~Bachelor Thesis Finalization~

Development and Simulation of Application Based on Quantum Key Distribution (QKD)

Takihara Yudai, 4th year Bachelor student
The University of Aizu
Supervisor: Prof. Anh T. Pham

Jun. 28, 2024

Outline

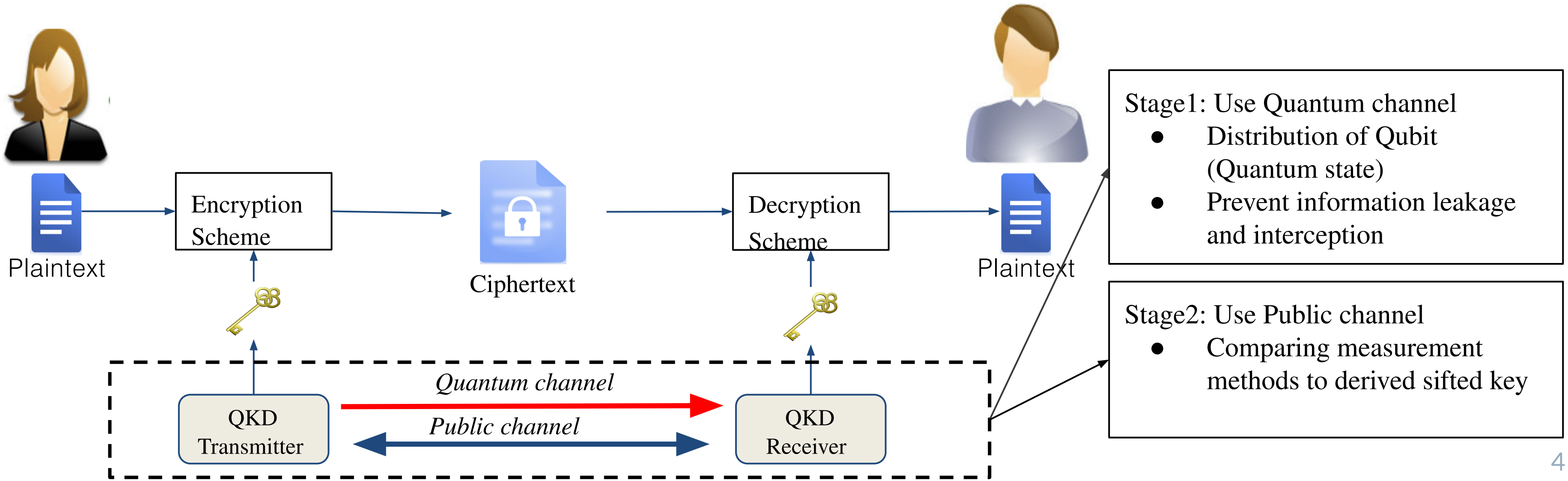
- **Research Background**
  - Quantum Key Distribution (QKD)
  - BB84 QKD Protocol
  - Secure Chat Application based on QKD
  - Research Motivation and Goals
- **System Description for QKD Application**
  - IBM Quantum simulator and Useful tool
  - Framework of QKD simulator Based IBM Platform
  - Flowchart of QKD Application
- **Implementation of the QKD Application**
  - Chat Application Demonstration
  - Implementation for key generation
  - Simulation Result of QKD application
- **Conclusion**

Outline

- **Research Background**
  - Quantum Key Distribution (QKD)
  - BB84 QKD Protocol
  - Secure Chat Application based on QKD
  - Research Motivation and Goals
- **System Description for QKD application**
  - IBM Quantum simulator and Useful tool
  - Framework of QKD simulator Based IBM Platform
  - Flowchart of QKD Application
- **Implementation of the QKD application**
  - Chat Application Demonstration
  - Implementation for key generation
  - Simulation Result of QKD application
- **Conclusion**

Quantum Key Distribution (QKD): Introduction

- QKD is a promising method to ***distribute secure keys*** secretly between legitimate users
  - It bases on the laws of quantum physics
  - First QKD protocol proposed by C. Bennett and G. Brassard in 1984, i.e., ***BB84 Protocol***
  - Some of best-known Japanese companies have been working on various QKD projects, e.g., Toshiba, NEC, and NTT



BB84 QKD Protocol

- BB84 uses photon polarization states to encode the bits of the key
- Each bit is encoded with a random polarization basis: + or ×

Base \ Bit	0	1
+	↕	↔
×	⤢	⤡

Alice

Bit	Base	Qubit State
0	+	↔
0	+	↔
1	+	↕
0	×	⤢

→

Bob

Base	Qubit State	outcome(bit)	Derived key
+	↔	0	0
×	⤡	1	discard
×	⤡	1	discard
×	⤢	0	0

BB84 QKD Protocol with Eavesdropper

- Eve intercepts Alice’s Qubit state to measure her Qubit. After that, she resends it to Bob.
  - Theoretically, Eve gets 50% sifted key information (50%: Probability of choosing the same basis as theirs)
    - If Eve has different basis as theirs, there is a 50% probability of a bit value error between Alice and Bob because of the change of Qubits states, .

Alice and Bob have about a 25% error rate in their sifted key.

➢ **They can detect the presence of Eve.**

Alice

Bit	Base	Qubit State
0	+	↔
0	×	↔
1	+	↕
0	×	⤡

Eve (eavesdropper)

Base	Qubit State	Bit
×	⤢	1
+	⤡	1
+	↕	1
×	⤡	0

Bob

Base	Qubit State	Bit
+	↕	0
×	⤡	1
+	↕	1
×	⤡	0

Secure QKD Application and Research Motivation

- Chat (e.g., WhatsApp, Line, and Skype) are the most popular Internet applications
  - Require a *high security level* against eavesdropping and surveillance by attackers
  - Current solution is the use of *public key cryptography* for sharing the key when starting the chat application -> be vulnerable to quantum computers

➡ **We want to apply the QKD for sharing the secret keys for each message exchange in chat applications**

❑ **Key question**: ***How to apply BB84 QKD protocol for Secure Chat Application?***

➢ Sender (Alice) encrypts the message using ***one-time pad (OTP) schemes*** for with the shared key by QKD system

  - *One-Time Pad*: The shared key is updated every few minutes to protect the risk from cryptanalysis and eavesdropping.

➢ Receiver (Bob) decrypts the ciphertext

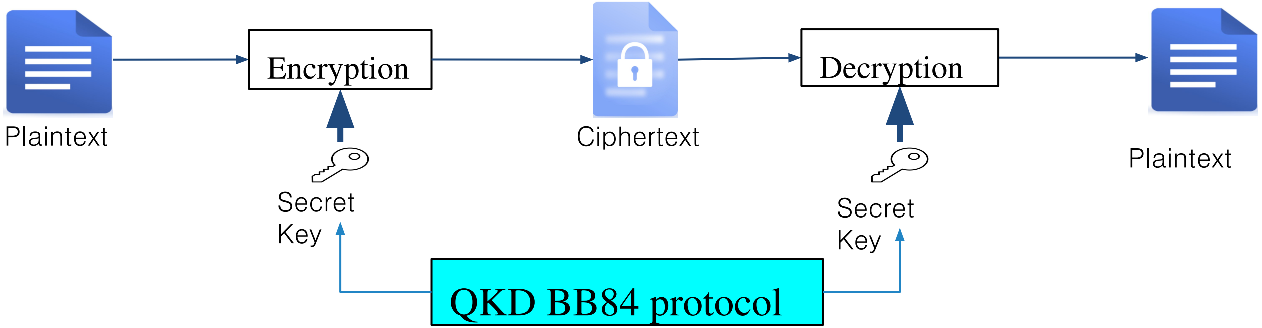
What my Goal?

Goal: To develop and simulate a **secure Chat Application based on QKD**

➢ ***The BB84 protocol*** can be applied for sharing secret keys between two legitimate users

Develop and simulate Secure Chat Application using quantum simulator from IBM

- ***Qiskit***, open source SDK for quantum computing and support to develop and simulate application
  - Generate Qubit (the basic unit information for quantum computing)
  - Develop secure application using Qubit according to the BB84 protocol.



Outline

- **Research Background**
  - Quantum Key Distribution (QKD)
  - BB84 QKD Protocol
  - Secure Chat Application based on QKD
  - Research Motivation and Goals
- **System Description for QKD application**
  - IBM Quantum simulator and Useful tool
  - Framework of QKD simulator Based IBM Platform
  - Flowchart of QKD Application
- **Implementation of the QKD application**
  - Chat Application Demonstration
  - Implementation for key generation
  - Simulation Result of QKD application
- **Conclusion**

IBM Quantum Experience (IQX)

- IQX: An open platform available for someone for simulation and development
  - Include a user interface that allows anyone to run experiments on both on a real quantum computer and a simulator for free.
  - Includes access to prototype quantum processing facility developed by the IBM along with the instructions, tutorials and interactive textbook for helping out the beginners.
  - Available with ***Qiskit***, an open source quantum computing framework and Python library developed by IBM.

IBM Quantum Experience (IQX): **Real quantum computer**

- Real quantum computer
  - The IBM-Q provides the facility to run quantum algorithm / circuits on real quantum computer. It can easily interact with the quantum computer by designing quantum circuit models of the calculations employed by IBM.

  - **Advantage**
    - Observe directly real-world behaviors and constraints that are difficult to reproduce in simulations, and gain more accurate and practical insight into quantum computation
      - Can be used to help develop the QKD application.
  - **Disadvantage**
    - Real quantum computers that are freely available access by **a lot of researchers of the whole world at a time**.
      - It can take several hours to run a quantum circuit once.

➢ We can instead use a simulator that can run quantum algorithms.

IBM Quantum Experience (IQX): **Simulator**

- Simulator (Local simulator)
  - Executable quantum algorithm / circuits on own local PC. The code to execute it is almost the same as when executing a real quantum computer.

  - **Advantage**
    - Verification of operation under different conditions
      - E.g. In QKD, it is possible to reproduce noise errors due to the environment of the quantum channel.

  - **Disadvantage**
    - Limitation by the capabilities of classical computers
      - The larger the number of quantum circuits to be run at one time, the exponentially longer the run time becomes.

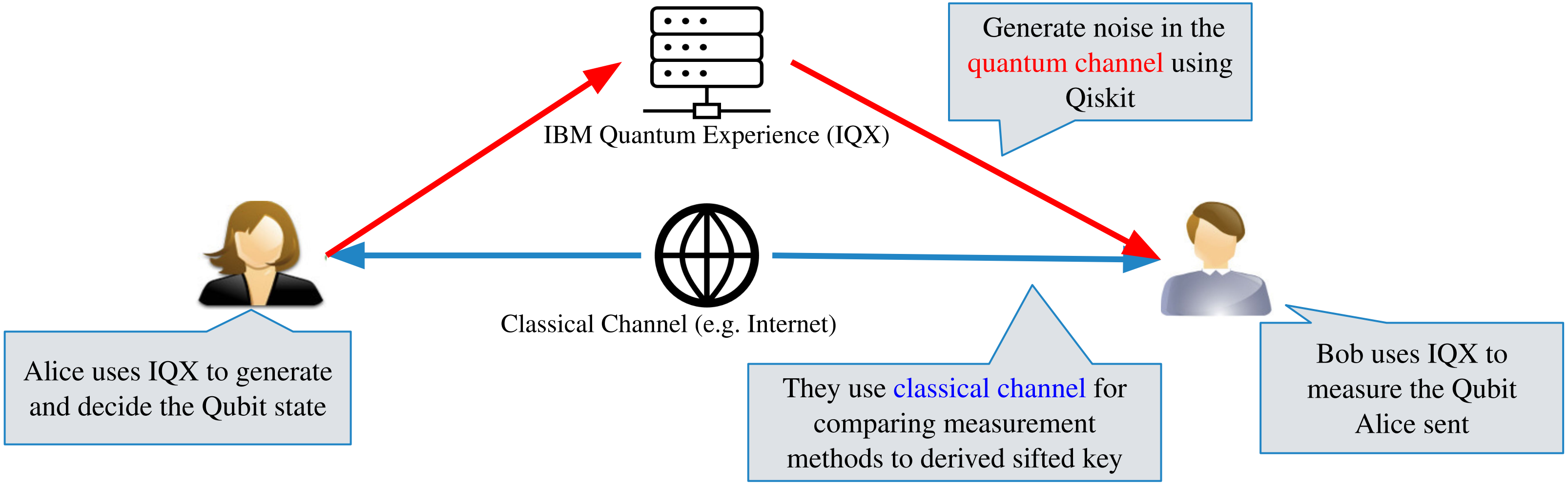
Useful tool: **Qiskit**

- ***Qiskit***: Open Source SDK for quantum computing from IBM and Qiskit community. It can be run on a local simulator or on a real quantum computer from IBM.
    - It supports researchers, developers, and students from learning quantum computing to developing real-world applications.
    - It is distributed as a Python library and can be used through the Python interface.
    - Quantum Circuit Design: Design quantum circuits using an easy to use intuitive API.

Qiskit

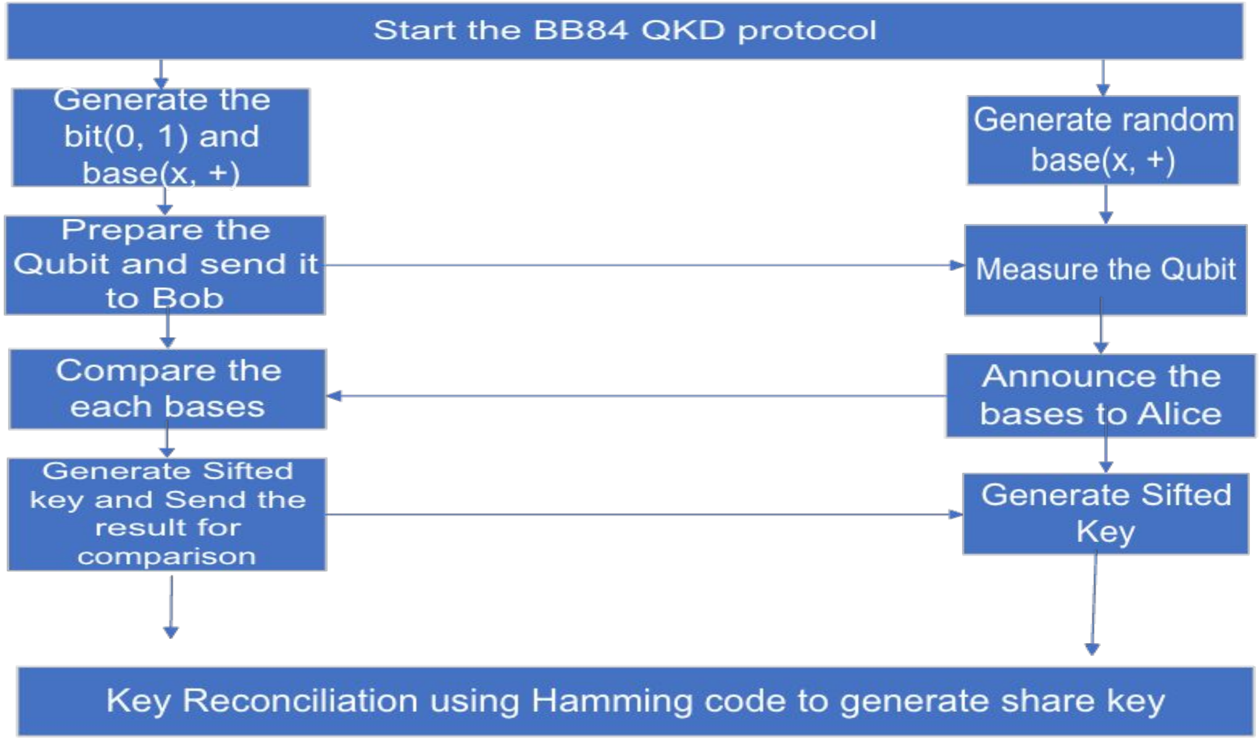
Framework of QKD Simulator Based IQX

- Alice and Bob use the Quantum Channel Simulator on the IQX and Classical channel for BB84 protocol
- Quantum Channel Simulator on the IQX is used for :
    - Process related to Qubit (decide and measure Qubit)
    - Simulation of noise in Quantum channel



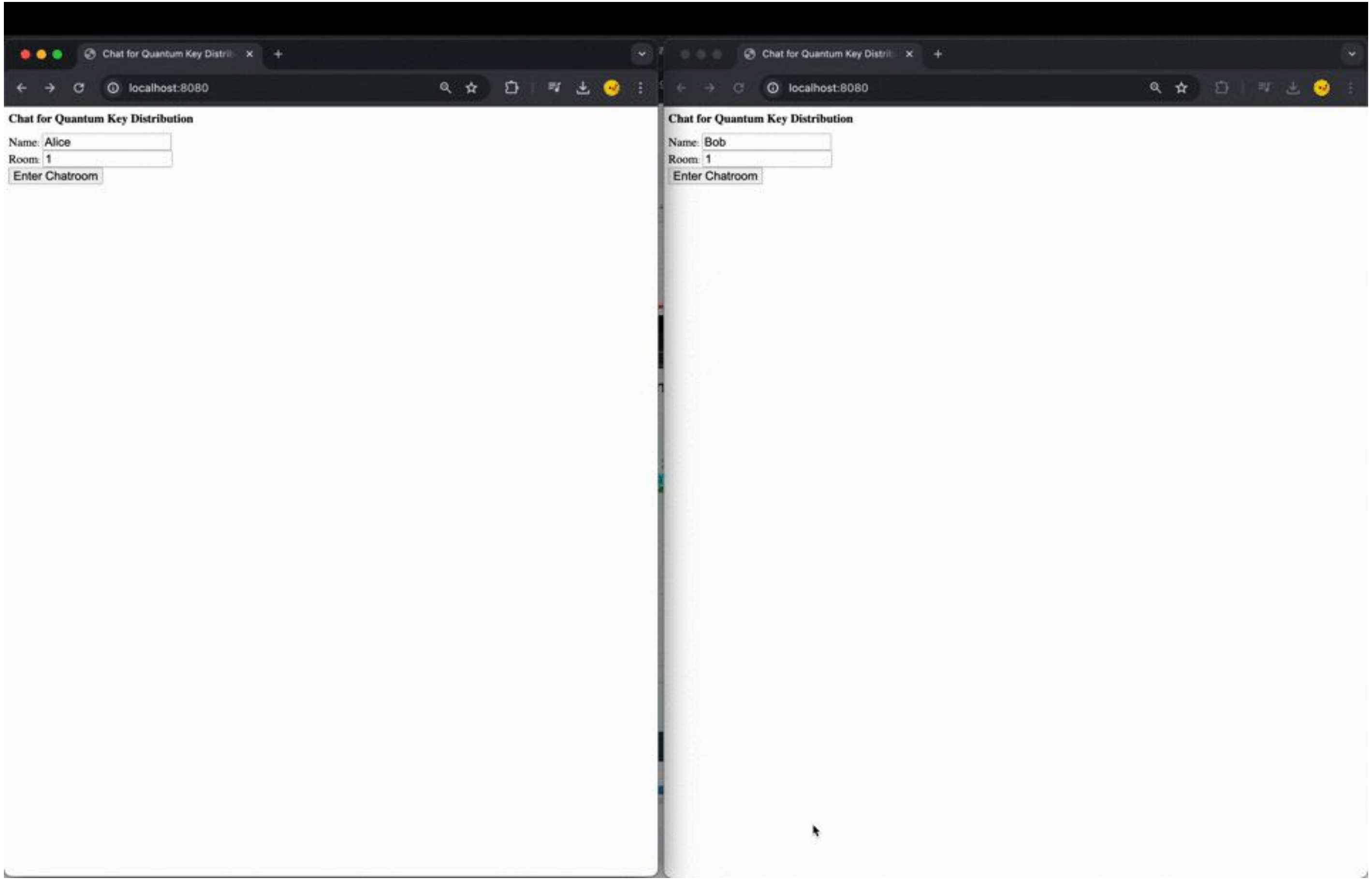
Flowchart of BB84 protocol on chat application

Assume that Alice sends a message to Bob



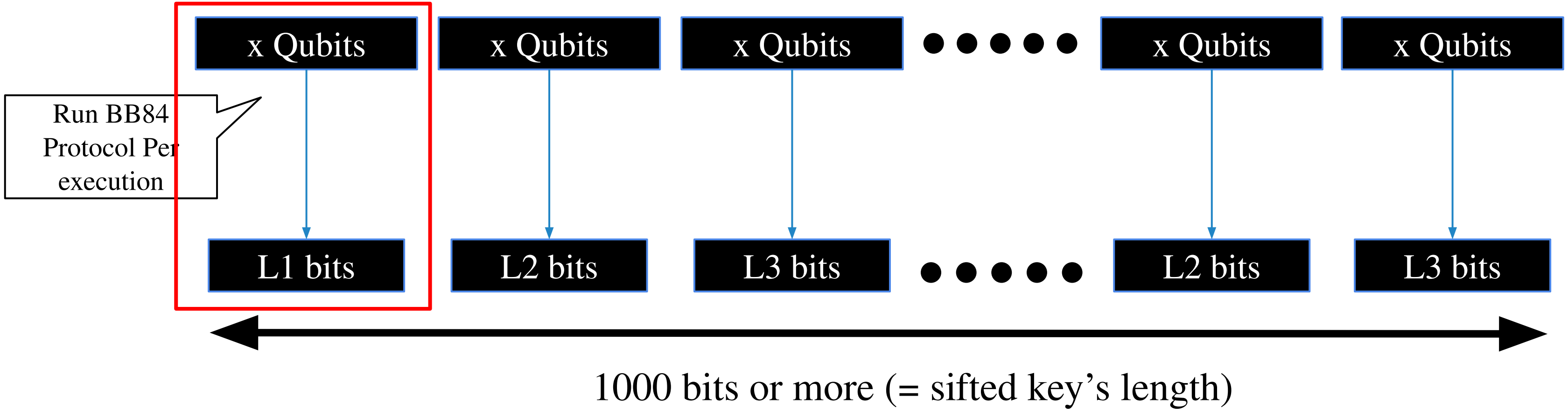
Outline

Demonstration of Secure Chat Applications



Implementation for key generation

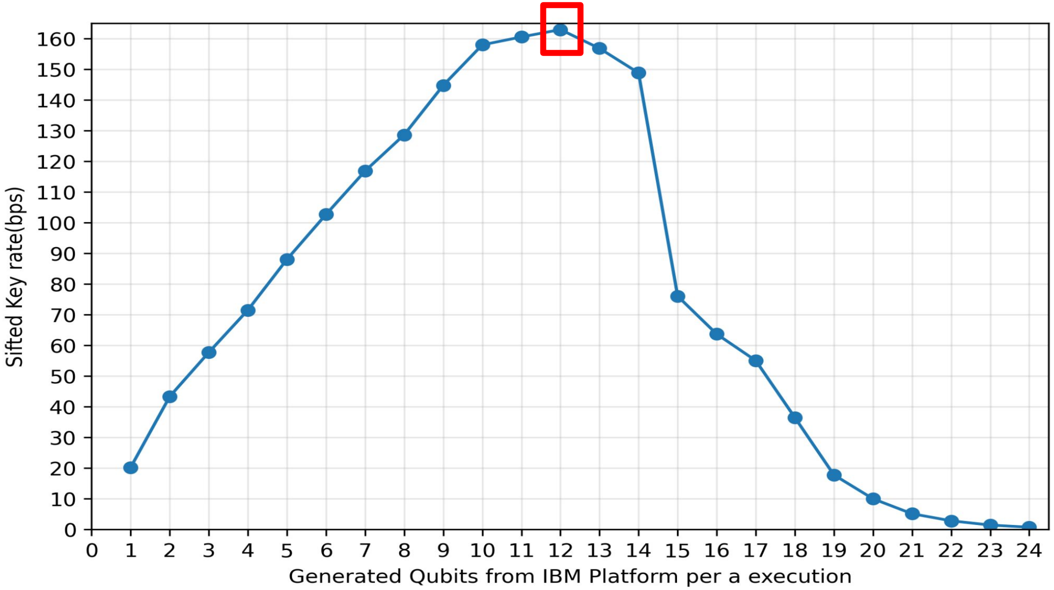
- An example of implementation for key generation



- The key length of the sifted key is 1000 bits or more.
  - Execute BB84 QKD Protocol multiple times
    - Execute it until the length of the sifted key reaches or exceeds 1000 bits.
  - Obtain a bit string of arbitrary length
    - E.g., When 12 qubits are supplied from the IQX simulator and BB84 is executed, it generates a key of about 6 bits from the 12 qubits.

Simulation of QKD Application (1)

Key Rate vs *Generated Qubits from IQX per a execution*

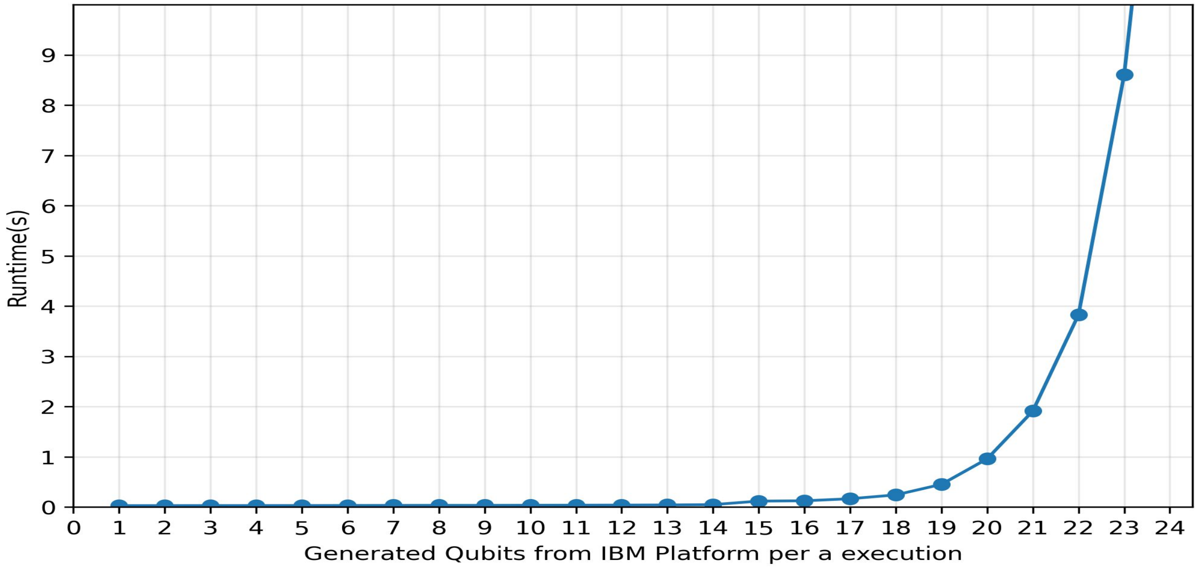


The highest Key rate was found to be achieved with 12 qubits.

➢ Required key length(e.g. 1000 bits) can be generated in the shortest possible time.

Simulation for QKD Application (1)

Runtime vs *Generated Qubits from IBM Platform per a execution*

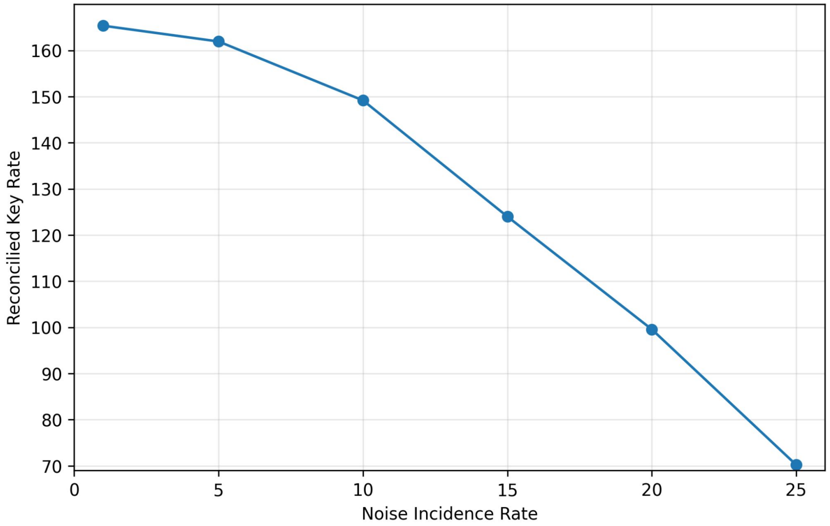


Quantum operations are represented as large matrices, and the dimension of these matrices grows exponentially with the number of qubits.

- ➢ Runtime is increasing exponentially with each increase in the number of generated qubits.

Simulation of QKD Application (2): Noise Error

Reconciled key rate vs Noise Incidence Rate

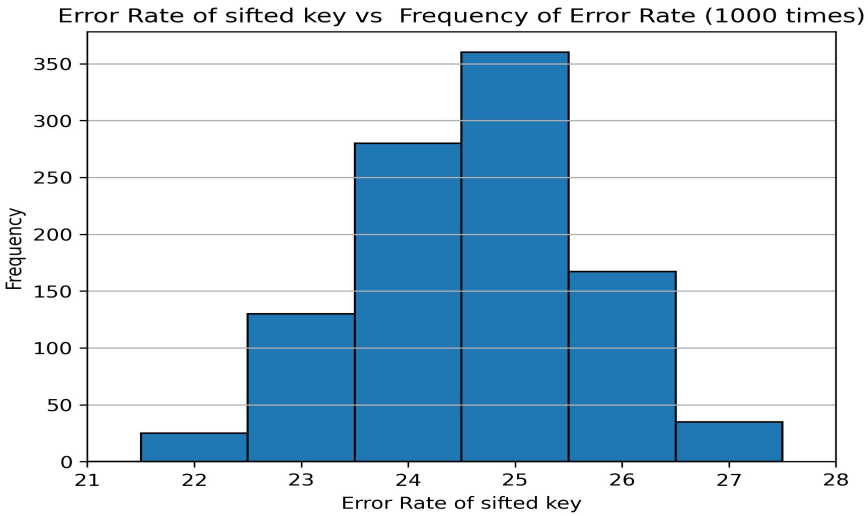


Generated Qubits from IBM Platform per a execution = 12 (Qubits)

- ➢ The reconciled key rate reaches 70 bps in all cases.
  - ○ It takes less than 20 seconds to generate a 1000-bit key.

Simulation of QKD Application (3): Eavesdropping

Error Rate of sifted key vs *Frequency of the error rate* (Run 1000 times)



Generated Qubits from IBM Platform per a execution = 12 (Qubits)

- ➢ Setting the error rate threshold for detecting the presence of Eve to 21~22% will allow Alice and Bob to detect the presence of Eve almost 100% of the time.

Outline

Conclusion

- We developed the secure chat application based on QKD
  - IBM Quantum Experience (IQX) support that development and simulation the application
  - The implementation of the BB84 protocol utilizes the principles of quantum mechanics to guarantee the security of communications

- We simulated the QKD application
  - *Sifted Key Rate* vs *Generated Qubits from IBM Platform per a execution*
    - Explored the number of Qubits supplied by IBM to generate the required key length (e.g. 1000 bits) in the shortest possible time.
  - *Reconciled key rate* vs *Noise Incidence Rate*
    - The key rate after key reconciliation reached at least 70 bps, regardless of what percentage of noise was applied to the Qubits.
  - *Error Rate of sifted key vs Frequency of the error rate*
    - Examined the threshold on the error rate of the sifted key for detecting the presence of an eavesdropper Eve.

Thank you for your listening!