Kuantum Anahtar Dağılımı

Oguz Narli

May 2022

1 Giriş

Alice ve Bob, güvenli olmayan bir kanal (internet gibi) üzerinden gizli bir mesajı (Bob'un çevrimiçi bankacılık detayları gibi) iletmek istediklerinde, mesajı şifrelemek esastır

Alice ve Bob, anahtarlarını paylaşmak için Eve'nin bulunduğu klasik iletişim kanalını kullanmak istiyorlarsa, Eve'in bu anahtarın bir kopyasını kendisi için elde edip edemediğini söylemek mümkün değildir. Eve'in dinlemediğine dair tam bir güven duymaları gerekir. Bununla birlikte, Eve ve diğerleri bir kuantum iletişim kanalı kullanmaktalarsa, Alice ve Bob'un artık Eve'ye rahatlıkla güvenebilir. Çünki Bob'un mesajını Alice'e ulaşmadan önce okumaya çalışıp çalışmadığını bileceklerdir.

2 Kuantum Anahtar Dağılımı İçin Matematiksel Ön Bilgilendirme

Konuyu lineer cebirsel yönü anlatılmadan önce bir takım kuantum bilişimi hakkında ön bilgilendirme yapılması gerekmektedir.

Bu çalışmada klasik sistem yerine kuantum sistemi kullanıldığından bir değişkeni belirtirken bit yerine kübitlerle ifade ederiz. Eğer bir kuantum sisteminde algoritma inşa edilmek istenirse kuantum kapıları kullanılması gerekmektedir. Bu çalışma içersinde ve kuantum bilişiminde kullanılan temel kuantum kapılarını niteleyen matrisler aşağıda yer almaktadır:

Eğerki biz kuantum durumlarını bir koordinat üzerinde göstermek istiyorsak bunun için Bloch Küresi kullanılır.Bloch Küresi ;

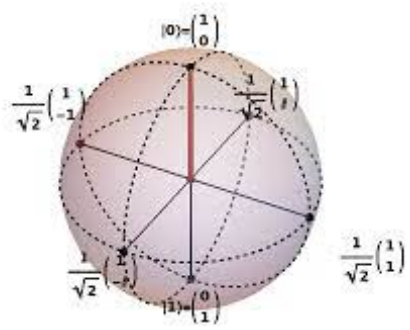

Figure 1: Bloch Küresi

Bloch küresinide kuantum durumlarını;

- $|0\rangle$ durumu +z ekseni.
- $|1\rangle$ durumu -z ekseni.
- $|i\rangle$ durumu +y ekseni.
- $|-i\rangle$ durumu -y ekseni.
- $|+\rangle$ durumu +x ekseni.
- $|-\rangle$ durumu -x ekseni.

temsil etmektedir.

Bitlere karşılık gelen $|0\rangle$ ve $|1\rangle$ durumları aşağıdaki vektörlerle ifade edilir:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$|1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Bu kübitler üzerinde işlemlerimizi gerçekleştireceğimiz kuantum kapıları aşağıda yer almaktadır.

X Kapısı;

X= $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Y Kapısı;

Y= $\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$

Z Kapısı;

Z= $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Hadamaard Kapısı

H=$1/\sqrt{2}$ $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Eğer ki bir kübiti üzerinde işlem gerçekleştirilmek istenirse kübit ile kuantum kapısı matris çarpımı yapılarak müdahale edilir

Hadamaard kapısının amacı kübiti süperpozisyon durumuna sokmayı amaçlamaktadır. Bir kübitin süperpozisyonda olmasının anlamı aynı anda hem 1 hemde 0 belirtmektedir.Şimdi 0 ve 1 durumlarını sırayla süperpozisyon durmuna sokalım:

$|+\rangle$=H $|0\rangle$;

$|+\rangle$= $1/\sqrt{2}$ $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$=$1/\sqrt{2}$ $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Eğer bu işlemi python üzerinde uygulamak istersek;

```
import numpy as np

zero_state=np.array(([1],[0]))
hadamaard_gate=np.array(([1,1],[1,-1]))/np.sqrt(2)
zero_result=hadamaard_gate@zero_state
```

Sonuç;

```
array([[0.70710678],
       [0.70710678]])
```

1 durumu için işlem yapalım;

$|-\rangle$=H $|1\rangle$;

$$|-\rangle = 1/\sqrt{2} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = 1/\sqrt{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Eğer bu işlemi python üzerinde uygulamak istersek;

```
import numpy as np

one_state=np.array(([0],[1]))
hadamaard_gate=np.array(([1,1],[1,-1]))/np.sqrt(2)
zero_result=hadamaard_gate@one_state
```

Sonuç;

```
array([[0.70710678],
       [-0.70710678]])
```

3 Kuantum Kriptolojide Bu Özelliklerinin Kullanımı

Kübitleri göz önüne alırken $|0\rangle$ ve $|1\rangle$ durumlarını klasik sistemde 0 ve 1 in karşılığı olarak görebiliriz. Ancak kuantum dünyasında bu durumlar aynı anda 0 ve 1 durumunda bulunabilirler. Aynı anda 0 ve 1 olma duruma süperpozisyon durmu denir. Önceki bölümde süperpozisyona geçişi Hadamaard kapısı ile yapabileceğimizden bahsettik. Peki bu özelliğin bizim için önemi nedir. Eğerki süperpozisyon durumundaki kübite tekrar Hadamaard kapısı uygularsak kübit ilk durumuna geri döner. Örneğin;

$|0\rangle$=H $|+\rangle$;

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} = 1/\sqrt{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} 1/\sqrt{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Python gösterimi;

```
import numpy as np

zero_state=np.array(([1],[0]))
hadamaard_gate=np.array(([1,1],[1,-1]))/np.sqrt(2)
zero_result=hadamaard_gate@zero_state #S perpozisyona sokulmakta
zero_result=hadamaard_gate@zero_result #Eski durumuna geri d nmekte
```

Sonuç;

```
array([[1],
       [0]])
```

İlk kısımda Hadaamard kapısı ile Alice göndereceği kübiti Hadamaard kapısı ile süperpoziyon durumuna sokmuş olusun. 2. Hadamaard uygulayan kişide Bob olsun. Yani Alice Hadamaard kapısı ile bilgisini gizlemiş Bob'ta tekrar Hadamaard kapısı uygulayarak çözmüş olur. Peki Eve bilgiyi ele geçirmek isterse nolur? Bir durumda ölçüm yapıldığında klasik sisteme geri dönüldüğünden kübit ya $|0\rangle$ yada $|1\rangle$ durumuna çökmesi gerekmektedir. Yani ölçüm yapıldığında matematiksel olarak olasılık her ikisi içinde % 50'dir. Yani Eve'nin doğru qubiti ölçme ihtimali % 50'dir. Bundan dolayı kuantum haberleşme kanalında bit yollanmak isterse Eve sisteme müdahale edicek ve doğru bilgiyi ölçmesi şansına kalıcaktır. Bu durumu matematikseel olarak göstermek istersek;

$|+\rangle = 1/\sqrt{2}\ |0\rangle + 1/\sqrt{2}\ |1\rangle$

$1/\sqrt{2}\begin{bmatrix}1\\1\end{bmatrix} = 1/\sqrt{2}\begin{bmatrix}1\\0\end{bmatrix} + 1/\sqrt{2}\begin{bmatrix}0\\1\end{bmatrix}$

Eğerki bir kübitin ölçülme olasığını ölçmek istiyorsanız onu bra vektörü ile çarpınız ve karesini alınız.Bra vektörü ket vektörünün transpoze alınmış halidir.Çıkan sonuç size olasılığı vermektedir.Yani;

$(\langle\phi|\phi\rangle)^2 \leq 1$

Eğer $|+\rangle$ için işlem yaparsak;

$\langle 0|+\rangle = 1/\sqrt{2}\ \langle 0|0\rangle + 1/\sqrt{2}\ \langle 0|1\rangle$

$|0\rangle$ ve $|1\rangle$ vektörler ortogonal olduklarından dolayı vektörel çarpım sonucu 0'a eşit olur.

$0 = (\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix})^2$

Python gösterimi;

```
import numpy as np

zero_state=np.array(([1],[0]))
one_state=np.array(([0],[1]))
np.sqrt(zero_state.T@one_state)
```

Sonuç;

```
array([[0.]])
```

Aynı işlemi süperpozisyon durumu içinde yapalım;

$(\langle 0|+\rangle)^2 = (1/\sqrt{2})^2$

$(\langle 0|+\rangle)^2 = 1/2$

Python gösterimi;

```
import numpy as np

zero_state=np.array(([1],[0]))
hadamaard_gate=np.array(([1,1],[1,-1]))/np.sqrt(2)
np.square(zero_state.T@hadamaard_gate@zero_state)
```

Sonuç;

```
array([[0.5]])
```

Şimdi ise Alice, Bob ve Eve'nin haberleşme kanalını Python'un qiskit aracında gösterelim.İlk olarak Eve bilgiyi ele geçirmeden sağlıklı şekilde bilgiyi karşı tarafa gönderilmesini gösterelim.

```
from qiskit import QuantumCircuit, Aer, transpile, assemble
from qiskit.visualization import plot_histogram, plot_bloch_multivector
from numpy.random import randint

qc = QuantumCircuit(1,1) # 1 tane k bit tan mlan r
qc.h(0)
qc.barrier()
qc.h(0)
```

```
qc.measure(0,0)

display(qc.draw())
aer_sim = Aer.get_backend('aer_simulator')
job = aer_sim.run(assemble(qc))
plot_histogram(job.result().get_counts())
```

Çıktı aşağıdaki gibi olucaktır.

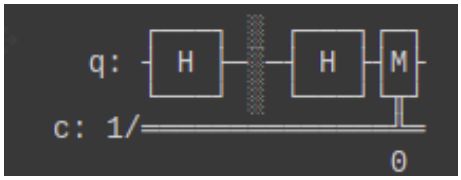

Figure 2: Kuantum Devresi

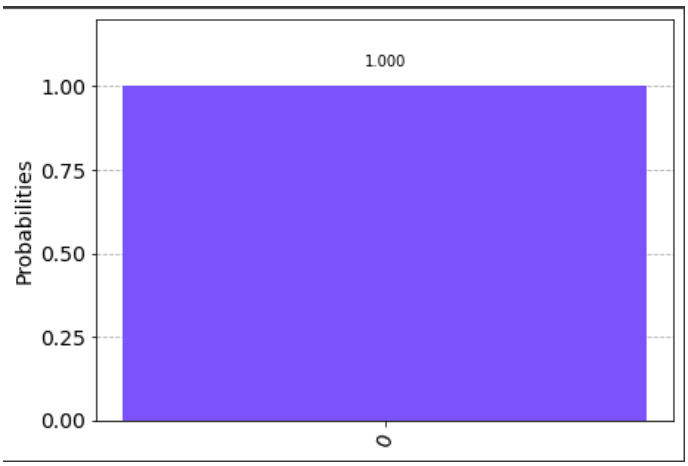

Figure 3: Ölçüm Sonucu Olasılık

Eğerki Eve sisteme müdahale ederse nasıl bir sonuç elde ederiz.

```
qc = QuantumCircuit(1,1)

qc.h(0)

qc.measure(0, 0)
qc.barrier()

qc.h(0)
qc.measure(0,0)

display(qc.draw())
aer_sim = Aer.get_backend('aer_simulator')
job = aer_sim.run(assemble(qc))
plot_histogram(job.result().get_counts())
```

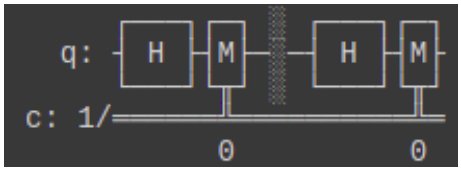

Figure 4: Kuantum Devresi

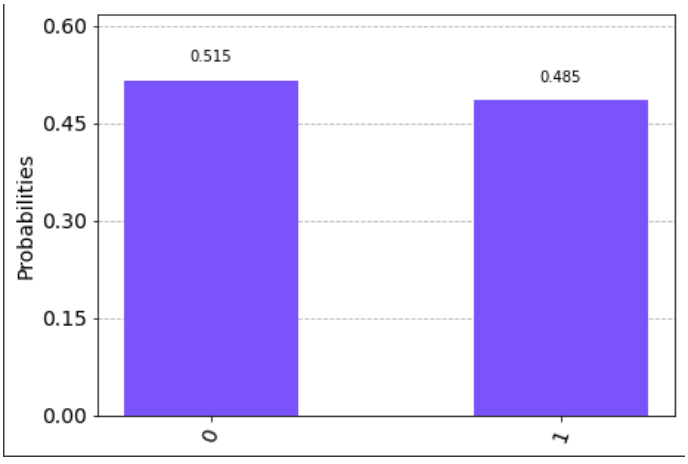

Figure 5: Ölçüm Sonucu Olasılık

Kuantum anahtar dağıtım protokolü, bu işlemi, bir dinleyicinin bu müdahaleden kurtulma şansının ihmal edilebilir bir şekilde olduğu kadar, yeterince tekrarlamayıda içerir. Kabaca şöyledir:

Adım 1

Alice bir dizi rastgele bit seçer, örneğin:

1000101011010100

Birde rastgele her bit için basis seçmektedir.

ZZXZXXXZXZXXXXXX

```
np.random.seed(seed=0)
n = 100
## Adim 1
# Alice bitleri turetir
alice_bits = randint(2, size=n)
# Alice basis turetir
alice_bases = randint(2, size=n)#0 X tabani 1 Z tabani
print(alice_bits)
print(alice_bases)
```

Çıktı:

```
[0 1 1 0 1 1 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1 1 0 0 1 1 1 1 0 1 0 1 0 1 1 0
 1 1 0 0 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 0 0 1 1 0 1 0 1 0 0 0 0 0 1 1 0 0
 0 1 1 0 1 0 0 1 0 1 1 1 1 1 1 0 1 1 0 0 1 0 0 1 1 0]

 [1 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 1 0 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 1 0
 0 0 0 1 1 0 0 1 0 1 1 1 1 0 0 0 1 0 1 1 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0
 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0]
```

Adım 2

Alice daha sonra seçtiği tabanı kullanarak her bir biti bir dizi kübit üzerine kodlar; bu, her kübitin rastgele seçilen $|0\rangle$, $|1\rangle$, $|+\rangle$ veya $|-\rangle$ durumlarından birinde olduğu anlamına gelir. Bu durumda, kübit dizisi şöyle görünür:

$|1\rangle$ $|0\rangle$ $|+\rangle$ $|0\rangle$ $|-\rangle$ $|+\rangle$ $|-\rangle$ $|0\rangle$ $|-\rangle$ $|1\rangle$ $|+\rangle$ $|-\rangle$ $|+\rangle$ $|-\rangle$ $|+\rangle$ $|+\rangle$

Encode işlemi aşağıda yapılmaktadır:

```
#Adim 2
def encode_message(bits, bases):
    message = []
    for i in range(n):
        qc = QuantumCircuit(1,1)
        if bases[i] == 0: # Kubit Z tabanina ayarlan r
            if bits[i] == 0:
                pass
            else:
                qc.x(0)
        else: # Prepare qubit in X-basis
            if bits[i] == 0:
                qc.h(0)
            else:
                qc.x(0)
                qc.h(0)
        qc.barrier()
        message.append(qc)
    return message
message=encode_message(alice_bits,alice_bases)
```

Adım 3

Bob daha sonra her bir kübiti rastgele ölçer, örneğin aşağıdaki tabanları kullanabilir:

XZZZXZXZXZXZZZXZ

Bob, ölçüm sonuçlarını gizli tutar.

```
#Adim 2
def measure_message(message, bases):
    backend = Aer.get_backend('aer_simulator')
    measurements = []
    for q in range(n):
        if bases[q] == 0: # measuring in Z-basis
            message[q].measure(0,0)
        if bases[q] == 1: # measuring in X-basis
            message[q].h(0)
            message[q].measure(0,0)
        aer_sim = Aer.get_backend('aer_simulator')
        qobj = assemble(message[q], shots=1, memory=True)
        result = aer_sim.run(qobj).result()
        measured_bit = int(result.get_memory()[0])
```

```
            measurements.append(measured_bit)
    return measurements
bob_bases = randint(2, size=n)
bob_results = measure_message(message, bob_bases)
```

Adım 4 Bob ve Alice daha sonra her bir kübit için hangi tabanı kullandıklarını herkese açık olarak paylaşırlar. Bob, bir kübiti Alice'in hazırladığı aynı tabanda ölçtüyse, bunu paylaşılan gizli anahtarlarını bir parçasını oluşturmak için kullanırlar, aksi takdirde o bit için bilgileri atarlar.

```
#Adim 4
def remove_garbage(a_bases, b_bases, bits):
    good_bits = []
    for q in range(n):
        if a_bases[q] == b_bases[q]:
            # If both used the same basis, add
            # this to the list of 'good' bits
            good_bits.append(bits[q])
    return good_bits
alice_key = remove_garbage(alice_bases, bob_bases, alice_bits)
bob_key = remove_garbage(alice_bases, bob_bases, bob_results)
print(bob_key)
```

Adım 5

Son olarak, Bob ve Alice anahtarlarının rastgele bir örneğini paylaşırlar ve eğer örnekler eşleşirse, aktarımlarının başarılı olduğundan (küçük bir hata payıyla) emin olabilirler.

```
#Adim 5
def sample_bits(bits, selection):
    sample = []
    for i in selection:
        i = np.mod(i, len(bits))
        sample.append(bits.pop(i))
    return sample
sample_size = 15
bit_selection = randint(n, size=sample_size)

bob_sample = sample_bits(bob_key, bit_selection)
print("  bob_sample = " + str(bob_sample))
alice_sample = sample_bits(alice_key, bit_selection)
print("alice_sample = "+ str(alice_sample))
```

Sonuç:

```
bob_sample = [0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1]
alice_sample = [0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1]
```

4 Sonuç

Bu çalışmada lineer cebirin öncemi kuantum bilişiminde kullanılan her işlem vektörel ve matris üzerinde gösterilmektedir.Kuantum kapıları ve kübitlerin durumlarının lineer cebirsel karşılığı mevcuttur. Bu çalışmada matematiksel işlemlerle temel kuantum bilişimi bilgisi ve güvenlikteki önemini anlattık.Ardından bu özellikleri kullanılarak qiskit aracı ile kuantum anahtar dağılımı algoritması geliştirdir.