my protocol 1 -- QSDC

量子直接通訊協定**(Quantum secure direct communication)**

Alice

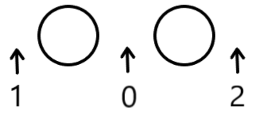
Bob

Pre-shared string (0,1,2)
(length : 3n/4)

0. message (length : n)
   message前n/2和後n/2做XOR
   →XOR_m (length : n/2)
   M = message + XOR_m (length : 3n/2)

1. depend on M, encode |0> to |+>,|->

2. generate 3n/4 decoy photon (|+> or |->)
   depend on pre-shared string,



3. Alice lock : M → X,Y,Z D → Rx,Ry,Rz

Send to Bob →

4. Bob lock : M → U3 D → No

← Send to Alice

5. Alice unlocks and checks decoy
   (X basis measurement)

6. if no Eve, unlock M

Send to Bob →

7. Bob unlocks and measures M
   (X basis measurement)

8. Check XOR(m’) == XOR(m)’
   if no Eve, get message.

Alice

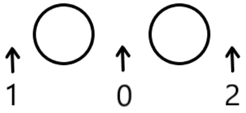
Bob

Pre-shared string (0,1,2)
(length : 3n/4)

0. message (length : n)
   Hash : (ax+by)%c
   → H(m) (length : n/2)
   M = message + H(m) (length : 3n/2)

1. depend on M, encode |0> to |0>,|1>

2. generate 3n/4 decoy photon (|0> , |1> ,|+> , |->)
   depend on pre-shared string,



3. Alice lock : M → X,Y,Z & H   D → U3

Send to Bob →

4. Bob lock : M → U3
   D → X,Y,Z

← Send to Alice

5. Alice unlocks and checks decoy

6. if no Eve, unlock M

Send to Bob →

7. Bob unlocks and measures M
   (Z basis measurement)

8. Check H(m') == H(m)'
   if no Eve, get message.

my protocol 2 -- QKD

量子金鑰分配協定 **(Quantum key distribution protocol)**

BB84 + Y基底

|+y> :

In [8]:

```python
# generate one qubit

q = QuantumRegister(1,'q')
c = ClassicalRegister(1,'c')

# Alice

Alice_circ5 = QuantumCircuit(q,c)   # |+y>
Alice_circ5.h(q[0])
Alice_circ5.s(q[0])
Alice_circ5.barrier(q)

Alice_circ6 = QuantumCircuit(q,c)   # |-y>
Alice_circ6.h(q[0])
Alice_circ6.sdg(q[0])
Alice_circ6.barrier(q)


Alice_circ5.draw()
#Alice_circ6.draw()
```

Out[8]:

```
q_0: |0>┤ H ├┤ S ├░─
 c_0: 0 ═══════════
```

|-y> :

In [9]:

```python
# generate one qubit

q = QuantumRegister(1,'q')
c = ClassicalRegister(1,'c')

# Alice

Alice_circ5 = QuantumCircuit(q,c)   # |+y>
Alice_circ5.h(q[0])
Alice_circ5.s(q[0])
Alice_circ5.barrier(q)

Alice_circ6 = QuantumCircuit(q,c)   # |-y>
Alice_circ6.h(q[0])
Alice_circ6.sdg(q[0])
Alice_circ6.barrier(q)


#Alice_circ5.draw()
Alice_circ6.draw()
```

Out[9]:

```
q_0: |0>┤ H ├┤ Sdg ├░─
 c_0: 0 ══════════════
```

Y measurement

In [10]:

```
# Bob measurement

B3 = QuantumCircuit(q,c) # Y basis
B3.rx((1/2)*np.pi,q[0])
B3.measure(q[0],c[0])


B3.draw()
```

Out[10]:

```
q_0: |0>┤ Rx(1.5708) ├┤M├
                       ╥
 c_0: 0 ═══════════════╩═
```

In [5]:

```
from qiskit import execute, BasicAer
backend_sim = BasicAer.get_backend('qasm_simulator')
```

In [11]:

```
circuit1=[]
circuit1.append(Alice_circ5+B3)
circuit1[0].draw()
```

Out[11]:

```
q_0: |0>┤ H ├┤ S ├░─┤ Rx(1.5708) ├┤M├
                                    ╥
 c_0: 0 ════════════════════════════╩═
```

In [12]:

```
job_sim1 = qiskit.execute(circuit1, backend_sim, shots=1024)
result_sim1 = job_sim1.result()
counts1 = result_sim1.get_counts(circuit1[0])
print(counts1)
```

```
{'0': 1024}
```

In [13]:

```
circuit2=[]
circuit2.append(Alice_circ6+B3)
circuit2[0].draw()
```

Out[13]:

```
q_0: |0>┤ H ├┤ Sdg ├░─┤ Rx(1.5708) ├┤M├
                                      ╥
 c_0: 0 ══════════════════════════════╩═
```

In [14]:

```
job_sim2 = qiskit.execute(circuit2, backend_sim, shots=1024)
result_sim2 = job_sim2.result()
counts2 = result_sim2.get_counts(circuit2[0])
print(counts2)
```

```
{'1': 1024}
```

my protocol 3

Measure-resend ASQKD with single photons

(報告用_原版)

Quantum channel

Alice

Pre-shared key :
K1 (長度n+m的01序列)
K2 (長度n+m的01序列)
K1, K2→ K3 (長度2n+2m的01序列)
(K3[i]=K1[i],K2[i])

Bob

Step1 : Key_A (長度n的01序列)
Hash(Key_A||K1) → (長度m的01序列)
Key_A||Hash(Key_A||K1)
→ (長度n+m的01序列) ,稱為Da

Step2 : Cb (隨機的長度n+m的01序列)
insert Cb into Da based on K1 (0 left, 1 right)
(長度2n+2m的01序列)
→量子訊息 Qa (長度2n+2m的|0>|1>序列)
operate I(0),H(1) based on K3
改變基底

Send Qa to Bob

Step3 : operate I(0),H(1) based on K3
還原為Z基底
Da: (Z) measure
get Key_A’ , Hash(Key_A||K1) ’
resend |0>,|1> based on result.
Cb: measure(1) or reflect(0) based on K2
measurement: if 0, resend |1>
if 1, resend |0>

before sending, operate I(0),H(1) based on K3.

Send Qa to Alice

Step4 : Z(0) or X(1) measure Qa based on K3
get resend Da and Cb
檢查Key正確性:
check resend Da == original Da
檢查回傳者是Bob:
check resend Cb and original Cb based on K2
(K2[i]=0 → resend_Cb[i] == original_Cb[i])
(K2[i]=1 → resend_Cb[i] != original_ Cb[i])

Step4 : check Hash(Key_A’||K1) == Hash(Key_A||K1) ’
get key : Key_A’

(實作用_原版)

Quantum channel

Pre-shared key
K1 (n+m)   K2 (n+m)
K1, K2 → K3 (2n+2m)
(K3[i]=K1[i],K2[i])

Alice

Bob

Step1 : Key : SK (n)
Hash(SK|| K1) = hsk (m)
SK||hsk = Sa (n+m)

Step2 : Ca (n+m) , Cb (=Ca)
insert Cb (|0>,|1>) into Sa (|0>,|1>)
based on K1 (0 left, 1 right)
→ Sa’ (2n+2m)
operate I(0),H(1) based on K3

Send Sa’ to Bob

Step3 : operate I(0),H(1) based on K3
Sa: (Z) measure Sa, get SK’ , hsk’
resend |0>,|1> based on result.
Cb: measure(1) or reflect(0) based on K2
measurement: if 0, resend |1>
if 1, resend |0>

before sending, operate I(0),H(1) based on K3.

Send Sa’ to Alice

Step4 : Z(0) or X(1) measure Sa’ based on K3
get resend Sa and Cb
check resend Sa = original Sa
check resend Cb and Ca based on K2
(K2[i]=0 → resend_Cb[i] == Ca[i])
(K2[i]=1 → resend_Cb[i] != Ca[i])

Step4 : check Hash(SK’|| K1) == hsk’
get key : SK’

my protocol 4 -- QD

量子對話 (Quantum dialogue)

Pre-shared key (length : 2n) → Reverse Pre-shared key (length : 2n)

Alice

1. message (length : 2n/3)
   Hash : (ax+by)%c →H(m) (length : n/3)
   M = message + H(m) (length : n → 偶數)

2. depend on M, encode |0> to |0>,|1> .
   depend on Pre-shared key, for every two M photons,
   01 or 10 swap, 00 or 11 not swap.

3. Generate n decoy photons (|0> , |1> ,|+> , |->)
   depend on Reverse Pre-shared key , insert decoy photons
   00 or 01 插在M photon前， 10 or 11 插在M photon後。

4. Alice lock :
   depend on 前n Pre-shared key : M → I (0) , Y (1) ; D → I (0) , H (1)
   depend on 後n Pre-shared key : M → I (0) , H (1) ; D → I (0) , Y (1)

5. Send to Bob

Bob

6. Depend on Reverse Pre-shared key,
   know M position.
   Depend on Pre-shared key,
   unlock Alice's M lock and swap.
   Z basis measure M, check H(m')==H(m)'
   If no Eve, get message m.

7. message2 (length : 2n/3)
   Hash : (ax+by)%c →H(m2) (length : n/3)
   M2 = message2 + H(m2) (length : n)

8. depend on M2, encode decoy : I (0) , Y (1)
   depend on Reverse Pre-shared key,
   for every two decoy photons,
   00 or 10 swap, 01 or 11 not swap.

9. Bob lock :
   depend on 前n Reverse Pre-shared key :
   M2 → I (0) , Y (1)
   depend on 後n Reverse Pre-shared key :
   M2 → I (0) , H (1)

10. Send to Alice

11. Depend on Reverse Pre-shared key ,
    unlock Bob's M2 lock and swap.
    unlock Alice's D lock and recover decoy state to |0> (in 3.)
    Z basis measure M2, check H(m2')==H(m2)'
    If no Eve, get message m2.