ENG1116 - PUC-Rio, 2020.2

Trabalho 4: ENG1116

28/10/2020

Professor: Guilherme Temporão e Thiago Guerreiro *Aluno: Rafael Vilela*

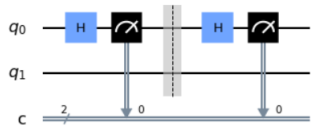
1 Questão 1 - Protocolo ideal

1.1 1.1 a 1.5

Os circuitos quânticos a seguir representam o circuito que recebe dois qubits e dois cbits, sendo que é possível combina-los para estados $|0\rangle$, $|1\rangle$, $|+\rangle$, $|-\rangle$ após aplicar a operação unitária no primeiro qubit.

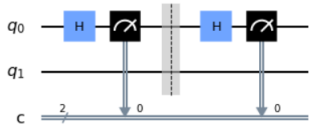
O outro recebe como parâmetro um bit e realiza medida no primeiro qubit.

In [683]:
```
q = QuantumCircuit(2,2)

def Ua(qc,qubit,cbit1,cbit2):
    if cbit2 == 0:
        if cbit1 == 1:
            qc.x(qubit)
    else:
        if cbit1 == 0:
            qc.h(qubit)
        else:
            qc.x(qubit)
            qc.h(qubit)
            qc.barrier()

Ua(q,0,1,1)
display(qc.draw(output='mpl'))
```



(1)

In [684]:
```
def Ub(qc,qubit,cbit):
    if cbit == 1:
        qc.h(qubit)
        qc.measure(0,0)
        qc.barrier()
    else:
        qc.measure(0,0)
        qc.barrier()

Ub(q,0,1)
display(qc.draw(output='mpl'))
```



(2)

```
np.random.seed(seed = 2)
n = 100


b1 = randint(2, size=n)  # bits alice
alicebases = randint(2, size=n)
mensagem = codificar(0, b1, alicebases)  #codifica

bobbases = randint(2, size=n)  #base escolhida por bob
bobresults = medir_mensagem(0, mensagem, bobbases) #medicao pelo bob

alicekey = remover_errados(alicebases, bobbases, b1)
bobkey = remover_errados(alicebases, bobbases, bobresults)

p = 20  #amostra

selecaobit = randint(n, size = p)

mostraalice, mostrabob = erro(alicekey, bobkey, selecaobit)
print(" alice = " + str(mostraalice))
print(" bob = " + str(mostrabob))
```

```
Erro = 0.0
 alice = [0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0]
 bob = [0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0]
```

(3)

```
def codificar(qubit, bits, bases):
    mensagem = []
    for i in range(len(bits)):
        qc = QuantumCircuit(2,2)
        Ua(qc,qubit,bits[i],bases[i])

        mensagem.append(qc)
    return mensagem

def medir_mensagem(qubit, mensagem, bases):
    backend = Aer.get_backend('qasm_simulator')
    medicoes = []
    for i in range(len(bases)):
        Ub(mensagem[i],qubit,bases[i])
        result = execute(mensagem[i], backend, shots=1000, memory=True).result()
        bit_medido = int(result.get_memory()[0])
        medicoes.append(bit_medido)
    return medicoes

def remover_errados(a_bases, b_bases, bits):
    bits_certos = []
    for i in range(len(bits)):
        if a_bases[i] == b_bases[i]:
            bits_certos.append(bits[i])
    return bits_certos

def escolha_bits(bits, selecionado):
    escolha = []
    for i in selecionado:

        i = np.mod(i, len(bits))
        escolha.append(bits.pop(i))
    return escolha

def erro(b1, b2, size):
    e1 = []
    e2 = []
    n = 0
    for i in range(len(b1)):
        if b1[i] != b2[i]:
            n = +1
    for i in size:
        i = np.mod(i, len(b1))
        e1.append(b1.pop(i))
    for i in size:
        i = np.mod(i, len(b2))
        e2.append(b2.pop(i))
    error = 100-100*((len(b1) - n)/(len(b1)))
    print('Erro = ' + str(error))
    return e1, e2
```

(4)

2 Questão 2 - Ataque Intercept-Resend

2.1 2.1

In [695]:
```
np.random.seed(seed = 3)
b1 = randint(2, size=n)  # bits alice
alicebases = randint(2, size=n)
mensagem = codificar(0, b1, alicebases)

evabases = randint(2, size=n)
evamensagem = medir_mensagem(0,mensagem,evabases)

bobbases = randint(2, size=n)
bobresults = medir_mensagem(1, mensagem, bobbases)

alicekey = remover_errados(alicebases, bobbases, b1)
bobkey = remover_errados(alicebases, bobbases, bobresults)


p = 10  #amostra

selecaobit = randint(n, size = p)

mostraalice, mostrabob = erro(alicekey, bobkey, selecaobit)
print(" alice = " + str(mostraalice))
print(" bob = " + str(mostrabob))
```

```
Erro = 2.3809523809523796
 alice = [1, 1, 0, 0, 1, 0, 0, 0, 0, 0]
 bob = [1, 1, 1, 0, 1, 0, 0, 1, 0, 0]
```

(5)

2.2 2.2

A nova taxa de erro encontrada está no print do código.

2.3 2.3

A maior fração (de qubits) para que Alice deve interceptar para que a taxa de erro esteja abaixo de 10% é de 40%. Porque a cada fração que ela intercepta metade da metade dos bits vão ser afetados, atrapalhando Bob de parear com sua base. 25% de 40 é 10%, a taxa de erro máxima. Ela conhecerá 50% dos bits interceptados.

2.4 2.4

O XOR é aplicado a cada par de 2 bits, fazendo com que a nova fração dos bits da chave que Eva conhece é de 25% da anterior, logo 2.5 % da (taxa máxima) chave, dessa forma essa medida aumenta a segurança do protocolo.

```
|: def xor(b1,b2):
       b = b1 + b2
       return b%2

   def fxor(key):
       new_key = []
       i = 0
       while i < (len(key)-1):
           result = xor(key[i],key[i+1])
           new_key.append(result)
           i= i + 2
       return new_key
```

(6)

References

[1] *https://qiskit.org/textbook/ch-algorithms/quantum-key-distribution.html*.

[2] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[2] [1]