BB84 Protocol : A Quantum Key Distribution Demonstration

Nick Coffman

October 2, 2024

1 Introduction

During the early years of the still largely theoretical domain of quantum computing, it had been become apparent that quantum mechanics could bear a significant impact on the information security. Classical cryptography, which had long relied on mathematical complexity to ensure security, had then been made to face a potentially severe vulnerability under the light of the computational power of quantum technologies. These emerging capabilities posed a substantial threat to the cypersecurity protocols in use at the time, which were designed for classical computers and were incapable of withstanding the exponential power of quantum algorithms.

Against this potential threat, Charles Bennett of IBM and Gilles Brassard of the University of Montreal authored their landmark paper, **"Quantum Cryptography: Public Key Distribution and Coin Tossing."**, wherein they had proposed the **BB84 protocol**. This protocol was a groundbreaking advancement in the field of **quantum cryptography**, proposing means of securely distributing cryptographic keys between two parties using principles of quantum mechanics. It had taken its name from the initials of its authors and of the year of its conception. The BB84 protocol enables two parties, typically referred to as "Alice" and "Bob", to securely share a cryptographic key over an insecure communication channel. Among the protocol's key innovations is its ability to detect the presence of any eavesdropper (in the paper referred to as "Eve"), through the disturbance caused in the measurement of quantum phenomena.

The BB84 protocol is considered one of the first practical applications of quantum mechanics in the field of cryptography. It utilizes the quantum properties of superposition and measurement in different bases to ensure that any attempt to intercept the communication will introduce detectable anomalies. By comparing a portion of their data, Alice and Bob can verify the integrity of the shared key and discard any information that may have been compromised. This ensures that the remaining key is secure for cryptographic purposes.

This document presents a practical demonstration of the BB84 protocol using IBM's **Qiskit** library, a powerful tool for simulating and executing quantum circuits on classical hardware. Through this demonstration, both scenarios - with and without an eavesdropper - are explored to highlight how the protocol operates and how the presence of an eavesdropper can be detected. The demonstration also illustrates the core concepts of **quantum key distribution (QKD)** and the unique benefits of quantum cryptography in securing communication against future quantum-enabled threats.

---

2 Initialization and Circuit Creation

In the BB84 protocol, Alice's first step is to prepare a set of quantum states that will be transmitted to Bob. These quantum states - often photons, but any qubit system can be used - are encoded based on randomly generated bits and measurement bases. The randomly chosen bit values represent Alice's message, while the randomly chosen measurement bases (either standard or Hadamard) introduce uncertainty that is key to ensuring the security of the protocol.

Below is a Python implementation that simulates this initial step using Qiskit's `AerSimulator`.

```
simulator = AerSimulator()

def create_bb84_circuit(bits, bases):
        num_qubits = len(bits)
        qc = QuantumCircuit(num_qubits, num_qubits)
        for i in range(num_qubits):
                if bases[i] == 1:
                    qc.h(i)
                if bits[i] == 1:
                    qc.x(i)
        if bases[i] == 1:
                    qc.h(i)
        qc.barrier()
        return qc, bits, bases
```

In this code, we initialize the `AerSimulator`, a powerful tool from Qiskit that allows quantum circuits to be simulated on classical hardware. This is essential for simulating quantum behavior without access to a physical quantum computer. The function `create_bb84_circuit` represents Alice's preparation of the quantum states, which she will transmit to Bob. The function encodes the qubits based on Alice's randomly selected bits

and measurement bases.

Breakdown of the code:

• **Quantum Circuit Setup:** Alice prepares a quantum circuit `(qc)` that contains a number of qubits equal to the length of her bit array. Each qubit is initialized in the $|0\rangle$ state by default.

• **Basis Selection:** For each qubit, Alice randomly selects a measurement basis - either the standard (Z) basis or the Hadamard (X) basis. If Alice chooses the Hadamard basis `(bases[i] == 1)`, she applies a Hadamard gate `(qc.h(i))` to put the qubit into superposition, allowing it to be measured in the diagonal basis ($|+\rangle$ and $|-\rangle$).

• **Bit Encoding:** Next, Alice encodes her message by flipping the qubit using the X gate if her randomly chosen bit is 1. The X gate, also known as the quantum NOT gate, flips the state of the qubit from $|0\rangle$ to $|1\rangle$ (or vice versa).

• **Final Hadamard Application:** If the qubit was prepared in the Hadamard basis, Alice reapplies the Hadamard gate to ensure the qubit is in the correct superposition state before sending it to Bob. This step is crucial for measuring the qubits in the same basis Bob will randomly choose later.

• **Barrier:** The barrier is added to visually separate the stages of the circuit and helps when visualizing the quantum operations in a circuit diagram.

In this step, we simulate one of the most important features of the BB84 protocol: the random selection of bits and measurement bases. This randomness is central to ensuring the security of the protocol, as both the message Alice wants to sent (encoded in bits) and the basis she chooses to encode those bits are chosen unpredictably.

Below is the Python code that generates the required random bits and bases for Alice and Bob.

```
num_qubits = 10
alice_bits = np.random.randint(2, size=num_qubits)
alice_bases = np.random.randint(2, size=num_qubits)
bob_bases = np.random.randint(2, size=num_qubits)
```

Breakdown of the code:

• **Number of Qubits:** The variable `num_qubits = 10` defines the number of qubits Alice will send to Bob. In this case, Alice is preparing 10 qubits, meaning that 10 bits of information will be transmitted. The number of qubits can be adjusted based on the needs

of the simulation, but 10 is used here for clarity and simplicity.

• **Random Bit Generation:** Alice generates a random sequence of bits, stored in the array `alice_bits`. Each bit in this array is either `0` or `1`. These bits represent the classical information Alice wants to encode and send to Bob using qubits. This randomness ensures that the information is unpredictable and secure.

• **Random Basis Selection for Alice:** Alice also randomly chooses a measurement basis for each bit she will encode. This is stored in the array `alice_bases`, where each value is either `0` (representing the standard basis, or Z-basis) or `1` (representing the Hadamard basis, or X-basis). These randomly chosen bases will determine how Alice encodes her bits into qubits. The variation in bases helps prevent any potential eavesdropping from going undetected.

• **Random Basis Selection for Bob:** Similarly, Bob randomly selects his own measurement bases for each qubit he receives from Alice, stored in the array `bob_bases`. Just like Alice, Bob will either use the standard basis (`0`) or the Hadamard basis (`1`) for his measurements. Since Bob doesn't know which basis Alice used to prepare each qubit, he also has to randomly choose a basis for each qubit when performing his measurements.

The Importance of Randomness: The random generation of bits and bases plays a critical role in the BB84 protocol's security. Alice and Bob do not initially know whether their bases match, which introduces uncertainty. Once Alice and Bob have completed the transmission, they will communicate over a public classical channel to compare their bases. Any qubits where their bases match will be used to generate the final shared key, while those where the bases do not match will be discarded.

This process ensures that even if an eavesdropper (Eve) were to intercept some of the qubits, the key generated will remain secure, as Eve's actions would introduce detectable errors if she guessed the wrong basis.

---

3 Simulation Without Eavesdropping

In this simulation, we demonstrate the BB84 protocol in the absence of an eavesdropper (Eve). Alice prepares the qubits according to her randomly chosen bits and bases, which she then sends to Bob over a quantum channel. Bob, in turn, measures the qubits using his own randomly selected bases. This process is visualized in the quantum circuit below.

Below is a visualization of the quantum circuit in its initial preparatory stage.

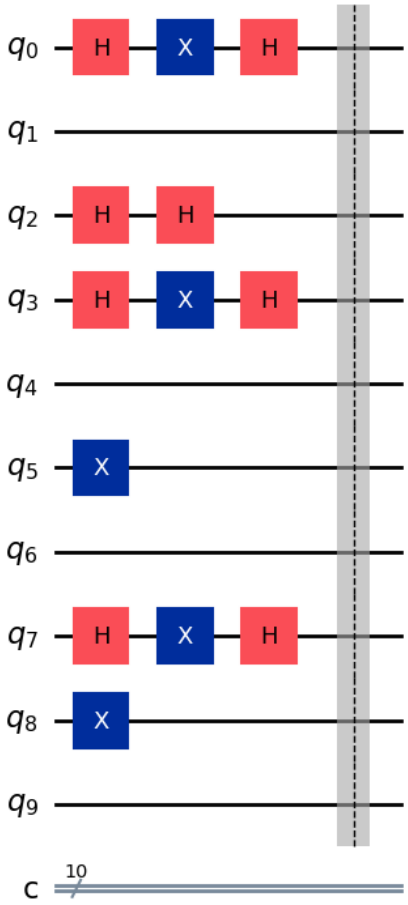

Figure 1: Visualization of the quantum circuit representing the qubits Alice sends to Bob. Each row corresponds to a different qubit, with Hadamard gates (H) and X gates applied based on Alice's chosen bits and bases. For instance, for qubit q_0, Alice selects the X-basis (represented by a Hadamard gate), flips the bit to 1 using the X gate, and applies another Hadamard gate to finalize the qubit for measurement in the same basis.

Breakdown of the circuit:

• **Qubit Initialization:** Each row in the circuit diagram corresponds to a single qubit, which Alice has prepared to send to Bob. The qubits start in the standard basis state $|0\rangle$, and Alice modifies them based on her chosen bits and measurement bases.

• **Hadamard (H) Gate:** Alice applies the Hadamard gate to qubits when she selects the Hadamard (or X) basis for that qubit. The Hadamard gate places the qubit into superposition, which is essential for encoding the bit in the non-standard basis. In the diagram, the Hadamard gate is applied to qubits such as q_0, q_2, and q_3, indicating that Alice has chosen the Hadamard basis for those qubits.

• **X Gate (Bit Flip):** For qubits where Alice's randomly generated bit is `1`, she applies the X gate, which flips the state of the qubit from $|0\rangle$ to $|1\rangle$. This operation is visible on qubits q_0, q_3, q_5, and q_7, where the X gate modifies the state to represent Alice's bit choice.

• **Hadamard Gate (Again):** After flipping the bit (if necessary), Alice applies another Hadamard gate to any qubits that are prepared in the Hadamard basis. This final Hadamard gate ensures that the qubits are ready to be sent in the chosen basis, aligning with Alice's encoding. For example, on qubit q_0, the Hadamard gate is applied again after the X gate, ensuring that the qubit is fully prepared for measurement.

Observing the Process: In this diagram, we observe Alice's preparation of the qubits, showing how each qubit is manipulated based on her random selection of bits and bases. Once prepared, Alice sends these qubits to Bob for measurement. Bob will then apply his own randomly chosen measurement bases to attempt to extract Alice's bits.

Next, we arrive at the critical stage in the BB84 protocol where Bob measures the qubits he receives from Alice. Since Bob does not know in which basis (either **Z-basis** or **X-basis**) Alice prepared each qubit, he must randomly select a basis for his measurements. The `measure_in_bases` function below simulates this process. The `bases` array passed to the function represents Bob's randomly selected bases. These are either:

• **0**: Bob will use the **Z-basis** (standard computational basis, which distinguishes between the states $|0\rangle$ and $|1\rangle$), or

• **1**: Bob will use the **X-basis** (Hadamard basis, distinguishing between the superposition states $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$).

Here's the code simulating Bob's measurement:

```
def measure_in_bases(qc, bases):
    for i, base in enumerate(bases):
        if base == 1:
            qc.h(i)
        qc.measure(i, i)
    qc.barrier()
```

• If Bob chooses the **X-basis** (i.e., if `bases[i] == 1`), he applies a Hadamard gate `qc.h(i)` to switch the qubit from the **Z-basis** (computational basis) to the **X-basis** (Hadamard basis) before measuring. This transformation is necessary to align the qubit with the basis in which Bob is measuring.

• After the appropriate transformation (or none if Bob is using the **Z-basis**), Bob measures the qubit by calling `qc.measure(i, i)`. This measurement collapses the

quantum state into a classical bit (0 or 1), and the outcome is recorded in the corresponding classical register.

• The barrier `qc.barrier()` ensures that no operations beyond this point interfere with the measurement process.

Once Bob measures all the qubits, he obtains a set of classical bits. However, at this stage, he does not yet know whether his chosen measurement bases align with those that Alice used to prepare the qubits. If Alice and Bob used the same basis for a given qubit, Bob's measurement result will correctly reflect Alice's prepared bit. If the bases differ, Bob's result is effectively random, and that qubit will be discarded later during the key sifting process.

The diagram below provides a visual representation of Bob's measurement process. Each row corresponds to a qubit sent by Alice, and the gates applied to each qubit are shown. After receiving the qubits, Bob measures them using either the **Z-basis** or **X-basis**, depending on his randomly chosen measurement basis.

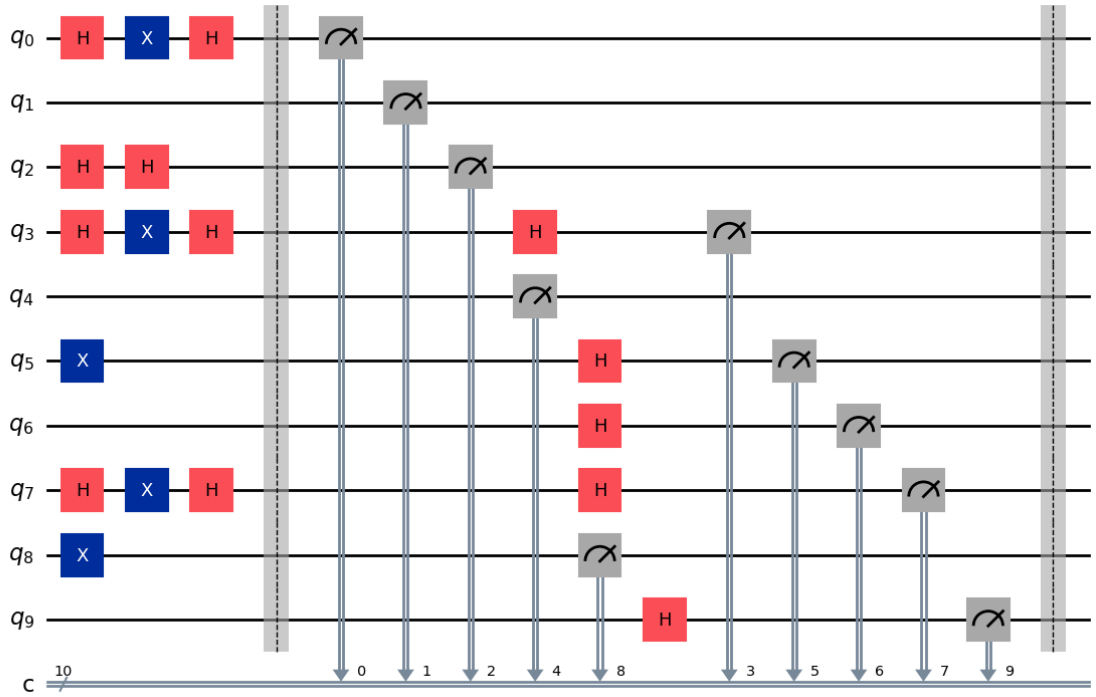

Figure 2: Visualization of Bob's measurement process. The qubits received from Alice are measured by Bob using either the **Z-basis** or the **X-basis**. Measurement gates (represented by the symbols with arrows) are applied at the end of each qubit's circuit. Depending on Bob's chosen measurement basis, he may apply a Hadamard gate (H) before measurement if measuring in the **X-basis**. The results of these measurements, either 0 or 1, are stored in classical bits.

In this diagram, the rows labeled `q0` to `q9` represent the ten qubits Alice sent to Bob. Bob's chosen measurement basis is indicated by the presence of Hadamard gates (H) for the **X-basis** or no additional gate for the **Z-basis**. After the measurement gates (shown by meter symbols) are applied, the qubits collapse into classical bits, which are stored in the corresponding classical registers, either as 0 or 1.

The function, `sift_and_visualize`, implements the key sifting process, a critical step in the BB84 protocol. This step occurs after Alice and Bob have completed their quantum measurements. To generate a shared secret key, Alice and Bob publicly compare the measurement bases they used for each qubit, while keeping the actual measurement results private. By comparing the bases, they can determine which qubits were measured using the same basis. Only the bits corresponding to these matching bases will be retained for the final secret key.

```python
def sift_and_visualize(bits, bases, bob_bases):
    sifted_key = []
    matching_bases = []
    for i in range(len(bases)):
        if bases[i] == bob_bases[i]:
            sifted_key.append(bits[i])
            matching_bases.append(i)
    return sifted_key
```

In the code above:

• `sift_and_visualize` accepts three arguments:

• `bits`: The classical bit values obtained from Alice's qubits.

• `bases`: The bases that Alice used when preparing her qubits.

• `bob_bases`: The bases that Bob randomly chose when measuring Alice's qubits.

• The function compares the bases used by Alice and Bob for each qubit. If Alice's basis matches Bob's (`if bases[i] == bob_bases[i]`), the corresponding bit is added to the `sifted_key`. The `sifted_key` stores the bits that were measured using the same basis, and these bits form the secret key that Alice and Bob can securely share.

• The indices of the matching bases are also recorded in `matching_bases`, though this is not returned here (but could be useful for visualization or debugging).

This key sifting process is essential to ensure that both Alice and Bob have identical bits, forming the foundation of their shared cryptographic key. Bits corresponding to mismatched bases are discarded since they introduce randomness and cannot be guaranteed to match.

At this stage in the BB84 protocol, Alice and Bob have completed the quantum transmission and measurement steps. The key-sifting process, which involves comparing the measurement bases used by both parties, has taken place, and now the matching bits are retained to form the final secret key. Below are the results of the key-sifting process in a scenario without eavesdropping.

In this table:

• **Alice's Bases** and **Bob's Bases** show the randomly chosen bases used by Alice to prepare the qubits and by Bob to measure them.

Alice's Bases:	[1 1 0 1 1 0 1 1 0 1]
Bob's Bases:	[0 1 0 0 1 1 0 1 1 0]
Matching Bases:	[2 4 7]
Sifted Key:	[1 1 0]

Table 1: Key-sifting process without eavesdropping. Alice and Bob compare their bases, and for the positions where their bases match (positions 2, 4, and 7), the corresponding bits are retained to form the final shared key.

• **Matching Bases** lists the positions where Alice and Bob's bases coincide. For example, in this instance, their bases match at indices 2, 4, and 7.

• **Sifted Key** represents the bits retained from Alice's original transmission, based on the matching bases. This forms the final shared key that can now be used for secure communication.

After performing the sifting process, Alice and Bob are left with their **final secret key**. Since there was no eavesdropping in this instance, the bits Alice sent and the bits Bob measured, where their bases matched, should be identical. These final bits form the secure shared key.

Initial Bits:	[0 1 1 0 1 0 1 0 1 1]
Final Bits:	[1 0 0]

Table 2: Comparison of Alice's initial bits and the final bits after sifting. The **Initial Bits** are Alice's original bit sequence, and the **Final Bits** represent the result of Bob's measurement where their bases matched. Since there was no eavesdropping, Alice and Bob's bits match perfectly for the sifted positions.

In this table:

• **Initial Bits** are the original bits that Alice encoded into the qubits.

• **Final Bits** are the bits that Bob measured after receiving the qubits and matching his bases with Alice. In a scenario without eavesdropping, these bits match perfectly at the positions where Alice's and Bob's bases were the same, resulting in a secure shared key.

---

4 Simulation With Eavesdropping

Now we simulate the BB84 protocol in the presence of an eavesdropper, "Eve." In this scenario, Eve attempts to intercept the qubits Alice sends to Bob, measuring them in a

randomly chosen basis. Since Eve does not know the basis Alice used to prepare the qubits, her measurement disturbs the qubit's state when she guesses incorrectly. This disturbance will eventually be detected by Alice and Bob when they compare their results.

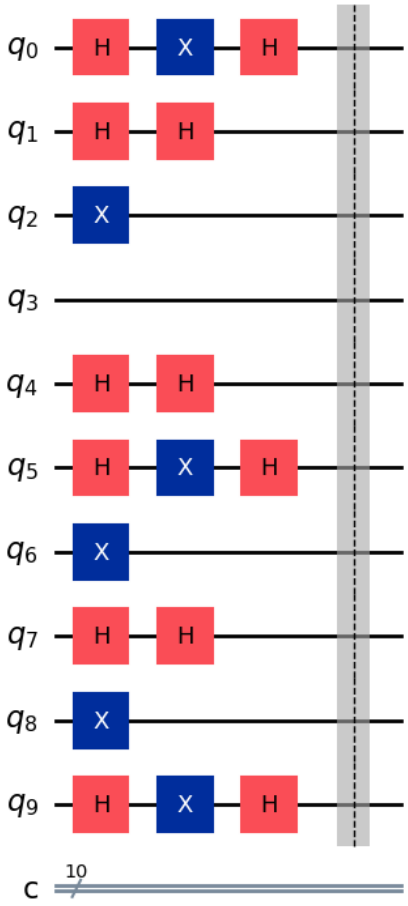

Figure 3: A visualization of the quantum circuit after Alice prepares the qubits for transmission to Bob. Alice selects the qubits' states based on her random choice of bits and measurement bases. The circuit shows the application of Hadamard gates and X gates, corresponding to her random choices.

As before, the circuit diagram illustrates Alice's process of preparing the 10 qubits. Each qubit is initialized according to a randomly chosen bit (0 or 1) and measurement basis (standard or Hadamard). The X gates represent bit flips (i.e., when Alice's bit is 1), while the Hadamard gates reflect the choice of the Hadamard basis (diagonal basis). The qubits are then sent to Bob, but this time Eve attempts to measure them in transit.

In this case, we will see how Eve's interference impacts the final key shared by Alice and Bob.

At this stage, Eve enters the protocol. The function `eavesdrop` simulates her attempt to intercept and measure the qubits Alice sends to Bob. Since quantum cryptography ensures that any measurement made on a qubit alters its state, Eve's intervention introduces errors when she chooses the wrong basis. The BB84 protocol is designed to detect these disturbances, ensuring the security of the key exchange.

Below is the Python code that simulates Eve's eavesdropping:

```
def eavesdrop(qc, num_qubits):
    eve_bases = np.random.randint(2, size=num_qubits)
    for i, base in enumerate(eve_bases):
        if base == 1:
            qc.h(i)
            qc.measure(i, i)
```

• The variable `eve_bases` is randomly generated using `np.random.randint`, which simulates Eve's attempt to guess the basis used to encode each qubit. Like Alice and Bob, Eve randomly chooses between the rectilinear (standard) basis and the diagonal (Hadamard) basis for each qubit.

• Inside the for-loop, Eve applies a Hadamard gate `(qc.h(i))` if her selected basis is 1 (diagonal basis). This gate puts the qubit into a superposition state, preparing it for measurement in the diagonal basis.

• Next, Eve measures the qubit using `qc.measure(i, i)`. This measurement collapses the qubit's state into either $|0\rangle$ or $|1\rangle$, depending on the result.

• If Eve guesses the correct basis (i.e., the same one Alice used to encode the qubit), she can measure the qubit without disturbing its state. However, if Eve chooses the wrong basis, her measurement will collapse the qubit into an altered state, introducing errors that can later be detected by Alice and Bob.

This eavesdropping process introduces potential errors into the system, which will be revealed later during the key comparison between Alice and Bob. The BB84 protocol is secure precisely because any attempt by an eavesdropper to measure qubits without knowing the correct bases creates detectable discrepancies.

The diagram below illustrates the moment in the BB84 protocol after Eve's eavesdropping has taken place. The qubits that Alice prepared and sent to Bob have been intercepted and

measured by Eve, and now Bob is measuring the received qubits.

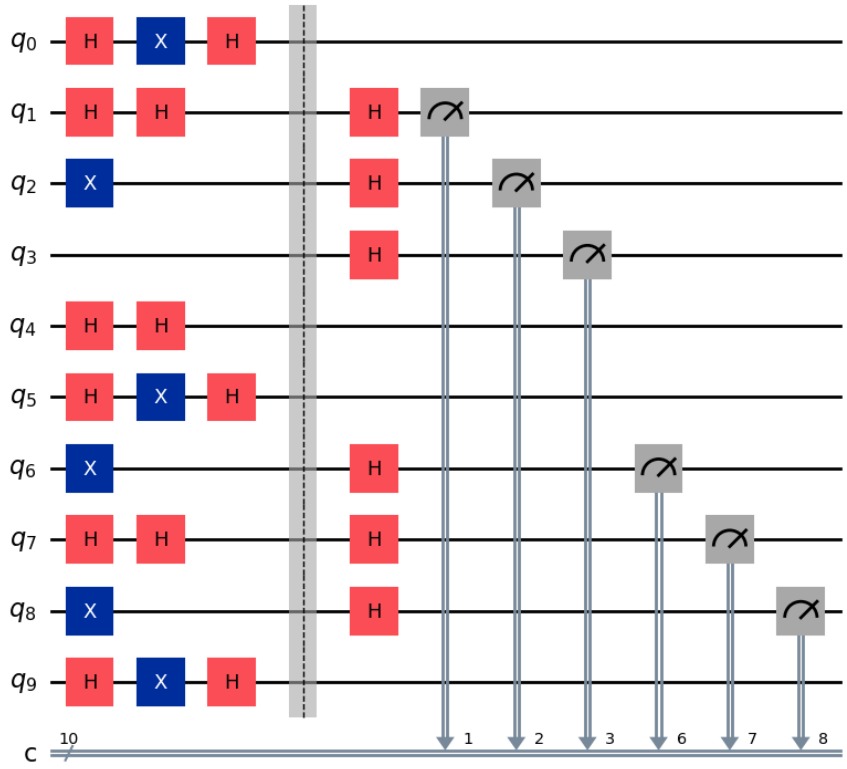

Figure 4: Prior to this stage, Alice had prepared her qubits using either the rectilinear or diagonal basis. The qubits were represented by Hadamard gates (H) for diagonal basis encoding, and X gates were applied to flip the qubit based on Alice's bit selection. The *left side of the gray barrier* represents Alice's preparation of the qubits. *After the gray barrier*, we see the qubits post-Eve's intervention and Bob's subsequent measurements.

• In this visualization, Eve has intercepted the qubits during their transmission from Alice to Bob. Since Eve does not know which basis Alice used to encode the qubits, she must randomly guess whether to measure in the rectilinear (standard) or diagonal (Hadamard) basis. Hadamard gates applied after the gray barrier represent Eve's random basis guesses.

• If Eve's guess was correct (i.e., if she used the same basis as Alice), she does not disturb the qubit's state, and it remains intact for Bob to measure. However, if Eve guesses the wrong basis, her measurement alters the qubit's state, introducing errors that Bob may detect when measuring the qubits.

• On the right side of the diagram, we see the *gray measurement gates* that represent Bob measuring the qubits after receiving them. Bob randomly chooses his measurement basis for each qubit, as he does not know which basis Alice (or Eve) used to prepare the qubits.

• Before measuring in the diagonal basis, Bob applies a **Hadamard gate (H)** to switch the qubit from the standard (rectilinear) basis to the diagonal (Hadamard) basis. This gate puts the qubit into the correct superposition state for measurement in the diagonal basis.

• If Eve guessed the same basis that Alice used for a qubit, Bob will receive the qubit in the correct state, and his measurement will match Alice's original bit. However, if Eve used the wrong basis, the qubit will have been altered by her measurement, and Bob may receive an incorrect result. These discrepancies will be detectable later in the protocol when Alice and Bob compare their measurement bases.

In this stage of the BB84 protocol, we compare the bit values that Alice initially prepared and sent to Bob with the bit values after Eve's interception. Eve's attempt to measure the qubits has introduced errors into the quantum states, and this table highlights the differences caused by her eavesdropping.

Initial Bits:	[0 0 1 0 0 1 1 0 1 1]
After Eavesdropping:	[0 1 0 0 0 1 0 1 0 1]

Table 3: Comparison of Alice's **Initial Bits** with the bits measured **After Eavesdropping**. The differences highlight the disturbance caused by Eve's attempt to intercept the qubits.

• The **Initial Bits** represent the qubits that Alice originally prepared and sent to Bob over the quantum channel. These bits correspond to the states of the qubits before any eavesdropping occurred.

• The **After Eavesdropping** row reflects the bit values after Eve's interception and measurement of the qubits. Since Eve does not know the bases that Alice used to prepare the qubits, her measurements have disturbed some of the qubits, causing errors in the transmitted bit values.

• In the table, we can see that several bits have been altered due to Eve's interference. For instance:

  • The bit at position 1 changed from **0** to **1**.

  • The bit at position 2 changed from **1** to **0**.

  • The bit at position 6 changed from **1** to **0**.

  • The bit at position 8 changed from **1** to **0**.

These discrepancies between the initial bits and the bits after eavesdropping demonstrate the key security feature of the BB84 protocol: any interception of the qubits will introduce

detectable errors in the transmitted information. Alice and Bob can use this to determine whether the communication channel has been compromised.

In this stage of the BB84 protocol, we observe how Bob measures the qubits that he receives from Alice, potentially affected by Eve's eavesdropping. Depending on whether Eve measured the qubits correctly or incorrectly, Bob may receive either an undisturbed or disturbed state, affecting the final measurements and, ultimately, the security of the key.

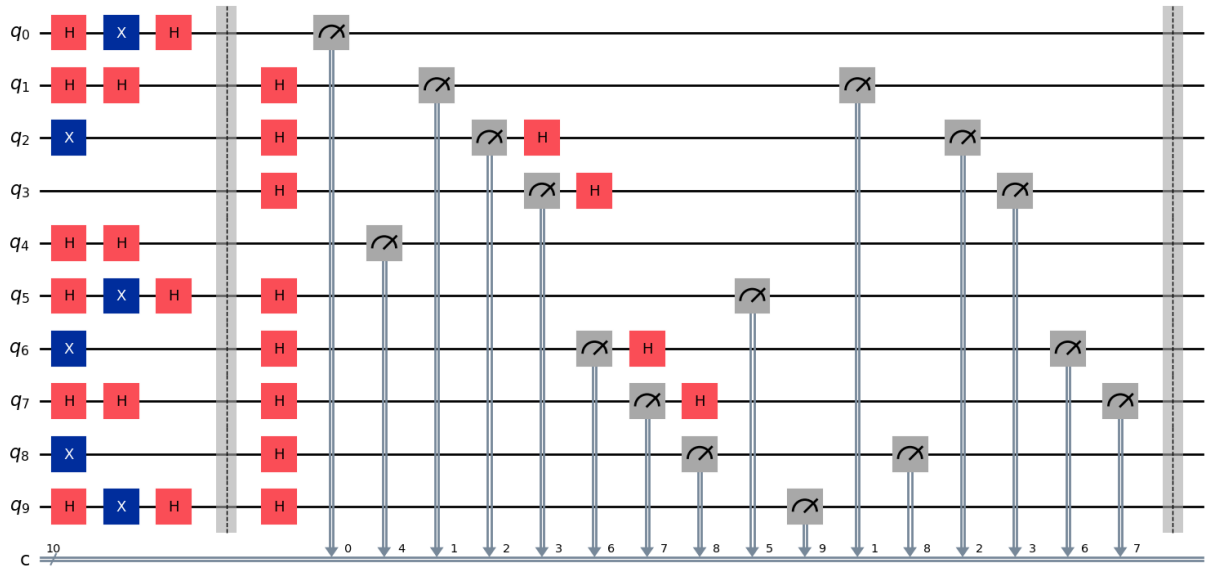

Figure 5: Visualization of Bob's qubit measurements after Alice has sent the qubits and Eve has potentially interfered. The *gray measurement boxes* on the right represent Bob's measurements. For each qubit, if Bob's basis matches Alice's, and there has been no disturbance, he measures correctly. Otherwise, the outcome may be random, particularly if Eve has incorrectly measured the qubits. Bob applies a **Hadamard gate (H)** before measuring in the diagonal (X) basis when required.

- In the diagram, each qubit undergoes measurement by Bob after transmission through the quantum channel. The qubits on the left show Alice's initial preparation, while the qubits on the right demonstrate Bob's measurement after potential interference from Eve.

- If Eve correctly guessed Alice's basis for a particular qubit, Bob will measure the qubit accurately. However, if Eve guessed incorrectly, the qubit's state might have been disturbed, causing Bob to measure an incorrect or random result. This could lead to mismatches when Alice and Bob compare their final keys.

• The **Hadamard gate (H)** is used by Bob to switch between the standard (Z) and diagonal (X) bases, depending on his measurement basis choice. If Bob uses the same basis as Alice for the corresponding qubit, his measurement will correctly reflect Alice's initial bit, provided there has been no interference from Eve.

• The presence of Eve's interference will be evident if Alice and Bob detect discrepancies between their sifted keys during the key reconciliation phase.

During the key sifting phase of the BB84 protocol, Alice and Bob publicly compare their bases (but not their measured bits) to determine which qubits can be used for generating a secure key. This process ensures that only the bits where their bases matched are used in the final key, improving the reliability and security of the communication, especially in the presence of an eavesdropper.

Alice's Bases:	[1 1 0 1 1 0 1 1 0 1]
Bob's Bases:	[0 1 0 0 1 1 0 1 1 0]
Matching Bases:	[1 2 4 7]
Sifted Key:	[0 1 0 0]

Table 4: Alice and Bob's sifted key generation after comparing their bases. The **Matching Bases** row shows the positions where Alice's and Bob's bases were the same, and the **Sifted Key** consists of the bits from those positions, which will form the final shared key.

Eve's eavesdropping introduces discrepancies into the communication. By measuring the qubits with randomly chosen bases, Eve disturbs their states when her basis does not match Alice's, leading to errors in Bob's final measurements. Below, we compare Alice's original bits, the bits after eavesdropping, and the final bits Bob measures:

Initial Bits:	[0 0 1 0 0 1 1 0 1 1]
After Eavesdropping:	[0 1 0 0 0 1 0 1 0 1]
Final Bits:	[0 1 0 0]

Table 5: Comparison of the original bits Alice sent, the altered bits due to Eve's interference, and the final bits measured by Bob after eavesdropping. The discrepancies introduced by Eve in the **After Eavesdropping** row are evident in positions where the final bits do not match the initial bits, indicating the presence of an eavesdropper.

In quantum key distribution, detecting eavesdropping is a critical aspect of security. By comparing the sifted key bits where Alice and Bob's bases matched, they can infer whether Eve attempted to intercept the communication. The presence of discrepancies in the sifted key is a clear indication of eavesdropping:

Eavesdropping Detected:	[True]

Table 6: Detection of eavesdropping. The comparison between Alice's original bits and Bob's final bits shows discrepancies, leading to the detection of Eve's interference. In this case, the protocol would signal that the communication channel is compromised.

5 Conclusion

This demonstration of the BB84 protocol illustrates the process of secure quantum key distribution in the presence and absence of an eavesdropper. The sifting process, combined with the ability to detect disturbances introduced by eavesdropping, ensures the security of the communication. The results show how discrepancies in the key generation process can reveal the presence of a third party attempting to intercept the communication, validating the robustness of quantum cryptography.