## Encryption and Decryption function

The encryption and decryption functions are used after the QKD scheme. This is because without the QKD scheme, a secure and shared key does not exist. My encryption function takes in two inputs, the message (I used “Quantum computing is cool!”) and a key generated by QKD such as [1, 0, 0, 1, 0, 1, 1, 1]. My encryption method involves converting each character in my message string to its binary representation (ASCII or Unicode depending on the character). Then, an XOR gate is applied on each character with its corresponding 1 or 0 in the key. The XOR gate is picked because it is easily reversable - applying two XOR gates will yield the original result, making decryption easy. Practically, the XOR gate is a terrible method because it’s computationally easy to decrypt. However, for the purposes of demonstration, its simplicity and reversibility led me to use it.

```Python
def encode(message, key):
    encoded_message = ""
    for i in range(len(message)):
        char = message[i]
        key_code = key[i % len(key)]  # Cycle through the key list
        char_code = ord(char) # Convert characters in message to ASCII/Unicode
        encoded_code = char_code ^ key_code # Perform  XOR on the codes
        encoded_char = chr(encoded_code) # Convert the encoded code back to character
        encoded_message += encoded_char # Add encoded letter to encrypted message
    return encoded_message

print(encode("Quantum computing is cool!", [1, 0, 0, 1, 0, 1, 1, 1]))
# Encrypted message: Puaottl!bomquuhof ir bnnm!
```

Since the encryption function uses the reversible XOR gate, the decryption function simply needs to run the XOR gate on the encrypted message again.

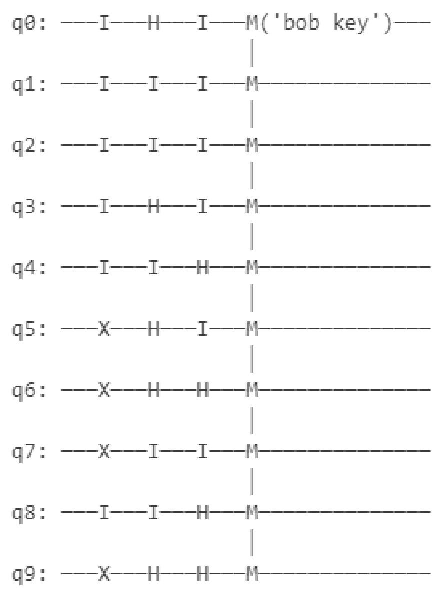
```Python
def decode(encoded_message, key):
    return encode(encoded_message, key)  # Reuse for decoding

print(decode("Puaottl!bomquuhof ir bnnm!", [1, 0, 0, 1, 0, 1, 1, 1]))
# Decrypted message: Quantum computing is cool!
```

## What happens when there is an eavesdropper?

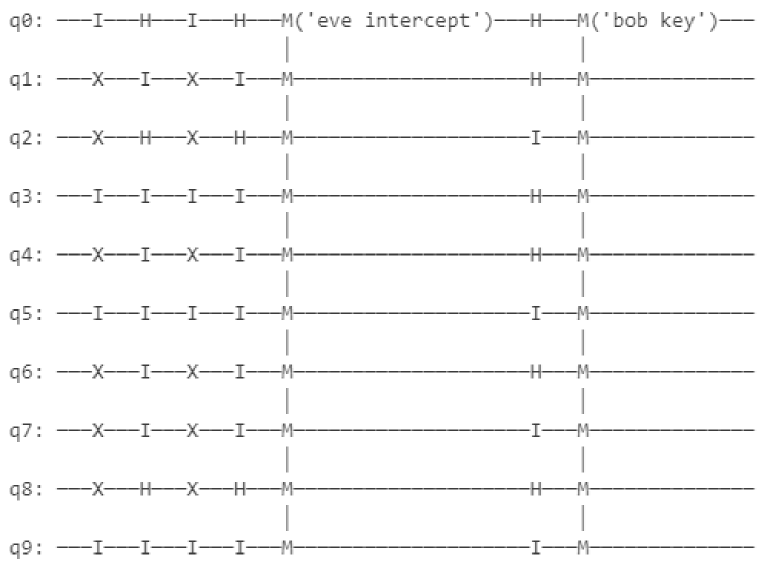
The presence of an eavesdropper would impact the key distribution process. In an ideal circuit without eavesdroppers (figure 1a), Alice and Bob should end up with the same shared key, which completes the key distribution scheme. Alice and Bob can then go on with their encryption/decryption business. However, the existence of an eavesdropper in the circuit (figure 1b) will invetivably ruin the key distribution process – according to the no-cloning theorem, the eavesdropper cannot detect quantum states without ruining them. When there is an eavesdropper, Alice and Bob will realize that their shared keys have differences. Statistically, 50% of Alice and Bob's keys wouldn't match due to the eavesdropper (figure 2b). Seeing these differences, Alice and Bob will realize that an eavesdropper is listening, which means their keys cannot be used. They would try QKD in another quantum channel, hoping that the eavesdropper won't listen again. Doing this repetitively, when Alice and Bob finally achieve a shared key undisrupted by an eavesdropper (like the cases in figure 2a), they can pass their messages and their shared key into the encode and decode functions.

**Figure 1a**: Full BB84 circuit without eavesdropper:



```
q0: ───I───H───I───M('bob key')───
                   │
q1: ───I───I───I───M──────────────
                   │
q2: ───I───I───I───M──────────────
                   │
q3: ───I───H───I───M──────────────
                   │
q4: ───I───I───H───M──────────────
                   │
q5: ───X───H───I───M──────────────
                   │
q6: ───X───H───H───M──────────────
                   │
q7: ───X───I───I───M──────────────
                   │
q8: ───I───I───H───M──────────────
                   │
q9: ───X───H───H───M──────────────
```

**Figure 1b**: Full BB84 circuit with eavesdropper's measurement attack:



```
q0: ───I───H───I───H───M('eve intercept')───H───M('bob key')───
                       │                        │
q1: ───X───I───X───I───M────────────────────H───M──────────────
                       │                        │
q2: ───X───H───X───H───M────────────────────I───M──────────────
                       │                        │
q3: ───I───I───I───I───M────────────────────H───M──────────────
                       │                        │
q4: ───X───I───X───I───M────────────────────H───M──────────────
                       │                        │
q5: ───I───I───I───I───M────────────────────I───M──────────────
                       │                        │
q6: ───X───I───X───I───M────────────────────H───M──────────────
                       │                        │
q7: ───X───I───X───I───M────────────────────I───M──────────────
                       │                        │
q8: ───X───H───X───H───M────────────────────H───M──────────────
                       │                        │
q9: ───I───I───I───I───M────────────────────I───M──────────────
```

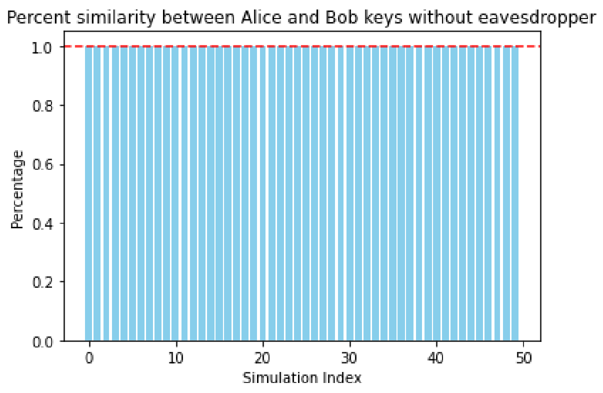


**Figure 2a**: Without a eavedropper, QKD flawlessly gives 100% similar shared keys.

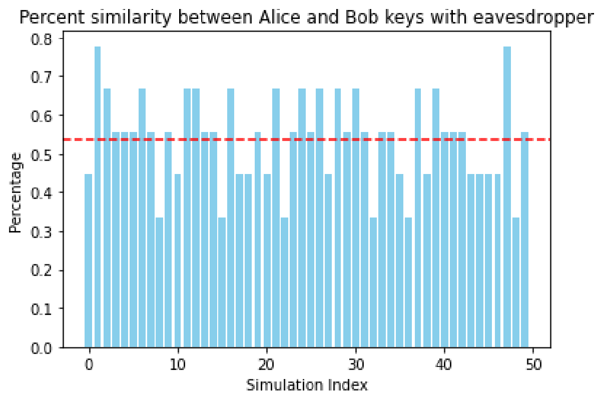


**Figure 2b**: With eavesdroppers, only around 50% of Alice and Bobs' keys align.

## Why is QKD, and quantum encryption in general, important?

Traditional encryption, the kind that secures most of our online communication today, relies on complex math problems. For instance, the RSA algorithm, extensively utilized across communication channels like web browsers, chats, and VPNs, operates under the assumption that no classical computer can practically factor the product of two large prime numbers. While effective currently, these methods can be cracked by sufficiently powerful quantum computers. Already, quantum computers can use the Shor algorithm to break the RSA algorithm. Thus, quantum computers are endangering important sectors such as online transactions, cryptocurrency, and national security.

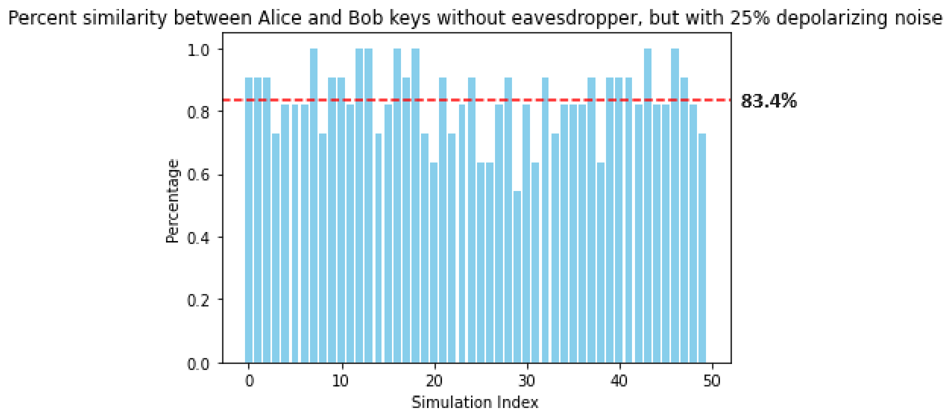
Quantum encryption promises unbreakable communication for the future. Built on the principles of quantum  mechanics such as superposition and entanglement, quantum encryption can transmit keys that are demonstrably tamper-proof. Quantum key distribution is a future-proof algorithm that can securely transmit encryption keys from one party to another. There also exists post-quantum cryptographic schemes such as SPHINCS+ and XMSS, which are proven secure against attacks from quantum computers.

As the quantum threat looms, the significance of quantum encryption in the scientific community is growing. Governments and regulatory bodies such as NISQ are recognizing the importance of quantum encryption to ensure the security of critical infrastructure and sensitive information. Standards and regulations have already been promoted in the adoption of quantum-safe encryption solutions in various industries.

## Next steps for my QKD model (github)

To better mirror current challenges in quantum computing, I plan to add 2 more things to my QKD model:

1) My current QKD model is completely noise-free. In reality, current quantum computers are very noisy – hence, the current stage of quantum computing is named Noisy Intermediate Scale Quantum. Additionally, QKD is meant to be done over large distances, through noisy mediums such as the atmosphere or optical fibers. This noise can compromise the security of QKD. For example, even without the presence of an eavesdropper, noise can create differences between Alice and Bob's shared keys (figure 3). This creates a dilemma for Alice and Bob: are the differences merely due to noise, or are they due to an eavesdropper? This dilemma makes it harder for Alice and Bob to detect the presence of an eavesdropper, which compromises the security of the QKD scheme. To reflect these challenges, I have already written a model that includes depolarizing noise in the QKD scheme. I plan to analyze the results of the noisy QKD circuit, and compare them to the results of a non-noisy QKD-with-eavesdropper circuit. This comparison can tell me how easily Alice and Bob can differentiate between the presence of noise and the presence of an eavesdropper, or both. I'm currently in the middle of writing this code!



2) To mitigate the effects of noise, I plan to include error correction in my circuit. In my original model, each logical qubit is represented by only one physical qubit. This means that noise cannot be detected. To improve my model, I can represent each logical qubit with three physical qubits. Then, with quantum repetition codes, I can add error correction circuits to my original QKD circuit. While this model of QKD requires 3 times more qubits, it can significantly mitigate the effects of noise, which can ensure the security of QKD.

## Thanks for reading!

---

Code (bare bb84 protocol, bb84 with eavesdropper, and bb84 with noise) found on:
https://github.com/xcuithetoe/qxq_final_project
Libraries used: Matplotlib, Numpy, Cirq, Random

Special thanks to Qubit by Qubit for providing many resources that I used to complete this final project.