Securing Communications with the BB84 Protocol on the IBM Quantum Platform

As computer science students, our work usually involves classical computing, which is based on bits and logic gates. However, the field of computing is expanding to include the principles of quantum mechanics. For this project, we were tasked with implementing the BB84 quantum key distribution protocol using IBM's Quantum Platform and the Qiskit library. This article describes our project, from the basic problem of secure communication to our experience running code on a real quantum computer.

Part 1: The Problem of Secure Communication

Encryption: Locking Digital Information

Imagine two people, Alice and Bob, who need to communicate over a public channel like the internet, where an eavesdropper, Eve, might be listening. To protect their messages, they use encryption—the process of converting readable information (plaintext) into an unreadable format (ciphertext). Reversing this process requires a secret piece of information: a key.

A common method is symmetric-key cryptography, where the same key is used to both encrypt and decrypt a message. This method is fast and efficient, but it has one critical vulnerability: the key itself. If Alice uses a specific key to encrypt a message, Bob must have the exact same key to decrypt it.

The Key Distribution Problem

The security of this entire system depends on keeping the key secret. If Eve gets a copy of the key, the encryption is useless. This creates the **key distribution problem**: How can Alice and Bob agree on a shared secret key without an eavesdropper intercepting it?

They cannot simply send the key over the public channel, as Eve could copy it. Meeting in person to exchange keys is often impractical, and using a trusted courier is slow and has its own security risks. The challenge is to establish a shared, secret key over an insecure channel in a way that ensures no one else has a copy. This is the first and most critical step for any secure communication system.

Part 2: The Classical Approach and Its Limitations

Classically, this problem was effectively solved with **public-key cryptography**, which allows Alice and Bob to establish a shared secret by communicating entirely in the open.

The Classical Solution: Security by Computational Hardness

Classical key exchange algorithms, such as Diffie-Hellman or RSA, don't hide the information being exchanged. Instead, their security relies on so-called **one-way functions** - mathematical operations that are easy to perform in one direction but extremely difficult to reverse. For example, it is simple to multiply two large prime numbers, but given only the product, finding the original prime factors is a task considered too difficult to solve in any practical amount of time, even for the most powerful supercomputers.

The security of our global digital infrastructure is therefore based on the assumption that these mathematical problems will remain hard to solve with current and foreseeable computing capabilities, rather than on a proven fundamental law.

The Quantum Threat

This assumption is now under threat by the development of quantum computers. In 1994, mathematician Peter Shor developed a quantum algorithm that can, in principle, solve both the prime factorization and discrete logarithm problems efficiently. Should large-scale quantum computers become a reality, the mathematical foundations of our current secure communication systems would become vulnerable.

This is where Quantum Key Distribution (QKD) offers a fundamentally different approach. Instead of relying on assumed computational difficulty, the security of QKD is guaranteed by the laws of physics. The core principle is that an observer cannot measure a quantum particle without risking a change to its state. Specifically, if an eavesdropper (Eve) intercepts a qubit and measures it using a different **basis** than the one it was prepared in, her measurement will alter the qubit's state. This change introduces detectable errors into the transmission. The guarantee of QKD is not that Eve cannot listen, but that any attempt to listen will almost certainly be detected.

Part 3: The BB84 Algorithm Explained

The (first) QKD protocol, BB84, was developed in 1984 by Charles H. Bennett and Gilles Brassard. It provides a **fundamentally different solution** to the key distribution problem by using the properties of quantum bits, or **qubits**.

The Quantum Foundation: Qubits and Bases

While a classical bit can only be a 0 or a 1, a qubit can exist in a **superposition** of both states simultaneously. The BB84 protocol uses two different sets of measurement settings, called **bases**.

A helpful analogy is to think of these bases as different polarizing filters, like the lenses in polarized sunglasses. Each filter is oriented to check for a specific alignment of light. After the light passes through the filter, it is aligned accordingly. This is the effect of the measurement.

1. **The Z-basis {|0⟩, |1⟩} (also called the computational basis)**: Analogous to a filter oriented to check for vertical and horizontal polarization.
2. **The X-basis {|+⟩, |-⟩}**: Analogous to a filter tilted 45 degrees to check for diagonal polarization.

Alice uses these bases to encode classical bits into quantum states:

- In the Z-basis, the bit 0 is encoded as |0⟩ and 1 is encoded as |1⟩.

- In the X-basis, the bit 0 is encoded as |+⟩ and 1 is encoded as |-⟩.

The protocol's security is based on a key principle of quantum mechanics:

- If you measure a qubit in the **same basis** it was prepared in, you get a definite answer.
- If you measure a qubit in the **wrong basis**, the outcome is random. For example, a qubit prepared as |0⟩ and measured in the X-basis has a 50% chance of being measured as |+⟩ and a 50% chance of being |-⟩. This measurement forces the qubit into the new, random state, destroying the original information.

This measurement disturbance is crucial. An eavesdropper cannot measure a qubit without risking a change in its state, which would reveal her presence.

The BB84 Protocol: A Step-by-Step Guide

Assume that Alice and Bob have two communication channels. The first is a classic non-private bidirectional channel with an authentication guarantee - Eve can intercept messages but cannot alter them or impersonate Alice or Bob. The second is a quantum channel in which Eve can interact with the qubits, including reading them.

Step 1: Alice Sends Qubits Alice generates two random sequences of classical bits: one containing the data (the potential key) and the other indicating which basis to use for each bit. For each bit, she prepares and sends a qubit according to her data and basis choice as mentioned above.

Step 2: Bob Measures the Qubits Bob receives the qubits but doesn't know which basis was used for each one. He generates his own random sequence of basis choices and measures each qubit accordingly. He then saves said measurements into his own sequence of data bits. If Bob does not receive the entire sequence of qubits, he terminates the protocol.

Step 3: The Sifting Process (The Public Discussion) Alice and Bob send each other their sequences of basis choices over the classical channel. They keep the data bits only for the positions where their bases match and discard all others. With high probability each of them ends up with a data sequence half the size of the original one. The resulting sequence is called the **sifted key**.

Step 4: Eavesdropper Detection To check if Eve was listening, Alice and Bob must look for errors. If Eve intercepted the qubits, she would have had to guess the bases to measure them. On the positions where she guessed the wrong basis, her measurement would alter the qubit's state. This means that even in the places where Alice and Bob's bases matched, Eve's interference could cause Bob's measurement to be different from Alice's bit.

To detect this, Alice randomly chooses s bits out of her remaining data sequence and sends them (along with their corresponding indices) to Bob. He then compares them to his data bits in the same indices.

- If the bits all match, they can be confident no one was listening. In this case they discard said s bits and use the remaining confirmed bits to form their final, secure key.
- Otherwise, they conclude that an eavesdropper was present, discard the key, and start over.

As Eve has a 50% chance to choose the wrong basis, and as in that basis she has a 50% chance to get a wrong measurement, then the probability of Eve to read incorrectly is $\frac{1}{4}$. Hence the probability for all the s bits of Alice and Bob's data sequences to match is $(\frac{3}{4})^s$. Ideally, we want to choose s that is sufficiently large so eavesdropping detection is almost certain, but small enough so there remains enough bits to make the key secure.

Part 4: Implementation and Reflections

From Theory to Qiskit Code: Crafting the Circuits

We built the protocol in a Jupyter notebook using Qiskit, a quantum computing library by IBM. A key feature of the BB84 protocol is its reliance on single, independent quantum transmissions. To simulate this, our implementation constructed a unique quantum circuit for every single bit Alice sent to Bob. Each circuit contained one qubit and one classical bit for the result.

Preparing the quantum states involved using a sequence of quantum gates. For example, a qubit starts in the $|0\rangle$ state. To encode a '1' in the Z-basis, we applied an **X gate** (quantum NOT gate). To prepare states in the X-basis, we used an **Hadamard (H) gate**, which creates a superposition, followed by a **Z gate** if the bit was a '1'. Bob's measurement process used a similar logic, applying an H gate before measurement to correctly read qubits sent in the X-basis.

Finally, we ran each circuit with shots=1. This instruction is critical, as it tells the system to execute each circuit only once. This mimics the real-world scenario where measuring a quantum state is a unique event that consumes the state.

A Successful Key Exchange: Simulator Results

We first ran our code on a local, noise-free simulator. For a reproducible test (using a fixed seed of 42), we aimed to generate a 100-bit key, using 10 additional bits to check for eavesdropping (n=100, s=10).

The results demonstrated a successful execution of the protocol:

1. **Raw Transmissions:** To generate the required 110 bits for the key and integrity check, the protocol started with a much larger set of qubits. Our implementation processed an initial batch of **1024** transmissions.
2. **Sifting:** After Alice and Bob compared their basis choices, **515** bits remained in the sifted key. This is very close to the expected 50% retention rate, reflecting the randomness of their choices.

3. **Integrity Check:** Alice and Bob then publicly compared 10 randomly chosen bits. On the noiseless simulator, the bits matched perfectly. This resulted in a **Quantum Bit Error Rate (QBER) of 0.0**, giving them confidence that no eavesdropper was present.
4. **Final Secure Key:** After the integrity check, the 10 sample bits were discarded. The first 100 of the remaining bits were taken as the final, secret key.

This successful run on the simulator confirmed our implementation was correct. The next step was to run the code on actual quantum hardware through the IBM Quantum Platform. This allowed us to observe the primary challenge of current quantum computers: **noise**. Real qubits are sensitive to their environment, and factors like temperature fluctuations or electromagnetic fields can corrupt their delicate quantum states, a phenomenon known as **decoherence**. This "noise" introduces errors into the computation.

When we ran our BB84 implementation on the ibm_torino hardware, one particular run yielded a significant error rate. In the integrity check phase, the Quantum Bit Error Rate (QBER) was measured at 10%. Because this value exceeded our 2% security threshold, our program deliberately aborted the key generation process by raising a ValueError. This result is a successful demonstration of the protocol's security feature, not a failure. The elevated QBER is a known characteristic of today's noisy quantum hardware. While other runs might yield a lower, acceptable QBER, this specific outcome highlights the protocol's ability to detect insecure channel conditions and prevent the creation of a compromised key. This single run gave us a firsthand appreciation for the difficulty of building fault-tolerant quantum computers.

This project was a practical introduction to the future of computation and security. We learned that the principles of quantum mechanics are not just abstract theories but emerging engineering tools as well. **For us, a key part of the learning process was translating these abstract principles into functional circuits using the Qiskit library.** While the road to large-scale, fault-tolerant quantum computing is long, projects like this provide a direct look into the challenges and potential of this transformative technology. Although the project itself was focused, it was fascinating to see the depth and diversity of the topics surrounding quantum computation, from number theory and linear algebra to cryptography and fundamental physics.

Acknowledgments

The learning and work presented in this project and article were built upon several key resources. We would like to acknowledge the lectures and summaries of Professor Shay Mozes, which provided the foundational knowledge for our work. We also consulted Wikipedia and other public sources for general information on the principles of the BB84 protocol and quantum key distribution.