eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

UNIVERSIDAD DEL PAÍS VASCO
FACULTAD DE INFORMÁTICA

# BB84 in Qiskit

Justas Remeikis

jremeikis001@ikasle.ehu.eus

Donostia
2025

# Contents

## 1.1. Overview of the protocol

In 19**84** the first quantum exchange protocol (**BB84**) was introduced by Charles **B**ennett and Gilles **B**rassard, which formed the basis of the quantum key exchange (QKE).

BB84 is the One-Time-Pad protocol (Yanofsky & Mannucci, 2008), which exploits quantum mechanics to securely transmit private keys.

In this protocol 2 different orthogonal bases (1.1), (1.2) are used:

$$+ = \{|\rightarrow\rangle, |\uparrow\rangle\} = \{[1,0]^{\mathrm{T}}, [0,1]^{\mathrm{T}}\} \tag{1.1}$$

$$X = |\nwarrow\rangle, |\nearrow\rangle = \left\{\frac{1}{\sqrt{2}}[-1,1]^{\mathrm{T}}, \frac{1}{\sqrt{2}}[1,1]^{\mathrm{T}}\right\} \tag{1.2}$$

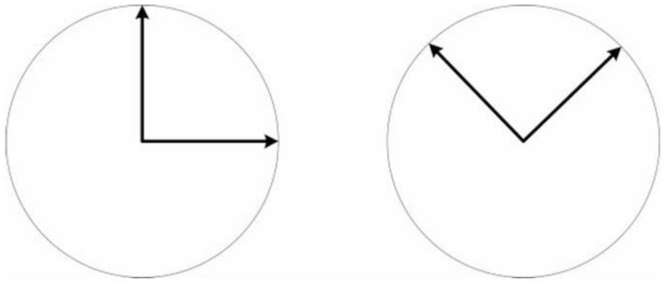


Figure 1.1 Bases used for BB84

Using these 2 bases, the states $|0\rangle$ and $|1\rangle$ are described by the following table:

Table 1.1 States description

| State / Basis | + | X |
|---|---|---|
| $\vert 0\rangle$ | $\vert\rightarrow\rangle$ | $\vert\nearrow\rangle$ |
| $\vert 1\rangle$ | $\vert\uparrow\rangle$ | $\vert\nwarrow\rangle$ |

Steps of the protocol:

1. Alice selects (or generates random) n bits length string which she wants to send and a random n bits length string which would determine in which of the two bases to send those bits. Using the table above (Table 1.1) she encodes the information and sends it to Bob.

2. Bob receives the bits, but he does not know which bases were used, so he measures them on a random basis.
   Bob's basis will match about 50% of the time, so Bob's measurements will agree with Alice's about half of the time.

3. Alice and Bob publicly compare used basis. Bits that are not on the same basis are removed. If nobody was listening, the remaining bits should be identical, since they were sent and received on the same basis.

4. Bob randomly chooses half of the bits, which have matching basis, and compares them with Alice. If they disagree by more than a tiny percentage, that means that Eve was listening. In that case the remaining bit string is unusable, and they need to try different way. But if there is no sign of listening, they can scratch off used bits and use remaining ones for private key.

Overall, remaining string should be ¼ length of the originally sent bit string, since Bob correctly guess only ½ of the basis and ½ of correctly guessed basis's bits are used for eavesdropping detection.

## 1.2. Implementation

The implementation process of the bb88.py script involves several steps to simulate the BB84 quantum key distribution protocol:

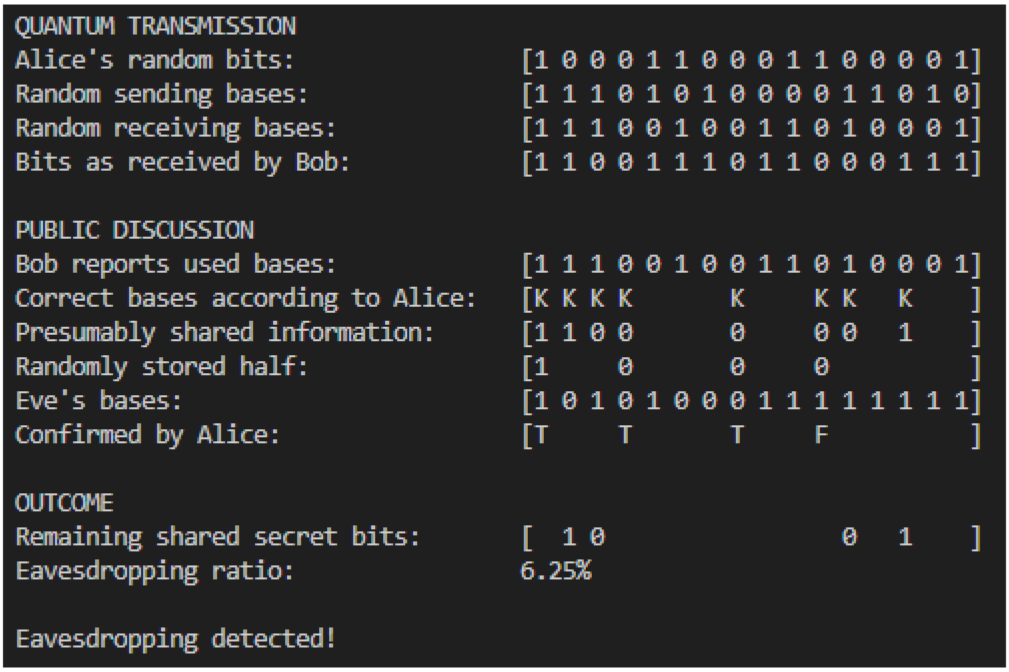
1. **Import Libraries**:
   - Import necessary libraries: qiskit for quantum circuit operations, numpy for numerical operations, and AerSimulator for simulating quantum circuits.
2. **Define Constants**:
   - num_qubits: Number of qubits used in the simulation.
3. **Generate Alice's Data**:
   - generate_alice_data(num_qubits): Generates random bits and bases for Alice.
4. **Prepare Qubits**:
   - prepare_qubits(alice_bits, alice_bases): Prepares quantum circuits based on Alice's bits and bases.
5. **Eve's Interception**:
   - eve_intercepts(circuits, num_qubits): Simulates Eve's interception by measuring and possibly altering the qubits.
6. **Bob's Measurement**:
   - bob_measures(circuits, num_qubits): Bob measures the qubits using his randomly chosen bases.
7. **Reconcile Bases**:
   - reconcile_bases(alice_bases, bob_bases, alice_bits, bob_bits): Determines the shared key by comparing Alice's and Bob's bases.
8. **Check Correct Bases**:
   - check_correct_bases(alice_bases, bob_bases): Checks which bases match between Alice and Bob.
9. **Format Correct Bases**:
   - format_correct_bases(correct_bases): Formats the correct bases for display.
10. **Create Correct Bits**:
    - create_correct_bits(bob_bits, correct_bases): Creates a list of bits that were correctly measured by Bob.
11. **Randomly Store Half**:
    - randomly_store_half(correct_bits): Randomly selects half of the correct bits to be stored.

12. **Confirm by Alice**:
    - confirm_by_alice(alice_bits, half_correct_bits): Alice confirms the correctness of the stored bits.
13. **Create Outcome**:
    - create_outcome(correct_bits, half_correct_bits): Creates the final shared secret bits.
14. **Calculate Eavesdropping Ratio**:
    - calculate_eavesdropping_ratio(confirmed_by_alice): Calculates the ratio of bits that were intercepted by Eve.
15. **Collect Data**:
    - collect_data(eavesdropping_enabled=True): Collects all the data from the simulation steps and stores it in a dictionary.
16. **Print Data**:
    - print_data(data): Prints the collected data in a formatted manner.
17. **Run Tests**:
    - run_tests(num_tests): Runs multiple tests to check the detection of eavesdropping.
18. **Main Function**:
    - main(): Executes the main process, including data collection, printing, and running tests.
19. **Entry Point**:
    - if __name__ == "__main__": main(): Ensures the script runs the main function when executed directly.

This script simulates the BB84 protocol, including the possibility of eavesdropping, and checks the integrity of the shared key between Alice and Bob.

## 1.3. Analysis

Runing the program a report is generated:



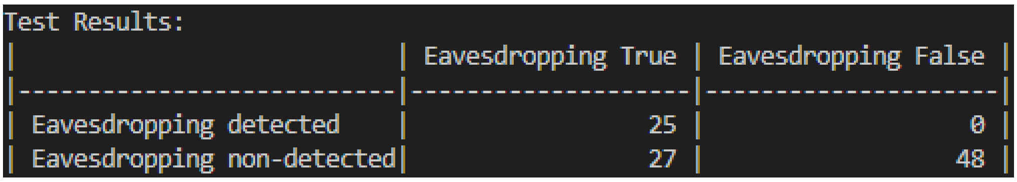
```
QUANTUM TRANSMISSION
Alice's random bits:                [1 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1]
Random sending bases:               [1 1 1 0 1 0 1 0 0 0 0 1 1 0 1 0]
Random receiving bases:             [1 1 1 0 0 1 0 0 1 1 0 1 0 0 0 1]
Bits as received by Bob:            [1 1 0 0 1 1 1 0 1 1 0 0 0 1 1 1]

PUBLIC DISCUSSION
Bob reports used bases:             [1 1 1 0 0 1 0 0 1 1 0 1 0 0 0 1]
Correct bases according to Alice:   [K K K K       K     K K   K    ]
Presumably shared information:      [1 1 0 0       0     0 0   1    ]
Randomly stored half:               [1     0       0     0          ]
Eve's bases:                        [1 0 1 0 1 0 0 0 1 1 1 1 1 1 1 1]
Confirmed by Alice:                 [T     T       T     F          ]

OUTCOME
Remaining shared secret bits:       [  1 0                 0   1    ]
Eavesdropping ratio:                6.25%

Eavesdropping detected!
```

Figure 1.2 Generated report

The report provides detailed information about the first protocol instance. Using the report, it is possible to see what bits and bases were generated, which bases Bob correctly guessed and what bits were used to check for eavesdropping.

Since simulated hardware was used, there was no unpredicted changes to bits, that means that all keys generated (which aren’t compromised) are usable.

The program also can run tests to check for correctly identified eavesdropping.

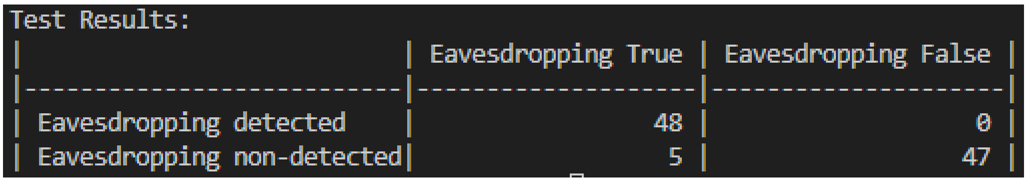


```
Test Results:
|                           | Eavesdropping True | Eavesdropping False |
|---------------------------|--------------------|---------------------|
| Eavesdropping detected    |                 25 |                   0 |
| Eavesdropping non-detected|                 27 |                  48 |
```

Figure 1.3 Eavesdropping test results for n = 16, num_tests = 100

In the results above (Figure 1.3) sending small number of bits is dangerous since eavesdropping could be not detected.

To combat this issue, the number of bits should be increased.



```
Test Results:
|                           | Eavesdropping True | Eavesdropping False |
|---------------------------|--------------------|---------------------|
| Eavesdropping detected    |                 48 |                   0 |
| Eavesdropping non-detected|                  5 |                  47 |
```

Figure 1.4 Eavesdropping test results for n = 32, num_tests = 100

By increasing bits used by 2, the non-detected eavesdropping falls 5 times (Figure 1.4).

## 1.4. Conclusion

The implementation of the BB84 protocol in Qiskit successfully demonstrates the principles of quantum key distribution and its effectiveness in detecting eavesdropping. The results highlight the importance of using enough qubits to reduce the risk of undetected eavesdropping and ensure secure key generation. Future work could focus on testing the protocol on real quantum hardware.

## Bibliography

Yanofsky, N. S., & Mannucci, M. A. (2008). *Quantum Computing for Computer Scientists.* New York: Cambridge University Press.