ARTICLE

Quantum Key Distribution - BB84 algorithm explained with a focus on the noise which affect communications

Francesco Aldo Venturelli*

Alma Mater Studiorum - Physics department A. Righi, Bologna

*Corresponding author: francesco.venturell3@studio.unibo.it

Abstract

The goal of the paper is to make a little illustration on how to do quantum communications nowadays, focusing on an old, primitive protocol, and the problems that communications are affected by. Since ancient times, when Werner Heisenberg formulated the uncertainty principle that has taken his own name (one of the started point to understand Quantum Mechanics), it was clear that adapting the real life to a new kind of technology, i.e. quantum would have generated quite a few problems yet. This paper starts with a brief introduction to cryptography, moving from classical part to the quantum one. Therefore, the BB84 protocol used to protect communications between two users in a quantum-like computing set-up is presented and discussed. The same treatment of the protocol will be modified by the introduction of a noise source that generally affect quantum computing simulations and a particular focus will be posed on the quantity of message Bob can understand by varying the probability with which errors occur and on the bit error rate that deals with the percentage of bit wrongly deciphered.

1. Introduction

1.1 Quantum mechanics: the starting point

Quantum mechanics is an innovative way of doing Physics that modified not only the theoretical aspects of the Science itself, but had a considerable impact on everyday life, bringing to light a new technology and applications, but also new issues that need to be solved. Computer Science is one of the biggest sectors which has been touched deeply by the "Quantum world" and still a lot of investigation and research activities arise from it. The enormous power of a supercomupter has been revealed and scientists think the time that separates humans from the remaining hidden part of the Nature they want to have access to, will drop drastically thanks to these up-to-date machines. The main difference between quantum and classical computers live in the intrinsic mathematical aspects of the basic unit of information needed for making a contact with them: the **bit**. If for a classical computer each request we ask corresponds to a precise sequence of bits but where any of them can only take the value of 0 or 1, then, for the quantum version, i.e. the quantum bit (**qubit**), takes again the values of 0 and 1 but at the same time. From this point we can understand the improvements a quantum computer will furnish to our works and one of those will result in an incredible speed-up in the computation and, coupled to that, an increasing computational complexity will arise.
As an example, imagine a bus that leaves from the airport of a metropolis and should move to the city-

center bringing lots of passengers already landed. Try to adapt this situation to a real one, i.e. where there is traffic between the two points. This is a typical complex-systems problem, where little few rules govern the scheme and his way of changing with time. A classical machine-learning algorithm has been tested to solve this problem, but it failed. On the contrary, a quantum machine-learning algorithm, which is characterized by a different way of processing and elaborating data, known as *data embedding*, has worked it out successfully updating every time the traffic changed his "form". We can understand that, with the introduction of quantum computers in real life, a considerable class of problems, known as *hard problems* will cease to exist. Unfortunately, at this moment there's much work to do because some aspects of quantum computation aren't already clear, some of quantum algorithms don't provide a significant improvement compared to the already existed classical ones, also, there's a non trivial difficultly to build quantum processors and quantum circuits and, finally, the last ones act like noise sources and will affect our results irrecoverably.

1.2 The communication

1.2.1 Classical antisymmetric cryptography

Telephone communications, online documents accessing, home banking, tax payment, online shopping and so on...are all examples of activities we do not make without cryptography or without willing to take a risk. Cryptography studies how to keep information secure, confidential and complete, in order to avoid corruption and eavesdropping by unknown people. A communication to be protect needs to be changed to something similar but not easy to bring back to the original - this is a process called "encryption". It's easy to imagine the opposite situation: where an object called key has used to have the access to the content of the message, that's now readable and intelligible (in a similar way to open a chest), known as **decryption**. A typical cryptography-process can be summarized with three components:

- ***message:*** the plain-text to be encrypted;

- ***encrypting key:*** the key used for the encryption of the message;

- ***encrypted message:*** the output of the communication (cipher-text).

Each of the users employed in the communication has a pair of keys: the **public key** that must be distributed and the **private key** that must be kept hidden. The mechanism of antisymmetric cryptography is based on the fact that, if one of the two keys is used for encryption, then the message will be deciphered by the other one. In a system which uses public keys, everybody can encrypt a message with the public key of the receiver, but that message can be deciphered only with his private key. To do so, the generation of two keys must be computationally easy, otherwise the process will be unfeasible. The power of a cryptography-system of public key is based on the difficultly of determining the private key corresponding to the public key. Security therefore depends only on keeping the private key secret, while the public key can be published without compromising security.

1.2.2 Quantum Cryptography

Quantum cryptography, as can be guessed by the reader, is a method of encryption that uses the naturally occurring properties of quantum mechanics to secure and transmit data in a way that cannot be hacked. This new method continues to rely on mathematics, but a considerable modification has been in principle introduced by Physics: either the message and the key (or for quantum mechanics, the basis state sequence the sender decided to encrypt the message with) do not correspond on sequence of 0 and 1 anymore, but they has been changed to polarization states of light (such as "up" and "down"). In a quantum computer, messages are represented by photons that can assume different orientations in the Bloch sphere, known as already anticipated polarization. Even if it's not so trivial to digest, for now on, every time a user should send a text or a password for example, makes use of packet of light that can be captured by the receiver that, with a backward process, can understand a message already sent. In fact, it uses individual particles of light, photons, to transmit data over fiber optic wire and these represent binary bits. Quantum cryptography is a system that is completely secure against being compromised without the knowledge of the message sender or the receiver. That is, it is impossible to copy (no cloning theorem) or view data encoded in a quantum state without alerting the sender or receiver. Quantum cryptography should also remain safe against those using quantum computing as well. The security of the system relies on quantum mechanics. These secure properties include the following:

- particles can exist in more than one place or state at a time;
- a quantum property cannot be observed without changing or disturbing it;
- whole particles cannot be copied.

These properties make it impossible to measure the quantum state of any system without disturbing that system. Photons are used for quantum cryptography because they offer all the necessary qualities needed: Their behavior is well understood, and they are information carriers in optical fiber cables. One of the best-known examples of quantum cryptography currently is quantum key distribution (QKD), which provides a secure method for key exchange.

2. One step back

2.1 The entanglement

The deep ways that quantum information differs from classical information involve the properties, implications, and uses of quantum entanglement. A bipartite pure state is entangled if its Schmidt number[1]is greater than one. If the vector ω can be expressed as a tensor product between basis vector then is said separable, otherwise is said entangled. From the Schmidt decomposition, we can see that ω is entangled if and only if ω has Schmidt rank strictly greater than 1. Entangled states are interesting because they exhibit correlations that have no classical analog. Not going to far on it, if Alice creates states called *Bell pairs* and measures them, the receiver will have no ambiguity on the measurement of his qubit that will be totally correlated to the one of the Alice. This is the real power of entanglement: the creation of a sort of correlation between states totally quantum like and that keep even tough particles are far apart from each other.

2.2 Quantum teleportation

[1] Quantum teleportation is a technique for moving quantum states around, even in the absence of a quantum communications channel linking the sender of the quantum state to the recipient. Suppose that two parties, named Alice and Bob, met long time ago but now live far apart. While together they generated an EPR [2] pair [3], each taking one qubit of the pair when they separated. Many years later, Bob is in hiding, and Alice's mission, should she choose to accept it, is to deliver a qubit $|\psi\rangle$ to him. She does not know the state of the qubit, and moreover can only send classical information to Bob. Should Alice accept the mission? Intuitively, things look pretty bad for Alice. She doesn't know the state $|\psi\rangle$ of the qubit she has to send to Bob, and the laws of quantum mechanics prevent her from determining the state when she only has a single copy of $|\psi\rangle$ in her possession. What's worse, even

[1] The Schmidt number is a particular parameter that expresses the degree of entanglement between two part of a bipartite system. Roughly speaking, taken two Hilbert spaces with different dimensions, for any vector ω in the tensor product $(H_1 \otimes H_2)$ there always exist orthonormal sets which $\omega = \Sigma_i \alpha_i u_i \otimes v_i$ where α_i are real and non-negative scalars.

[2] Einstein-Podolsky-Rosen.

[3] An EPR pair is a particular case of entangled pair of qubits: groups of particles (photons in this case) are generated or interact in ways such that the quantum state of each of them cannot be described independently of the state of the other.

if she did know the state $|\psi\rangle$, describing it precisely takes an infinite amount of classical information since $|\psi\rangle$ takes values in a continuous space. So even if she did know $|\psi\rangle$, it would take forever for Alice to describe the state to Bob. It's not looking good for her. Fortunately, quantum teleportation is a way of using the entangled EPR pair in order to send $|\psi\rangle$ to Bob, with only a small overhead of classical communication. In outline, the steps of the solution are as follows: Alice interacts the qubit $|\psi\rangle$ with her half of the EPR pair, and then measures the two qubits in her possession, obtaining one of four possible classical results, 00, 01, 10, and 11. She sends this information to Bob. Depending on Alice's classical message, Bob performs one of four operations on his half of the EPR pair. The state to be teleported is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, while the input state of the circuit is $|\psi_0\rangle = |\psi\rangle \frac{|00\rangle + |11\rangle}{\sqrt{2}}$ where the second state is already entangled. The whole calculation is left to the reader, what is important for us, is that as we explained previously, Alice's second qubit and Bob's qubit start out in an EPR state. She sends her qubits through a CNOT gate and then sends the first qubit through a Hadamard gate getting $|\psi_2\rangle = \frac{1}{2}[|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)]$. This expression naturally breaks down into four terms. The first term has Alice's qubits in the state $|00\rangle$, and Bob's qubit in the state $\alpha|0\rangle + \beta|1\rangle$ which is the original state $|\psi\rangle$. If Alice performs a measurement and obtains the result 00 then Bob's system will be in the state $|\psi\rangle$. Similarly, from the previous expression we can read off Bob's post-measurement state, given the result of Alice's measurement. This is the power of the entanglement. Letting quantum states interact to each other and then sending to someone very far apart from us, he will recover the state thanks to our measurement's result![1]

3. BB84-PROTOCOL

Figure 1. (From left to right) Charles Bennet (physicist) and Gilles Brassard (computer scientist), the inventor of the BB-84 protocol.

3.1

3.1.1 Basic set up

The use of general quantum states to represent information and encode messages would appear, from what we have said, to have some significant drawbacks compared to the use of classical states! Firstly (a), a received unknown quantum state cannot be reliably identified (unless it has been promised to belong to a specified orthonormal set) so the receiver cannot reliably read the message and secondly, (b), any attempt to read the message (in the context of general signal states) results in only partial information and is always accompanied by irrevocable corruption ("measurement collapse") of the quantum state and correspondingly, part of the sent message is necessarily irretrievably destroyed. Remarkably these seemingly negative features can be used to positive effect to provide valuable benefits for some cryptographic and information security issues, which in some cases turn out to be impossible to achieve with classical signals. For example, in communication between distant parties, (b) implies that any attempted eavesdropping on the message en route must leave a mark on the signal which can then in principle be detected by actions of receiver in (public) discussion with the sender. It turns out that this can be used to provide communication that is provably secure against eavesdropping. Classical messages on the other hand can always be read en route and sent on to the receiver perfectly intact. Also it turns out that the effects of (a) for the communicators can be circumvented by a suitably clever (non-obvious) protocol involving further (public) discussion between them. It is now known that quantum effects can provide benefits for a wide variety of cryptographic tasks beyond just secure communication.

3.1.2 Introductory to the process

In 1984, Charles Bennett and Gilles Brassard published a protocol based on Heisenberg's uncertainty principle. The protocol is named **BB84** after the authors' names and the year it was published. It is one of the most prominent quantum protocols ever made until nowadays. The BB84 protocol is an algorithm of quantum cryptography aimed to keep the communication between two users secure from the presence's of an hidden spy. Let's see a brief introductory to its mechanism. Imagine Alice wants to send a secret message to Bob with a quantum device and is wondering about the security of the text inside. She generates a precise[4] message and uses a key for the encryption, as we've already anticipated in the cryptography paragraph. In the following picture you can appreciate the total random behaviour of Alice generating the bit string to deliver to Bob.

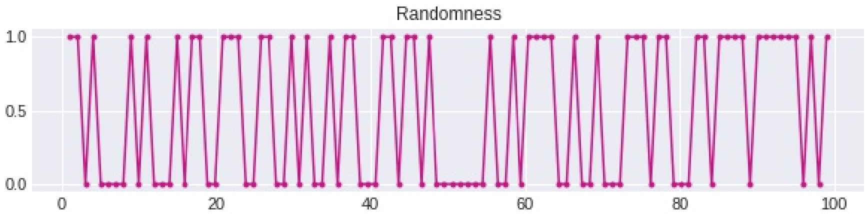
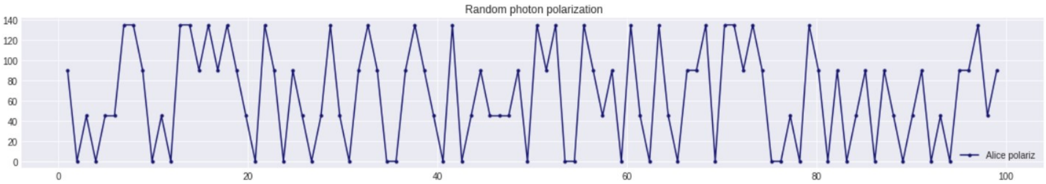


Figure 2. Randomness of Alice's creation of the message to send. As already said, we've considered a total random message generation avoiding the complexity of the corrispondence between a real message and the bit string associated. In this particular example a sequence of length 100 bits is used.

Once she chosen, she needs to polarize photons that contain the message, i.e. the classical sequence of bits. By doing so, Bob remains to apply a reversed process: guess the Alice's key, hope to have caught enough photons in the same polarization state and read the remaining message. These steps are discussed better in the following. Here you can have an idea of Alice's random generation of the encrypting key.

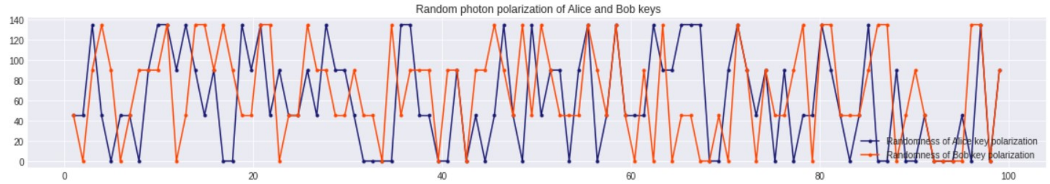


[2]

Figure 3. Randomness of Alice's creation of the key to encrypt the message. In this particular example a sequence of length 100 bits is used.

[4] In this article all the sequences are generated randomly and it's assumed they refer to a precise message

This time, instead of only two values (0 and 1), the blue dots assume four different values corresponding to the four angles of polarization: 0, 45, 90 and 135 degrees. If the classical bit is let's say 0, then it'll be encoded in the $|0\rangle$ or $|1\rangle$ key state, while the message bit corresponding to 1 will be encoded in the $|\pm\rangle$ state. To have a complete idea of the comparison between the key's bit an esplicative graph is presented below.



[2]

Figure 4. Comparison of Alice and Bob's key's bits.

3.1.3 The steps

[2] Let's see in detail what happens in the BB84 protocol. As before, Alice generates two uniformly random binary strings: the classical message $\mathbf{x} = x_1x_2x_3...$ and the encrypting key $\mathbf{y} = y_1y_2y_3...$ that is still written in terms of 0 and 1 only. Subsequently she prepares m qubits in the states $|\psi_{x_1y_1}\rangle\,|\psi_{x_2y_2}\rangle\,...\,|\psi_{x_my_m}\rangle$ and sends these m qubits over to Bob. When Bob receives the qubits they may no longer be in the states $|\psi_{x_iy_i}\rangle$ that Alice sent, since the quantum channel may have been **noisy** or eavesdropping may have occurred. To understand how the protocol works let us imagine first that there is no eavesdropping and that the channel is perfectly noiseless i.e. Bob receives precisely the states$|\psi_{x_iy_i}\rangle$ that Alice sent. Bob chooses a uniformly random bit string $\mathbf{y'} = y'_1y'_2y'_3...$ and measures the i^{th} received qubit in the y'_i basis to get a result x'_i. Note that y'_i is Bob's guess at Alice's choice of encoding basis and x'_i is his guess at Alice's bit value x_i. Let $\mathbf{x'} = x'_1x'_2x'_3...$ be the string of Bob's measurement outcomes. It's worth to highlight that if $y'_i = y_i$ (Bob correctly guessed Alice's encoding basis) then $x'_i = x_i$ and he will learn her message bit correctly with certainty. But if $y'_i \neq y_i$ then x'_i is completely uncorrelated with x_i. After the completion of the previous step Alice and Bob publicly reveal and compare their choice of bases i.e. their strings y and y' (but they do not reveal the strings $\mathbf{x}$ and $\mathbf{x'}$). They discard all bits x_i and x'_i for which $y_i \neq y'_i$ leaving shorter strings of expected length $m/2$. For making everything clear we call these strings $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}'}$. Under our assumptions of no noise and no eavesdropping in the quantum channel, these bit strings would provide the desired outcome of a shared secret key. In reality there will always be some noise in transmissions and we need to deal with possible eavesdropping too. In particular, this paper will focus briefly on the influence of the noise in the transmission of messages. To address these issues the BB84 protocol concludes with a method we discuss below[3].

3.2 Error detection

3.2.1 BER - Bit Error Rate

Alice and Bob now have in their hands $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}'}$ strings. They want to estimate *the proportion of bits in* $\hat{\mathbf{x}}$ *and* $\hat{\mathbf{x}'}$ *that are **not** equal to each other*, i.e. the **bit error rate (BER)**. To do so, they publicly compare a random sample of their strings (say half of their bits chosen at random positions in order to expose themselves to any dangers), and then discard all the announced bits. They assume that the remaining bits have about the same proportion of errors as those checked. Next they want to correct these remaining errors (albeit at unknown positions) to obtain two strings that agree in a high percentage of positions with high probability. Remarkably this can be done (at the expense of sacrificing some more bits) without giving everything away, if the bit error rate is not too large[3][4].

3.2.2 Privacy Amplification

From the estimated bit error rate Alice and Bob can estimate the maximum amount of information that an eavesdropper is likely to have obtained about the remaining bits. From this information estimate they use techniques of so-called privacy amplification from classical cryptography to replace their strings by even shorter strings about which the eavesdropper can have practically no knowledge whatever (with high probability). This concludes the BB84 quantum key distribution protocol.

3.3 The noise problem

The problem for a quantum computer is noise. At the beginning of this treatment, we've said that quantum computers operate by exploiting the laws of quantum mechanics that provide an increasingly amount of computational complexity and speed. Unfortunately, performing quantum computations creates quantum decoherence, or noise as its commonly called. Until now we have dealt almost solely with the dynamics of closed quantum systems, that is, with quantum systems that do not suffer any unwanted interactions with the outside world. Although fascinating conclusions can be drawn about the information processing tasks which may be accomplished in principle in such ideal systems, these observations are tempered by the fact that *in the real world there are no perfectly closed systems*. Real systems suffer from unwanted interactions with the outside world. These unwanted interactions show up as noise in quantum information processing systems. We need to understand and control such noise processes in order to build useful quantum information processing systems. What is the distinction between an open and a closed system? For the first one, everything that surround it, say the *environment* has an influence on it. An open system interacts with other systems while the other one does not. No quantum systems are ever perfectly closed, and especially not quantum computers, which must be delicately programmed by an external system to perform some desired set of operations. For example, if the state of a qubit is represented by two positions of a electron, then that electron will interact with other charged particles, which act as a source of uncontrolled noise affecting the state of the qubit. An open system is nothing more than one which has interactions with some other environment system, whose dynamics we wish to neglect, or average over. Since qubits have multiple possible states, their combined computational ability scales exponentially as you add more of them. This means they become lousy with errors once they're operating with just a handful of qubits. Unfortunately we need a lot of qubits to do anything useful with a quantum computer and again it seems that nature has put obstacles in our path that must be overcome.

4. Simulation and results

4.1 Noisless version

This is the penultimate part of the work, where the simulation and results are shown. Before starting, it's a good idea to point out that the code has been written in Python3 and the *Google-Colab* environment has been used for writing and testing the model. Going back to BB84 protocol, the reader will have got an idea about the process itself and its application. In this paper, as anticipated, the spy is missing and the only noise's presence is considered. The article focuses totally on the effect of the noise on a quantum circuit realized for the main purpose: guaranteeing security during a communication between two users. Although the most captivating thing this algorithm owns, i. e. the spy, we can observe how the communication gets corrupted and how strategically the two guys should reconstruct the entire message to understand it!

At the starting point, Alice, who sends the message, should produce a random classical bit string to share. Again in this case, a complete random sequence of 10 bits is generated in order to avoid too much complexity on this simple model, that's not the object of the presentation. By using a random number generator inside *Numpy* and set the boundaries to be 0 and 1 we get the desired string:

$$[1, 1, 0, 1, 1, 0, 0, 0, 0, 0]$$

.
The same process has been done for Alice's encrypting key:

$$[0, 1, 0, 1, 1, 0, 1, 0, 1, 0] \qquad .$$

Note that in the model only 0 and 1 values of the keys are used instead of the polarization angles. Once message and key are generated, it's time to realize a circuit and to connect the two parties and see what happens.

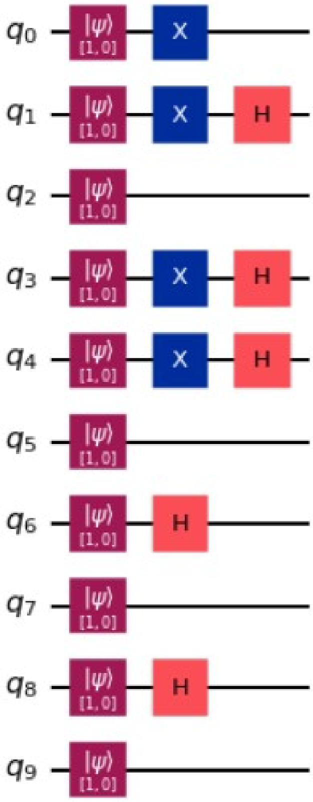


Figure 5. Alice's part of the quantum circuit.

The figure above represents the quantum circuit referred to Alice only. It's worth highlighted that for the current model, the following qubits are used: $|\psi_{00}\rangle = |0\rangle, |\psi_{10}\rangle = |1\rangle, |\psi_{01}\rangle = |+\rangle$ and $|\psi_{11}\rangle = |-\rangle$. In this way, every time the message's bit and the key's bit coincide, if they are all 0 then nothing happens and, while if they are all in 1 state, a gate flip and an Hadamard gate are applied. Instead only one flip is applied if the key's bit is in 0 and the one belonging to the string is in 1. In the remaining case only an Hadamard is applied. This is done because, once the message is encrypted, Bob needs to apply an inverse process to reconstruct what has been sent to him, remembering that he will measure in the computational basis, so qubits in the $|\pm\rangle$ states must be converted to $|0\rangle$ and $|1\rangle$. Now it's Bob's turn. Remember he communicates with Alice only by a quantum device avoiding any classical channel that can be stormed by an eavesdropper. He applies the inverse transformation in order to recover perfectly the message which, in this noiseless case, shouldn't be varied. Here is printed the guessed key:

$$[0, 1, 1, 1, 1, 1, 0, 1, 0, 1]$$

We can ascertain that some encoded qubits are totally wrong-placed with respect to Bob's framework and therefore they are colorful with green and red to emphasize the failure.

$$[0, 1, 1, 1, 1, 1, 0, 1, 0, 1]$$

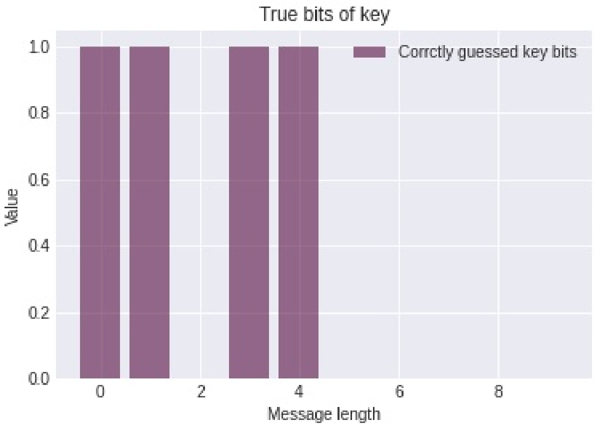


Figure 6. True key bits respectively in positions 0,1,3 and 4 as indicated in the histogram.

This is intuitive since, if Bob guesses the basis qubit he has only 50 percent of probability of guessing correctly, thus, for the law of large numbers, on average Bob will guess $\frac{m}{2}$ of the key.

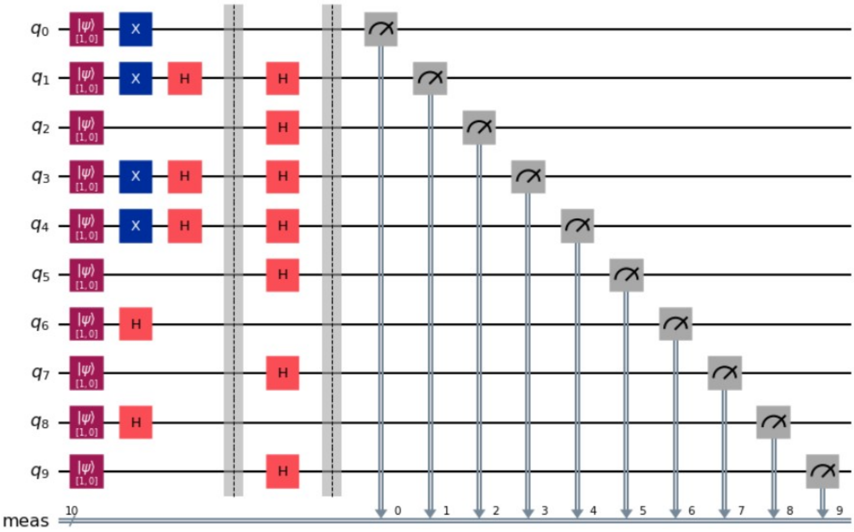


Figure 7. Total circuit comprehending either Alice, either Bob

It's evident that if the code had been run many times, the circuit would have been different as well as the operations Bob must fulfill to have access to the content of the message. Finally, only the bit error rate remains to be evaluated, but in this simple starting case, it's easy to guess what it's should be: 0. This means that Bob has been able to reconstruct the entire message that has not been corrupted by any sort of error or noise in this case. The communication did well.

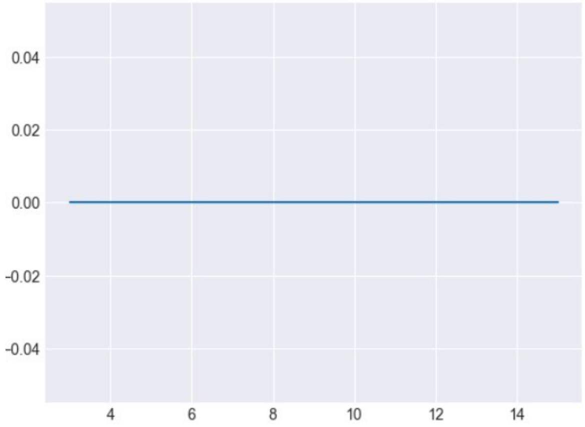


Figure 8. Bit error rate (BER) as a function of message's dimension without noise contamination. The message has been totally recovered and nothing has been interposed between Alice and Bob.

4.2 Real simulation with noise

In this last section we examine the effect of the noise on the circuit used for the communication between Alice and Bob. The idea was to create a noise model with the standard functionality that Qiskit offers and realize a graph showing the quantity of pure message Bob had access to. Let's see in detail what already described. Since a new function aimed to show the noise has been written, a new bit string of message has been generated and has nothing to do with the previous one. The starting point for this case is a standard practice when deal with noise, i. e. using a provider and simulate a real quantum circuit with an IBM server. The *provider* module contains the classes used to build external providers for Qiskit Terra. A provider is anything that provides an external service to Terra and the typical example of this is a Backend provider which provides Backend objects which can be used for executing *QuantumCircuit* and/or schedule objects. This module contains the abstract classes which are used to define the interface between a provider and Terra. A provider class serves a single purpose: to get backend objects that enable executing circuits on a device or simulator. The backend classes are the core to the provider. These classes are what provide the interface between Qiskit and the hardware or simulator that will execute circuits. This includes providing the necessary information to describe a backend to the compiler so that it can embed and optimize any circuit for the backend. As an example of what has been said a *Ibm Nairobi* of 7 qubits was selected to simulate a real circuit. By doing this, a bit string of message with length 7 has been adapted for the purpose. Then the *Qasm Simulator*[5] has been invoked to visualize the result through an histogram.

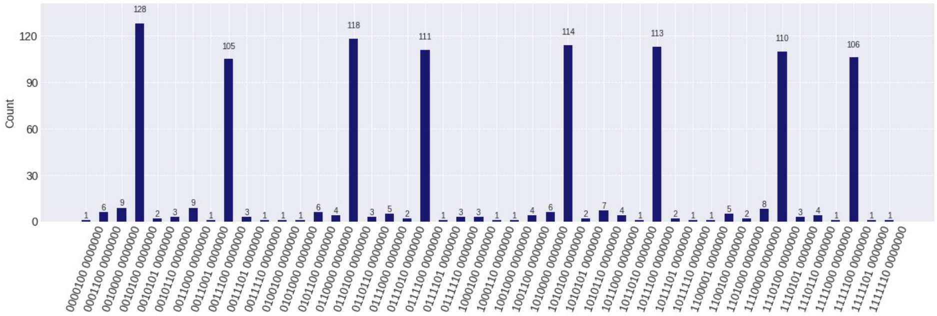


Figure 9. Result of the *getcount()* function of the result of the noise simulation on the Ibm Nairobi with a bit message of 7 bits.

The last part of the work is probably the most important. Now it's assumed that something strange may happen during the transmission of the message from Alice to Bob. In particular, a *pauli error* function has been used to generate two kinds of error that can occur: the bit flip and the phase flip. After having read Qiskit's documentation on how to generate a selected type of error on the qubits, a single function has been created and then repeated many times to plot different results associated to error's combinations. In the function named *generating noise*, plus a string that identifies the error selected, is reported the creation of the circuit and the transmission of the bit message in the presence of the noise. Here a brief summarize of the passages inside the code is represented here:

- **pass** message's length, list of probabilities which errors are applied with, maximal number of repetitions (max count);
- **random generation** of Alice message, encrypting key every time count is less then its maximum (unfortunately a small number of max count has been chosen in order to not overload the laptop too much);
- Bob's guessing decryption key;
- **build** noise model with the standard line of code presented inside Qiskit's documentation and add the error (bit flip for now) to all qubits and then measure;

[5] The Qasm Simulator is the main Qiskit Aer backend. This backend emulates execution of a quantum circuits on a real device and returns measurement counts. It includes highly configurable noise models and can even be loaded with automatically generated approximate noise models based on the calibration parameters of actual hardware devices.

- **creation** of an array initialized to negative 1 in order to keep track of how many times Bob gets the right bit of message;
- **loop** inside this array to get the right bit message to append to Bob's empty message;
- **return** the fraction of correct message guessed by Bob.

4.2.1 Bit flip error

The mathematical aspects of the theory behind quantum computation is a bit complicated and long. Since this paper wants to illustrate a specific small event inside quantum computing science, it's been preferred to leave the dirty work behind and just insert a few pills of theory here and there. Up to this choice, it's necessary to introduce, in a very fast way, the meaning of a *quantum operation*. A quantum operation is an equation of this kind $\rho' \longrightarrow \mathcal{E}(\rho)$ where map each density matrix to its endomporphism defined above. These are unitary transformations and measurements, for which $\mathcal{E}(\rho) = U\rho U^{\dagger}$ and $\mathcal{E}_m(\rho) = M_m\rho M_m^{\dagger}$, respectively. The quantum operation captures the dynamic change to a state which occurs as the result of some physical process; ρ is the initial state before the process, and $\mathcal{E}(\rho)$ is the final state after the process occurs, possibly up to some normalization factor. The bit flip channel flips the state of a qubit from $|0\rangle$ to $|1\rangle$ (and vice versa) with probability 1 − p. It has operation elements of this kind: $E_0 = \sqrt{p}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$ and $E_1 = \sqrt{1-p}\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$.

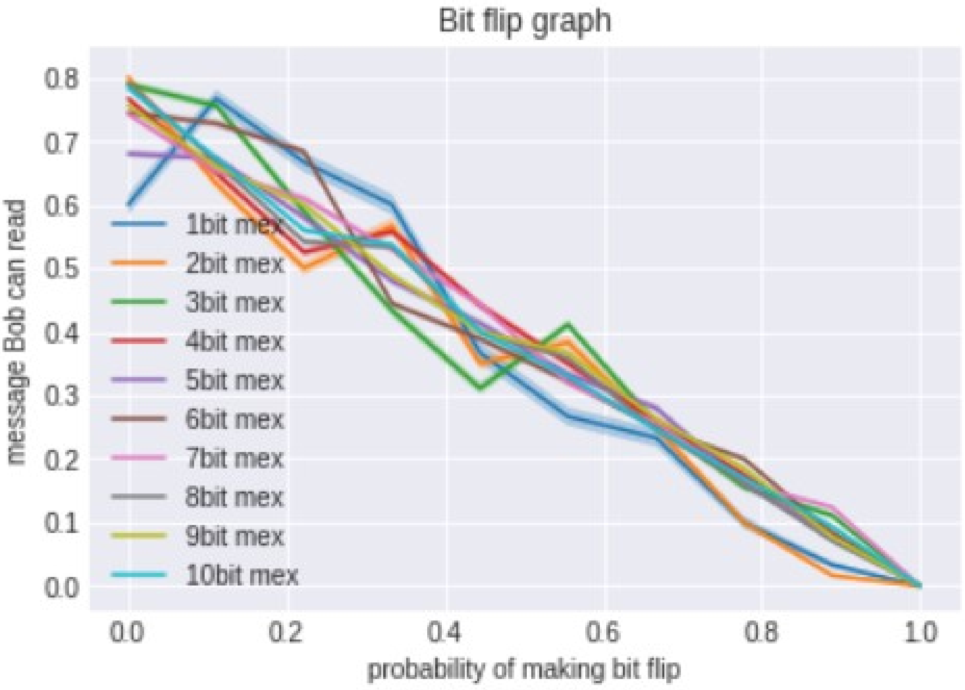


Figure 10. Graph representing the fraction of learned message by Bob as a function of the probability to make a bit flip during the communication.

In the graph above the portion of corrected deciphered message by Bob depending on how much the bit flip error occurs in the sending-and-receiving process is plotted. It's worth highlighted that, even if the probability of bit flip is near 0 or completely 0, the learned message has still not entirely recovered because a small number of repetition (message sent many times by Alice) has been used. As clarified before, this is a simple model illustration and for that, it was agreed not to weigh too much the running notebook. As the probability increases, as can be expected, the learned message approaches zero. Already from here, the reader can have an idea of how devastating is the effect of a noisy device on the simple message delivering. You can see 10 different lines which are equivalent to messages of variable length: from 1 bit only to 10 bits. After that, once in the plot, it was decided to create a list on which to repeat the entire process, thus creating a statistic. Therefore, the correctly identified message that you see in the graph is more of an average, surrounded by its standard deviation divided by the average itself.

4.2.2 Phase flip

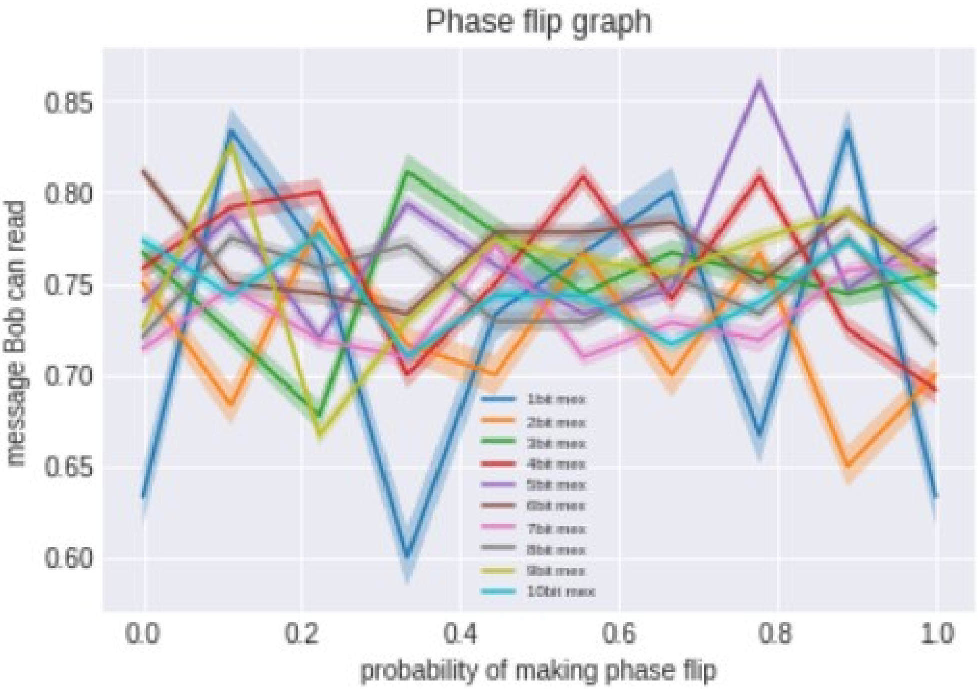


Figure 11. Learned message by Bob as a function of the probability that a phase flip occurs

In the currently graph is plotted the phase flip instead. As usual, the phase flip channel has operation elements $E_0 = \sqrt{p}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $E_1 = \sqrt{1-p}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$.

In this case the only difference inside the function is the fact that the phase flip Pauli error is used, composed by a Z operator that changes the phase of the $|1\rangle\langle 1|$ matrix element and leave all the others untouched. Something strange has happened. The phase flip does not have a destructive effect on the message Bob received. This makes sense because every time Bob receives key's qubits encoded in the $|\pm\rangle$ state he performs the Hadamard gate that restores completely the phase, so the fraction of the understand remains almost the same, or around 75%. It's necessary to modify the code in order to have an idea about the quantity of the understand message after a phase flip on each qubit.

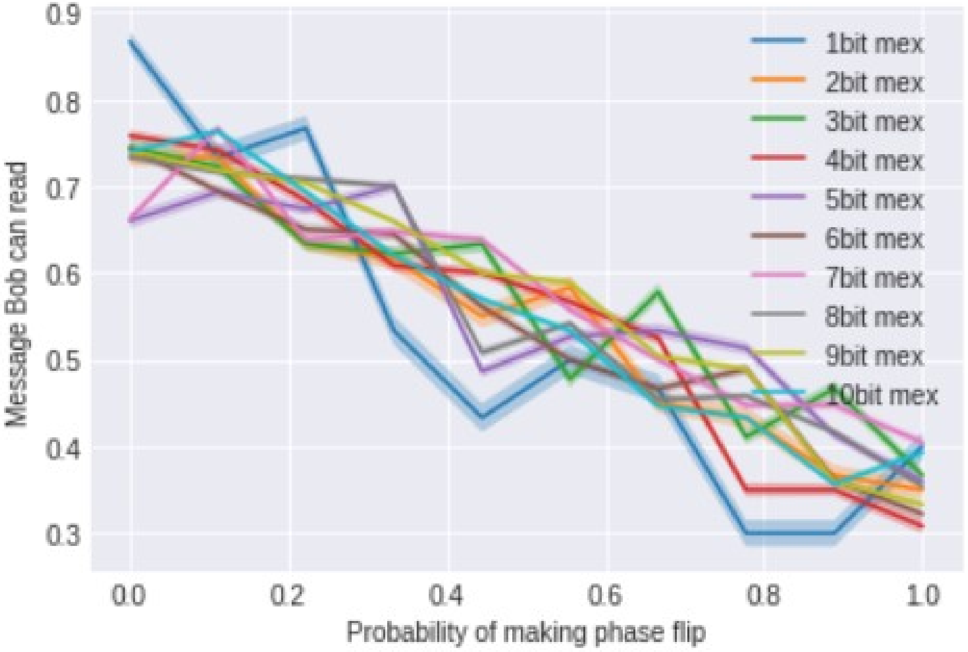


Figure 12. Phase flip without the Qiskit's instruction *measure*.

Here the idea of making a combination of these two kinds of errors is shown in the next page. In the first case the Z error is applied during the measurement process, so it's not detected and only the bit flip results as the main source of error. In the second picture instead, the phase flip is applied on each qubit before the measurement and then it's combined with the bit flip, leading to a curious behaviour. It seems that the quantity of understandable message increases a little bit as the probability of making these two errors approaches to 100%. It' like the two errors compensate each other, leading to a small improvement in the message Bob has access to. This phenomena is totally random, it would be convenient to enlarge the number of message bits in order to have a much bigger statistic on it and have a much complete result.

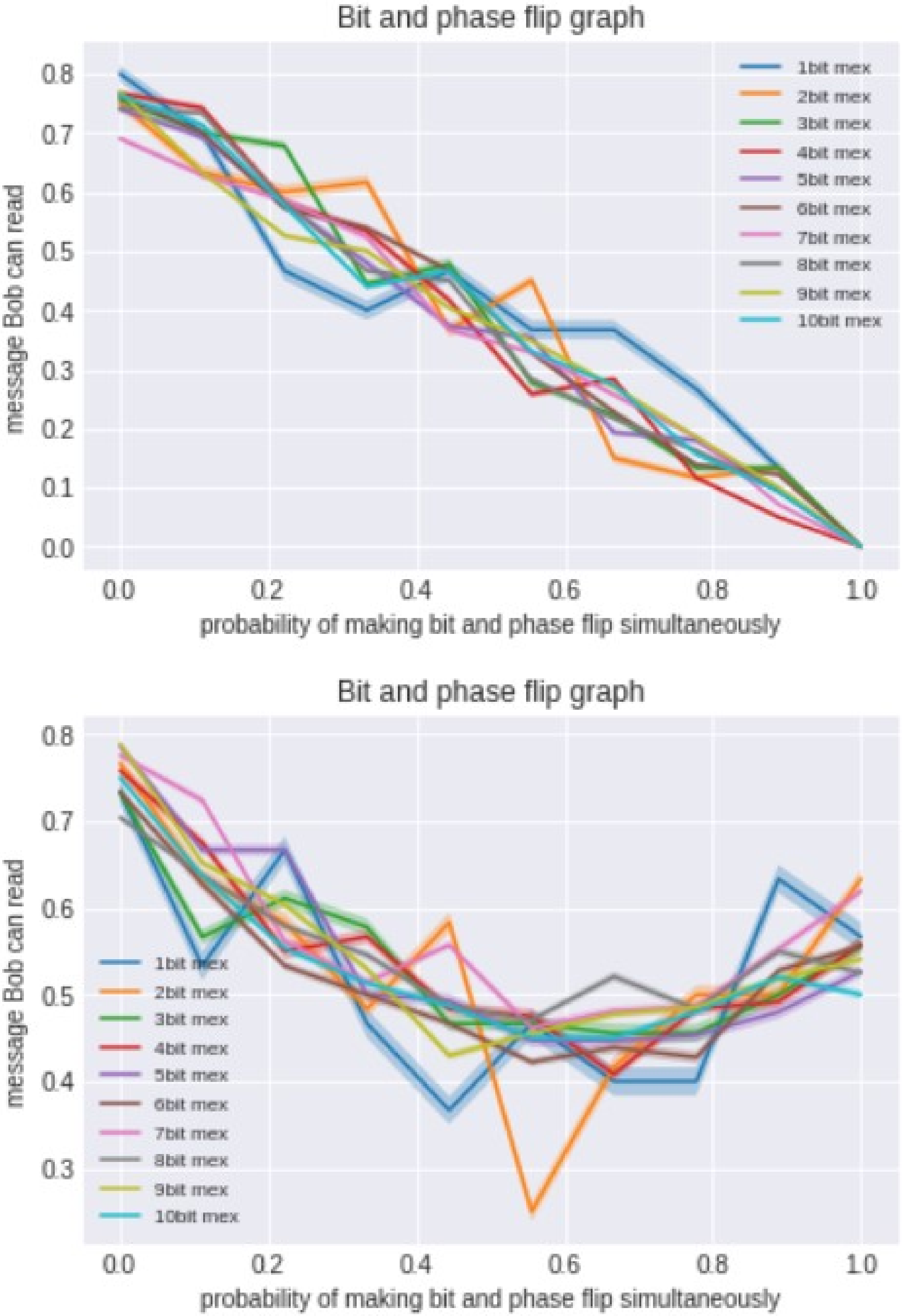


Figure 14. Bit phase flip not during the measurement.

4.3 Bit error rate graphs

To be more complete, in this page the bit error rate as function of the probability of making such errors has shown. Note an increasing bit error rate as the probability of making an error gets bigger.

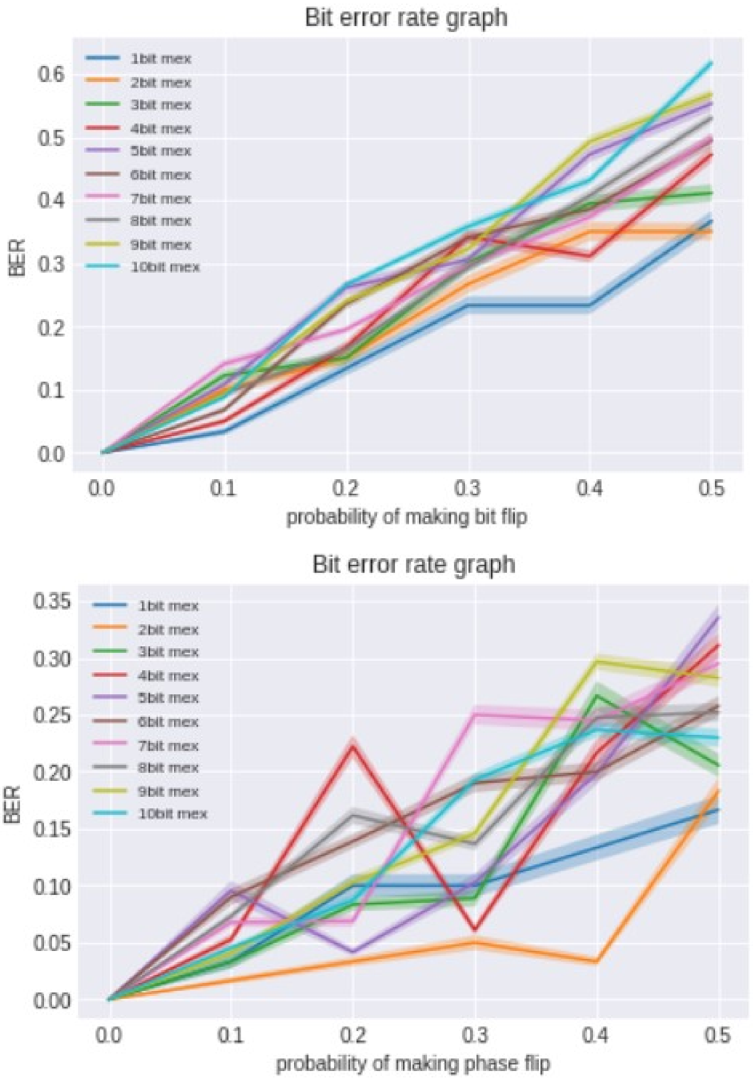


Figure 15. Bit error rate graphs for bit and phase flip respectively.

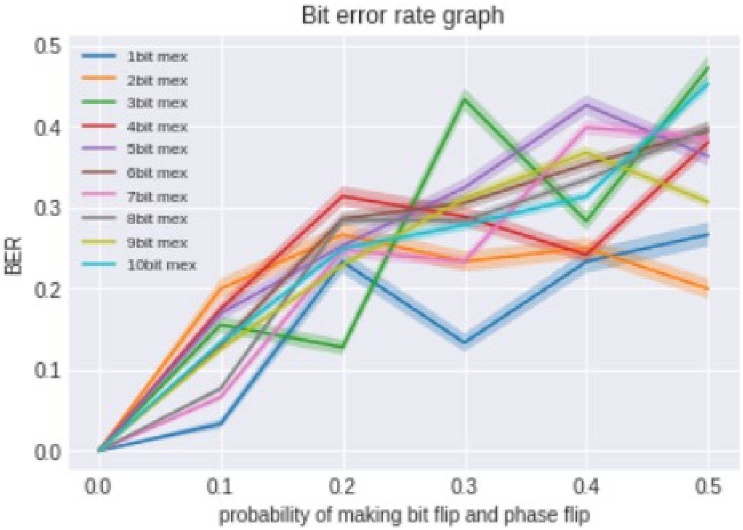


Figure 16. Bit error rate for bit and phase flip error combined together.

5. Conclusions

As we have seen, the possibility of making not only one, but many errors due to a noisy quantum circuit plays a fundamental role in quantum key distribution. In addition, since Alice should repeat the entire process in order to allow Bob to recognize as much message quantity as possible, it can never be free from errors and will be a source of noise itself! Qiskit gives the opportunity to add other kinds of errors such as *depolarizing channel*, *Krause error*... but in this article, as it has shown, a combination on only two types of error are used. This can be considered as a good starting point for a more deep study in the large quantum key distribution set that is pure quantum. In fact, the reader interested to go further could extend the message's bit string and see what happens for long messages. Furthermore, can use a different number of repetition to get a more fine statistics, use a greater number of shots in the simulation process for the circuit and the measurement. Here instead, it was used a lower number in order to not to stress to much the laptop's power. For that it's strongly suggested to move to *Google Colab* in order to have some helps by the GPU. Another improvement that can be done would be to plot the bit error rate and the learned message as a function of the message's length, instead of probability values, and then fix a single value of probability as a hue and see what happens. It would be be also interesting to see the effect of an eavesdropper, that place between the two parties, and study the quantity of error that arises from him, translated in missed bits' Bob message. Still, it could study the more complicated case where the two parties suffer the presence of a noisy circuit and an eavesdropper at the same time. If you would like to see the code in detail and making modifications go to https://github.com/poporubeus/QKD_BB84_repo to grab what you want. Hope this article has clarified the problem of noise in quantum communications and be a starting point for a much complete forward work.

Thanks.

References

[1] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[2] Akwasi Adu-Kyere, Ethiopia Nigussie, and Jouni Isoaho. "Quantum Key Distribution: Modeling and Simulation through BB84 Protocol Using Python3". In: *Sensors* 22.16 (2022), p. 6284.

[3] Eric P Hanson, Vishal Katariya, Nilanjana Datta, and Mark M Wilde. "Guesswork with Quantum Side Information: Optimal Strategies and Aspects of Security". In: *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2020, pp. 1984–1989.

[4] Peter W Shor and John Preskill. "Simple proof of security of the BB84 quantum key distribution protocol". In: *Physical review letters* 85.2 (2000), p. 441.

[5] Robin Harper, Steven T Flammia, and Joel J Wallman. "Efficient learning of quantum noise". In: *Nature Physics* 16.12 (2020), pp. 1184–1188.