Qubit by Qubit:
Introduction to Quantum Computing
Final Project

Encryption in Python

Lázaro Raúl Díaz Lievano
lazaro.diazl@alumno.buap.mx
+52 221 264 5041
LinkedIn

April 21, 2024

1 Project Descryption

In the Encryption in Python Project, students will write code to encrypt and decrypt a message of their choice using a key created through Quantum Key Distribution (QKD). They will analyze their work and write a short paper discussing the steps of encrypting and decrypting your data and the effect an eavesdropper would have on the success of transmitting encrypted data.

2 Project

This project consists of two parts: first, we need to create a key that will be used to encrypt and decrypt messages, for this purpose, we will use the BB84 quantum protocol. Second, we will define two functions; one for encrypting and another for decrypting messages. For this task, we utilize binary addition, in this way, we need to transform the message into binary format before encrypting it.
All the code is hosted in Github Repository.

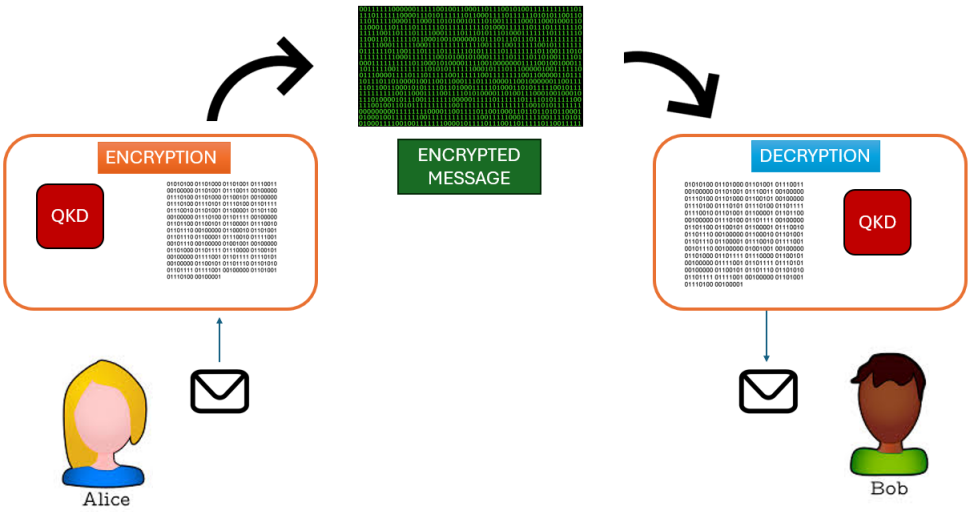

Figure 1. General structure of the project.

Note that, sending the message is only possible if both Alice and Bob have the same key generated by the BB84 quantum key distribution. Otherwise, they should repeat the BB84 quantum protocol to generate a secure key.

2.1 Creating the key

BB84 is a quantum key distribution (QKD) protocol developed by Charles Bennett and Gilles Brassard in 1984. It was the first quantum cryptographic protocol, utilizing the principles of quantum mechanics, particularly the no-cloning theorem, to offer provably secure key generation.[BB84]

The foundation of BB84 lies in the impossibility of gaining information to distinguish two non-orthogonal states without disturbing the signal.

The protocol involves two parties, Alice and Bob, connected by a classical communication channel. Additionally, Alice can prepare qubits in a specific state and transmit them to Bob through a unidirectional quantum channel.

Alice initiates the protocol by generating two random binary strings, a and b, of the same length n. String a encodes the state, while string b encodes the basis. She then prepares n qubits according to the following scheme:

$$\begin{aligned}
|q[i]\rangle = |0\rangle \quad &\text{if} \quad a[i] == 0 \quad \text{and} \quad b[i] == 0 \\
|q[i]\rangle = |1\rangle \quad &\text{if} \quad a[i] == 1 \quad \text{and} \quad b[i] == 0 \\
|q[i]\rangle = |+\rangle \quad &\text{if} \quad a[i] == 0 \quad \text{and} \quad b[i] == 1 \\
|q[i]\rangle = |-\rangle \quad &\text{if} \quad a[i] == 1 \quad \text{and} \quad b[i] == 1
\end{aligned}$$

where $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$.

Alice sends her qubits to Bob. Bob then generates a random binary string c of length n. He measures the qubit $|q[i]\rangle$ in the $\{|0\rangle, |1\rangle\}$ basis (computational basis) if c[i] == 0 and in the $\{|+\rangle, |-\rangle\}$ basis (Hadamard basis) if c[i] == 1 and stores the result in a string m. Alice and Bob then announce the strings b and c, which encode the random basis choices of Alice and Bob respectively.

The strings a and m match in the places where b and c are the same. This happens because the state was measured on the same basis in which it was prepared. For the remaining bits, the results are uncorrelated. The bits from strings a and m where the bases match can be used as a key for cryptography.

BB84 is resilient against intercept-and-resend attacks due to the no-cloning theorem, which guarantees that a qubit in an unknown state cannot be cloned. Consequently, any measurement will destroy the initial state of the qubit. If an eavesdropper intercepts Alice's qubits, measures them on a randomly chosen basis, prepares another qubit in the measured state, and resends it to Bob, the state measured by the eavesdropper may not be the same as the state prepared by Alice. Therefore, Alice and Bob may detect eavesdropping by comparing some bits from their obtained keys.

2.1.1 No eavesdropper

n the absence of an eavesdropper, the BB84 protocol operates smoothly. Alice prepares her qubits according to the random binary strings a and b and sends them to Bob. Bob measures each qubit in the appropriate basis c and stores the results in string m. Since there is no interference from an eavesdropper, Alice and Bob's measurements perfectly align. When they publicly announce their basis choices (b and c), they find that they used the same basis for most qubits. By comparing these bits, they derive a secure cryptographic key, as their measurements are consistent and unaltered. For example, if Alice and Bob have a different basis, then Bob doesn't measure anything.

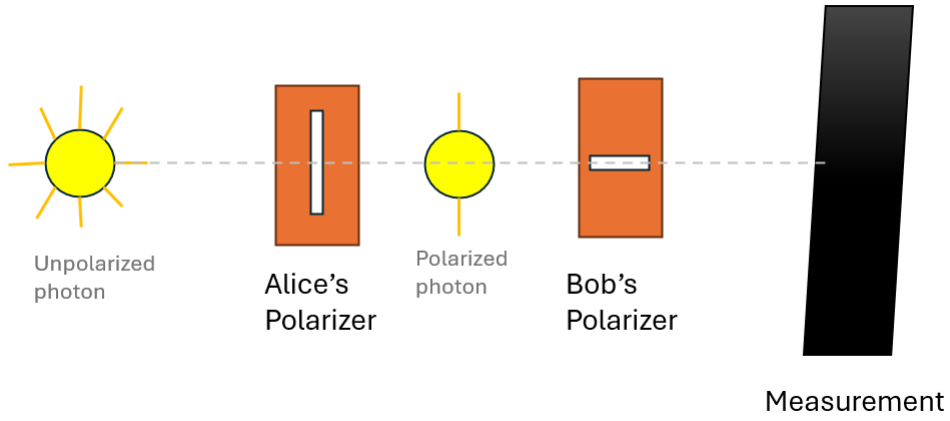

Figure 2. A secure BB84 quantum protocol with different Alice's and Bob's basis.

On the other hand, if Alice and Bob have the same basis then, Bob will measure a 1.

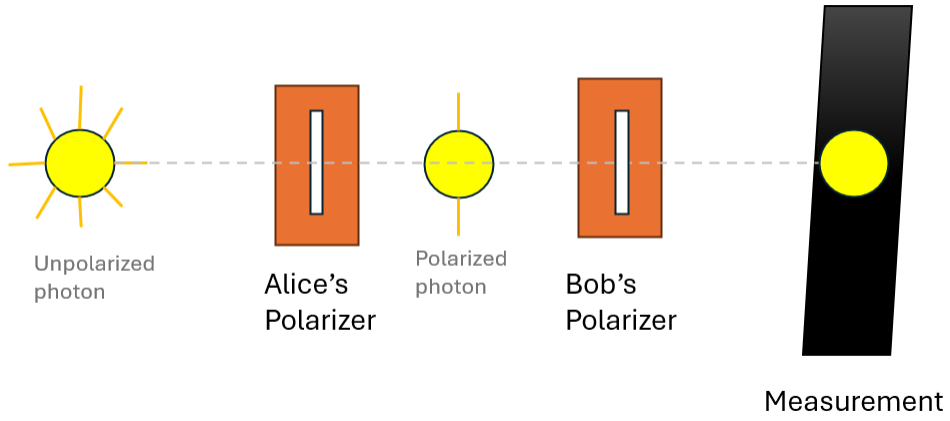

Figure 3. A secure BB84 quantum protocol with the same Alice's and Bob's basis.

2.1.2 Eavesdropper

When an eavesdropper, Eve, is present, the scenario becomes more complex. Eve intercepts the qubits sent from Alice to Bob and attempts to measure them to gain information about the key. However, the act of measurement disturbs the quantum states of the qubits, introducing errors into the system. Eve then sends modified qubits to Bob, hoping to remain undetected. However, since her measurements have disrupted the original states, Bob's measurements no longer match Alice's as consistently. When Alice and Bob compare their bases (b and c), they find discrepancies in their measurements, indicating the presence of an eavesdropper. By detecting these inconsistencies, Alice and Bob can take measures to ensure the security of their communication, such as discarding the compromised key and initiating a new key exchange process.

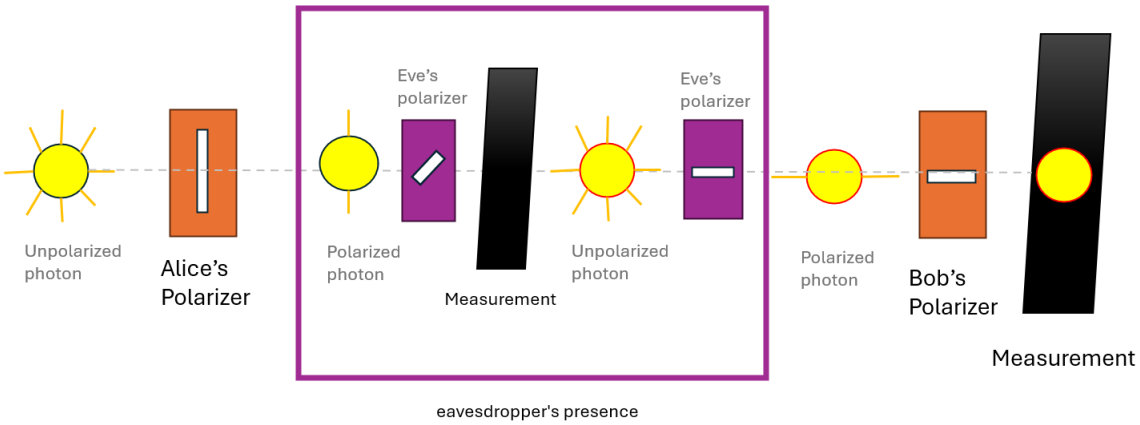

Figure 4. BB84 quantum protocol with eavesdropper's presence.

In other words, the idea is to encode every bit of the secret key into the polarization state of a single photon. Because the polarization state of a single photon cannot be measured without destroying this photon, this information will be 'fragile' and unavailable to the eavesdropper. Then, Eve will have to detect the photon, and then she will either reveal herself or will have to re-send this photon. But then **she will inevitably send a photon with a wrong polarization state.** This will lead to errors, and again **the eavesdropper will reveal herself.** For instance, in Figure 4, the polarization of the photon sent by Alice was vertical, but Eve did not know this and sent a photon with horizontal polarization, so when Alice and Bob compare their basis they will find an error, in this way, Eve will be detected.

The Github Repository contains the full implementation of the BB84 quantum protocol, including examples of both cases.

2.2 Encypt-Decrypt

The encrypt and decrypt tasks are functions utilized to transform a message using a specific key, in this way, the message is secure to share in a channel, and, in theory, only the person who has the key can decrypt it and read the original message. This is the way that currently cryptography works, all the sensible data that is on the internet has to be encrypted and then, decrypted once the message arrives at the destination. [BFMR22]

The encrypt and decrypt tasks consist of binary addition, for this, we need to transform the message into binary format, this is to transform regular messages like numbers or text into 0s and 1s. For instance,

Original Message	Binary Format Message
12345	11000000111001
"Google"	01000111011011110110111101100111011010110001100101
"HswrTY@K"	01001000011100110111011101110010010101000101100101000000001001011
"I am a future quantum leader (:"	0100100100100000011000010110110100100000011000010010000001100110011101010111010001110101011100100110010100100000011100010111010101100001011011100111010001110101011011010010000001101100011001010110000101100100011001010111000100100001000000010100000111010

Table 1: Original message (left) and its transform into binary format (right).

Binary addition

Binary addition is a fundamental operation in binary arithmetic, the number system used in computers and many digital communication systems. In binary addition, binary digits (bits) are added together like addition in the decimal system, but with only two possible values for each digit: 0 and 1. [FS03]

In binary addition, there are four possible combinations for adding two bits:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 0$$

Binary addition is used in cryptography to encrypt messages in various ways, including the following:

- Stream cipher: In a stream cipher, each bit of the original message is combined with a corresponding bit from a randomly generated bit sequence (keystream). Binary addition is used to combine the bits of the original message with the bits of the keystream, producing the ciphertext.
- Block cipher: In a block cipher, the message is divided into fixed-size blocks, and a cipher function is applied to each block individually. Binary addition may be part of this cipher function, combining the bits of the plaintext block with the bits of an encryption key.
- One-Time Pad (OTP): OTP is an encryption method where each bit of the original message is exclusively combined with a corresponding bit from an encryption key that is as long as the message. Binary addition is used to combine each bit of the message with the corresponding bit of the key, resulting in a ciphertext that is essentially impossible to decipher without the proper key.

In this project, we are using the One-Time Pad (OTP). This means that the key generated by the BB84 quantum protocol will have the same length as the message we want to encrypt/decrypt.

Functions
For those reasons, we define the following functions:

- $shorten\_key(data, key)$
- $encryption(data, key)$
- $decryption(data, key)$

2.2.1 $shorten\_key(data, key)$

The function $shorten\_key(data, key)$ ensures that the length of the key matches the length of the data before performing the encryption operation. If the length of the key is shorter than the length of the data, the key is repeated to match the data's length. If the key is longer than the data, it is simply returned unchanged.

Function 2.1:

Function $shorten\_key(data, key)$:

- The function adjusts the length of the key to match the length of the data before performing the encryption operation.
- If the length of the key is shorter than the length of the data:
  1. A repetition factor is calculated by dividing the length of the data by the length of the key and adding 1 to ensure sufficient length.
  2. The key is repeated this number of times and then truncated to match the length of the data.
- If the length of the key is greater than or equal to the length of the data, the key is returned unchanged.
- Finally, the function returns the adjusted key.

2.2.2 *encryption*(*data*, *key*)

Now, let's move on to the *encryption* function, which performs the encryption process on the data using the provided key. Here's the detailed description:

The *encryption*(*data*, *key*) function takes the data and key as input and produces the encrypted message using binary addition. If the length of the key does not match the length of the data, the *shorten_key* function is used to adjust the length of the key. Then, bitwise binary addition is performed between the data and the key.

Function 2.2:

Function encryption(data, key):

- This function takes the data and key as input and produces the encrypted message using binary addition.
- It checks if the length of the data matches the length of the key. If they do not match: The length of the key is adjusted using the *shorten_key* function to match the length of the data.
- Initializes a list to store the encrypted message.
- Performs bitwise binary addition between the data and the key and stores the result in the list of the encrypted message.
- Converts the list of the encrypted messages into a string and returns it as the result.

2.2.3 *decryption*(*data*, *key*)

Finally, let's describe the *decryption* function, which decrypts the message using the same logic as the *encryption* function. Since it utilizes the same binary addition operation as the *encryption* function, the *decryption* function simply calls the *encryption* function with the encrypted message and the provided key, resulting in the reconstruction of the original message.

Function 2.3:

Function decryption(data,key)

- This function recreates the original message using the same key used for encryption.
- Since the decryption operation is the same as the encryption operation (binary addition), it simply calls the *encryption* function with the encrypted message and the provided key.
- Returns the result of the *encryption* function, which is the original message.

3 Conclusion

In conclusion, the project to implement the BB84 protocol has been successful in generating secure keys for data encryption. We have demonstrated the effectiveness of this protocol by successfully encrypting targets ranging from simple five-digit passwords to words and even complete sentences. Quantum key distribution offers a highly secure way to generate keys that are only accessible to Alice and Bob, ensuring the confidentiality of communication between them.

Furthermore, the detection of any interception attempts becomes an essential feature of the BB84 protocol. If any discrepancies are detected between the keys generated by Alice and Bob, it can be inferred that a potential intruder is present. This allows users to know when a channel is not secure and take steps to protect their communication, such as resetting the key generation process. However,

we only proved this using Cirq's quantum simulator, this implies that there is no error for decoherence that may cause an error in Eve's detection.

As quantum computing technology advances, quantum cryptography will become a cornerstone of information security on the Internet. With its ability to ensure the security of communications even in the increasingly complex environment of the digital age, quantum cryptography promises to be a significant milestone in the evolution of computer security.

References

[BB84] G Brassard and Charles H Bennett. Quantum cryptography: Public key distribution and coin tossing. In *International conference on computers, systems and signal processing*, pages 175–179, 1984.

[BFMR22] Yuval Bloom, Ilai Fields, Alona Maslennikov, and Georgi Gary Rozenman. Quantum cryptography—a simplified undergraduate experiment and simulation. *Physics*, 4(1):104–123, 2022.

[FS03] Niels Ferguson and Bruce Schneier. *Practical cryptography*, volume 141. Wiley New York, 2003.