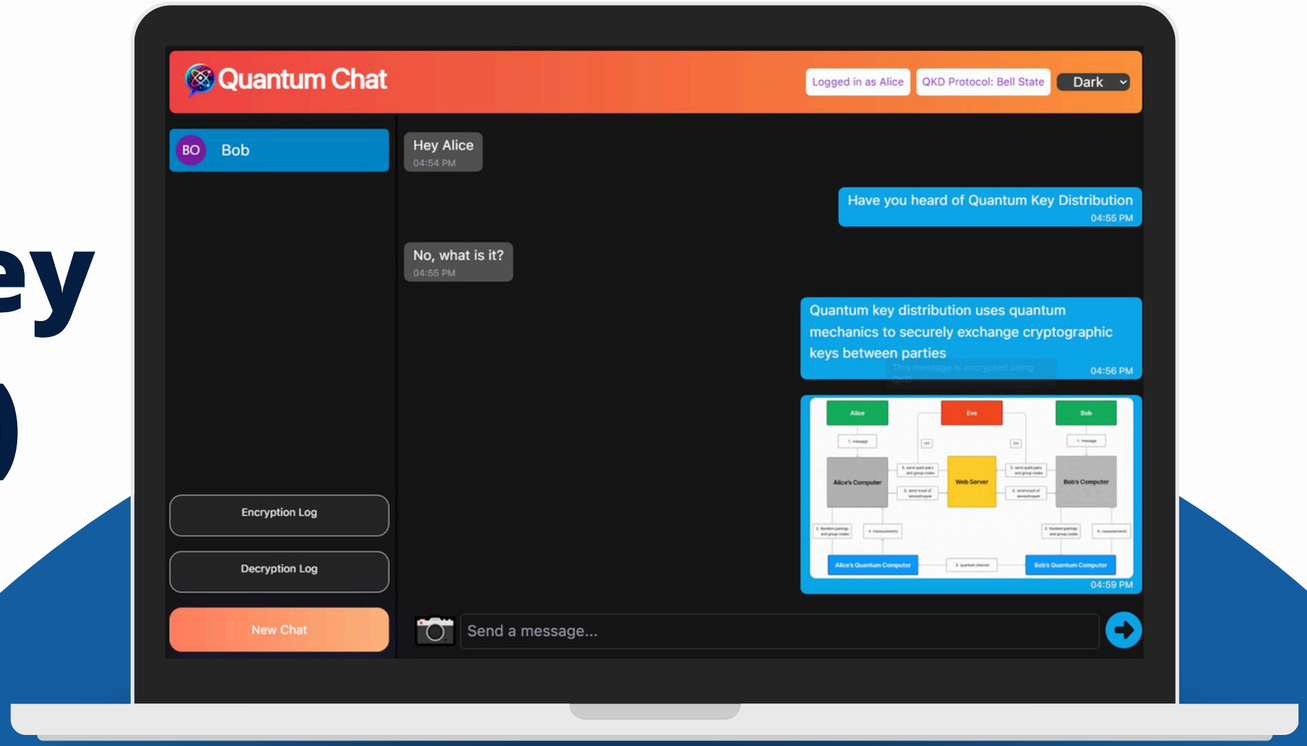
Quantum Chat

Secure Messaging using Quantum Key Distribution (QKD)

Theme: Quantum Mechanics

Team: Quantum Cuircuit Breakers



Overview

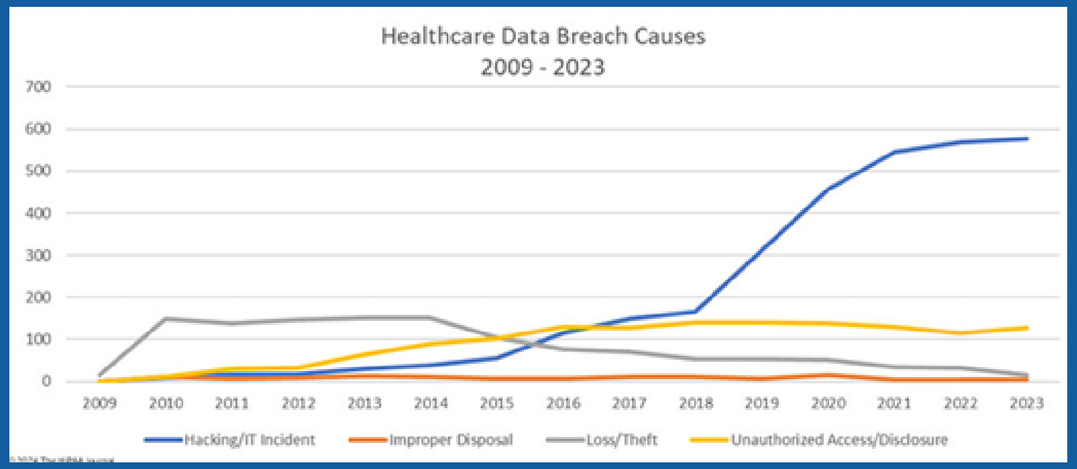
Introduction

Cybersecurity Challenges in Healthcare

- The healthcare sector faces an alarming surge in cyberattacks, with 725 large breaches reported in 2023 (US DHHS).
- Recent incidents such as the MOVEit Transfer and GoAnywhere MFT vulnerabilities underscore vulnerabilities in healthcare data security.
- Advances in quantum computing, particularly Shor's algorithm, highlight the urgency for quantum-resistant encryption.

Project Overview

- This project implements a Quantum Key Distribution (QKD) system, using the BB84, E91 and T22 QKD protocols. The system is integrated into "**QuantumChat**," a secure chat application designed for the healthcare industry.
- Innovative Integration:
  - Simulates patient vital data (e.g., heart rate, temperature) using Arduino sensors.
  - Patient data is transmitted securely via the QuantumChat application using QKD protocols.
  - Ensures sensitive medical information remains protected from cyber threats.



PROBLEM STATEMENT

- The healthcare industry increasingly relies on digital communication for sharing sensitive patient data.
- However, existing cryptographic methods, reliant on computational complexity, are becoming vulnerable to advancements in quantum computing.
- This vulnerability jeopardizes the confidentiality and integrity of patient data, potentially leading to privacy breaches and hindering the adoption of digital healthcare solutions.

BACKGROUND INFORMATION

- The healthcare industry increasingly relies on digital communication for sharing sensitive patient data.
- However, existing cryptographic methods, reliant on computational complexity, are becoming vulnerable to advancements in quantum computing.
- This vulnerability jeopardizes the confidentiality and integrity of patient data, potentially leading to privacy breaches and hindering the adoption of digital healthcare solutions.

Objectives

Objective 01

- Use a quantum circuit simulation to generate a shared key via Quantum Key Distribution (QKD), ensuring the key is securely distributed to both users.
- Evaluate the security advantages of the T22 protocol compared to other QKD protocols, particularly BB84 and E91.

Objective 02

- Integrate the implemented QKD protocol into a secure chat application ("QuantumChat") designed for communication between healthcare professionals.
- Demonstrate the feasibility of the system for real-world use by simulating key exchange and secure message transmission.

Objective 03

- Demonstrate the potential applications of QuantumChat within various healthcare scenarios, including teleradiology and remote patient monitoring.
- Receive live patient vitals data (e.g., heart rate, temperature) using Arduino and sensors.

LITERATURE SURVEY

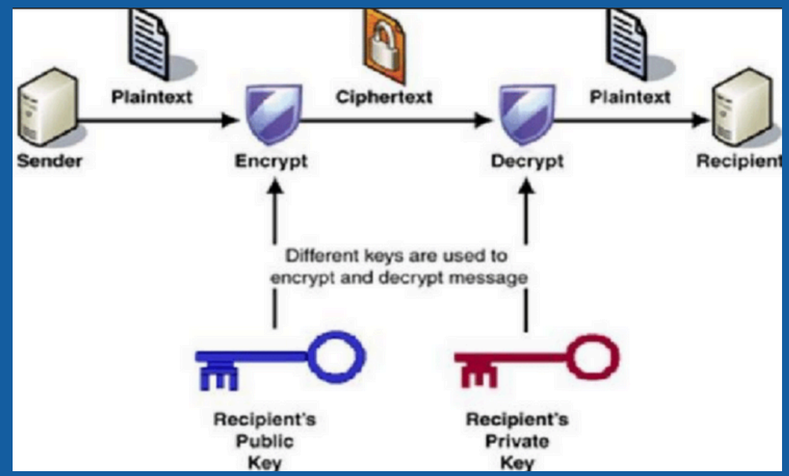
Author	Paper Title	Summary
RAMONA WOLF	QUANTUM KEY DISTRIBUTION	This book is intended to serve as an introduction to the fast-developing field of quantum key distribution. It requires only basic knowledge of quantum mechanics and linear algebra but no prior knowledge of quantum information theory.
D. Song and D. Chen, "Quantum Key Distribution Based on Random Grouping Bell State Measurement," in IEEE Communications Letters, vol. 24, no. 7, pp. 1496-1499, July 2020, doi: 10.1109/LCOMM.2020.2988380	Quantum Key Distribution Based on Random Grouping Bell State Measurement	In the field of cryptography, strings of values called keys are used for encrypting, decrypting and transmitting messages securely. In their simplest form, keys are sequences of numbers that, when added to another string of numbers, produce a new one. Keys are generated through random processes, so encrypting or decrypting the message is only possible with direct access to the key.
NAYANA TIWARI	Quantum key distribution using bell states entaglement	To communicate information securely, the sender and recipient of the information need to have a shared, secret key. Quantum key distribution (QKD) is a proposed method for this and takes advantage of the laws of quantum mechanics. The users, Alice and Bob, exchange quantum information in the form of entangled qubits over a quantum channel as well as exchanging measurement information over a classical channel

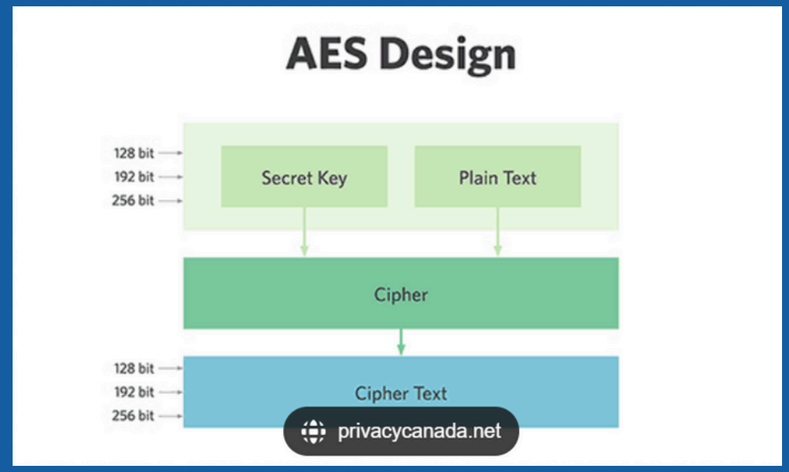
Classical Encryption Techniques

Cryptography is the practice of securing communication and information through the use of codes. today we use 2 major classical encryption algorithms.

- **RSA** – The sender uses the recipient's public key to encrypt data. The recipient then uses their private key to decrypt it. The keys are generated using two large prime numbers. The public key consists of the product of these two primes and an arbitrary small exponent

- **AES** – AES, which stands for "Advanced Encryption Standard," is a specification for the encryption of electronic data. AES is a symmetric block cipher algorithm, meaning it uses the same key for both encryption and decryption



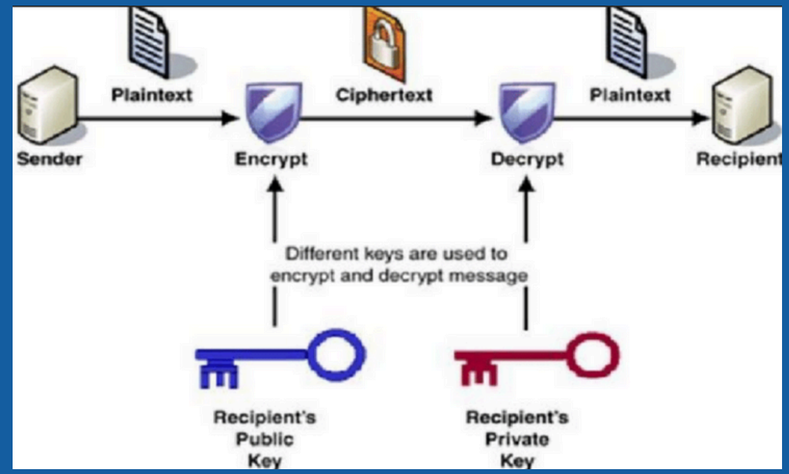


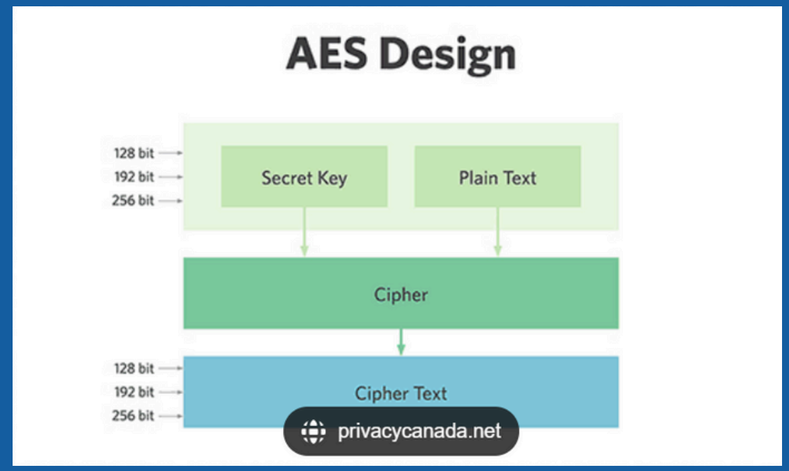
Why are classical encryption methods not safe anymore?

RSA encryption, which is widely used for securing online data, could potentially be threatened by the advent of quantum computers

- **Shor's Algorithm**: Quantum computers can use Shor's algorithm, a quantum algorithm that factors large numbers efficiently. Since the security of RSA is based on the difficulty of factoring large numbers, this makes RSA vulnerable to quantum attacks
- **Quantum Computing Power**: A sufficiently powerful quantum computer could break 2048-bit RSA encryption in as little as 8 hours. AES encryption is also not 100% safe to a online data breach

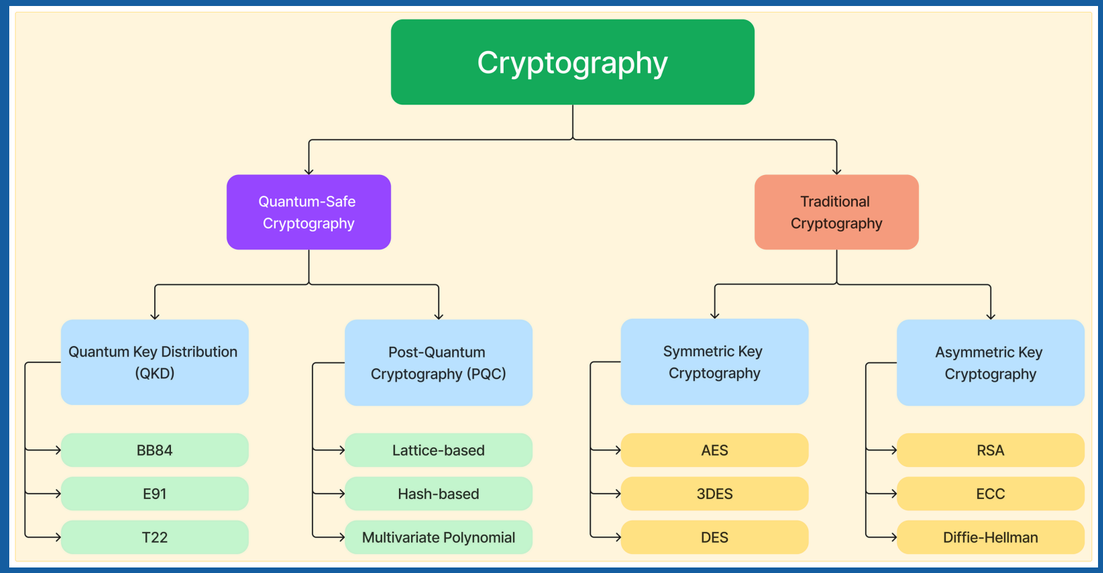
- **Grover's Algorithm**: Quantum computers can use Grover's algorithm, which can reduce the brute force attack time to its square root32. This means that an AES-256 encryption would take half the time to break using a quantum computer





Introduction to QKD

- Traditional encryption techniques like RSA and AES, while currently secure, are becoming increasingly vulnerable to advances in computing power, especially with the potential rise of quantum computers.
- These classical methods could be broken by quantum algorithms, threatening the security of encrypted data.
- Quantum key distribution (QKD) is a secure communication scheme for sharing symmetric cryptographic keys based on the laws of quantum physics
- Unlike classical cryptography, which relies on complex mathematical problems, QKD guarantees security through the laws of physics



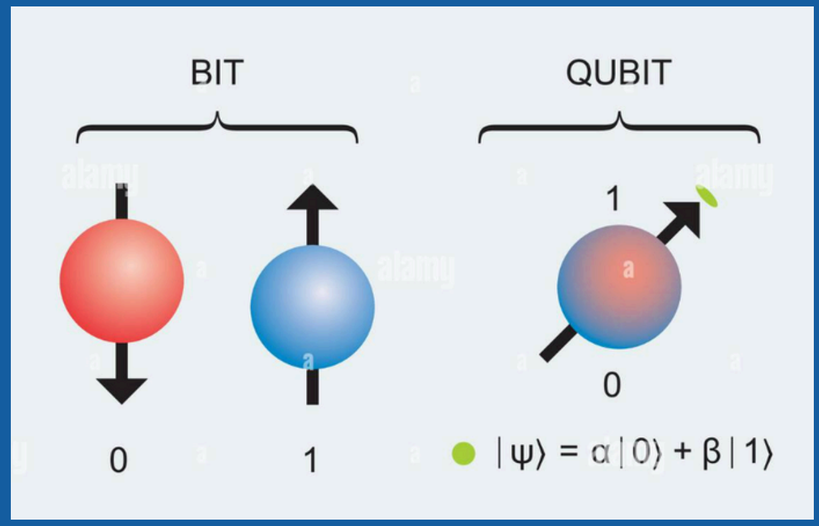
Role of Quantum Physics in QKD

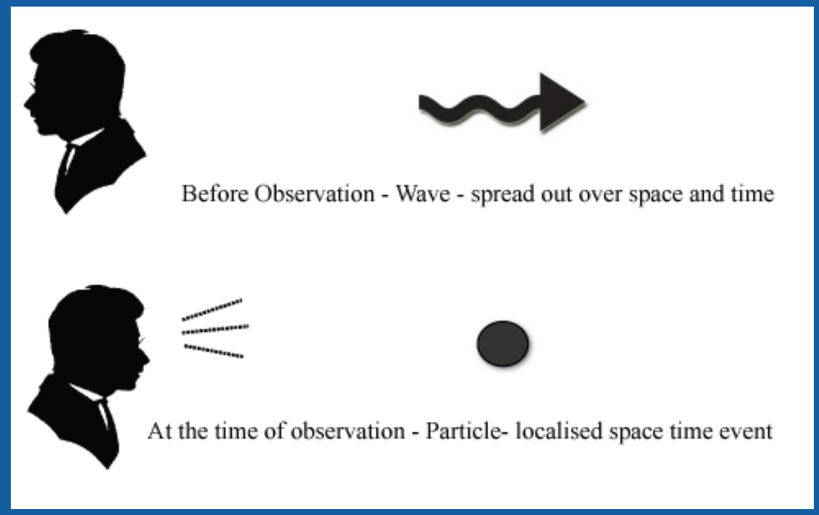
- **Quantum Superposition and Measurement**: a quantum system can exist in multiple states simultaneously, but when a measurement is made, the system collapses into one of its possible states. This property is used in QKD to create and share keys

- **Quantum Entanglement:** This is a phenomenon where two or more particles become linked and the state of one particle is directly related to the state of the other. used to create correlated qubits.

- **Observer Effect:** In quantum mechanics, measuring a quantum state will alter state. This principle is used in QKD to detect eavesdropping. If a third party tries to measure the quantum key, they can be detected.

- **Use of Photons:** QKD often uses photons (particles of light) as information carriers. The quantum states of these photons (such as their polarization or phase) form the basis of the quantum key





Quantum Gates

Quantum gates are the fundamental operations used in quantum computing to manipulate quantum bits, or qubits. They are the quantum equivalent of classical logic gates and form the basis for quantum algorithms.

1. **Nature of Quantum Gates**: Unlike classical gates, quantum gates allow a superposition of states and must satisfy certain criteria such as **reversibility**. They enable transformations that are not possible in classical computing, thanks to the principles of superposition and entanglement.
2. **Representation:** Quantum gates are represented by unitary matrices. A gate that acts on ‘n’ qubits is represented by a 2^n x 2^n unitary matrix. The quantum states that the gates act upon are unit vectors in complex dimensions, with the complex Euclidean norm (the 2-norm)
3. Here are the major types of quantum gates

- Identity Gate
- Pauli Gates
- Hadamard Gates
- Phase Gates
- SWAP Gates

Quantum Gates

- **Identity gate** - often represented as I, is a fundamental quantum gate used in quantum computing. The Identity gate is a single-qubit operation that leaves the basis states |0⟩ and |1⟩ unchanged. It is the simplest quantum gate and performs no operation on the qubit, leaving it in its original state.

- **phase gates** - The application of a phase gate to a qubit results in a change in the phase of the qubit's state. This change in phase can have significant effects when the qubit is entangled with others, leading to a variety of uses in quantum algorithms.

- **SWAP gates** - After applying the SWAP gate, the state of the first qubit becomes the state of the second qubit, and vice versa. The SWAP gate is reversible, meaning you can apply it again to restore the original states of the qubits

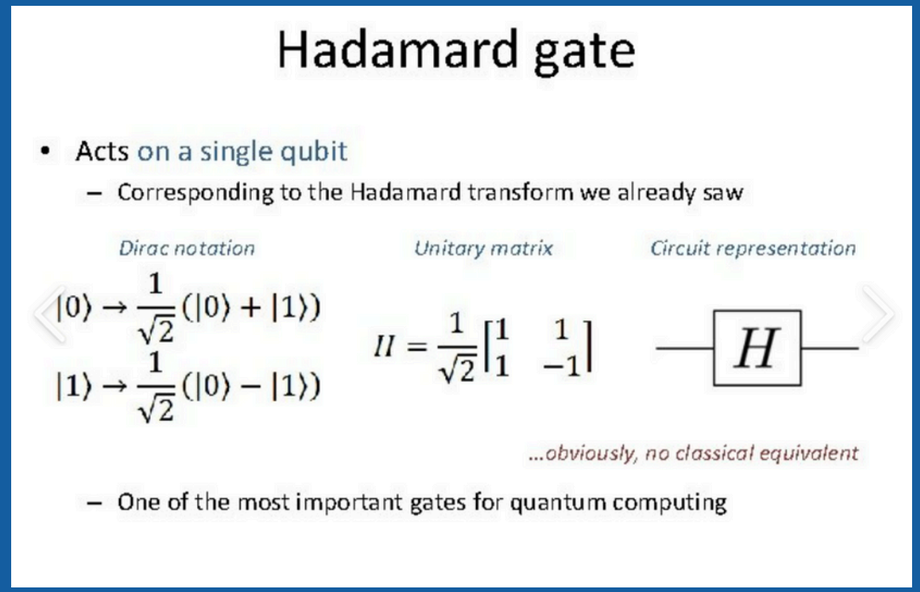
$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Quantum Gates

- **Hadamard gate** - The Hadamard gate is a fundamental operation in quantum computing. The Hadamard gate is a single-qubit operation that puts a qubit into a superposition of states. This means that when the qubit is measured, it will collapse to either |0⟩ or |1⟩ with equal probability. The Hadamard gate maps the basis state |0⟩ to (|0⟩ + |1⟩)/√2 and |1⟩ to (|0⟩ - |1⟩)/√22. This creates an equal superposition of the two basis states. In terms of rotations, this corresponds to a rotation of π radians around the Z-axis followed by π/2 (90 degrees) around the Y-axis. The Hadamard gate is one of the most important gates in quantum computing. It is often used to create superposition states, which are crucial for many quantum algorithms.



Quantum Gates

- **Pauli X gate** - Also known as the bit-flip gate, the Pauli-X gate performs a bit-flip operation on a qubit, changing a |0⟩ to |1⟩ or a |1⟩ to |0⟩. The X gate follows the same logic as the classical NOT operator.

- **Pauli Y gate** -_The Pauli-Y gate performs both a bit-flip and a phase-flip operation on a qubit. The Y gate describes the combined effect of both the bit- and the phase-flip.

- **Pauli Z gate** - Also known as the phase-flip gate, the Pauli-Z gate causes rotation around the z-axis by π radians. The Z gate flips |+⟩ to |-⟩ and |-⟩ to |+⟩. It does not affect the |0⟩ and |1⟩ states as they lie on the z-axis.

These gates are essential building blocks for more complex quantum circuits and algorithms. These operators are also called sigma operators (usually when we use the notation σx, σy, σz) or (when written as matrices in the standard basis, as we have done) as Pauli spin matrices.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Methodology

- Theoretical Implementation of QKD: Simulating QKD protocols to generate secure keys, including procedures for qubit preparation, transmission, measurement, and key distillation.
- Practical Application Development: Developing a secure chat application ("QuantumChat") integrating the simulated QKD system for secure key exchange and utilizing established cryptographic algorithms for message encryption.

1. Literature Review:
Research on quantum cryptography, QKD protocols (BB84, E91,T22), and their potential applications in healthcare

↓

2. QKD Protocol Simulation: Implementing the T22 protocol using programming languages like Python and libraries like Qiskit. This involved simulating the generation and measurement of entangled Bell states to generate shared secret keys.

↓

3. Secure Chat Application Development: Developing "**QuantumChat**" using a client-server architecture. The client-side, responsible for user interface and message handling, was developed using Svelte, while the server-side, managing connections and key distribution, was built with TypeScript and Node.js.

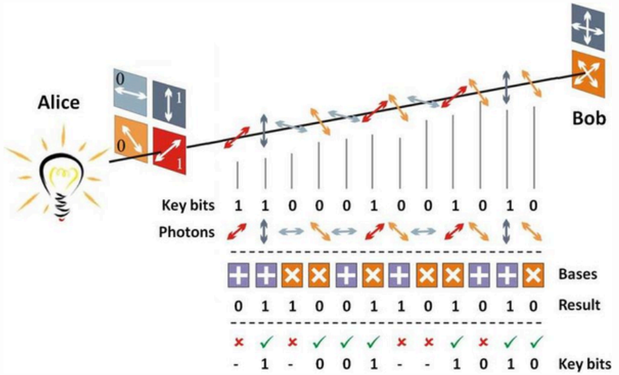
↓

4. Integration and Testing: Integrating the simulated QKD system with the chat application to enable secure key exchange for each new conversation. This involved rigorous testing to ensure the accuracy and security of the key generation and distribution process.

↓

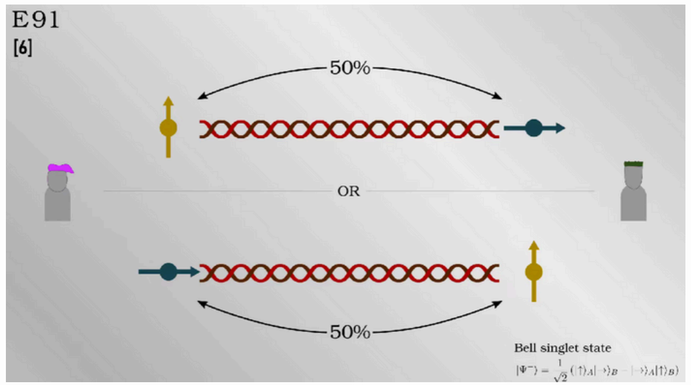
5. Evaluation and Analysis: Evaluating the performance and security of the implemented system through simulation and analysis of its resistance to eavesdropping attempts.

QKD Protocols



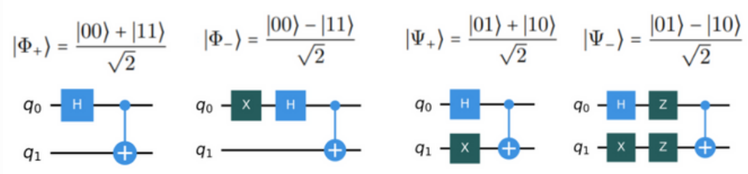
BB84 Protocol

Developed in 1984, Charles Bennett and Gilles Brassard developed the first quantum key distribution protocol called the BB84 protocol, which uses the encoding of classical bits into qubits, i.e., two-level quantum systems.



E91 Protocol

The E91 protocol utilizes Bell's inequality to verify the entanglement and detect eavesdropping. Violations of Bell's inequality confirm the presence of quantum entanglement and ensure the security of the key exchange process.



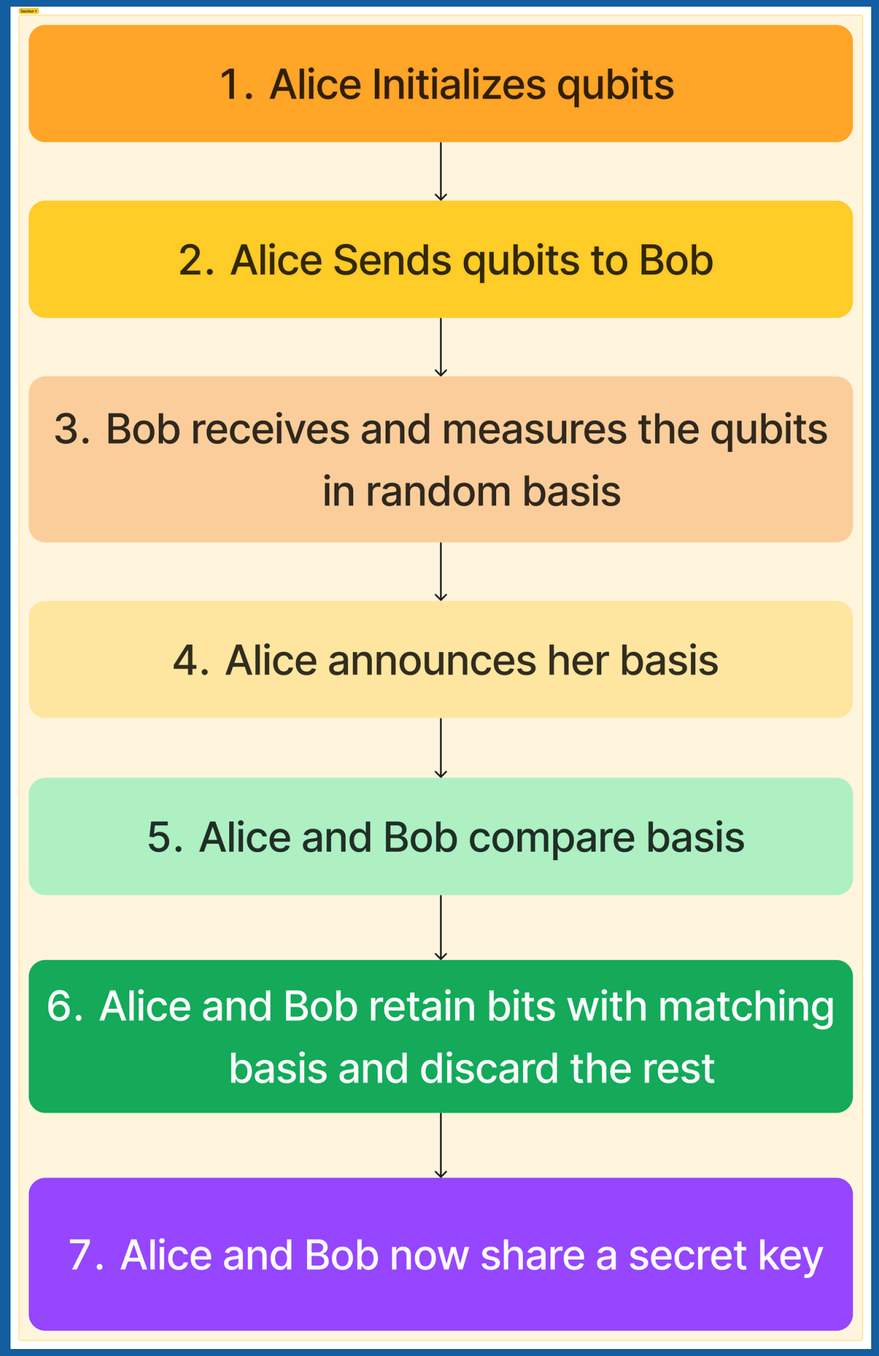
Bell States (T22)

The T22 protocol uses entangled Bell states to generate a quantum key and is based on a theoretical proposition by Dan Song and Dongxu Chen

BB84 QKD Protocol

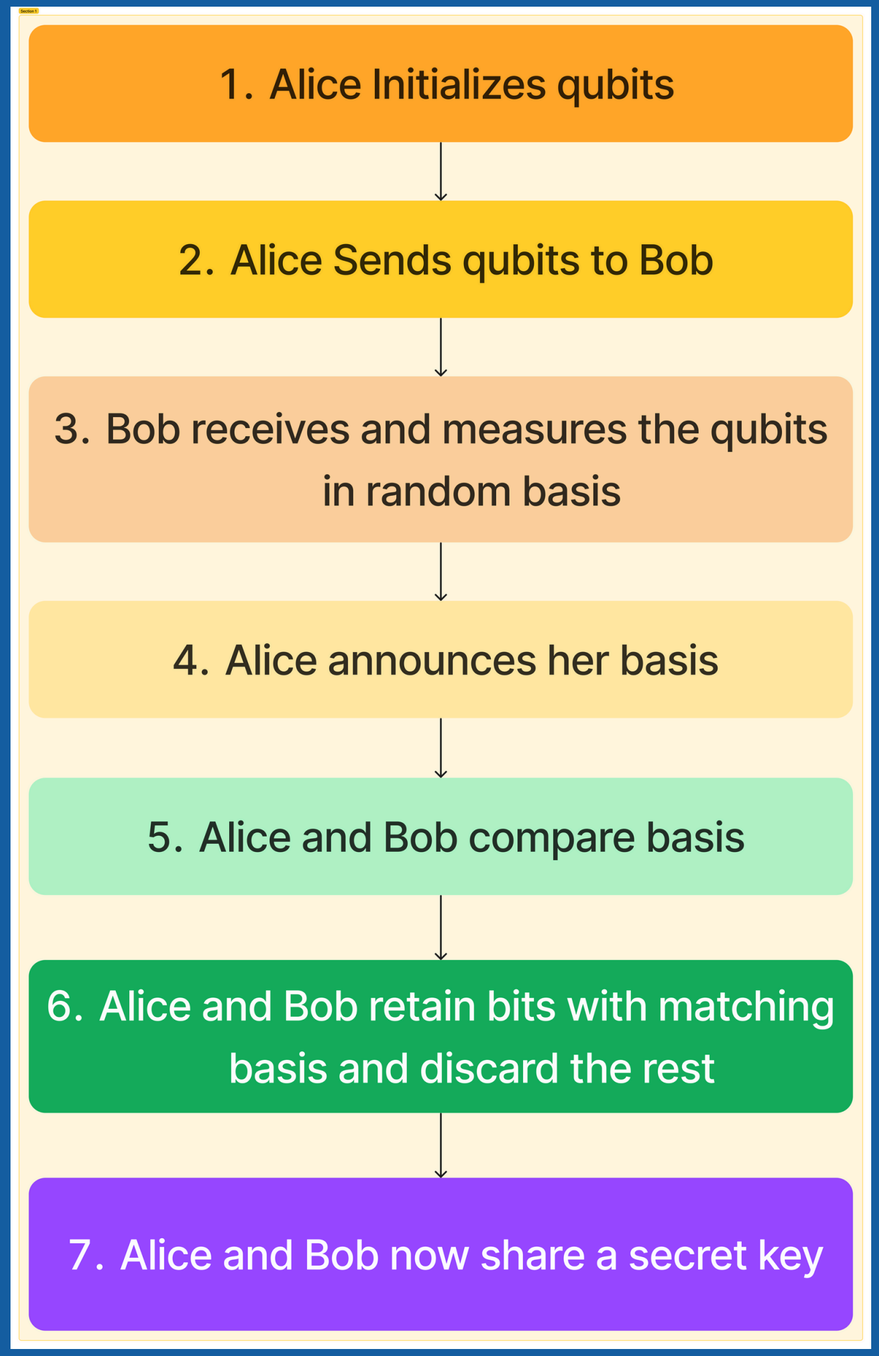
The BB84 protocol uses single qubits that can be in either a |0⟩ or a |1⟩ state in the Z-basis or in the |+⟩ or |-⟩ state in the X-basis.

- **Alice's initialization:**
  - Alice chooses two random bit strings a and b
  - a = 1010, b = 1100
- **Encoding**:
  - Alice encodes each bit in a using the basis specified by b:
    - if b = 0, use Z-basis. If b =1, use X-basis
    - a1=1, b1=1 (X-basis) → $|+\rangle$
    - a2=0, b2=1 (X-basis) → $|-\rangle$
    - a3=1, b3=0 (Z-basis) → $|1\rangle$
    - a4=0, b4=0 (Z-basis) → $|0\rangle$
- **Transmission**:
  - Alice sends the qubits $|+\rangle, |-\rangle, |1\rangle, |0\rangle$ to Bob over quantum channel



BB84 QKD Protocol

- **Bob's Measurement:**
  - Bob receives the qubits and measures each one in a randomly chosen basis:
    - Measurement bases (chosen randomly by Bob):
      - $b1'=0$ (Z basis)
      - $b2'=1$ (X basis)
      - $b3'=0$ (Z basis)
      - $b4'=1$ (X basis)
    - Measurement results:
      - $1a1'=1$ (measured in Z basis)
      - $a2'=0$ (measured in X basis)
      - $a3'=1$ (measured in Z basis)
      - $a4'=1$ (measured in X basis)
- **Basis Announcement:**
    - Alice publicly announces her basis string $b=1100$.
- **Bit Comparison:**
  - Alice and Bob compare their basis strings b and b′:
    - They keep bits where $bi=bi'$:
      - $i=3$ (Both used Computational basis, $a3=1$, $a3'=1$)
    - They discard the rest.
- **Result**: Alice and Bob now share a secret key, for example, "1", after going through the protocol steps.
- In summary, Alice encodes and sends qubits, Bob measures them, they compare bases, keep matching bits, estimate errors, and then finalize a secure key.



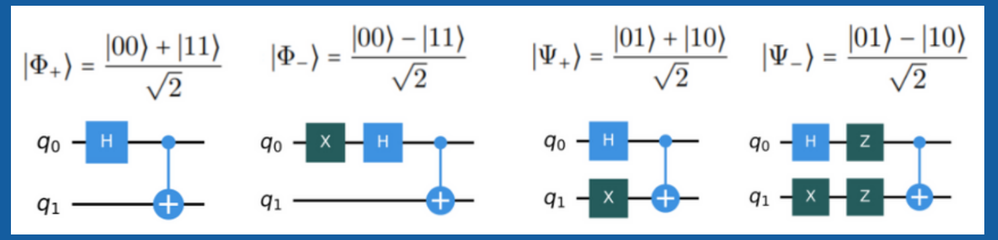
T22 QKD Protocol

- **Bell states:**
  - Maximal superpositions that entangle two bits: **|Φ+⟩, |Φ−⟩, |Ψ+⟩, |Ψ−⟩**
- **Qubit Entanglement Pairing:**
  - Using 4 qubits, 3 pairings of qubits are possible.
- **Bell State Groups:**
  - Groups indicate which Bell state operations they use on the qubit pairings

- Pairings decide how the four qubits will be entangled using CNOT gates.

- Groupings indicate which Bell state operations they use on the qubit pairings

- The same pairing and grouping from both will create a symmetric circuit yielding a |0000> measurement for Bob.



Pair 1	Pair 2	Label
(q0, q1)	(q2, q3)	A
(q0, q2)	(q1, q3)	B
(q0, q3)	(q1, q2)	C

Pair 1	Pair 2	Groupings
$\|\Phi_+\rangle$	$\|\Phi_-\rangle$	00
$\|\Phi_-\rangle$	$\|\Phi_+\rangle$	01
$\|\Psi_+\rangle$	$\|\Psi_-\rangle$	10
$\|\Psi_-\rangle$	$\|\Psi_+\rangle$	11

T22 QKD Protocol

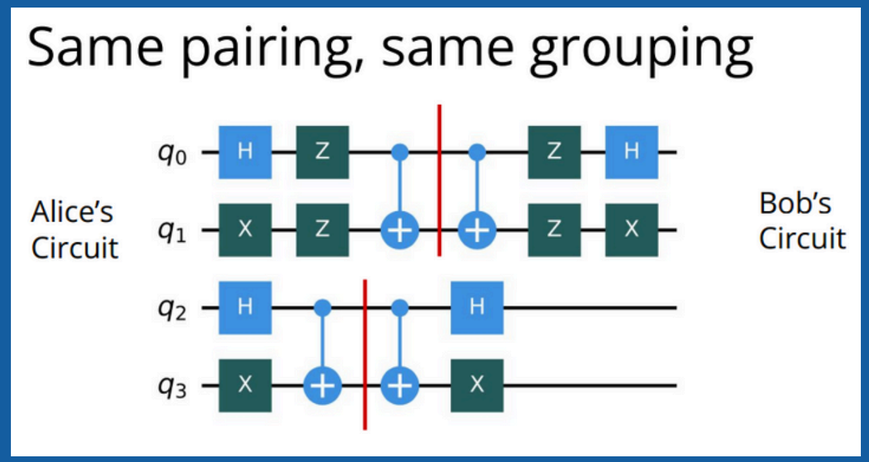
Steps:

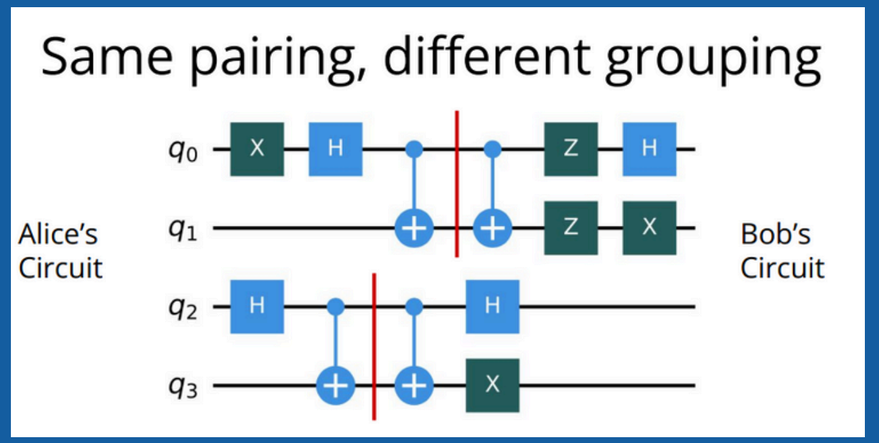
1. Alice chooses a random pairing and grouping
2. Alice sends their entangled qubits to Bob
3. Bob chooses a random pairing and grouping and performs the reverse bell state circuit
4. Bob measure the results in the computational basis
5. Bob knows Alice chose the same pairing and grouping if Bob measures all 0s
6. Alice and Bob classically discuss some of the measurements
7. Alice and Bob now have a shared secret key

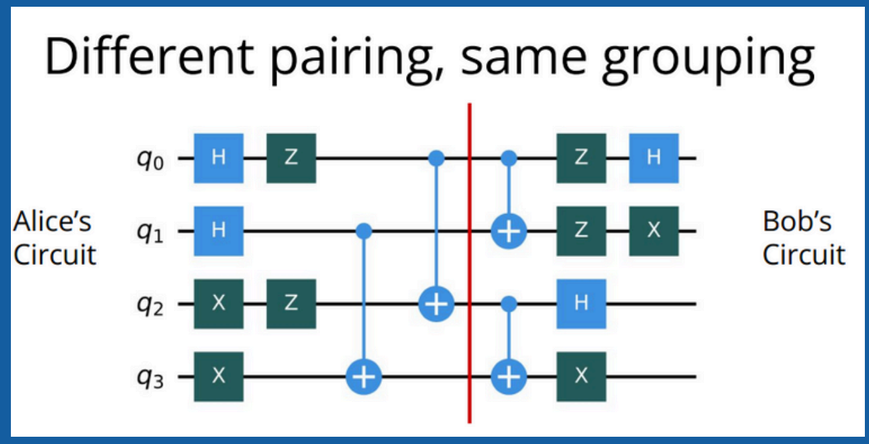
- **Example:**

Alice pairing	A	B	C	A	B	C	A
Alice group	11	10	01	00	11	10	01
Bob pairing	B	B	C	A	B	C	A
Bob group	11	10	01	00	00	10	01
Eve detection		Verified				Verified	
Final Key			01	00			01

Final Key is 010001

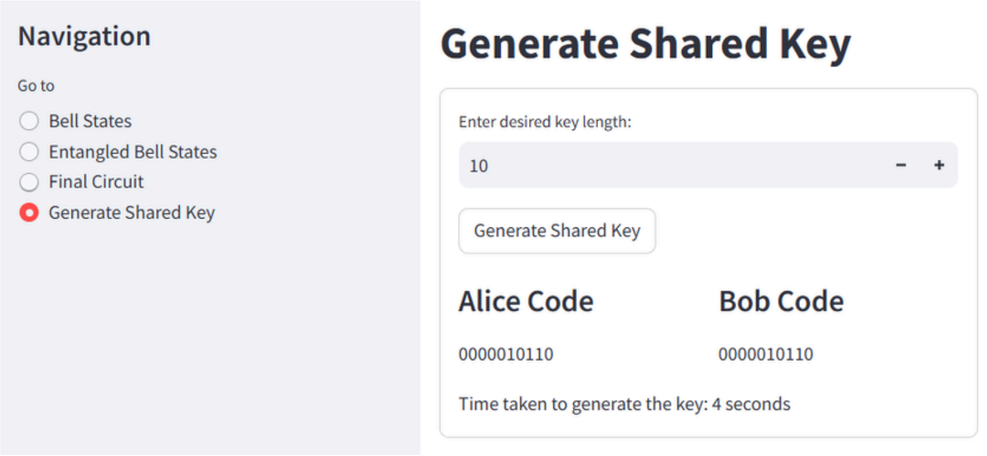






Simulation of QKD

- QKD model of Alice and Bob was modified to accommodate a software simulation of QKD rather than individuals with two quantum computers and a quantum channel.
- In the simulation, users are represented by JSON files to keep the flow of information the same – for example, the JSON file representing Alice will never contain Bob's measurements.
- Both the BB84 protocol and the T22 protocol were run for 1000 samples using the IBM Qiskit Aer software simulator backend and the IBM Qiskit API for Python.
- A Streamlit web interface was built to showcase the shared key generation using QKD

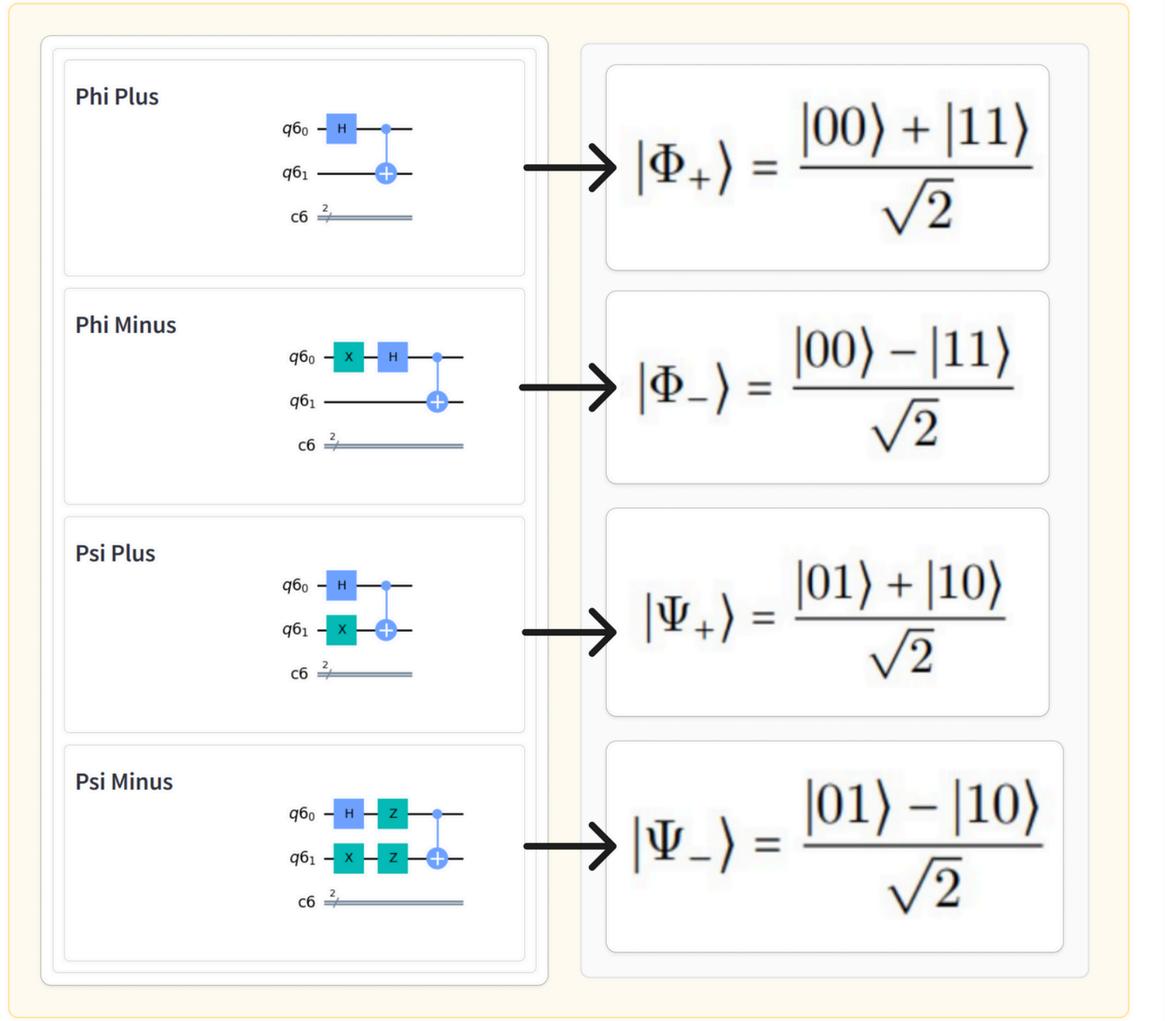


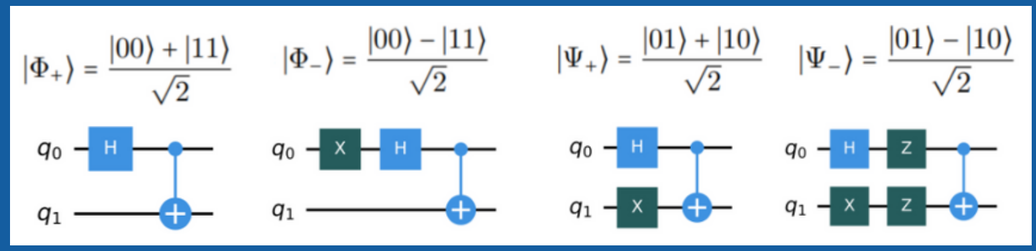
qkd_sim.py

```python
from qiskit import Aer
from qiskit import execute
from qiskit.circuit import QuantumRegister, ClassicalRegister, QuantumCircuit


def quantum_compute(alice_data, bob_data):
    alice_qpairs = alice_data["pairings"]
    alice_groupcodes = alice_data["groupings"]

    bob_qpairs = bob_data["pairings"]
    bob_groupcodes = bob_data["groupings"]

    # Keeps track of the indices in Bob's list that were correct guesses
    correct_guesses = []

    for i in range(len(alice_qpairs)): # iterate over  qubit-pairs

        # Obtain both users' group codes
        alice_gc = alice_groupcodes[i]
        bob_gc = bob_groupcodes[i]

        qpairs = Pairing(alice_qpairs[i][0], alice_qpairs[i][1])

        # Build Alice's circuits based on her inputs
        qc, q = entangling_circuit_map[alice_gc](qpairs)

        qpairs = Pairing(bob_qpairs[i][0], bob_qpairs[i][1])

        # Run Bob's test circuits on top of Alice's input
        reversal_circuit_map[bob_gc](qpairs, q, qc)

        # Add the index to the list of correct guesses to return to the users
        if (verify_circuit(qc)):
            correct_guesses.append(i)
            print("Bob's guess was correct")

    alice_data["correct_measurements"] = correct_guesses
    bob_data["correct_measurements"] = correct_guesses

    return alice_data, bob_data

def generate_code(groupings):
    code = ""
    for i in groupings:
        if i == 0:
            code += "00"
        if i == 1:
            code += "01"
        if i == 2:
            code += "10"
        if i == 3:
            code += "11"
    return code
```

Simulation of QKD

Quantum Circuits built using Qiskit Python Library





gates.py

```python
from qiskit import Aer
from qiskit import execute
from qiskit.circuit import QuantumRegister, ClassicalRegister, QuantumCircuit

def phi_plus(bit0, bit1, qc):
    qc.h(bit0)
    qc.cx(bit0, bit1)

def phi_minus(bit0, bit1, qc):
    qc.x(bit0)
    qc.h(bit0)
    qc.cx(bit0, bit1)

def psi_plus(bit0, bit1, qc):
    qc.h(bit0)
    qc.x(bit1)
    qc.cx(bit0, bit1)

def psi_minus(bit0, bit1, qc):
    qc.h(bit0)
    qc.x(bit1)
    qc.z(bit0)
    qc.z(bit1)
    qc.cx(bit0, bit1)
```

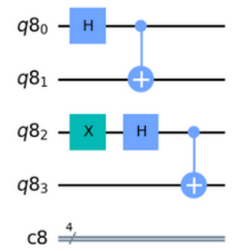
Simulation of QKD

Quantum Circuits built using Qiskit Python Library

circuits.py

```python
def circuit00(pairs):
    q = QuantumRegister(qbits)
    b = ClassicalRegister(qbits)
    qc = QuantumCircuit(q, b)

    phi_plus(q[pairs.bit0], q[pairs.bit1], qc)
    phi_minus(q[pairs.bit2], q[pairs.bit3], qc)
    print("Alice chose group 00")
    return qc, q
```

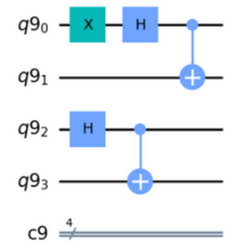
Group 00



circuits.py

```python
def circuit01(pairs):
    q = QuantumRegister(qbits)
    b = ClassicalRegister(qbits)
    qc = QuantumCircuit(q, b)

    phi_minus(q[pairs.bit0], q[pairs.bit1], qc)
    phi_plus(q[pairs.bit2], q[pairs.bit3], qc)
    print("Alice chose group 01")
    return qc, q
```

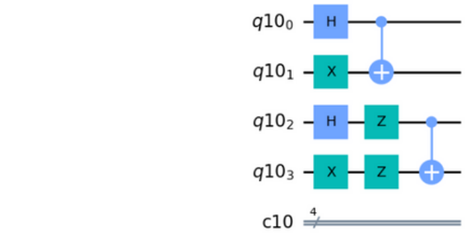
Group 01



circuits.py

```python
def circuit10(pairs):
    q = QuantumRegister(qbits)
    b = ClassicalRegister(qbits)
    qc = QuantumCircuit(q, b)

    psi_plus(q[pairs.bit0], q[pairs.bit1], qc)
    psi_minus(q[pairs.bit2], q[pairs.bit3], qc)
    print("Alice chose group 10")
    return qc, q
```

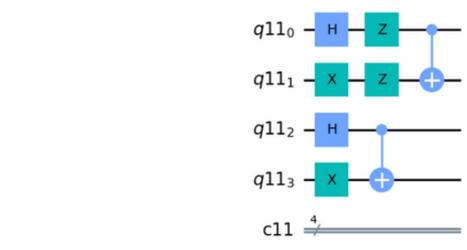
Group 10



circuits.py

```python
def circuit11(pairs):
    q = QuantumRegister(qbits)
    b = ClassicalRegister(qbits)
    qc = QuantumCircuit(q, b)

    psi_minus(q[pairs.bit0], q[pairs.bit1], qc)
    psi_plus(q[pairs.bit2], q[pairs.bit3], qc)
    print("Alice chose group 11")
    return qc, q
```

Group 11



Simulation of QKD

Simulation Results of all three QKD protocols

- The T22 protocol enhances security by reducing the probability of successful eavesdropping attempts. The probability of generating a correct circuit is calculated as: $P_{\text{correct}} = \left(\frac{1}{3}\right) \times \left(\frac{1}{4}\right) \times 1 \times 1 = \frac{1}{12}$
- Eavesdropper (Eve) Success Probability:

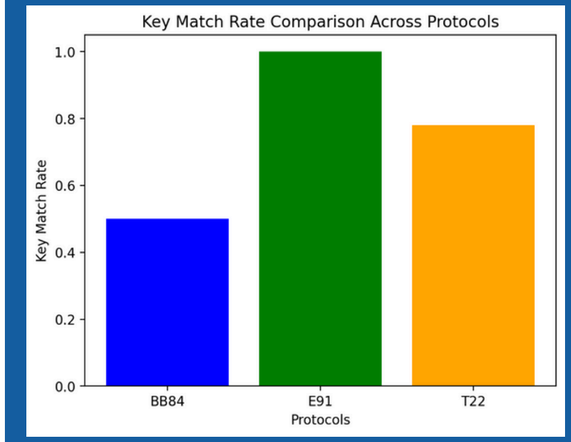
$$P_{\text{infiltrate}} = (P_{\text{correct}}) \times (\text{Eve same pair and group})$$

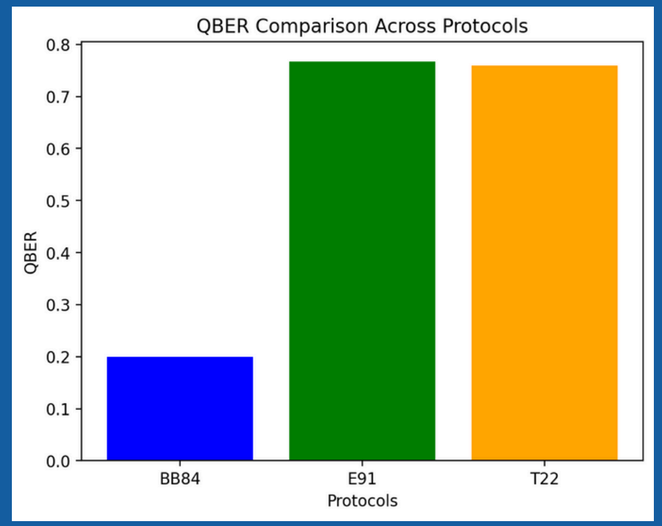
$$P_{\text{infiltrate}} = \frac{1}{12} \times 1 = \frac{1}{12}$$

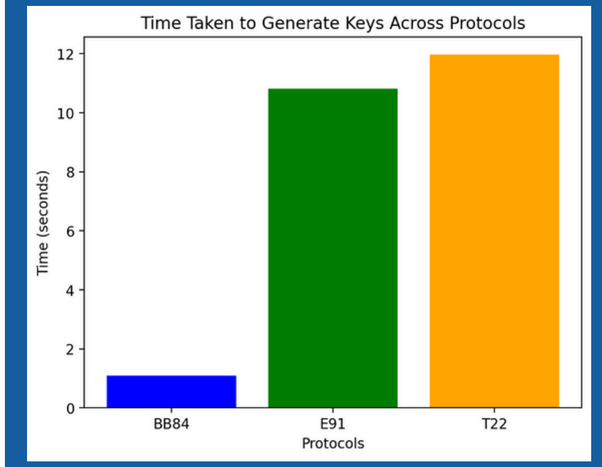
- This decreased probability, compared to protocols like BB84, makes it significantly more challenging for an eavesdropper (Eve) to infiltrate the key generation process undetected.
- This T22 protocol offers enhanced security against eavesdropping compared to BB84, at the cost of increased complexity and lower key generation efficiency.

Comparison with BB84

- T22 uses 4 qubits per circuit to generate a 2-bit key, while BB84 uses 1 qubit for 1 bit.
- T22 has a lower correct circuit probability, enhancing security against eavesdropping.
- In the simulation, both protocols use 256 shots per circuit and measure only in the Z basis for consistency.
- The T22 protocol uses Bell states and their measurements, adding complexity to the key distribution process.







Simulation of QKD

Simulation Results of all three QKD protocols

- The T22 protocol enhances security by reducing the probability of successful eavesdropping attempts. The probability of generating a correct circuit is calculated as: $P_{\text{correct}} = \left(\frac{1}{3}\right) \times \left(\frac{1}{4}\right) \times 1 \times 1 = \frac{1}{12}$
- Eavesdropper (Eve) Success Probability:

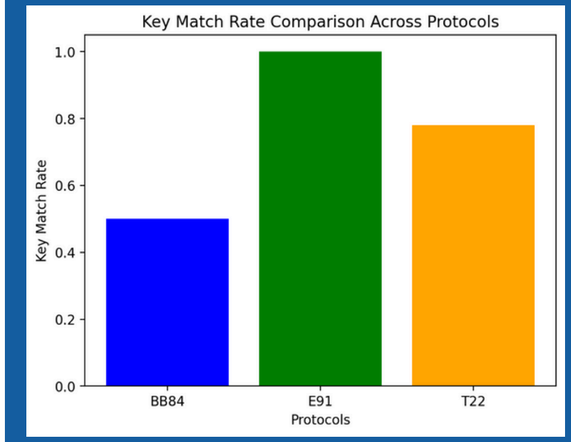
$$P_{\text{infiltrate}} = (P_{\text{correct}}) \times (\text{Eve same pair and group})$$

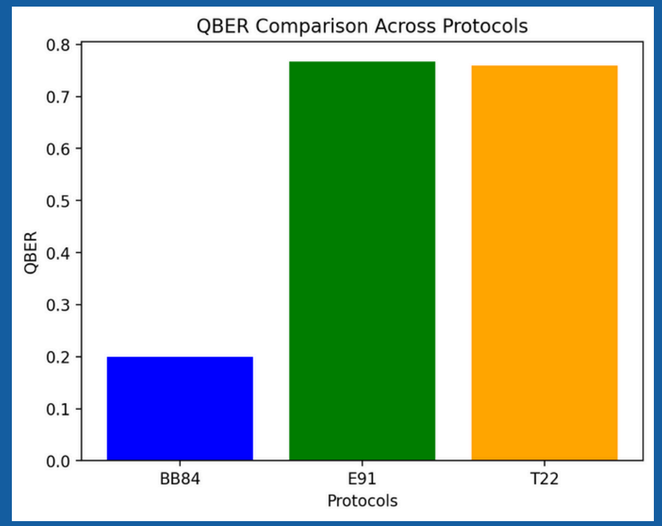
$$P_{\text{infiltrate}} = \frac{1}{12} \times 1 = \frac{1}{12}$$

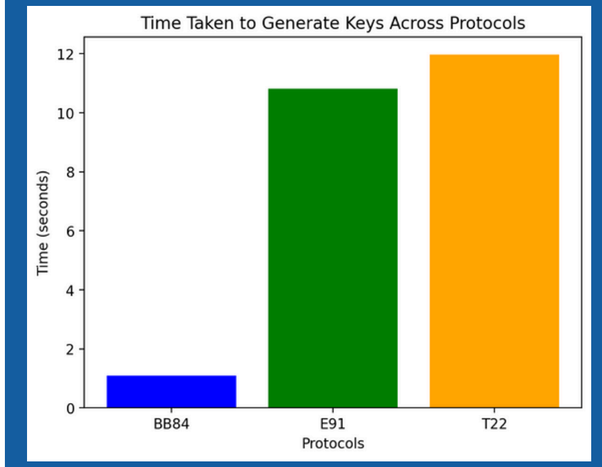
- This decreased probability, compared to protocols like BB84, makes it significantly more challenging for an eavesdropper (Eve) to infiltrate the key generation process undetected.
- This T22 protocol offers enhanced security against eavesdropping compared to BB84, at the cost of increased complexity and lower key generation efficiency.

Comparison with BB84

- T22 uses 4 qubits per circuit to generate a 2-bit key, while BB84 uses 1 qubit for 1 bit.
- T22 has a lower correct circuit probability, enhancing security against eavesdropping.
- In the simulation, both protocols use 256 shots per circuit and measure only in the Z basis for consistency.
- The T22 protocol uses Bell states and their measurements, adding complexity to the key distribution process.





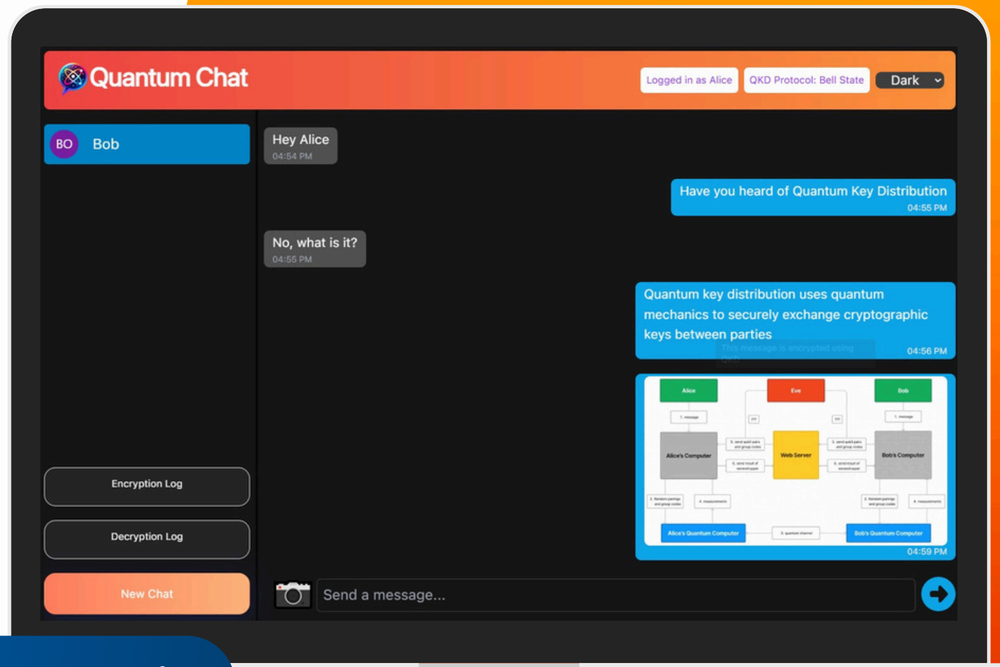


Implementation in Quantum Chat

Quantum Chat is a secure messaging platform that uses Quantum Key Distribution (QKD) to keep conversations safe.

- **Server-Side**: Built with TypeScript to handle WebSocket connections and key distribution.
- **Client-Side**: Uses Svelte for user interface and IndexedDB for local message storage.
- **Quantum Keys**: Keys are generated with QKD, making them extremely secure.
- **Hybrid Encryption**: Combines AES-CTR for message encryption and RSA for key exchange.

.



QKD

- Generates encryption keys using 3 QKD protocols
- Ensures keys are virtually impossible to intercept or duplicate.

Hybrid Encryption

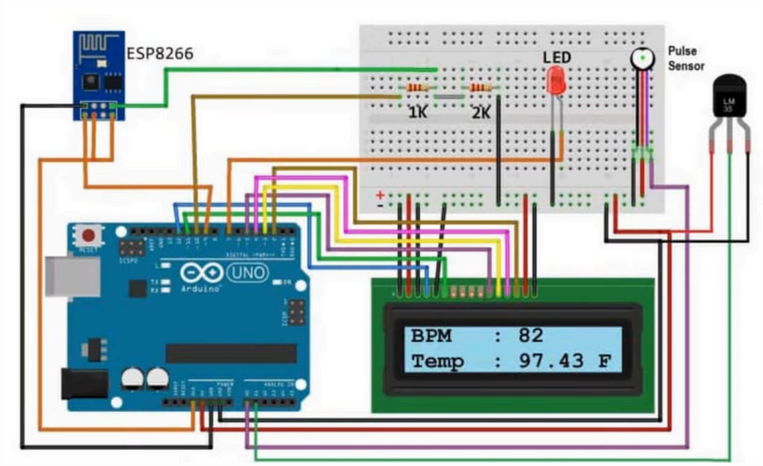
- Combines AES-CTR for encrypting messages and RSA for secure key exchange.
- Provides end-to-end encryption for all communications.

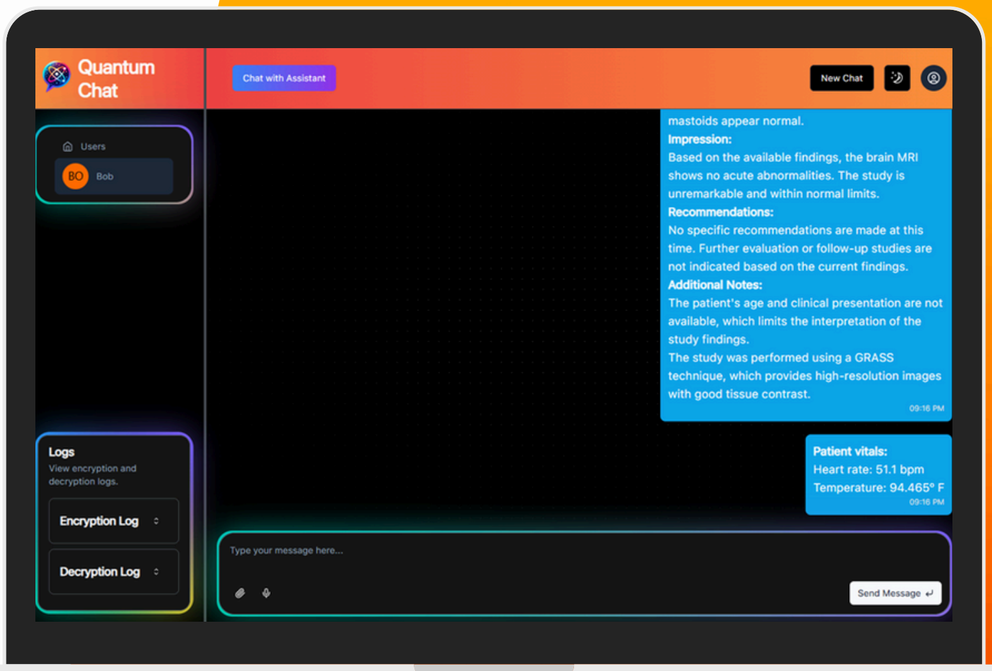
Image Encryption

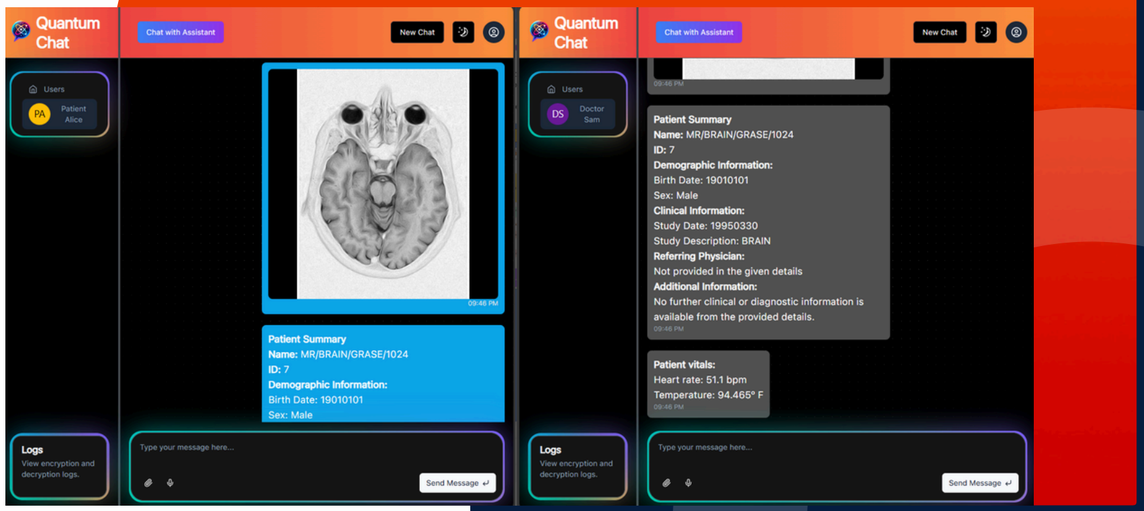
- Securely encrypts and transmits images along with text messages.
- Protects media content from unauthorized access.

Features of Quantum Chat

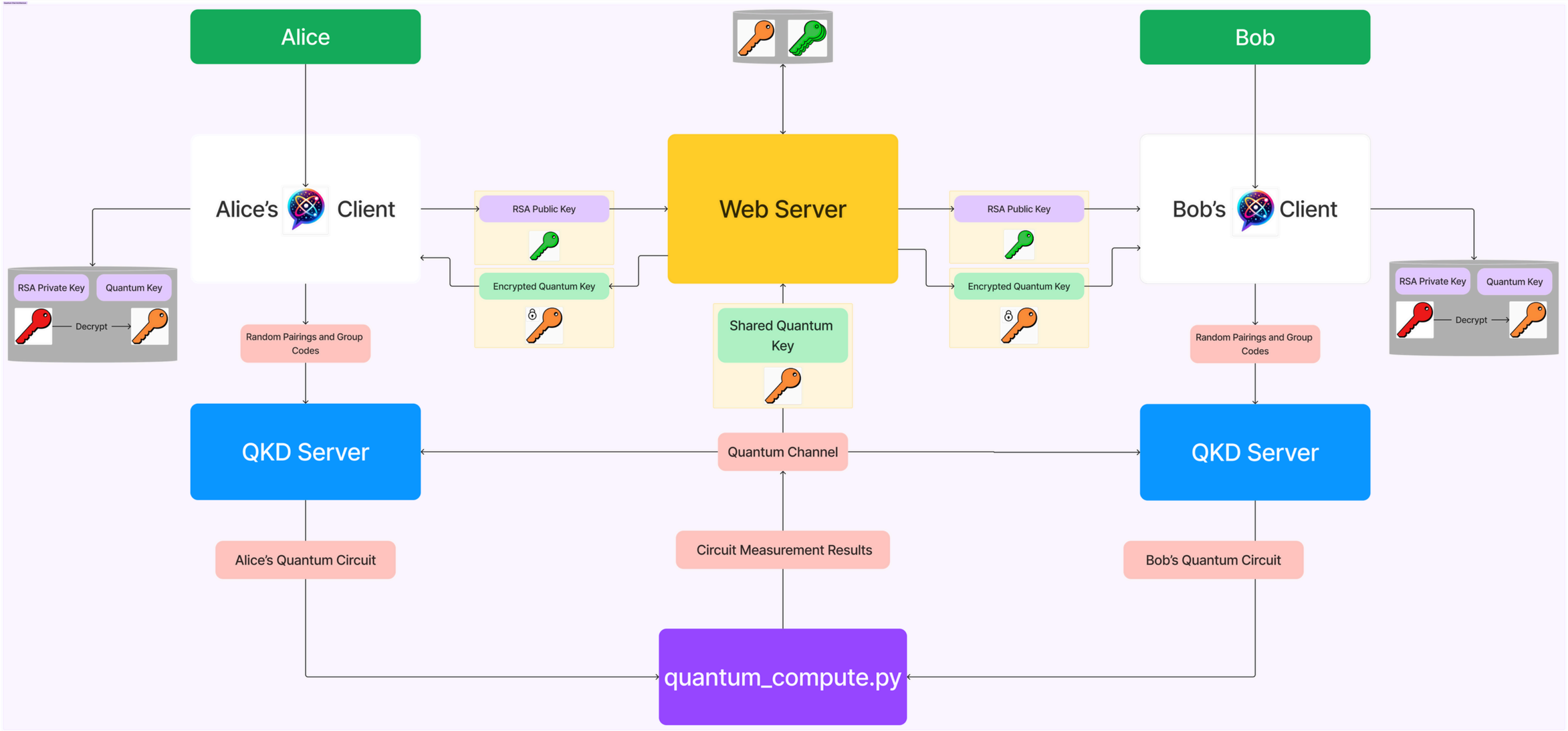
1. **Quantum-secured communication**
   - Ensures highest level of data protection for sensitive medical information
2. **Secure DICOM file sharing**
   - Enables safe transmission of **DICOM** medical imaging files
3. **Automatic JPEG conversion**
   - Allows quick preview of DICOM files without specialized software
4. **AI-powered metadata analysis**
   - Provides instant insights from image metadata using **Llama-3 7B LLM** running locally
5. **Real-time vital signs integration**
   - Incorporates live patient data from **Arduino**-based heart rate monitors via ESP8266 to Python database



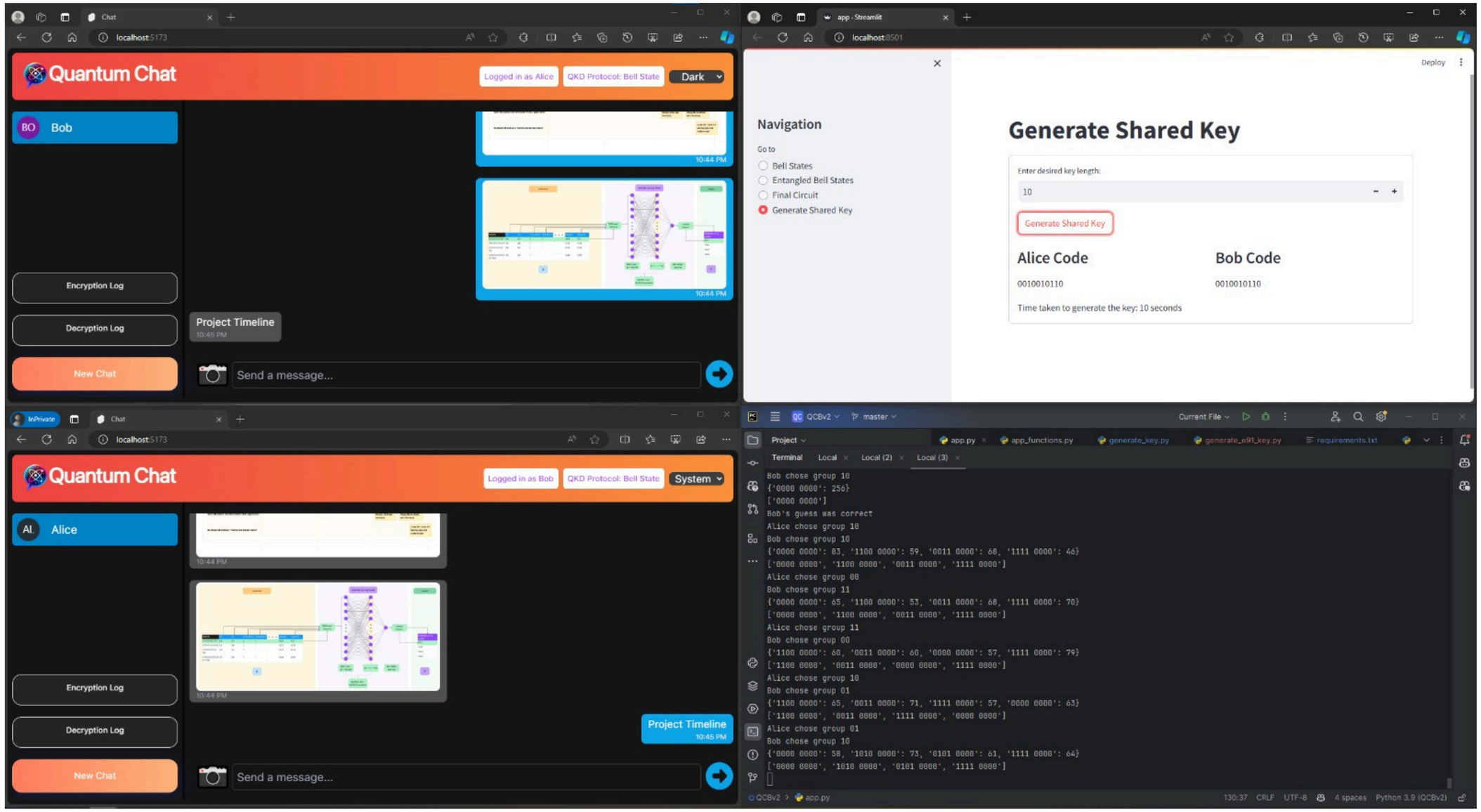




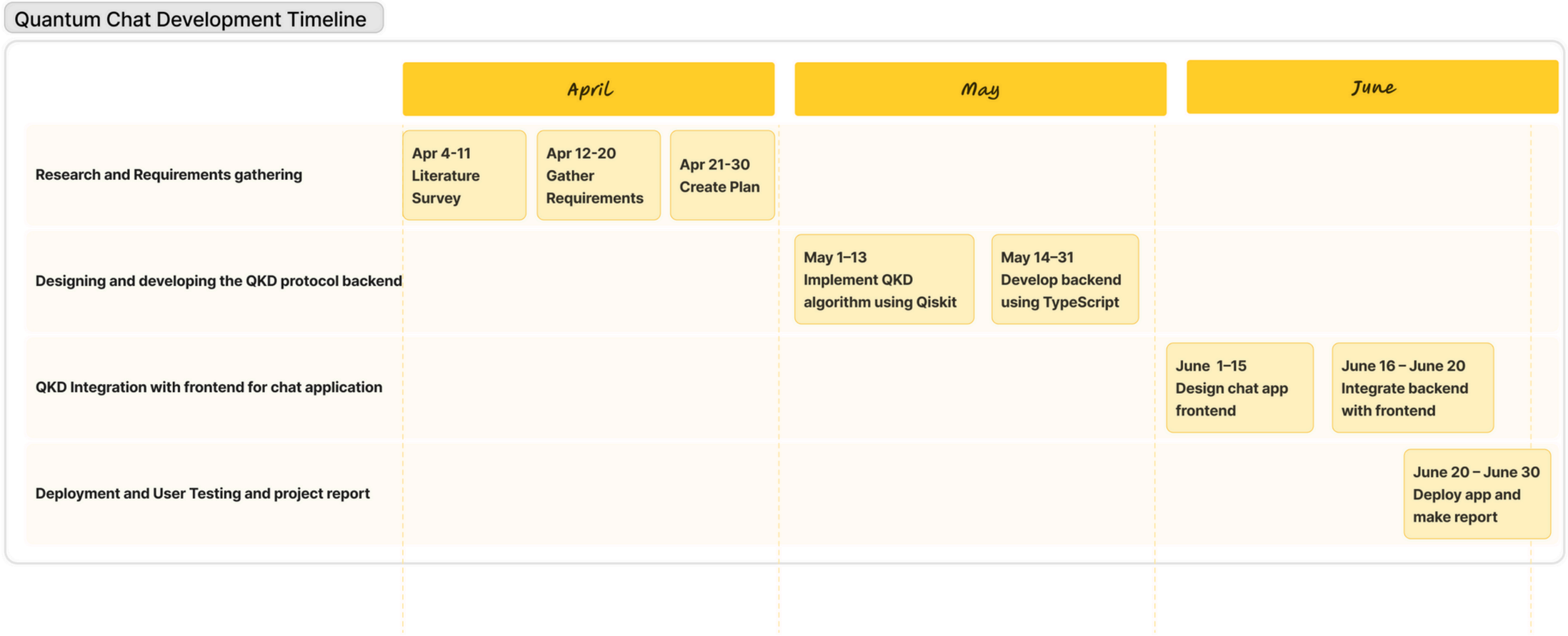
Quantum Chat Architecture



Quantum Chat Demo



Project Timeline



THANK YOU!