice,L) == 0
            M = floor(longrawAlice/L);
        else
            M = floor(longrawAlice/L)+1;
        end
        keyFinalBob = [];
        keyFinalAlice = [];
        for R = 1:M-1
```

```
        keyFinalBob = horzcat(keyFinalBob,depuraError2(rawAlice(
            L*(R-1)+1:L*R),rawBob(L*(R-1)+1:L*R)));

    end

    keyFinalBob = horzcat(keyFinalBob, depuraError2(rawAlice(L*
        M+1:longrawAlice),rawBob(L*M+1:longrawBob)));

    for R = 1:M-1

        keyFinalAlice = horzcat(keyFinalAlice,depuraError2(
            rawBob(L*(R-1)+1:L*R),rawAlice(L*(R-1)+1:L*R)));

    end

    keyFinalAlice = horzcat(keyFinalAlice, depuraError2(rawBob(L
        *M+1:longrawAlice),rawAlice(L*M+1:longrawBob)));
```

5.2.7 Privacy amplification

We will divide the remaining key into a number of subsets whose parities will form the final key. So we first calculate the number of these subsets according to the following equation $subsets = n - k - s$. n is the number of bits of the remaining key, k will be calculated depending on the previous error rate (k will represent the possible bits Eve could have) and$s$ is the parameter we chose at the beginning of the simulation. Finally, we concatenate all the parities.

Code 5.9 Privacy amplification.

```
keyLength = length(keyFinalAlice);
    k = floor(2*errorRate*keyLength);
    subsets = keyLength - k - s;
    L = floor(keyLength/subsets);
    definitiveKeyBob = [];
    definitiveKeyAlice = [];
    for R = 1:subsets-1
        definitiveKeyBob = horzcat(definitiveKeyBob,
            calculaParidad(keyFinalBob(L*(R-1)+1:L*R)));
    end
    definitiveKeyBob = horzcat(definitiveKeyBob,calculaParidad(
        keyFinalBob(L*R+1:keyLength)));

    for R = 1:subsets-1
        definitiveKeyAlice = horzcat(definitiveKeyAlice,
            calculaParidad(keyFinalAlice(L*(R-1)+1:L*R)));
    end
    definitiveKeyAlice = horzcat(definitiveKeyAlice,
        calculaParidad(keyFinalAlice(L*R+1:keyLength)));
```

5.2.8 Final results

In every single simulation we will save the initial error rate, the final key length and the possible final key errors. Finally we will calculate the mean of these results to have a global view of all the simulations.

6 Simulation results

Once we have explained how the simulation works, we are going to simulate some possible scenarios to see the results. Also, we will see how the modification of every parameter affects the simulation. Finally, we will repeat the simulations with different Alice's bits string lengths and numbers of the bits we use to compare in order to get an optimum value of this parameter.

6.1 Scenarios

6.1.1 Scenario 1

There is no error system and the eavesdropper is not attacking the communication.

Table 6.1 Parameters value.

Parameter	Variable	Value
Simulations	simulations	50
Alice's bits	N	5000
System error	error	0
Comparison bits	comp	1500
Eve's attack start	startEve	0
Eve's attack end	endEve	0
Threshold	threshold	5
Security parameter	s	0

After simulating, we can see that we can decrement the comparison bits without any problem due to the non-existence of error (no eavesdropper and no system error).

We start introducing some system error to the simulation to see what happens. With a 0.1% system error we will need at least 2400 bits to compare, this result show the difficulty of finding the wrong bit. If we raise the error to a 3% that could represent a maximum, knowing that the threshold is 5%, we see that the mean of the resulting key length after the information is approximately only 4. With a 4% or more we get an empty key in the most cases. The causes are: the communication stop in most cases and the high error affecting which will make the information reconciliation discard so many bits.

6.1.2 Scenario 2

Now we are introducing a significant Eve's attack.

Table 6.2 Parameters value.

Parameter	Variable	Value
Simulations	simulations	50
Alice's bits	N	5000
System error	error	0
Comparison bits	comp	1500
Eve's attack start	startEve	0
Eve's attack end	endEve	2000
Threshold	threshold	5
Security parameter	s	0

After simulating, we check that the system detects the presence of Eve. The error rate obtained is about a 16%, much higher than the threshold, so the communication is immediately stopped.

6.1.3 Scenario 3

Now we are introducing an Eve's attack that the system could accept.

Table 6.3 Parameters value.

Parameter	Variable	Value
Simulations	simulations	50
Alice's bits	N	5000
System error	error	0
Comparison bits	comp	1500
Eve's attack start	startEve	0
Eve's attack end	endEve	200
Threshold	threshold	5
Security parameter	s	0

The error rate is lower that the threshold so the communication continues. The resulting key is shorter than the one in the Scenario 1. If we now introduce some system error, this forces us to increment the comparison bits to avoid final key errors. So, we get a shorter key again.

6.2 Parameters optimization

We will repeat the simulations done with the Scenario 2 where we have an acceptable error rate introduced by the eavesdropper. In this case, we will modify the bits used to compare. We will see the optimum value of this parameter which gives us the longest key without including any error. The code used to draw these graphics is at the Appendix B.

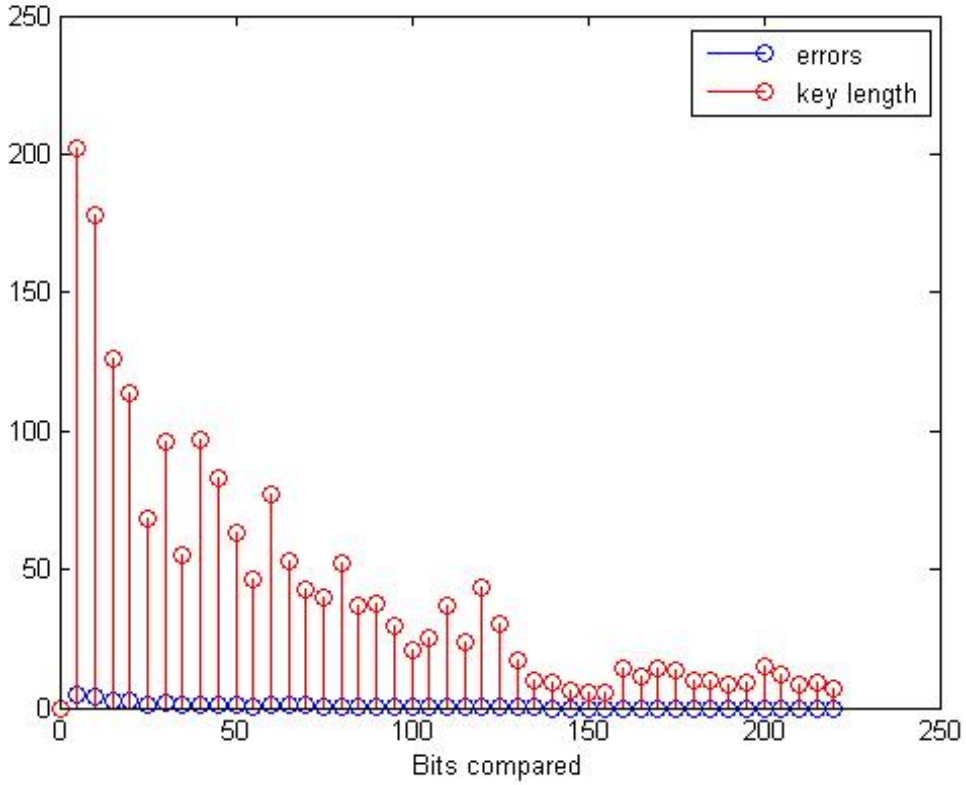

Figure 6.1 Final key errors and Length. 500 bits sent.

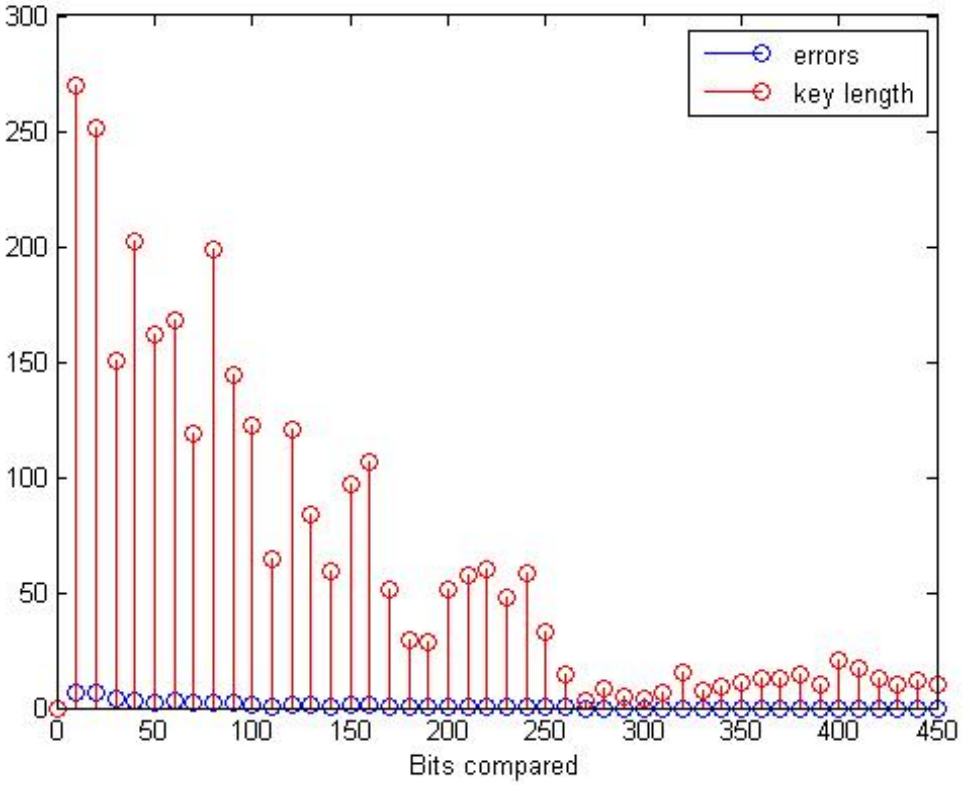

Figure 6.2 Final key errors and Length. 1000 bits sent.

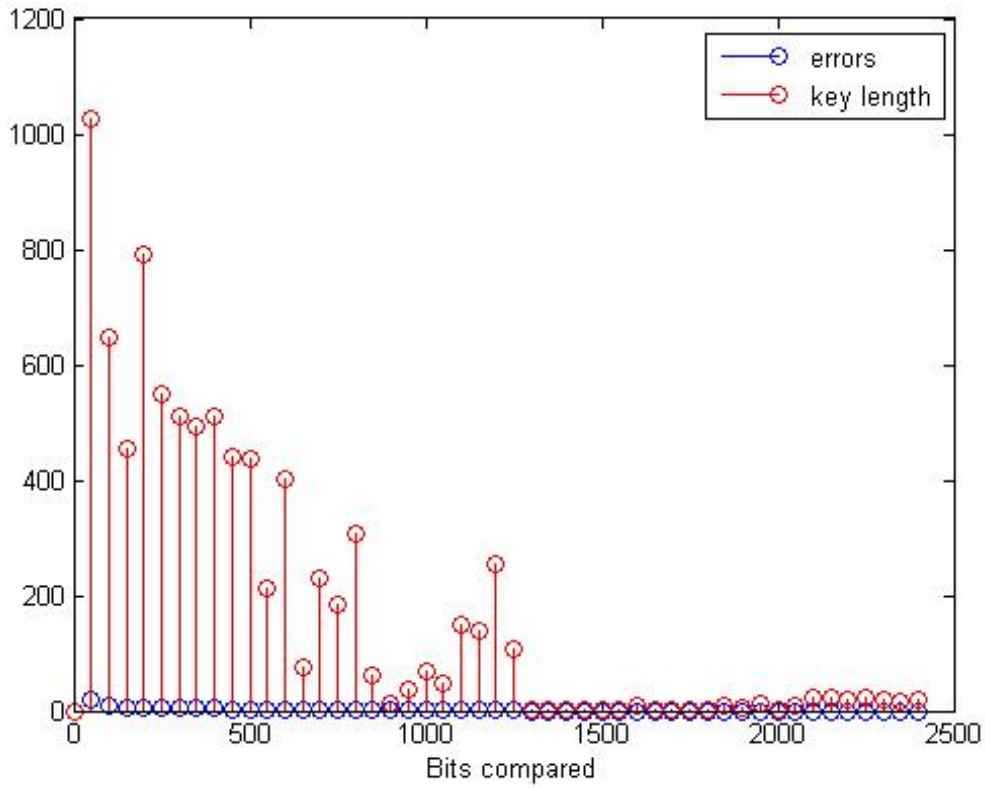

Figure 6.3 Final key errors and Length. 5000 bits sent.

We can see a similarity between the graphics where the optimum number of bits to compare is between the 40% . This result makes us think that is better to compare more bits rather than trying to remove the exact bits in the information reconciliation.

7 Quantum Cryptography Nets in Spain

Recently this year (2018) in Spain, Huawei, UPM(Universidad Politécnica de Madrid) and Telefónica have worked in common to demonstrate the viable use of quantum cryptography in commercial optic nets. They have created a net where the communications are encrypted using Quantum Key Distribution. The results obtained show the viability of using Quantum Cryptography, this could mean the start of the Quantum Cryptography era.

8 Conclusions

We have seen the problem of vulnerability the quantum computation brings to cryptography. Nevertheless, we have also seen that quantum cryptography can solve this problem. In order to see how the protocol works, we have simulated some possible scenarios that are similar to real channel conditions where we have checked that the protocol works as expected. The only problem we have faced is the fact that a minimum error can significantly affects the final key. Finally, the successful experiment made in Spain shows the real possibility of quantum cryptography implementation.

Appendix A
Simulation code

Code A.1 BB84simulation.m.

```
function y = BB84simulation(simulations, N, error, comp, startEve,
   endEve, threshold, s)
threshold = threshold/100;
errorRateTotal=0;
finalErrorTotal=0;
keyLengthTotal=0;
for P = 1:simulations
    biterrors=0;
    polAli = randi(2,1,N)-1;\%random Alice polarization where 0 is
       rectilinear and 1 diagonal
    polBob = randi(2,1,N)-1;\%random Bob polarization where 0 is
       rectilinear and 1 diagonal
    polEve = randi(2,1,N)-1;\%random Eve polarization where 0 is
       rectilinear and 1 diagonal

    qbitAli = randi(2,1,N)-1;
    qbitBob = zeros(1,N);
    qbitEve = zeros(1,N);

    qbitAli1 = qbitAli; \%qbits which affected by system error will
        arrive to Eve and Bob
    errorSystem = error*10^5;
    prob = randi(10000000,1,N);

    for R = 1:N
        if prob(R) < errorSystem
            if qbitAli1(R) == 0
                qbitAli1(R) = 1;
            else
                qbitAli1(R) = 0;
            end
        end
    end
```

```
\%bits obtained by Eve
for R = 1:N
    if R > startEve && R < endEve
        if polAli(R) == polEve(R)
            qbitEve(R) = qbitAli1(R);
        else
            qbitEve(R) = randi(2)-1;
        end
    end
end

\%bits obtained by Bob
for R = 1:N
    if R > startEve && R < endEve
        if polBob(R) == polEve(R)
            qbitBob(R) = qbitEve(R);
        else
            qbitBob(R) = randi(2)-1;
        end
    else
        if polAli(R) == polBob(R)
            qbitBob(R) = qbitAli1(R);
        else
            qbitBob(R) = randi(2)-1;
        end
    end
end

i=0;
for R= 1:N
    if polAli(R) == polBob(R)
        i= i+1;
        rawAlice(i) = qbitAli(R);
        rawBob(i) = qbitBob(R);
    end
end


\%bits comparison
longrawAlice = length(rawAlice);
longrawBob = length(rawBob);
step = floor(longrawBob/comp);
if step > 1
    i = 1;
    limit = longrawAlice;
    while i <= limit
        if rawAlice(i) ~= rawBob(i)
            biterrors = biterrors +1;
```

```
            end
            rawAlice(i) = [];
            rawBob(i) = [];
            i = i + step - 1;
            limit = limit-1;
        end
    else
        j=0;
        while j < comp
            if rawAlice(1) ~= rawBob(1)
                biterrors = biterrors +1;
            end
            rawAlice(1) = [];
            rawBob(1) = [];
            j= j+1;
        end
    end


    longrawAlice = length(rawAlice);
    longrawBob = length(rawBob);
    errorRate = biterrors/comp;
    errorRateTotal = errorRateTotal + errorRate;
    \%if the error rate is lower than the threshold we start the
        information
    \%reconciliation
    if errorRate < threshold
        if errorRate == 0
            keyFinalBob = rawBob;
            keyFinalAlice = rawAlice;
        else
        L = floor(0.05/errorRate);
        \%number of subsets
        if rem(longrawAlice,L) == 0
            M = floor(longrawAlice/L);
        else
            M = floor(longrawAlice/L)+1;
        end
        keyFinalBob = [];
        keyFinalAlice = [];
        for R = 1:M-1

            keyFinalBob = horzcat(keyFinalBob,depuraError2(rawAlice(
                L*(R-1)+1:L*R),rawBob(L*(R-1)+1:L*R)));

        end

        keyFinalBob = horzcat(keyFinalBob, depuraError2(rawAlice(L*
            M+1:longrawAlice),rawBob(L*M+1:longrawBob)));
```

```
for R = 1:M-1

    keyFinalAlice = horzcat(keyFinalAlice,depuraError2(
       rawBob(L*(R-1)+1:L*R),rawAlice(L*(R-1)+1:L*R)));

end

keyFinalAlice = horzcat(keyFinalAlice, depuraError2(rawBob(L
   *M+1:longrawAlice),rawAlice(L*M+1:longrawBob)));

end
biterrors= 0;
for R= 1:length(keyFinalAlice)
    if keyFinalBob(R) ~= keyFinalAlice(R)
        biterrors = biterrors +1;
    end
end


\%amplificacion de la privacidad calcular k,
keyLength = length(keyFinalAlice);
k = floor(2*errorRate*keyLength);
subsets = keyLength - k - s;
L = floor(keyLength/subsets);
definitiveKeyBob = [];
definitiveKeyAlice = [];
for R = 1:subsets-1
    definitiveKeyBob = horzcat(definitiveKeyBob,
       calculaParidad(keyFinalBob(L*(R-1)+1:L*R)));
end
definitiveKeyBob = horzcat(definitiveKeyBob,calculaParidad(
   keyFinalBob(L*R+1:keyLength)));

for R = 1:subsets-1
    definitiveKeyAlice = horzcat(definitiveKeyAlice,
       calculaParidad(keyFinalAlice(L*(R-1)+1:L*R)));
end
definitiveKeyAlice = horzcat(definitiveKeyAlice,
   calculaParidad(keyFinalAlice(L*R+1:keyLength)));
\%
\%       display(definitiveKeyAlice);
\%       display(definitiveKeyBob);
biterrorsfinales = 0;
for R= 1:length(definitiveKeyAlice)
    if definitiveKeyBob(R) ~= definitiveKeyAlice(R)
        biterrorsfinales = biterrorsfinales +1;
    end
end
```

```
        keyLengthTotal = keyLengthTotal + length(definitiveKeyBob);
        finalErrorTotal = finalErrorTotal + biterrorsfinales;
    else if errorRate == 0

        end
    end

end
errorRateMedia = errorRateTotal/simulations;
biterrorsFinalesMedia=finalErrorTotal/simulations;
keyLengthMedia=keyLengthTotal/simulations;

y= [errorRateMedia biterrorsFinalesMedia keyLengthMedia];
end
```

Appendix B
Graphics code

Code B.1 graphics.m.

```
function graphics (N, step)
comp = [0:step:(2*N/5)+step];
i=0;
for comp1 = 0:step:(2*N/5)+step
    i= i+1;
results = BB84simulation(50, N, 0, comp1, 0, floor(N/10), 5, 0);
finalErrors(i) = results(2);
finalLength(i) = results(3);
end
clf reset
figure(1)
stem(comp,finalErrors), hold on, stem(comp,finalLength,'r'), legend
   ('errors','key length'), xlabel('Bits compared');
end
```

Bibliography

[1] Giuliano Benenti, Giulio Casati, and Giuliano Strini. *Principles of Quantum Computation and Information*. World Scientific, 2004.

[2] Daniel T. Gillespie. *A Quantum Mechanics Primer*. Intertext Books, 1974.

[3] Dirk Bouwmeester, Arthur Ekert, and Anton Zeilinger. *The Physics of Quantum Information*. Springer, 2000.

[4] A. Einstein, B. Podolsky, and N. Rosen. Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? *Physical Review*, pages 777–780, 1935.

[5] J.S. Bell. On The Einstein Podolsky Rosen Paradox. *Physics Publishing*, pages 195–200, 1964.

[6] J.F. Clauser, M.A. Horne, A. Shymony, and R.A. Holt. Proposed experiment to test local hidden-variable theories. *Physics Review*, pages 880–884, 1969.

[7] Jian-Wei Pan, Zeng-Bing Chen, Chao-Yang Lu, Harald Weinfurter, Anton Zeilinger, and Marek Zukowski. Multiphoton entanglement and interferometry. *Reviews of Modern Physics*, 84:777–838, 2012.

[8] William K. Wootters and Wojciech H. Zurek. The no-cloning theorem. *Physics Today*, pages 76–77, February 2009.

[9] Ron Rivest. *Introduction to Modern Cryptography*. Elsevier, 2005.

[10] David Salomon. *Elements of Computer Security*. Springer, 2010.

[11] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *International conference on computers, systems & signal processing*, pages 175–179, December 1984.

[12] Omer K. Jashim, Safia Abbas, El-Sayed M. El-Horbaty, and Abdel-Badeeh M. Salem. Quantum key distribution: Simulation and characterizations. In *ICCMIT 2015*, April 2015.

[13] Matthias Scholz. *Quantum Key Distribution via BB84 An Advanced Lab Experiment*. PhD thesis, Humboldt University of Berlin, December 2006.

[14] Edmund optics. *LDM Focusable Laser Diode*.

[15] Thorlabs. *Liquid Crystal Noise Eaters User Guide*.

[16] Thorlabs. *Single Photon Counting Module. SPCMxxA Operation Manual*.

[17] Alfonsa García, Francisco García, and Jesús García. Lección 2. criptografía cuántica, 2014.

[18] Colin P. Williams and Scott H. Clearwater. *Explorations in Quantum Computing*. Springer, 1998.

[19] Jesús Martínez Mateo. *Criptografía cuántica aplicada*. PhD thesis, Universidad Politécnica de Madrid, 2008.