Quantum Key Distribution

Takihara Yudai 3d year undergraduate student
The University of Aizu

Contents

1. Why do we need the Quantum Key Distribution(QKD)?

1 Why do we need the Quantum Key Distribution?

・Problem of Symmetric cryptography (The method that use secret key for encryption and decryption)

・Key exchanging ： How can Alice and Bob can agree on the secret key?

・Solution

>> Asymmetric cryptography (The method uses two types of keys : private key and public key)

・ Computational secure : guarantees the security with reasonable assumptions about an adversary’s capabilities

But in the near future …….

>> Quantum computers (Computers with unusually fast processing speeds)

>>The algorithm of complex mathematical calculations(eg RSA) are easily deciphered.

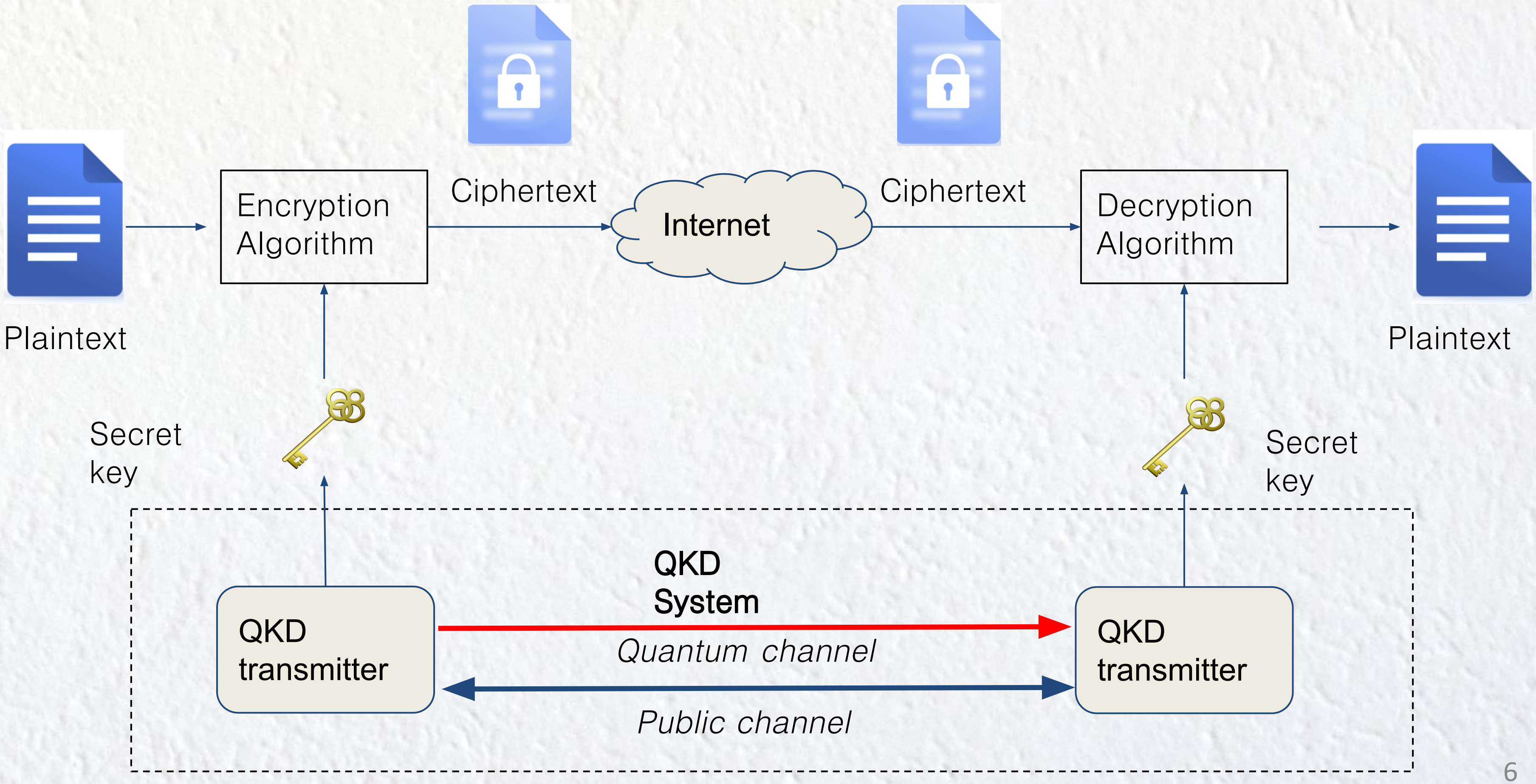
→ *New key distribution method needed*

2. What is Quantum Key Distribution(QKD)?

2 What is the Quantum Key Distribution?

Quantum Key Distribution is a technology that relies on quantum physics to secure the distribution of symmetric encryption keys

- **Quantum channel**
  - Used for distribution of Qubit.
  - Prevent information leakage and interception
- **Public channel**
  - share basis
  - announce the result of measurement



3. How QKD works?

3 How QKD works?

- **Operating scheme**
  - Prepare and measure
  - Entanglement-based
- **Implementation**
  - Discrete-variable
  - Continuous-variable
  - Non-coherent CV

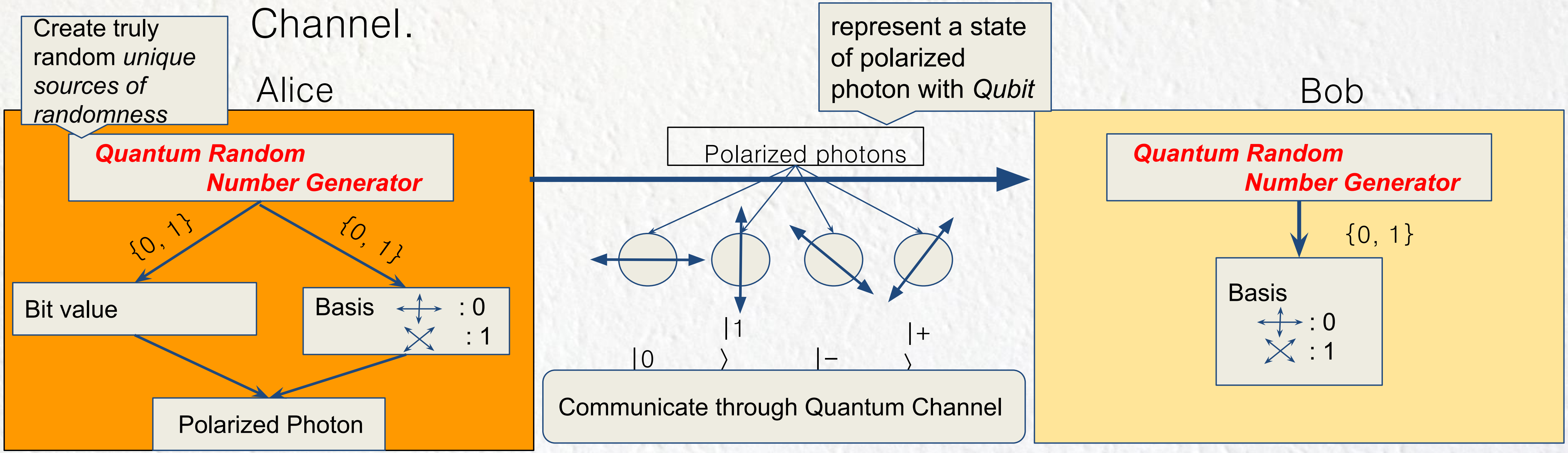
BB84

The popular protocol for Quantum Key Distribution

This protocol is named after the initials of the two developers and the year this protocol was published

3.2 How the BB84 Protocol works?

Step1: Alice sends a photon(Qubit) to Bob through Quantum Channel.

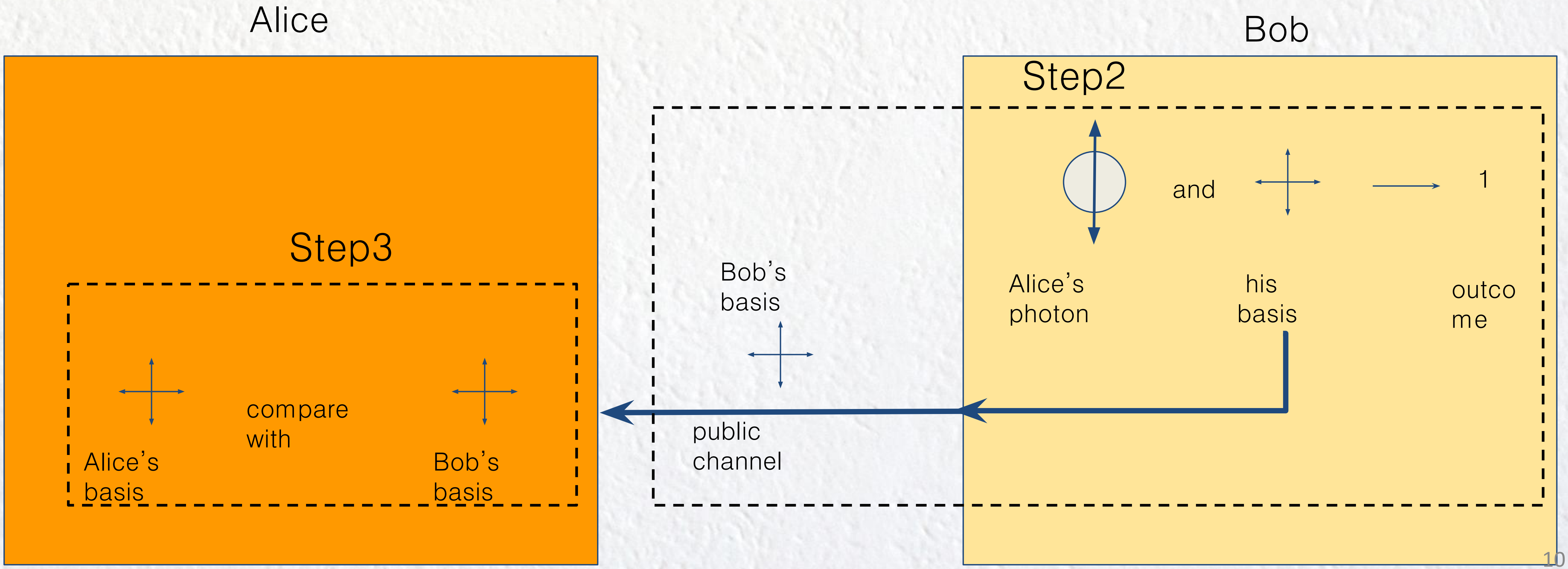


Bit value	Basis	Polarization	Qubit state
0			$\vert 0\rangle$
0			$\vert +\rangle$
1			$\vert 1\rangle$
1			$\vert -\rangle$

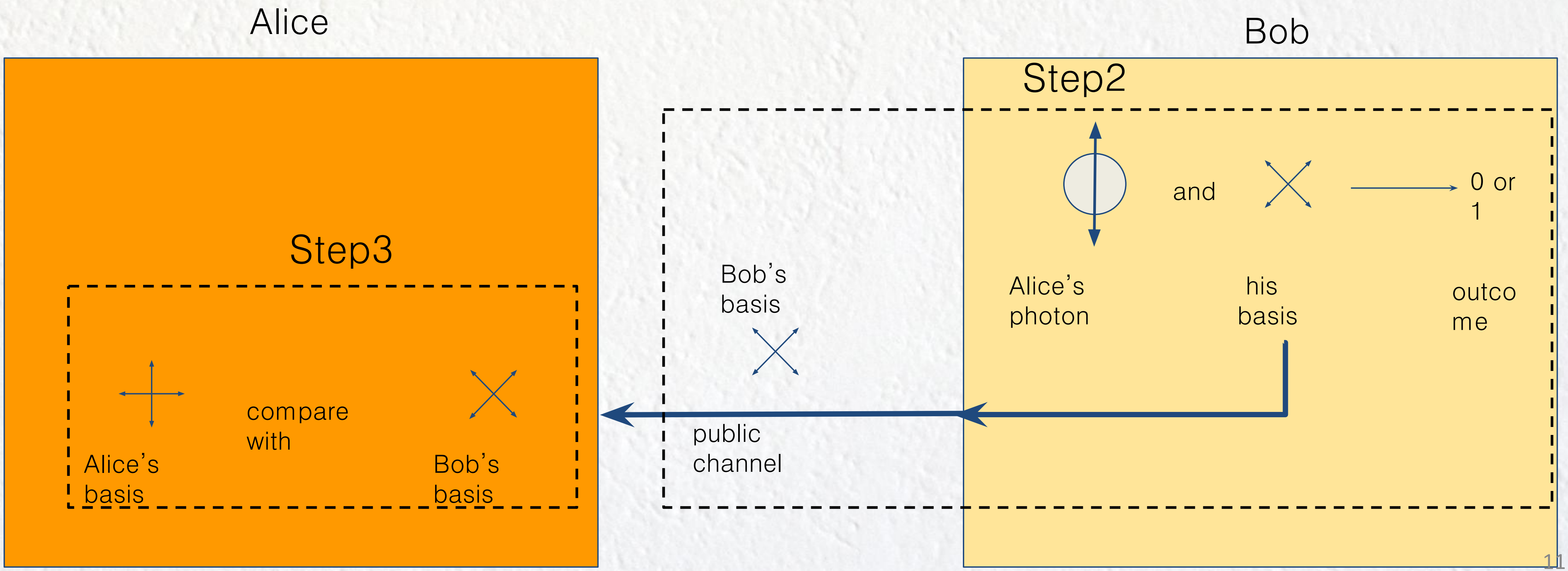
3.2 How the BB84 Protocol works?

Step2 : Bob measure Alice's photon based on his basis and derive bit value as outcome. After that, Bob sends his basis to Alice through public channel.

Step3 : Alice compares Bob's basis to her own basis.



3.2 How the BB84 Protocol works?



3.2 How the BB84 Protocol works?

Step4 : Alice discards bit value she prepared in Step1 if Bob's basis is different from her.
Bob keeps the outcome derived in Step2 if his basis is the same as Alice's.
=> This process is called sifted key.

Alice

Bit	Basis	State
0	+	↔
1	+	↕
0	+	↔
1	+	↕
0	×	⤢
1	×	⤡
0	×	⤢
1	×	⤡

Bob

Basis	State	outcome(bit)	sifted key
+	↔	0	0
+	↕	1	1
×	⤢	0	*discarded*
×	⤡	1	*discarded*
+	↔	0	*discarded*
+	↕	1	*discarded*
×	⤢	0	0
×	⤡	1	1

3.2 How the BB84 Protocol works?

Step5: Alice and Bob perform post-processing procedures to correct error in their keys and increase the secrecy of their key.

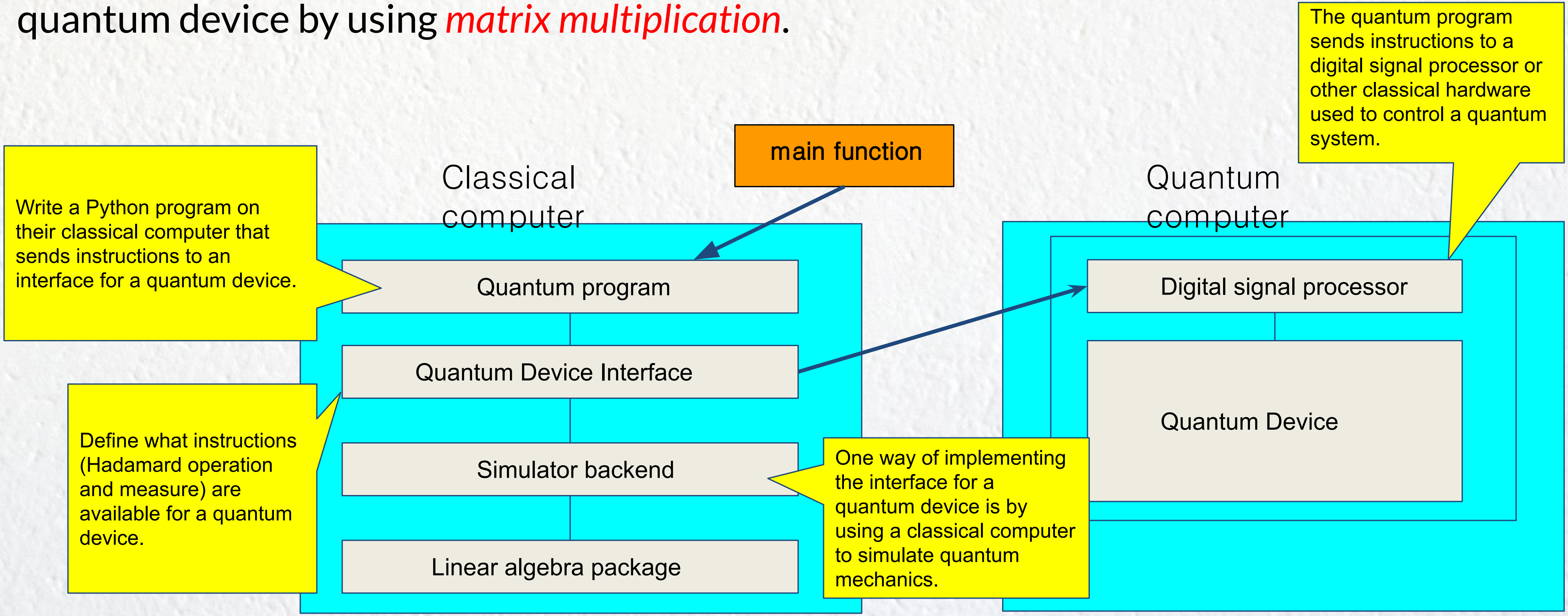
Step6 : Alice and Bob share a secret key.

4. The Algorithm and simulation in Python

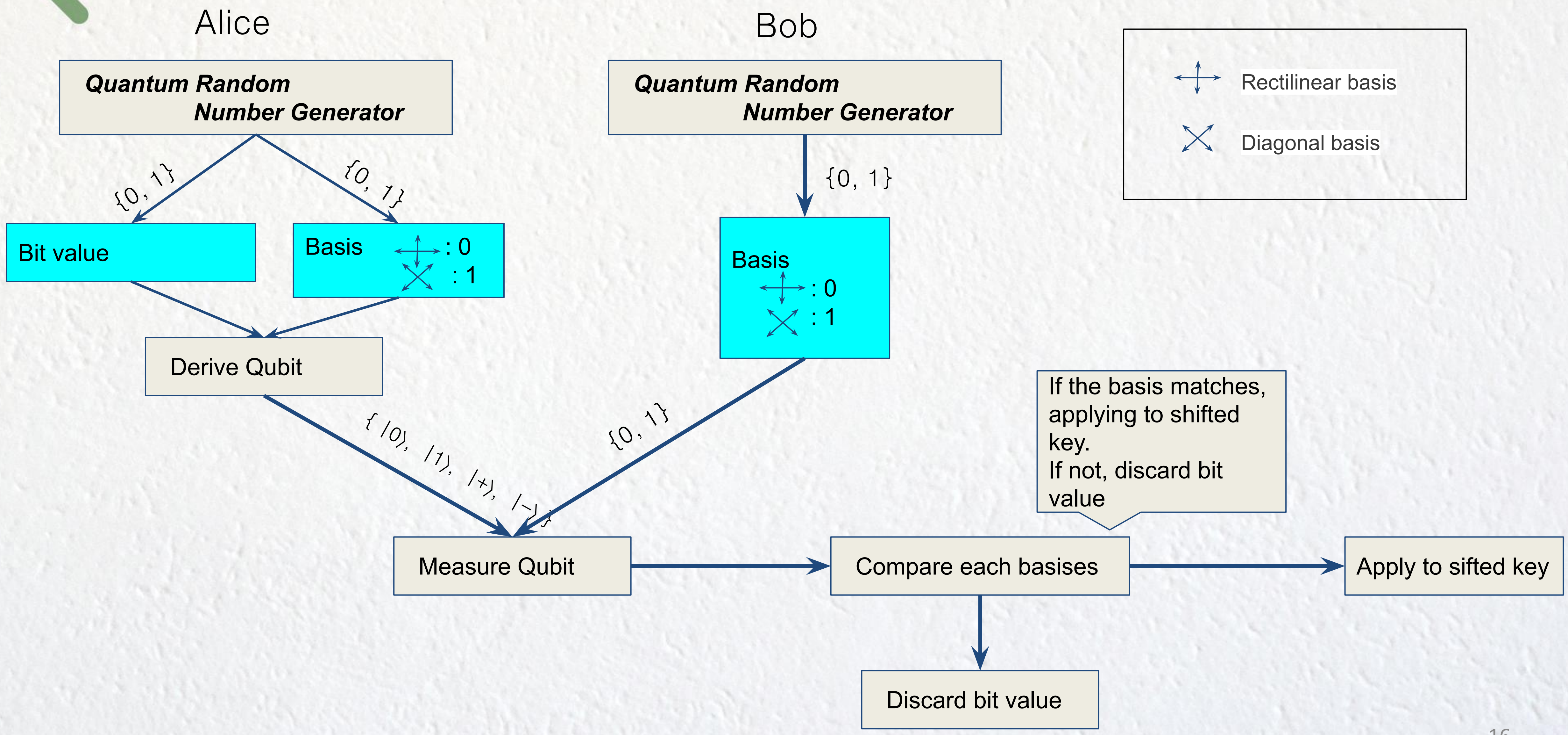
4 The algorithm and simulation in python

- Represent Qubit as vector.

>> Use a classical computer to simulate how BB84 programs would act on an ideal quantum device by using *matrix multiplication*.

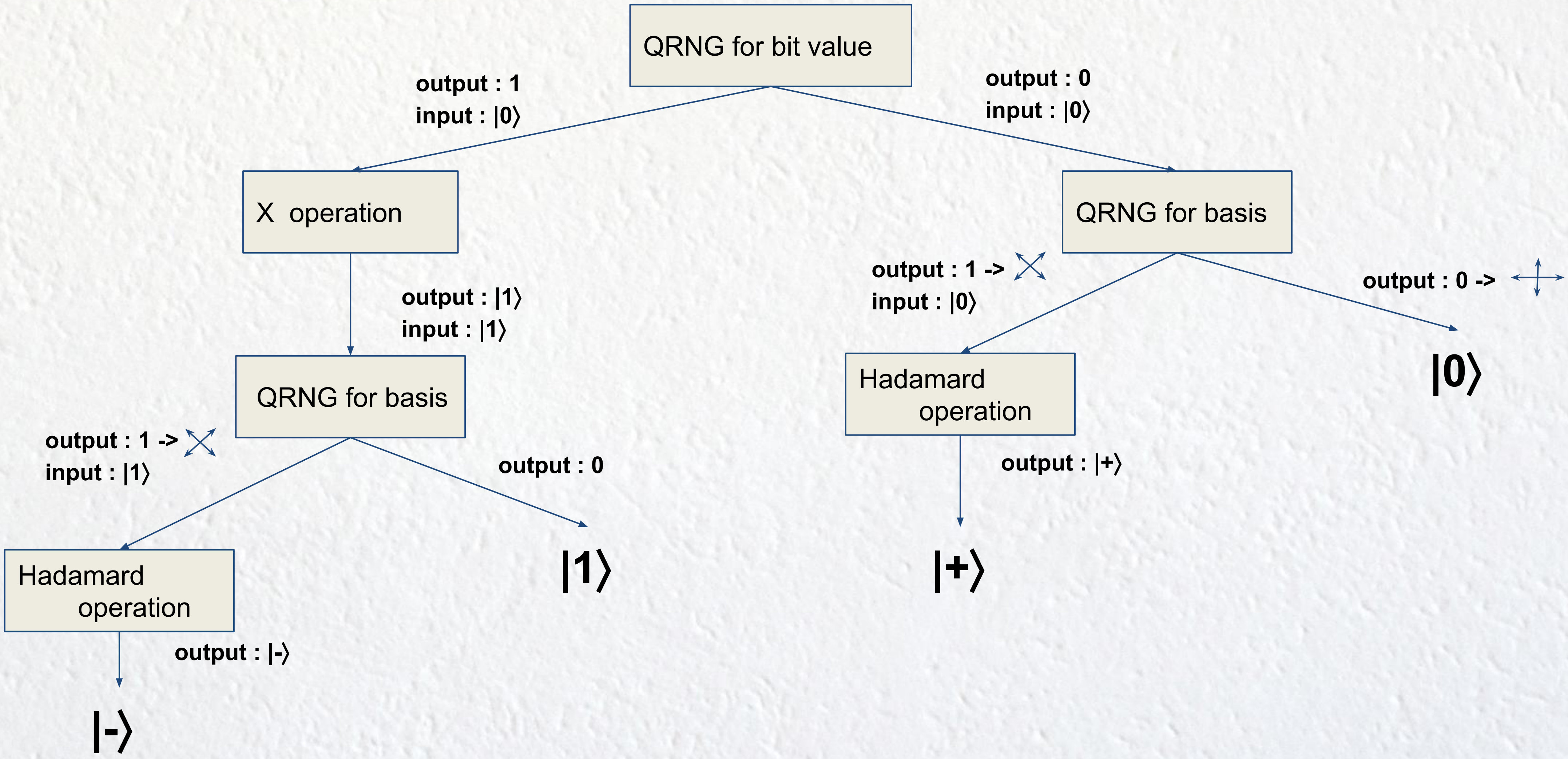


4.1 The algorithm and simulation for BB84



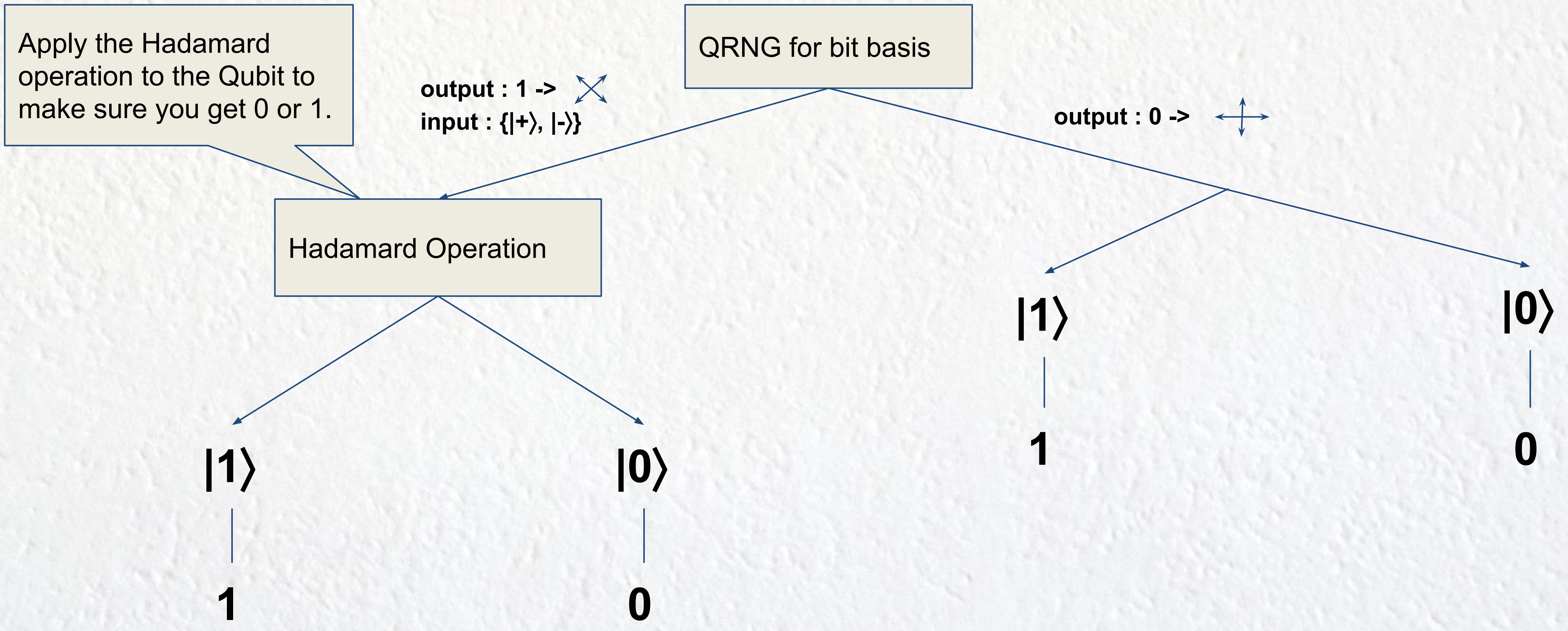
4.1 The algorithm and simulation for BB84

Alice side's



4.1 The algorithm and simulation for BB84

Bob side's



4.1 The algorithm and simulation for BB84

Simulation in python code

Alice applies a one-time-pad to message(plaintext) to be sent to Bob using a secret key

- One-time-pad -> XOR with message and private key

message XOR secret key → ciphertext

Bob applies a one-time-pad to ciphertext using a secret key >> Get message(plaintext)

ciphertext XOR secret key → message

4.2 The algorithm and simulation for BB84

Result of simulation

```
[yudai@wlan-napt-005] ~/qkd/chapter3 (main) ⚡
❯ python3 bb84.py
Generating a 96-bit key by simulating BB84...
Took 173 rounds to generate a 96-bit key.
Got key                          0xb0a8f2516bb3fc7736f8702.
Using key to send secret message: 0xd83ddc96d83ddc0dd83ddcbb.
Encrypted message:               0xd33753b3ce86e3caab525bb9.
Bob decrypted to get:            0xd83ddc96d83ddc0dd83ddcbb.

[yudai@wlan-napt-005] ~/qkd/chapter3 (main) ⚡
❯ python3 bb84.py
Generating a 96-bit key by simulating BB84...
Took 198 rounds to generate a 96-bit key.
Got key                          0x76afbc81a54c09c420f8438c.
Using key to send secret message: 0xd83ddc96d83ddc0dd83ddcbb.
Encrypted message:               0xae9260177d71d5c9f8c59f37.
Bob decrypted to get:            0xd83ddc96d83ddc0dd83ddcbb.

[yudai@wlan-napt-005] ~/qkd/chapter3 (main) ⚡
❯ python3 bb84.py
Generating a 96-bit key by simulating BB84...
Took 192 rounds to generate a 96-bit key.
Got key                          0xd16e246a1dd874677bf4e6a6.
Using key to send secret message: 0xd83ddc96d83ddc0dd83ddcbb.
Encrypted message:               0x953f8fcc5e5a86aa3c93a1d.
Bob decrypted to get:            0xd83ddc96d83ddc0dd83ddcbb.
```

Thank for listening