Table of content

Abstract

In today's era, the use of the cloud to store and send important information is a must. However, security to provide secure access to send and store information is a must. This is where cryptography comes in handy to provide security for such purposes. The four aspects of cryptography such as Data Confidentiality, Data Integrity, Authentication and Non-repudiation provide safeguards in respect of data being stolen or altered in the process. With the advancement of technology, Quantum Key Distribution has taken over cryptography and has been playing an important role in securing data. Quantum key Distribution(QKD) is a protocol where two parties communicate using a secret private key using a public channel, which is proven to be secure. Relying on the law of physics and properties of quantum information, the security of the resulting key is hence guaranteed. One of the schemes of the QKD is the BB84 protocol. In the BB84 protocol, instead of using a public channel to send the secret key, one can use a string of photons encoded with the polarization of the secret key. The goal will be to demonstrate that quantum key distribution is secure against man-in-the-middle attacks (MITM) since it is easier to detect if the qubits have been tampered with or changed. [1][5]

Introduction

In today's world, it is important to maintain security, privacy and confidentiality when it comes to handling information, data and communication. Many methods exist to achieve these goals but one of the most common practices is cryptography. Cryptography is the field of study focused on encryption, decryption and transforming data into encoded messages to make it hard for third parties to solve. Cryptography aims to provide data security, confidentiality, integrity and privacy. It does this by analyzing existing encryption and decryption methods used, identifying the weaknesses of the algorithm being analyzed and trying to make it more robust to attacks. In other words, cryptography aims to protect against sharing information with unwanted parties. Thus it is important that key distribution methods are safely implemented and protected against attacks. [2]

The two types of key distribution that exist are asymmetric and symmetric. The symmetric key uses the same key for encryption and decryption. For asymmetric key distribution or public key, it is used with a different key for encryption and decryption. Common protocols that are used in cryptology are Riverts Shamir-Adleman (RSA) and Advanced Encryption Standard (AES). RSA uses asymmetric keys and relies on the complexity of solving the mathematical equation to generate the keys. AES, on the other hand, uses symmetric key distribution but it can be more vulnerable to a man-in-the-middle attack that could easily intercept and obtain the key for encrypting the message. [3]

As technology begins to improve so does the computational power of computers and the reality of having quantum computers. Having quantum computers poses a great threat to traditional protocols and algorithms like RSA since quantum computers would be able to crack the mathematical formulas faster and easier compared to using a classical computer. It is important to find and come up with new methods and ways to help protect against quantum computer attacks. One method that will be discussed in this paper will be the BB84 quantum key distribution (QKD) protocol. The paper will aim to demonstrate how the BB84 protocol helps to protect against man-in-the-middle attacks.

The structure of the paper is broken down into the following sections:
The first section is relevant works and research, which will focus on similar work being done with QKD. The next section is research methodology, which will give you a bit more information on QKD and AES. The next section will be the design specification, where the setup of the experiment will be talked about. Then afterwards the main result section will talk about the discoveries and main findings from the experiment. Then the simulation/numerical results will show the simulation results such as pictures and calculation(s) needed for the experiment. Finally, the conclusion section will provide a brief recap and the findings and reflection from this experiment.

Relevant works and research

This section will give a brief overview of some studies and information related to managing cryptographic keys, such as key generation, key protection and key distribution.

Existing Cryptography Algorithms

As mentioned before cryptographic has two main types of encryption: asymmetric and symmetric encryptions. Asymmetric encryption uses two different keys. One key would be public for encryption, anyone can see the key and one private key, is kept secret and only known to the owner for decryption. On the other hand, a symmetric key uses the same key for both encryption and decryption. The key in a symmetric algorithm must be kept private and only be known between the sender and receiver. Figure 1 represents a picture view of asymmetric encryption and Figure 2, shows the symmetric encryption.

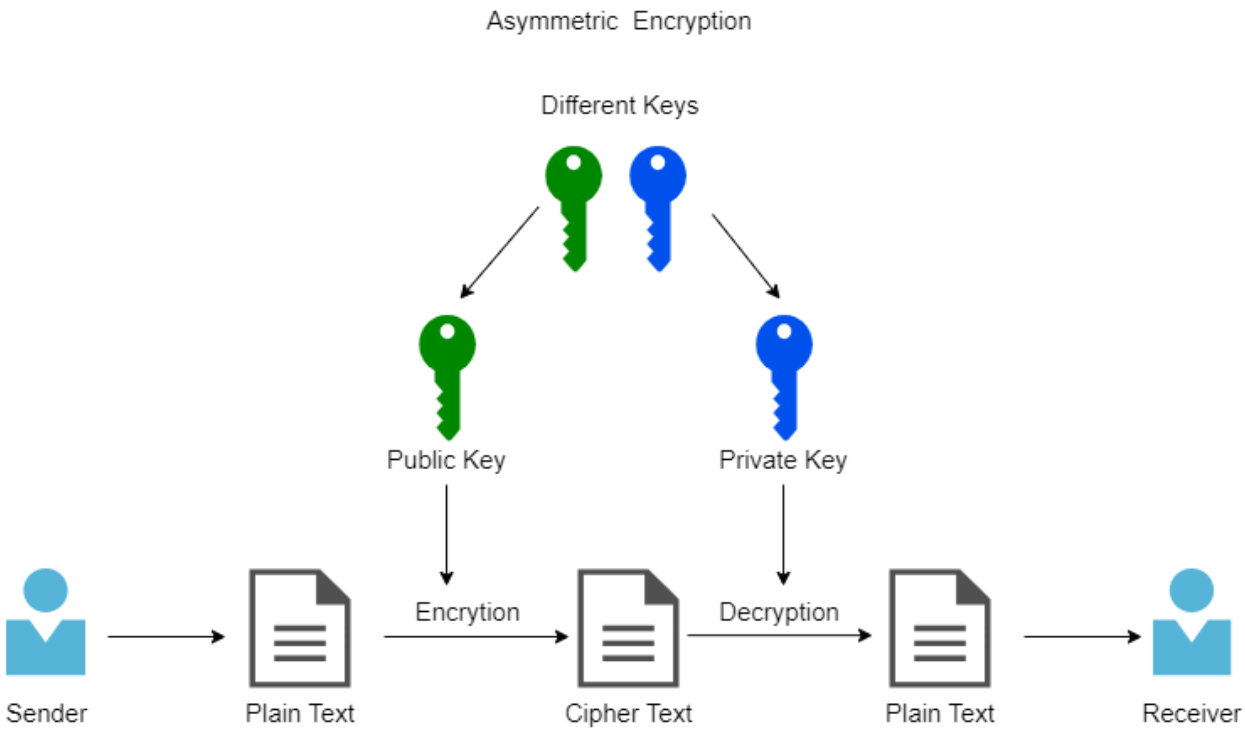

Figure 1: Asymmetric Encryption

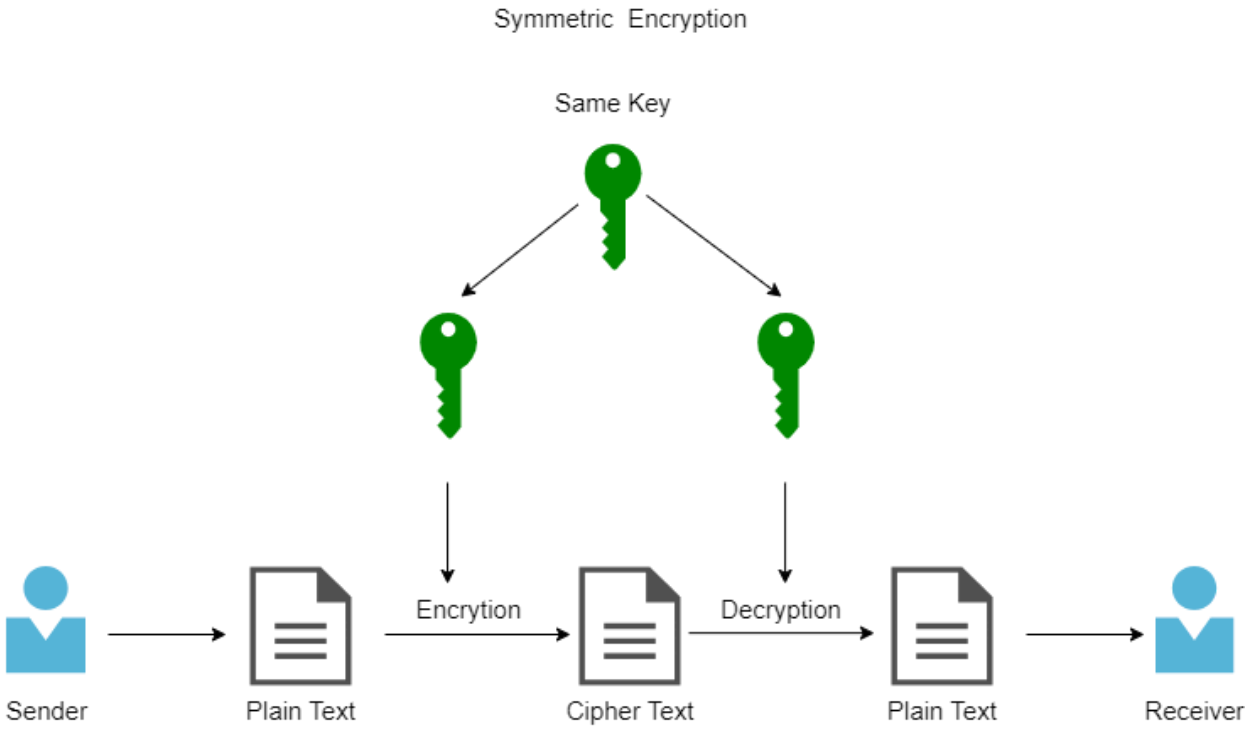

Figure 2: Symmetric Encryption

There already exist many different cryptographic algorithms which fall into asymmetric or symmetric key encryption categories. Each algorithm's security has its own strengths and weaknesses. Vishal Choudhary has done various comparisons and analyses of cryptographic algorithms. He states that RSA and AES are often the most used for security but lacks a method for secure key distribution. A potentially more commonly used protocol in the future might be elliptic curve cryptography (ECC), due to its smaller key size and faster operation. However, ECC has not been proven to provide enough security for smaller device systems like sensor networks and mobile devices. In addition, additional studies show that RSA and random key cryptography rely on the complexity of the algorithm for security, such as key size and data blocks. This could easily be broken in the future with more powerful computers that would be able to perform more complex tasks easier and faster. [3]

Key Generation

August Kerckoff's principle states that it is better to keep the key hidden rather than the algorithms themselves. In one study, the author modified the Diffie-Hellman(DH) algorithm to generate the symmetric key by using a method called the ABC conjecture. This method proved to be very robust and improved on the weaknesses of DH, but it was too challenging to implement in the real world. Another study proposed the generation of cryptographic keys by using biometric features. This method would implement a bit error correction method to handle known errors involved with biometric data. The users would evaluate the performance and

decide whether the results will be accepted or rejected. Although this method relies on the acceptance and rejection rates, it was found other biometric techniques performed better for longer key generation. Another method proposed was using cancellable fingerprint templates. This method involved the sender and receiver creating a cancellable fingerprint template from their original fingerprints and sharing this template via encryption with each other. Then with their original and shared template, it would create a new master template for the two users. This master template would be the shared secret key. This method provides higher security advantages but requires a lot of computation and storage for the templates. [3]

Key Protection

Key protection plays an important role in maintaining secure communication between parties. Often keys will be stored in an unsecured manner which leaves them vulnerable to attacks. One method to try to solve this issue was using a key wrap technique to secure keys. Another method proposed was to just store the keys in a scattered matter on a cloud-based virtual RAM, which would prevent the attacker from getting entire access to the key should the virtual memory get exposed to the attacker. This method would require more storage which would lower the efficiency of this technique. [3]

Key Distribution

Key distribution is important in order to relay and share keys between parties in a secure manner without the shared key being intercepted by a third party. It is important to find protocols to properly authenticate the sender and receiver to ensure that they are indeed the desirable recipient and sender of the messages. One method proposed for achieving this is by using strong complex mathematical functions to generate the shared key (private key). The issue with this method is that it is vulnerable to the man-in-the-middle attack if authentication errors occur.

Another method proposed for key distribution was the use of nodes to distribute public keys. This method was aimed more toward mobile devices and wireless networks. Each node in the system would store information about the public key in the networks. The only drawback of this method was that the security relies on the security of each node, which means if one node in the system got hacked then the entire system is at risk of being hacked as well.

To address the issues mentioned above the development of quantum key distribution (QKD) was proposed to securely distribute keys. QKD is more secure against eavesdropping since it uses qubits. Existing QKD protocols such as BB84, B92 and SARG4 use unique quantum physical properties to generate and share keys. For this experiment, the BB84 protocol will be used. [3]

Research Methodology

The man-in-the-middle attack is one method that can be used to try to intercept information secretly without being detected. The man-in-the-middle attack works with the belief that the two parties are interacting with each other, in reality, the attacker is receiving all the information between them and sending information back to them. It can be difficult to detect a man-in-the-middle attack since the attacker is making it seem like the two parties are directly interacting with each other, below Figure 3 illustrates the man-in-the-middle-attack concept. Notice how Eve (the attacker) is intercepting the message and information that Alice and Bob are sending to each other without being detected.

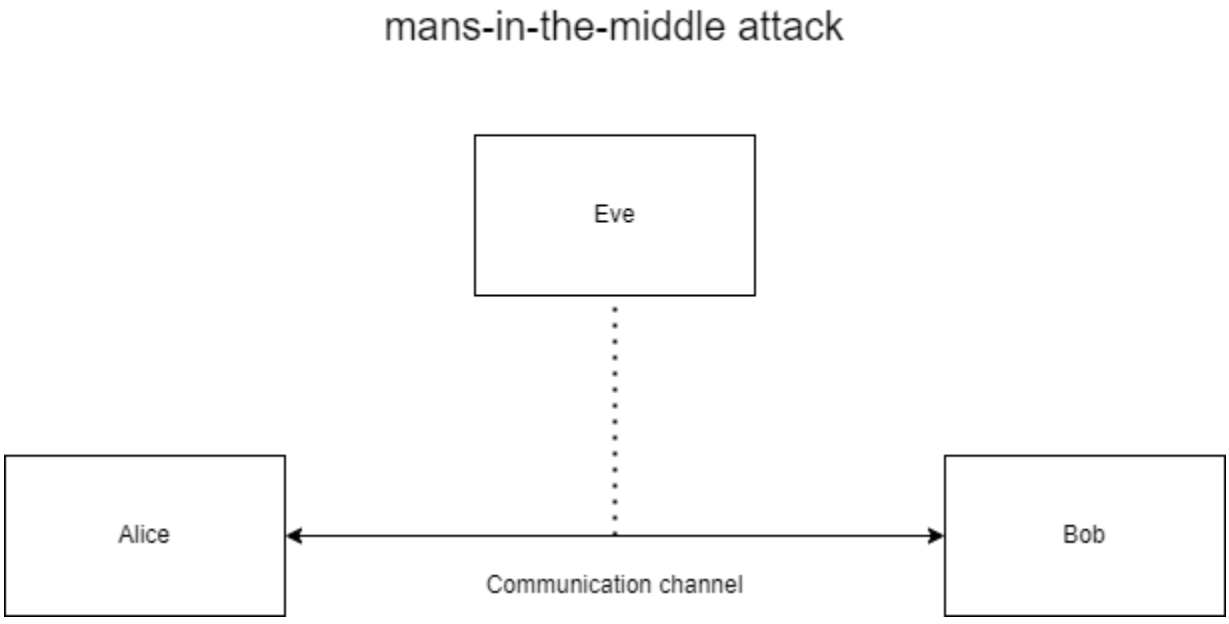

Figure 3: Man-in-the-middle attack

Cryptography relies on keeping the keys secret from the attackers, as the method of the algorithms would eventually become known. By hiding the key, it would be easier to change the key values rather than trying to come up with a completely new algorithm, if the key values are public and the algorithms were kept hidden. The method of quantum key distribution (QDK) was invented to help share key values in an insecure channel between two parties. The data being sent with this communication should be encrypted and decrypted. AES algorithm would be used to encrypt and decrypt the information in QDK since it is more efficient.

Quantum Key Distribution

The experts explain the safety features and mechanism of how secure the communication protocol is, specifically the BB84 protocol developed by Charles Bennett and Gilles Brassard in

1984. The quantum key distribution is a secure methodology against any sort of attack like the man-in-the-middle attack as it uses the cryptographically secure key to establish communication between any two parties by using quantum mechanics and Heisenberg's uncertainty principle. [3]

In terms of establishing communications between two parties, this is how the quantum key distribution methodology works. Let's assume, the two parties are Bob and Alice. Alice wants to send information to Bob. Therefore, Alice encodes some information known as qubits in the state of 0 or 1. Then Alice sends this information to Bob using the channel. Bob receives the information and decodes the information using a secret key established between them.

The quantum key distribution is the most secure method in cryptography as the key is randomly generated by the two parties. QKD is highly resistant to man-in-the-middle attacks because of its following features: [3][4]

1. Measurement Detection:
Every Time an attacker tries to attack the communication channel, the quantum bits will change their current state and result in alerting both parties of the presence of an attacker.

2. Uncertainty Principle:
It is impossible to predict the objects of quantum mechanics cannot be certainly measured or known at the same time.

3. No Cloning theorem:
Quantum mechanics prohibits the copying of the unknown state of a qubit, resulting in preventing the attacker from cloning a qubit state. Hence the attacker can never replicate an exact copy of a qubit.

4. Non- Orthogonality Principle:
There is no quantum measurement, which is capable of separating two non-orthogonal states.

AES Encryption

AES Encryption (Advanced Encryption Standard) was invented to overcome the flaw of the security issue in DES(Data Encryption Standard) by the National Institute of Science and Technology in 2000. AES is an algorithm used to encrypt and decrypt data by layering the data going through the process of substitution, transposition and matrix of bytes. Figure 4 represents the block diagram process of AES encryption.

Another interesting process of AES encryption is that the number of rounds for layering the data depends on the number of keys. For instance, 128 bits for 10 rounds, 192 bits for 12 rounds and 256 bits for 14 rounds. In the above AES system has proven to be the fastest and secure. [3]

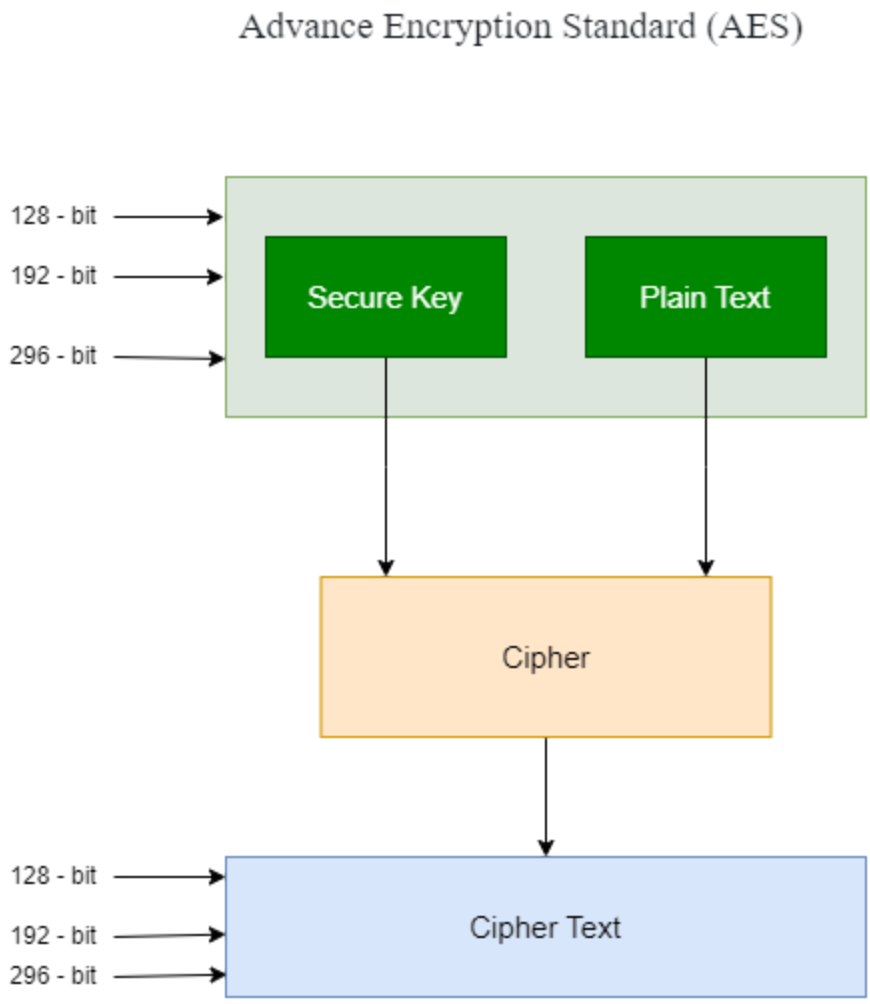

Figure 4: Shows AES Encryption Design

Designing Specification

The design will mimic a BB84 protocol, to share the key between Bob and Alice and AES protocol for the encryption and decryption. Figure 5 below represents a diagram that will represent the functionality flow of how the code for the experiment should work. At the start of the program, it will ask the user if they want to simulate an attacker (Eve). The process will follow the same steps regardless if an eavesdropper is present or not. For this experiment, Alice and Bob would be using two channels. One channel sends each other a qubit to measure and another communication channel tries to derive the shared key between each other. In the first part of the experiment Alice sends random generated sequence of qubits to Bob with four different outcomes (ex.45°, -45°, 0°, 90° and -90°). Bob will measure the receiving qubits and take notes of his results and the measurements he made to obtain those results. Alice will also take note of the qubit she sent to Bob. Now using the communication channel Bob and Alice will try to see if the results they obtain are acceptable. This is done with Alice communicating some

random qubits that she chose to Bob and Bob telling Alice the measurements he used to obtain the results for those qubits. Alice and Bob will compare the results to see if the qubits match. If Eve was present then the qubit that Bob obtain will be different from the one Alice sent. The program flow shows that when there is a presence of an eavesdropper then the system should give the wrong result. The code for the project will be done using Python in the ide, Visual Studio Code. The simulation of using 6 bits will be used for this project. [3][5]

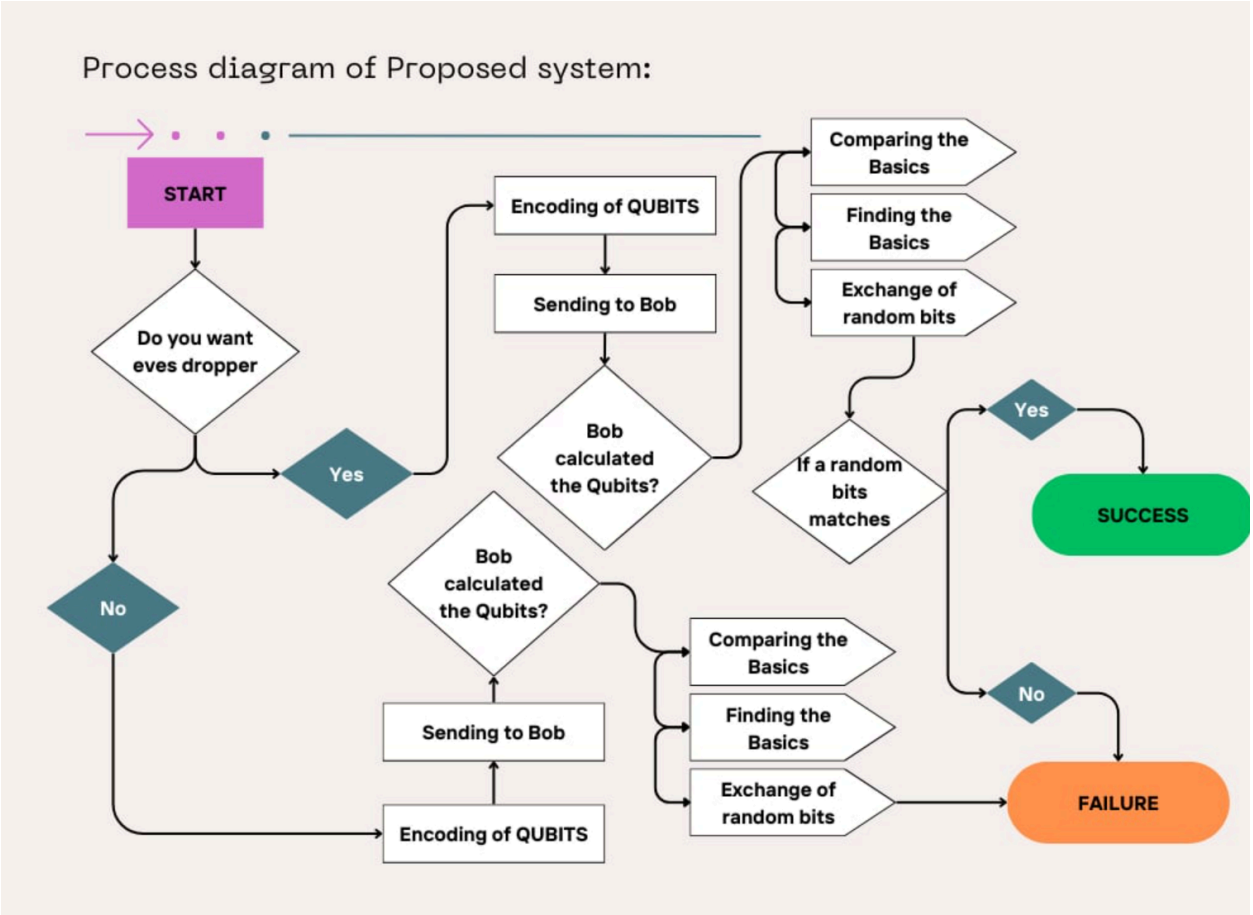

Figure 5: Flow Chart of Code for experiment

Simulations / Numerical Results

Note that for the sake of limited computation power and the purpose of proof of concept, we decided to simulate the results using six qubits being sent in the program. The language used for the simulation was Python. Ideally, the use of more qubits like 128-bit in a standard AES encryption would be used in a QKD BB84 protocol.

```
Do you want an eavesdropper in the simulation? (yes/no): no
Encoded qubits: 000000
Sending qubits to Bob without interference.
Bob received and is measuring qubits with his bases: XZZZZZ
Alice's bases: XXZZZZ
Bob's bases: XZZZZZ
Matching bases at positions: [0, 2, 3, 4, 5]
Secret key generated (partial due to basis mismatch): 00000
No inteference detected.
```

Figure 6: Result from simulation without eavesdropper

```
Do you want an eavesdropper in the simulation? (yes/no): yes
Encoded qubits: 000000
Eavesdropper is present. Interfering with the qubits...
Bob received and is measuring qubits with his bases: XXZXZX
Alice's bases: ZZZZXX
Bob's bases: XXZXZX
Matching bases at positions: [2, 5]
Secret key generated (partial due to basis mismatch): 01
```

Figure 7: Result from simulation with eavesdropper

Error propagation

1. Error Calculation on the channel:
The chances of having one correct qubit state is ½ i.e there is a 50% chance of sending one correct qubit to Bob.
There is also a 50% chance of having one correct base, i.e ½ chance of a correct base.
The percentage of finding an eavesdropper on the channel is ½ *½ = ¼ = 25% on the channel.
The percentage of not finding an eavesdropper on the channel is (100 - 25) % = 75% i.e 3/4

2. Error calculation:
For the sake of simplicity, we used a 6-bit key implementation. Out of which, half of the key has been discarded because of the selection of the wrong bases.

Therefore, the remaining bit keys = (6- 6*½) = 3 bit keys

There is a 50% chance of finding an eavesdropper on the remaining bit key i.e 3*½ = 1.5 bits
Hence, only 1.5 bits will be used for encryption as there is no chance of finding an eavesdropper.

The total calculation of having a secure channel is 1.5 bits * 3/4 = 1.125
Therefore, for a 3-bit key size, there is a 1.125 chance an attacker will not even get identified.

Main Results

This section will discuss the main findings from the simulation performed. In Figure 6, it can be seen that when an eavesdropper (eve) is not present the probability of Bob and Alice deriving the same key has a higher chance of occurring. Whereas, when Eve (attacker) for the man-in-the-middle attack makes it more likely to change the bases of the qubit (Figure 7). Therefore this would result in a higher chance of deriving the wrong keys and not having matching bases between Alice and Bob. However, when Bob and Alice do not have matching bases, it can indicate that an eavesdropper is present in the communication and quantum channels. There is also the likelihood of error propagation and other external factors that could cause the difference between what Alice sent and what Bob measured and calculated. Overall the QKD BB84 protocol can be seen as a good method of deriving and sharing keys between two parties from the man-in-the-middle attack in an unsecured channel. It will be important for Bob and Alice to determine what is an acceptable matching rate which would still make the key reliable.

Conclusion

After finishing this experiment it can be seen that QKD with BB84 protocol can be used to distribute share keys amongst users in an unsecured channel. It shows that the detection of an eavesdropper can be seen since the qubit being sent and received from the two parties would be different. This would alert the users that an eavesdropper is present. In addition, due to quantum key distribution principles like the no-cloning theorem, and measurement detection amongst others, enhance security and increase the difficulty for the attacker to obtain any information about the qubits being sent. However, factors such as noise and the qubit interaction with the environment could cause Alice and Bob to derive the same key. For this reason, it would be important for Alice and Bob to determine an acceptable error rate to determine the reliability of the key.

Looking forward, it is important to develop and test for other secure methods that will be robust against classical and quantum attacks. As technology advances, it will become important to find secure methods to share keys amongst parties, given the vulnerabilities of traditional communication algorithms. Although QKD has been proven to be secure against man-in-the-middle attacks, it can be hard to set up given the challenges involved with using quantum computers, such as cost and maintainability.

Reference

[1] Y.-Y. Fei, X.-D. Meng, M. Gao, H. Wang, and Z. Ma, "Quantum Man-in-the-middle attack on the calibration process of quantum key distribution," *Nature News*, 09-Mar-2018. [Online]. Available: https://www.nature.com/articles/s41598-018-22700-3. [Accessed: 15-Jun-2024]

[2] Class Notes, "Lesson 7: Modern Cryptography," 2024.

[3] S. K. Shrikant Jagdale, *Secure sharing of secret key on insecure channel using Quantum key distribution*, 2019. [Online]. Available: https://norma.ncirl.ie/4155/1/shirishkumarshrikantjagdale.pdf. [Accessed: 15-Jun-2024]

[4] MR.Asif, "Quantum key distribution and BB84 protocol," *Medium*, 10-May-2022. [Online]. Available: https://medium.com/quantum-untangled/quantum-key-distribution-and-bb84-protocol-6f03cc6263c5. [Accessed: 16-Jun-2024]

[5] Class Notes, "Lesson 10: Quantum Cryptography," 2024.

Appendix

Code used for this report in Python to simulate BB84 protocol

```python
import random

#Maya Ostiguy Hopp - 40175258
#Nabila Tabassum - 40109039

def generate_fixed_qubits(length=6):
    """Generate a fixed 6-bit binary string."""
    return '0' * length  # Fixed qubit string of zeros

def choose_random_bases(length=6):
    """For each bit, randomly choose a basis ('X' for rectilinear, 'Z' for diagonal)."""
    return ''.join(random.choice('XZ') for _ in range(length))

def simulate_eavesdropping(qubits):
    """Simulate the potential eavesdropping which may alter the qubits."""
    qubits_list = list(qubits)
    num_flips = random.randint(0, 3)  # Flip up to 3 bits to simulate interference
    for _ in range(num_flips):
        idx = random.randint(0, len(qubits) - 1)
        qubits_list[idx] = '1' if qubits_list[idx] == '0' else '0'
    return ''.join(qubits_list)

def send_qubits(qubits, eavesdropper):
    """Simulate sending qubits to Bob, possibly with eavesdropping."""
    if eavesdropper:
        print("Eavesdropper is present. Interfering with the qubits...")
        return simulate_eavesdropping(qubits)
    else:
        print("Sending qubits to Bob without interference.")
        return qubits
```

```python
def measure_qubits(qubits, bases):
    """Simulate Bob measuring the qubits based on his bases."""
    print(f"Bob received and is measuring qubits with his bases: {bases}")
    return qubits  # In actual QKD, the measurement would depend on alignment with bases

def basis_comparision(alice_bases, bob_bases):
    """Determine indices where Alice's and Bob's bases match."""
    print(f"Alice's bases: {alice_bases}")
    print(f"Bob's bases: {bob_bases}")
    matching_indices = [i for i in range(len(alice_bases)) if alice_bases[i] == bob_bases[i]]
    print(f"Matching bases at positions: {matching_indices}")
    return matching_indices

def qkd_simulation():
    eavesdropper_input = input("Do you want an eavesdropper in the simulation? (yes/no): ").lower()
    eavesdropper = eavesdropper_input == 'yes'

    # Generate and encode qubits
    qubits = generate_fixed_qubits()
    print(f"Encoded qubits: {qubits}")
    alice_bases = choose_random_bases()
    bob_bases = choose_random_bases()

    # Send and measure qubits
    received_qubits = send_qubits(qubits, eavesdropper)
    measured_qubits = measure_qubits(received_qubits, bob_bases)
```

```python
    # Basis comparision and key generation
    matching_indices = basis_comparision(alice_bases, bob_bases)
    key = ''.join(measured_qubits[i] for i in matching_indices if i < len(measured_qubits))
    print(f"Secret key generated (partial due to basis mismatch): {key}")

    #checking that Alice and Bob key match when no eavedropping occurs
    if not eavesdropper:
        exppected_key = ''.join(qubits[i] for i in matching_indices if i < len(qubits))
        if key == exppected_key:
            print("No inteference detected.")
        else:
            print("Keys do not match despite no interference")

    return key

# Run the QKD simulation
qkd_simulation()
```