MISR UNIVERSITY
FOR SCIENCE & TECHNOLOGY
College of Information Technology

جـامعـة مـصـر
للعلـوم والتكنـولـوجيـا
كليـة تكنولوجيـا المعلومـات

CRYPTOGRAPHY USING QUANTUM COMPUTING

Prepared By

Mahmoud Karem Mahmoud 200045706
Fouad Mahmoud 200043850
Mohamed Khaled 200037834
Youssef Mohamed Hassan 200043911

Under Supervision

Nehal Abdelsalam
Fares

1. Relevance:

This project connects directly to the concepts studied in Theory of Computation, particularly the study of computational models and their limitations. Classical cryptographic systems rely on the assumption that certain mathematical problems—such as integer factorization or discrete logarithms—are computationally hard for classical Turing machines. However, the emergence of quantum computation introduces new computational models that outperform classical ones in several key tasks.

Quantum algorithms like Shor's Algorithm are capable of breaking classical encryption by solving these hard problems in polynomial time. This threatens widely used cryptographic systems. As a result, quantum cryptography provides a new form of security based not on computation, but on the laws of quantum mechanics. The BB84 protocol is one of the earliest and most important examples of Quantum Key Distribution (QKD), demonstrating how quantum concepts can establish secure communication resistant to quantum attacks.

2. Effort:

- **Conducted research on the differences between classical and quantum computational power.**

- **Studied quantum cryptography principles, including superposition, quantum states, and measurement theory.**

- **Implemented a complete C++ simulation of the BB84 protocol, modeling random bit generation, basis selection, quantum measurement, and key extraction.**

- **Wrote a structured and detailed report explaining the protocol and its relation to computational theory.**

- **Prepared diagrams and documentation to demonstrate how the protocol achieves information-theoretic security.**

3. Introduction:

Cryptography ensures secure communication and protects data from unauthorized access. Traditional cryptographic systems rely on the assumption that certain problems are extremely hard to solve using classical computational models. For example, RSA security depends on the difficulty of factoring large integers—a task that classical computers struggle with.

However, quantum computing introduces a new paradigm where qubits can represent multiple states simultaneously through superposition, enabling exponentially faster computation for certain tasks. This threatens classical encryption systems and motivates the development of quantum-safe alternatives.

Quantum cryptography, specifically the BB84 protocol, leverages fundamental principles of quantum mechanics—such as the fact that measurement disturbs the state of a qubit—to achieve secure key distribution. This project demonstrates a simulation of the BB84 protocol using a classical programming language (C++) to illustrate how quantum-based security works.

4. Classical vs Quantum Cryptography:

Quantum Crypto	Classical Crypto	Feature
Physics (measurement changes qubits)	Hard math problems (factorization)	Security relies on
Only to physical tampering	Shor's Algorithm	Vulnerable to
BB84, E91	RSA, AES	Example

- Classical cryptography relies on computational complexity. If a problem takes too long to solve on a classical computer, it is considered secure. Examples include RSA, AES, and Diffie-Hellman. However, quantum computers challenge this assumption by providing new algorithms that can solve these problems efficiently.
- Quantum cryptography, on the other hand, does not rely on computational difficulty. Instead, it uses physical properties of quantum particles to ensure that any attempt at eavesdropping introduces detectable disturbances. This creates a form of security that remains valid even in a post-quantum world.

5. The BB84 Protocol:

The BB84 protocol is a quantum key distribution method proposed by Charles Bennett and Gilles Brassard in 1984. It enables two parties—Alice and Bob—to generate a shared secret key securely.

Protocol Steps (Expanded):

1- Alice generates random bits.
These bits represent the states she wants to encode.

2- Alice chooses random bases (Z or X).
Z basis (+): classical bit behavior
X basis (×): diagonal polarization
This randomness ensures unpredictability.

3- Alice sends encoded qubits to Bob.
The encoding depends on both bit value and basis.

4- Bob chooses random bases to measure.
If Bob's basis matches Alice's basis, he gets the correct bit.

5- Bob measures the qubits.
If the bases differ, quantum mechanics forces the measurement to be random.

6- Alice and Bob announce their bases publicly.
They do NOT reveal the bits.

7- They keep bits where bases matched.
These become the raw key.

8- They check for eavesdropping.
If an attacker measures the qubits, the error rate increases.

9- They extract the final shared key.

6. Implementation (C++ Simulation):

```
#include <iostream>           // Used for input/output operations
#include <cstdlib>            // Used for rand() and srand()
#include <ctime>              // Used to seed random number generator with current
time

using namespace std;

int main() {
    srand(time(0));           // Initialize random generator using current time

    const int n = 8;          // Number of bits to simulate
    int aliceBits[n];         // Array to store Alice's random bits
    int aliceBases[n];        // Array to store Alice's random bases (0 = Z, 1 = X)
    int bobBases[n];          // Array to store Bob's random bases
    int bobResults[n];        // Array to store Bob's measured bits
    int finalKey[n];          // Array will hold the final shared key
    int keyIndex = 0;         // Tracks how many matching bits are added to the key

    // ----- Step 1: Alice generates random bits and bases -----
    for (int i = 0; i < n; i++) {
        aliceBits[i] = rand() % 2;    // Alice picks a random bit (0 or 1)
        aliceBases[i] = rand() % 2;   // Alice picks a random basis (0=Z, 1=X)
    }

    // ----- Step 2: Bob chooses random bases and measures -----
    for (int i = 0; i < n; i++) {
        bobBases[i] = rand() % 2;     // Bob randomly chooses a basis (0=Z, 1=X)

        if (bobBases[i] == aliceBases[i]) {
            bobResults[i] = aliceBits[i];     // If bases match, Bob gets the correct
bit
        }
        else {
            bobResults[i] = rand() % 2;       // If bases differ, Bob gets a random
result
        }
    }

    // ----- Step 3: Compare bases to generate final key -----
    for (int i = 0; i < n; i++) {
        if (aliceBases[i] == bobBases[i]) {   // Keep only matching-basis positions
            finalKey[keyIndex++] = bobResults[i];  // Add bit to key
        }
    }

    // ----- Print everything -----
    cout << "Alice Bits:      ";
    for (int i = 0; i < n; i++) cout << aliceBits[i] << " ";
    cout << endl;

    cout << "Alice Bases:     ";
    for (int i = 0; i < n; i++) cout << aliceBases[i] << " ";
    cout << endl;
```

```
        cout << "Bob Bases:       ";
        for (int i = 0; i < n; i++) cout << bobBases[i] << " ";
        cout << endl;

        cout << "Bob Results:     ";
        for (int i = 0; i < n; i++) cout << bobResults[i] << " ";
        cout << endl;

        cout << "\nFinal Shared Key: ";
        for (int i = 0; i < keyIndex; i++) cout << finalKey[i] << " ";
        cout << endl;

        return 0;    // End of program
    }
```

7. Results & Discussion:

The C++ simulation shows the expected behavior of the BB84 protocol:

- **When Alice and Bob choose the same bases, Bob's measurement matches Alice's original bit.**
- **When they choose different bases, the measurement becomes random.**
- **Only the bits with matching bases are kept to form the shared key.**
- **This demonstrates how quantum uncertainty provides inherent security.**

Additionally, if the error rate becomes high, it indicates the presence of an eavesdropper (Eve). This is a unique aspect of quantum cryptography, allowing automatic detection of interception attempts.

8. Conclusion:

Quantum cryptography represents a fundamental shift in how secure communication can be achieved. Instead of relying on computational difficulty, it uses the principles of quantum mechanics to ensure that a key can be exchanged securely—even in the presence of future quantum computers.

This project provides a clear demonstration of these ideas through the BB84 protocol simulation. The implementation shows how randomness, measurement disturbance, and basis selection combine to produce a secure cryptographic key.

9. Github and README File:

- **Repo Link :** https://github.com/mahhmoudkarem/Quantum-Cryptography.git