Institute of Business Administration

Department of Computer Science

Quantum Key Distribution Using the BB84 Protocol: A Simulation-Based Approach

Milestone 02

BB84 Protocol Implementation and Secure Communication

MOHAMMAD KAMIL (29464)

Supervisor:

Dr. Jibran Rashid

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this milestone, I focused on implementing the BB84 quantum key distribution protocol using Python and Qiskit, an open-source quantum computing framework developed by IBM that allows us to easily create, simulate and run quantum circuits. This was a continuation of the theoretical understanding I developed in the previous milestone, but the goal was to build and simulate the protocol step by step. BB84 is one of the most well-known and widely studied methods in quantum cryptography, and it offers a way for two parties to securely share a key by using the unique behavior of qubits.

One of the main strengths of the BB84 protocol is that it allows the detection of any third-party interference. If someone tries to intercept or measure the qubits during transmission, it causes disturbances that can be noticed later when Alice and Bob compare parts of their data. This property makes quantum key distribution fundamentally more secure than classical encryption methods. During this milestone, I also simulated an attacker (Eve) to see how much error her interference introduced, and I measured this using something called Quantum Bit Error Rate (QBER). The results clearly showed how the protocol reacts when eavesdropping occurs. Along with the core BB84 steps, I also added a secure key refresh mechanism using HMAC to make the implementation closer to real-world systems where keys are regularly updated for better security. Finally I used the key generated through BB84 to encrypt and decrypt a message using a one-time pad. This completed the cycle from secure key exchange to practical message encryption.

I used Qiskit in Google Colab throughout the implementation to write and test my code. I started from fundamental steps, like generating random bits and encoding them using quantum gates, and gradually moved towards more advanced concepts like error rate analysis, Eve simulation, and key-based encryption. I made sure to visualize the results wherever possible by generating circuit diagrams, retention graphs, and QBER comparisons. These visuals not only helped me verify my code but also gave me a clearer understanding of what was happening behind the scenes in each step.

This milestone was not just about getting the code to work but it is more about learning how quantum principles like superposition, basis measurement, and entanglement come together to form the foundation of secure communication. Seeing how even a small change in measurement basis or the presence of Eve affected the final key helped me appreciate the sensitivity and power of quantum cryptography. Overall this experience gave me a much deeper insight into how BB84 can be implemented from scratch and how it connects theoretical quantum concepts with practical, real-world applications in secure messaging.

Chapter 2

BB84 Protocol Simulation

In this chapter, I focused on building the core functionality of the BB84 quantum key distribution protocol. The main idea was to simulate how Alice and Bob could securely share a secret key using quantum bits and randomly chosen measurement bases.

To begin with I created different helper functions to generate random binary strings for Alice and Bob. These strings included the raw bits that Alice wanted to send and the basis choices for both parties. A '0' represented the Z basis (uses the standard vertical and horizontal polarization — $|0\rangle$ and $|1\rangle$), while a 1 represented the X basis (it includes the diagonal states — $|+\rangle$ and $|\rangle$). I used Hadamard gates in Qiskit to put the qubits into superposition and produce random outcomes. *Table 2.1* below shows a sample output from BB84 simulation, displaying the bits Alice prepared, her chosen encoding bases, and Bob's measurement bases for each qubit.

Index	Alice Bit	Alice Basis	Bob Basis
1	1	0	0
2	0	1	0
3	1	0	1
4	1	0	0
5	1	0	0
6	0	1	1
7	0	0	1
8	0	0	1

Table 2.1: Alice's bits and bases along with Bob's basis choices during BB84 simulation

Once the random bits and bases were generated, Alice encoded her bits using quantum gates. If her bit was 1, she applied an X gate; if her basis was X, she also applied a Hadamard gate. Bob then received each qubit and measured it using his randomly selected basis. If Bob's basis matched Alice's, the measurement gave the correct bit with high probability. If the bases didn't match, the result was essentially random.

After the measurements, Alice and Bob publicly compared their basis strings and kept only the bits matching their basis choices. These bits formed the sifted key, a shared sequence of 0s and 1s that could later be used for encryption. This process ensures that no information about the actual bits is shared directly, and any mismatch caused by an incorrect basis is naturally filtered out. In addition to the numeric outputs, I also generated a quantum circuit diagram using Qiskit to visually represent the full BB84 process. As shown in *Figure 2.2*, it includes Alice's encoding operations, the barrier, and Bob's measurements. The circuit helped

me verify that the qubits were being correctly prepared using X and H gates, and that Bob was measuring them based on his randomly chosen bases.

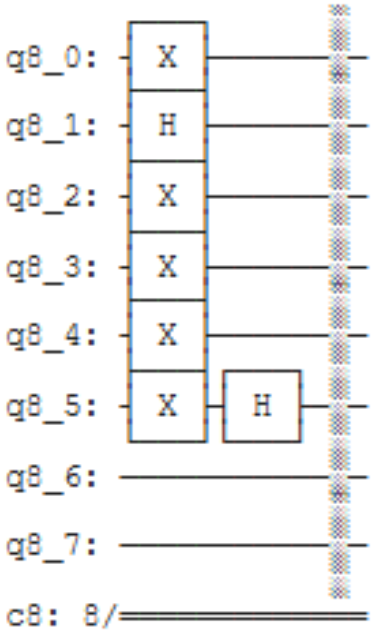

Figure 2.1: Quantum circuit showing Alice's encoding of qubits using X and H gates, followed by Bob's measurement using randomly chosen bases

To better understand and confirm that my implementation was correct, I also created a quantum circuit diagram. This circuit includes all the essential parts of the BB84 flow: Alice's preparation of qubits using X and H gates based on her random bits and bases, a barrier that represents the quantum channel, and Bob's measurement step.

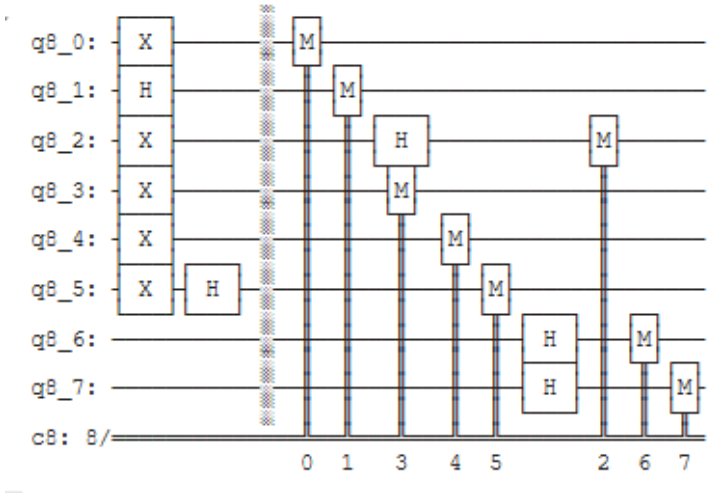

Figure 2.2: Complete BB84 circuit with Alice's encoding and Bob's basis dependent measurements

This chapter helped me understand how quantum randomness, measurement rules, and basis matching work together to enable secure key exchange.

Chapter 3

Secure Key Refreshing and Forward Secrecy

Once I have successfully generated the BB84 protocol, a secure shared key between Alice and Bob, the next goal was to make the communication system more realistic by introducing a safe way to update that key. Secure communication doesn't just depend on one good key, but it also relies on regularly refreshing that key to protect against long-term exposure or attacks. This is where the concept of forward secrecy becomes essential.

To address this, I implemented a method where the shared secret is updated after each round of communication. Instead of continuing to use the same key over multiple sessions, the key and a shared reference value were both refreshed using cryptographic techniques. This approach ensures that even if one key becomes known to an outsider, the previous and future keys remain safe and unrelated. Each communication round generated a new random looking string from the current key and seed value. This latest string was then used to create the next round's qubit bases. Once the round ended, the key and the seed were updated again before moving forward. This cycle repeated after every round, helping both parties maintain a continuously evolving shared secret that could not be reused or backtracked.

I recorded how the key and seed changed over three rounds to show this process more clearly. The *table 3.1* presents an example of this evolution using binary strings. Although the actual values are longer in the real simulation, they have been shortened here for simplicity.

Round	New Basis (T)	Updated Key (K)	Updated Seed (S)
1	1100101010010110	00101110100011110	1110001001001011
2	0110011001110001	1010101111000101	0101101010001100
3	1001110001100100	1110010110001010	0110111101110101

Table 3.1: Secure evolution of basis string, shared key, and seed across three communication rounds

This process gave the protocol a much stronger structure for secure communication over multiple sessions. Instead of relying on a one-time exchange, the shared secret was kept fresh and dynamic. Any interference or leak in one round would not compromise the others.

Working on this part of the project helped me realize that real security is not just about generating strong keys but also about how those keys are managed and updated over time. It added another important layer to the BB84 implementation and brought it closer to how modern secure communication systems are designed.

Chapter 4

One-Time Pad Encryption Using BB84 Key

Once I had generated a shared key using the BB84 protocol, I wanted to apply it meaningfully. To do this, I used the key to encrypt a short message using a method known as the One-Time Pad (OTP). This method is known for its perfect security when the key is truly random, used only once and kept completely secret between the sender and receiver.

Before moving to the encryption part it was important to ensure the generated key was long enough. Each character in a message requires 8 bits so a short key wouldn't be sufficient. To solve this I ran the BB84 simulation repeatedly and tracked how many bits were retained after the basis comparison between Alice and Bob. This process was repeated until the key had enough bits to match the message length I wanted to encrypt.

Once I had a suitable key, I encrypted the message by converting it into binary and then performing a bit-wise XOR with the key. The result was an encrypted binary string that looked completely random. To decrypt it, I applied the same XOR operation again with the same key, which perfectly recovered the original message.

For Exammple

- Original Message: `Hello quantum world!`
- Encrypted Binary: `100011110100110011000010101O...`
- Decrypted Message: `Hello quantum world!`

To understand how often the BB84 protocol gives a usable key, I ran 100 trials and recorded the retention rate, the percentage of bits that were kept after basis comparison. The results showed that about 50% of the bits were typically retained, which is in line with the expected behavior of BB84 as shown in *Figure 4.1*. The graph below shows how the retention rate varied across these 100 trials.

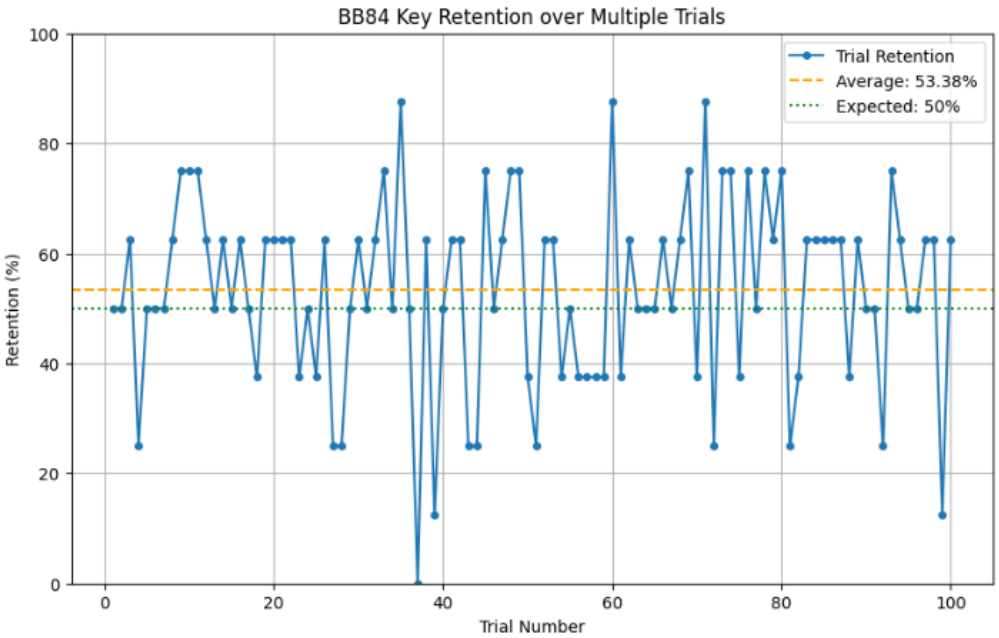

Figure 4.1: Retention rate of key bits across 100 BB84 trials

This step helped to connect everything i had worked on so far into something practical and meaningful. Seeing the BB84 protocol go from theoretical steps and circuit diagrams to actually encrypting a real message made the whole concept feel more complete. It showed me how the key generated through quantum communication isn't just random data — it can be used directly to secure messages in a way that's mathematically unbreakable, as long as the rules of the One-Time Pad are followed.

I learned it's not always about getting a perfect key on the first try. Sometimes, the BB84 simulation doesn't give a key long enough for the message, but that's expected due to how basis matching works. Still, getting a usable key didn't take many tries, and the process felt efficient overall. What made a difference was visualizing the retention rate. That graph in front of me clearly showed how much of the key was being kept in each trial. It helped me understand what to expect from the protocol and gave me confidence that the system was working as intended.

Chapter 5

QBER Comparison: Detecting Eve Through Error Analysis

One of the most fascinating parts of this project was testing how well the BB84 protocol could detect the presence of an eavesdropper. In theory, BB84 promises to expose any third-party interference simply by looking at the error rate between the sender and receiver. To put this theory to the test, I conducted a comparison using repeated trials , both with and without Eve attempting to intercept and resend the qubits.

In each trial, I calculated the QBER, which shows how many bits were mismatched after Alice and Bob applied their basis filters and kept only the bits with aligned bases. When Eve is not part of the process, the QBER stays low, usually well below 10%, since most mismatches come only from random basis differences. But once Eve tries to measure the qubits and resend them without knowing the correct basis, the error rate starts rising sharply.

To clearly see this effect, I ran multiple trials and plotted the QBER values on a graph. In *Figure 5.1*, the green line shows the QBER values when there is no eavesdropping, while the red line shows the values when Eve is active. A dashed line is included in the graph to mark the approximate threshold where interference becomes statistically noticeable.

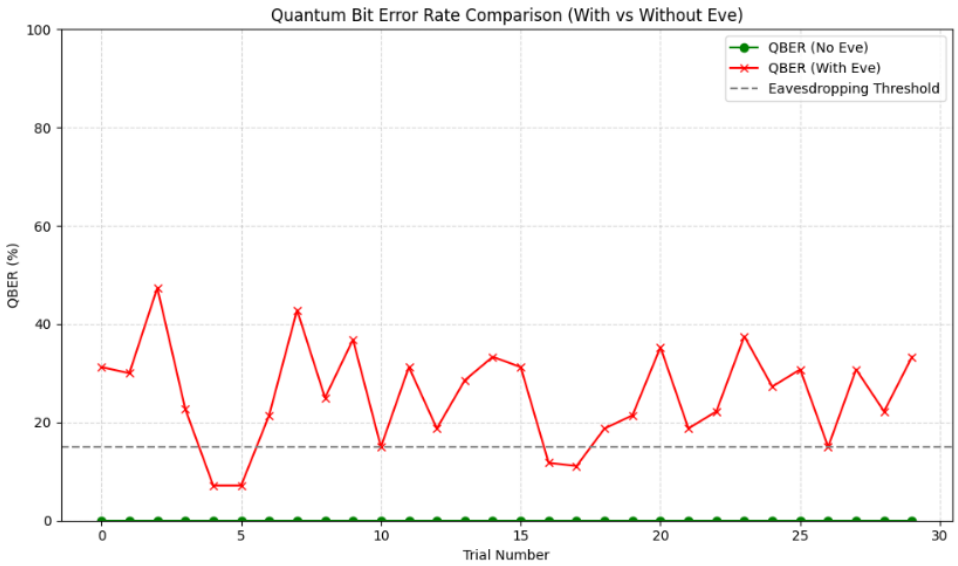

Figure 5.1: QBER comparison across multiple trials

The results I observed from this comparison were not only noticeable but also extremely

consistent across all trials. In the case where no external interference occurred, the QBER remained low and stable. This made sense as most discrepancies in such trials came only from natural mismatches in measurement bases between Alice and Bob, which are expected and filtered out during the sifting process. On average, the QBER without eavesdropping stayed below 10%, often even lower.

However, when i simulated Eve's presence someone trying to intercept the quantum communication by measuring each qubit and resending it — the difference was dramatic. The QBER values immediately increased, frequently exceeding 25% and sometimes approaching 40%. This sharp rise in the error rate was not random; it directly resulted from Eve's inability to guess the correct basis for each qubit. Since quantum measurements disturb the state of a qubit if done incorrectly, Eve's interference consistently introduced noise into the transmission, and this was reflected clearly in the measurement results on Bob's side.

Visualizing this difference through plotted graphs made the impact even more obvious. The QBER plots as shown in *Figure 5.1*, visually demonstrate how sensitive BB84 is to eavesdropping. Even though Eve operates silently without announcing her presence, the quantum nature of the protocol ensures that her actions cannot go unnoticed. The sudden shift in the error rate acts as a signal that the transmission is no longer secure, allowing Alice and Bob to immediately discard the affected session and avoid using the compromised key.

This makes BB84 incredibly powerful and trustworthy in real-world scenarios where data sensitivity and security are critical. Understanding and simulating this behavior helped bridge the gap between theoretical learning and practical confidence. I now see clearly how BB84 can generate secret keys and act as an early warning system for detecting unwanted observers. That level of reliability is what makes quantum cryptography such an exciting and promising area of study.

Chapter 6

Conclusion

In this milestone i have move beyond theoretical understanding and actually implement a complete quantum key distribution system based on the BB84 protocol. It was a hands-on opportunity to explore how quantum properties can be used not only to exchange secret keys, but also to detect interference in a way that classical cryptography cannot offer. Starting from random bit generation and basis selection, I simulated Alice's encoding process, the qubit transmission, and Bob's measurement , all using quantum circuits created in Qiskit.

One of the most valuable parts of the experience was observing how basis mismatches naturally filtered out incorrect measurements, which led to the formation of a sifted key. By running multiple simulations, I also realized that the protocol usually retains around half of the initial bits after filtering and this percentage was confirmed through visualizations such as the retention rate plot. Another key learning moment came when I introduced an eavesdropper into the simulation. By simulating Eve's interference — even though she simply measured and resent the qubits without altering their meaning, the error rate between Alice and Bob increased sharply. This confirmed what I had learned in theory: that quantum systems are inherently sensitive to measurement, and any third-party involvement leaves a visible trace. The QBER comparison graph clearly showed this pattern and demonstrated how BB84 can act as both a key generation tool and a security detection mechanism.

I also extended the protocol by implementing a secure key refreshing system using hash-based logic. This allowed the shared key and seed to evolve after each round, improving the long-term security of the communication. Using the one-time pad method, I applied the final shared key in a real encryption task. This brought everything together from key creation to message protection, and gave me a complete picture of how quantum-generated keys can be used in practice.

In the final milestone, I will focus on optimizing and polishing the entire project. This will include cleaning and organizing the code for better readability, improving the circuit structure and output formatting, and refining visualizations for clarity.I will also implement all the changes suggested by my supervisor. I will also add short conceptual exercises and record a brief presentation video that walks through the key parts of the project. The goal is to make the final version more efficient, well-explained, and presentation-ready