Qudit Implementation

Amir Mohammad Moghaddam 9823144

April 2024

Chapter 1

Qudit Theory

1.1 Introduction

In this chapter, we delve into the theory behind leveraging the spatial freedom offered by orbital angular momentum (OAM) to access an infinite set of discrete bases for implementing quantum protocols. As part of our project, we explore Qudits and their implementation methodologies. Subsequently, in the next chapter, we will delve into simulations and analyze the results obtained.

In order to proceed, our approach entails initially understanding the concept of a qudit and the tools at our disposal for its implementation. Following this, we will explore the utilization of these tools to effectively implement qudits. Subsequently, we will identify two bases suitable for implementing our protocols and calculate the theoretical error rates in the presence of Eve. Finally, we will address the issue of Cross-talk and integrate a rudimentary scheme to mitigate its effects within our simulation.

1.2 General Qudit state

To address the question of implementing a qudit, we must first comprehend what a qudit is. The general state of a d-dimensional qudit can be expressed as:

$$|\Psi\rangle_d = a_0|0\rangle + a_1|1\rangle + a_2|2\rangle + a_3|3\rangle + ... + a_d|d\rangle \tag{1.1}$$

Where $a_0, a_1, ..., a_d$ are the probability amplitudes and their sum of squares must be normalized to 1.Now that we have qubits at our disposal, the question arises: how can we employ them to represent the states described above? To accomplish this, we leverage the joint tensor product space of qubits. When $d = 2^n$, where n represents the number of qubits, we allocate n qubits to precisely match the size of our qudit. However, in cases where $d \neq 2^n$, we opt for the closest n value such that $d < 2^n$. In such instances, we treat the remaining $d < 2^n$ eigenstates as a single basis for our qudit. For example in case of a uniform qudit with $d = 4$ we can write :

$$\begin{aligned}
&|\Psi\rangle_4 = a_0|0\rangle + a_1|1\rangle + a_2|2\rangle + a_3|3\rangle \\
&|\Psi\rangle_4 = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\
&|\Psi\rangle_4 = \frac{1}{2}(|0\rangle \otimes |0\rangle + |0\rangle \otimes |1\rangle + |1\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle) \\
&\rightarrow |q_0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad |q_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)
\end{aligned} \tag{1.2}$$

As demonstrated in eq.(1.2), our simulation involves a d = 4 qudit, for which we utilize 2 qubits to emulate its behavior. So the next step is to find the bases we want to use.

1.3 Basis Implementation

In this project, the implementation of the BB84 protocol necessitates the utilization of Mutually Unbiased Bases (MUBs). The choice of the number of MUBs is flexible, with a minimum requirement of 2 bases, which can scale up depending on the dimensionality of the system. To commence our project, we opt for 2 MUBs, allowing for a comparison between the BB84 protocol in 4 dimensions and its counterpart in 2 dimensions. The primary distinction lies in the expansion of our alphabet size and the increase in entropy for each symbol. In selecting our bases, we will consider that the eigenstates are OAM (Orbital Angular Momentum) states($|l\rangle$), with the winding number($l = 0, \pm1, \pm2, ...$) serving as the Degree of Freedom (DOF).

For our bases, we will designate one as the pure OAM basis, comprising the following four OAM states:

$$|\Psi_{OAM}\rangle \in \{|0\rangle = |l = -3\rangle, |1\rangle = |-1\rangle, |2\rangle = |1\rangle, |3\rangle = |3\rangle\} \tag{1.3}$$

And another as the Fourier transform of the first basis, consisting of:

$$\begin{aligned}
&|\Phi^n_{ANG}\rangle = \frac{1}{\sqrt{d}} \sum_{k=0}^{4} |\Psi^k_{OAM}\rangle e^{\frac{j2\pi kn}{d}} \\
&|\Phi_{ANG}\rangle \in \{|0\rangle_{ANG}, |1\rangle_{ANG}, |2\rangle_{ANG}, |3\rangle_{ANG}\} \\
&\rightarrow |0_{ANG}\rangle = \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle) \\
&\rightarrow |1_{ANG}\rangle = \frac{1}{2}(|0\rangle + j|1\rangle - |2\rangle - j|3\rangle) \\
&\rightarrow |2_{ANG}\rangle = \frac{1}{2}(|0\rangle - |1\rangle + |2\rangle - |3\rangle) \\
&\rightarrow |3_{ANG}\rangle = \frac{1}{2}(|0\rangle - j|1\rangle - |2\rangle + j|3\rangle)
\end{aligned} \tag{1.4}$$

Now that we have defined our bases, it's time to examine the theoretical outcomes. Initially, we will explore the concept of Secret Key Rate, followed by its interpretation.

1.4 Probabilities

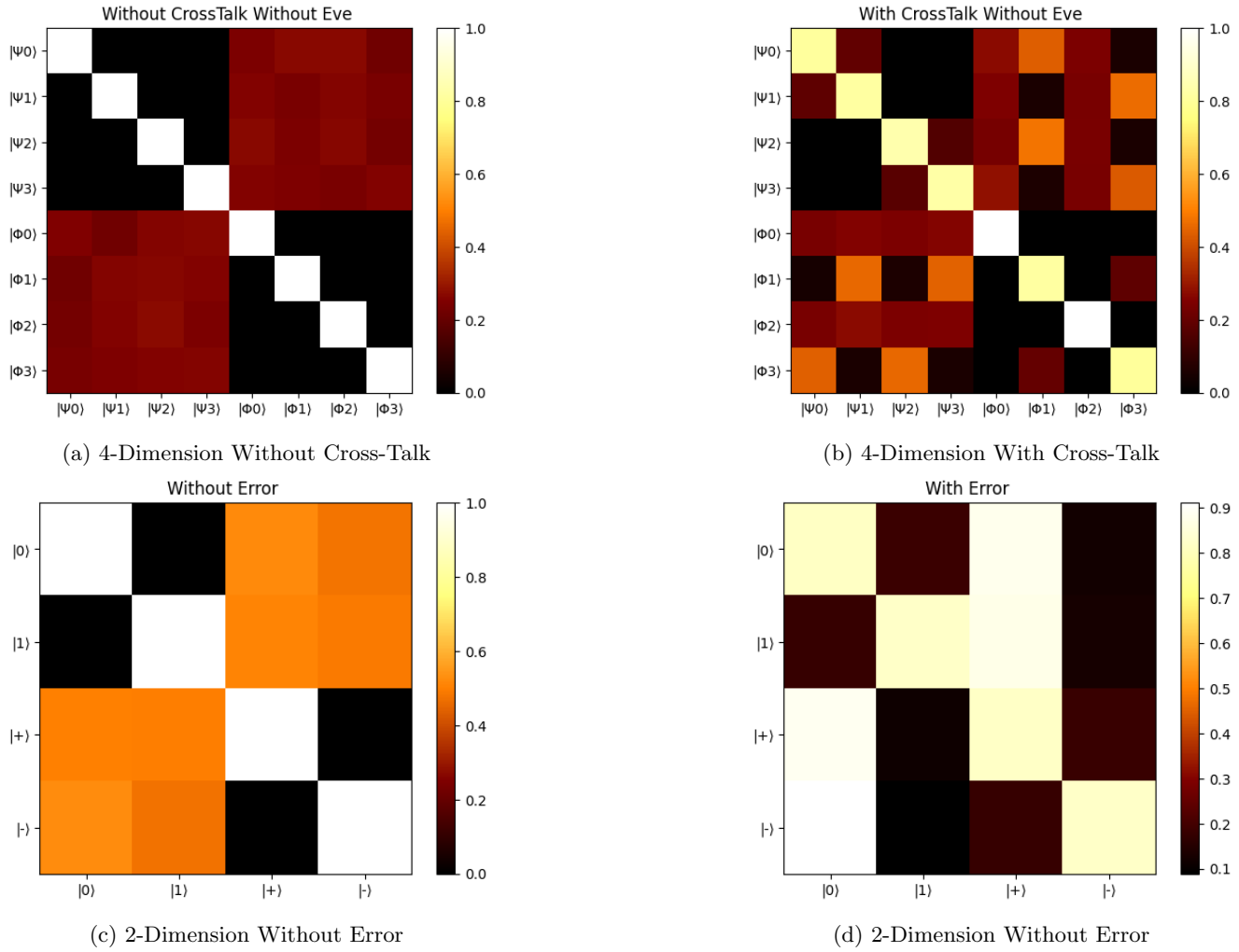
Now, we will analyze the probabilities of outcomes and assess the behavior of error probabilities with and without Cross-Talk. Furthermore, we will compare these results with those obtained from a 2-dimensional BB84 algorithm. To accomplish this, we will utilize our simulation, running it 1000 times to generate probability matrices as output. For the scenario without Cross-Talk and where Eve is not eavesdropping, the probability matrix is as follows:

$$\begin{bmatrix}
\langle\Psi_0|\Psi_0\rangle & \langle\Psi_1|\Psi_0\rangle & \dots & \langle\Phi_0|\Psi_0\rangle & \dots & \langle\Phi_4|\Psi_0\rangle \\
\langle\Psi_0|\Psi_1\rangle & \langle\Psi_1|\Psi_1\rangle & \dots & \langle\Phi_0|\Psi_1\rangle & \dots & \langle\Phi_4|\Psi_1\rangle \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\langle\Psi_0|\Phi_0\rangle & \langle\Psi_1|\Phi_0\rangle & \dots & \langle\Phi_0|\Phi_0\rangle & \dots & \langle\Phi_4|\Phi_0\rangle \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\langle\Psi_0|\Phi_4\rangle & \langle\Psi_1|\Phi_4\rangle & \dots & \langle\Phi_0|\Phi_4\rangle & \dots & \langle\Phi_4|\Phi_4\rangle
\end{bmatrix} =
\begin{bmatrix}
1. & 0. & 0. & 0. & 0.243 & 0.268 & 0.267 & 0.222 \\
0. & 1. & 0. & 0. & 0.257 & 0.241 & 0.261 & 0.241 \\
0. & 0. & 1. & 0. & 0.266 & 0.243 & 0.262 & 0.229 \\
0. & 0. & 0. & 1. & 0.255 & 0.249 & 0.239 & 0.257 \\
0.252 & 0.222 & 0.261 & 0.265 & 1. & 0. & 0. & 0. \\
0.222 & 0.26 & 0.262 & 0.256 & 0. & 1. & 0. & 0. \\
0.227 & 0.257 & 0.273 & 0.243 & 0. & 0. & 1. & 0. \\
0.235 & 0.248 & 0.257 & 0.26 & 0. & 0. & 0. & 1.
\end{bmatrix} \tag{1.5}$$

The matrix visualization for the scenario without Cross-Talk is depicted in Figure 1.1a. Additionally, the probability matrix for the case with Cross-Talk is as follows:

$$\begin{bmatrix} \langle\Psi_0|\Psi_0\rangle & \langle\Psi_1|\Psi_0\rangle & \dots & \langle\Phi_0|\Psi_0\rangle & \dots & \langle\Phi_4|\Psi_0\rangle \\ \langle\Psi_0|\Psi_1\rangle & \langle\Psi_1|\Psi_1\rangle & \dots & \langle\Phi_0|\Psi_1\rangle & \dots & \langle\Phi_4|\Psi_1\rangle \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \langle\Psi_0|\Phi_0\rangle & \langle\Psi_1|\Phi_0\rangle & \dots & \langle\Phi_0|\Phi_0\rangle & \dots & \langle\Phi_4|\Phi_0\rangle \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \langle\Psi_0|\Phi_4\rangle & \langle\Psi_1|\Phi_4\rangle & \dots & \langle\Phi_0|\Phi_4\rangle & \dots & \langle\Phi_4|\Phi_4\rangle \end{bmatrix} = \begin{bmatrix} 0.806 & 0.194 & 0. & 0. & 0.27 & 0.435 & 0.244 & 0.051 \\ 0.185 & 0.815 & 0. & 0. & 0.247 & 0.054 & 0.235 & 0.464 \\ 0. & 0. & 0.837 & 0.163 & 0.231 & 0.477 & 0.24 & 0.052 \\ 0. & 0. & 0.173 & 0.827 & 0.282 & 0.056 & 0.236 & 0.426 \\ 0.238 & 0.255 & 0.246 & 0.261 & 1. & 0. & 0. & 0. \\ 0.046 & 0.456 & 0.056 & 0.442 & 0. & 0.813 & 0. & 0.187 \\ 0.235 & 0.27 & 0.246 & 0.249 & 0. & 0. & 1. & 0. \\ 0.439 & 0.051 & 0.457 & 0.053 & 0. & 0.197 & 0. & 0.803 \end{bmatrix} \tag{1.6}$$

The visualization of this matrix is also depicted in Figure 1.1b.



(a) 4-Dimension Without Cross-Talk

(b) 4-Dimension With Cross-Talk

(c) 2-Dimension Without Error

(d) 2-Dimension Without Error

Figure 1.1: Matrix Visualization

Now, let's examine the probability matrices for the 2-dimensional case under ideal transmission conditions and with alignment errors.For the scenario without Alignment Error and where Eve is not eavesdropping, the probability

matrix is as follows:

$$\begin{bmatrix} \langle 0|0\rangle & \langle 1|0\rangle & \langle +|0\rangle & \langle -|0\rangle \\ \langle 0|1\rangle & \langle 1|1\rangle & \langle +|1\rangle & \langle -|1\rangle \\ \langle 0|+\rangle & \langle 1|+\rangle & \langle +|+\rangle & \langle -|+\rangle \\ \langle 0|-\rangle & \langle 1|-\rangle & \langle +|-\rangle & \langle -|-\rangle \end{bmatrix} = \begin{bmatrix} 1. & 0. & 0.522 & 0.478 \\ 0. & 1. & 0.511 & 0.489 \\ 0.503 & 0.497 & 1. & 0. \\ 0.524 & 0.476 & 0. & 1. \end{bmatrix} \tag{1.7}$$

And for the scenario with Alignment Error , the probability matrix is as follows:

$$\begin{bmatrix} \langle 0|0\rangle & \langle 1|0\rangle & \langle +|0\rangle & \langle -|0\rangle \\ \langle 0|1\rangle & \langle 1|1\rangle & \langle +|1\rangle & \langle -|1\rangle \\ \langle 0|+\rangle & \langle 1|+\rangle & \langle +|+\rangle & \langle -|+\rangle \\ \langle 0|-\rangle & \langle 1|-\rangle & \langle +|-\rangle & \langle -|-\rangle \end{bmatrix} = \begin{bmatrix} 0.817 & 0.183 & 0.881 & 0.119 \\ 0.176 & 0.824 & 0.874 & 0.126 \\ 0.886 & 0.114 & 0.818 & 0.182 \\ 0.911 & 0.089 & 0.177 & 0.823 \end{bmatrix} \tag{1.8}$$

The visualizations for these matrices are shown in Figure 1.1c and 1.1d.

Chapter 2

Simulation

2.1 Qudit Implementation

To incorporate qudits into our simulation, we designed an object that utilizes **Qistit's QuantumCircuit** with two qubits. Within this object, we integrated methods for measurements(**measure()**) and execution(**run()**). Additionally, we included a translation method to convert the results from the two-qubit form into the qudit form.

```
class QuditCircuit:
    def __init__(self, d, n):
        # Calculate the total number of qubits
        self.num_qudits = n
        self.dimension = d
        self.num_qubits = n * d/2
        self.Counts = []
        self.Counts2 = []
        # Create a QuantumCircuit object with the specified dimensions
        self.circuit = QuantumCircuit(self.num_qubits, self.num_qubits)
    def measure(self):
        # Add a barrier to separate the measurement operations
        self.circuit.barrier()
        # Measure all qubits
        qubit_indices = list(range(int(self.num_qubits)))
        self.circuit.measure(qubit_indices, qubit_indices)
    def CT(self,Gate):
      CT_Gate = qi.Operator(Gate)
      for i in range(0, int(self.num_qubits), 2):
        self.circuit.unitary(CT_Gate, [i,i+1], label="CrossTalk")
    def Translate(self):
      # Translate each key into the desired format
      translated_data = {}
      for key, value in self.Counts.items():
          translated_key = ''.join([str(int(key[i:i+2], 2)) for i in range(0, len(key), 2)])
          translated_data[translated_key] = value
      sorted_data = {k: v for k, v in sorted(translated_data.items(), key=lambda item: base4_sort(
item[0]))}
      self.Counts2 = sorted_data
    def run(self,Shot):
      # Use Aer's qasm_simulator
      simulator = Aer.get_backend('qasm_simulator')
      # Execute the circuit on the qasm simulator
      new_circuit = transpile(self.circuit, simulator)
      job = simulator.run(new_circuit, shots = Shot)
      # Grab results from the job
      result = job.result()
      # Return counts
      self.Counts = result.get_counts(self.circuit)
      self.Translate()
```

```
}
```

Listing 2.1: Qudit Object

There is a method named ***CT()*** in this object where using this method we can Add **Cross-Talk** to our Simulation.

In the BB84 protocol functions section, I included various state modulation functions tailored for qudits.

```
def Modulation(Qudit,Dit,AliceBasis):
  if AliceBasis == 0:
    #print("Basis 0")
    if Dit == 1:
      Qudit.circuit.x(0)
    if Dit == 2:
      Qudit.circuit.x(1)
    if Dit == 3:
      Qudit.circuit.x(0)
      Qudit.circuit.x(1)
  else:
     #print("Basis 1")
     if Dit == 0:
      #print("Entered 0")
      Qudit.circuit.h(0)
      Qudit.circuit.h(1)
     elif Dit == 1:
      #print("Entered 1")
      Qudit.circuit.x(1)
      Qudit.circuit.h(1)
      Qudit.circuit.h(0)
      Qudit.circuit.s(0)
     elif Dit == 2:
      #print("Entered 2")
      Qudit.circuit.x(0)
      Qudit.circuit.h(0)
      Qudit.circuit.h(1)
     elif Dit == 3:
      #print("Entered 3")
      Qudit.circuit.x(0)
      Qudit.circuit.h(0)
      Qudit.circuit.s(0)
      Qudit.circuit.x(1)
      Qudit.circuit.h(1)
  return Qudit
```

Listing 2.2: Modulation Function

In case of the first Basis the modulation process is clear. But for the second basis we can show them as Figure 2.1

(a) $|\Phi_0\rangle$ State

(b) $|\Phi_1\rangle$ State

(c) $|\Phi_2\rangle$ State

(d) $|\Phi_3\rangle$ State

Figure 2.1: ANG State Modulation circuits

Also there is a function added for Measurements in the second basis:

```
def ReciviengSinglePhoton(Qudit,BobBasis,Shot):
  csgate = SdgGate().control(1) # the parameter is the amount of control points you want
  if BobBasis == 1:
     #print("Measure Basis 1")
     Qudit.circuit.h(1)
     Qudit.circuit.append(csgate, [1, 0])
     Qudit.circuit.h(0)
     Qudit.circuit.cx(1,0)
     Qudit.circuit.cx(0,1)
     Qudit.circuit.cx(1,0)
  Qudit.measure()
  Qudit.run(Shot)
```

Listing 2.3: Measurement Function

The qubit circuit corresponding to this function is illustrated in the figure below:

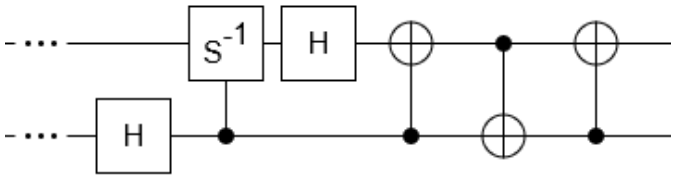


Figure 2.2: Measurement Function's circuit

2.2 Results

In this section, we will analyze various simulation outcomes, including average error rates, maximum error rates, and minimum error rates. We will conduct 100 runs of the simulation for different cases, which include:

1. 2-dimensional BB84 :
   (a) No Error No Eve
   (b) No Error with Eve
   (c) with Alignment Error No Eve
   (d) with Alignment Error and Eve
2. 4-dimensional BB84 :
   (a) No Error No Eve
   (b) No Error with Eve
   (c) with Cross-Talk Error No Eve
   (d) with Cross-Talk Error and Eve

Below are the results for each of these cases with Bit(Dit) Sequence length of 64:

2.2.1 2-dimensional BB84

```
######Without Eve and Allignment Error######
Max Error rate: 0.0
Min Error rate: 0.0
Average of error :  0.0
```

```
######With Eve Without Allignment Error######
Max Error rate: 0.40625
Min Error rate: 0.037037037037037035
Average of error :  0.24524262257710286
Eve's Max Error rate: 0.47058823529411764
Eve's Min Error rate: 0.037037037037037035
Eve's Average of error :  0.25941218312901915


#####Without Eve With Allignment Error#######
Max Error rate: 0.34285714285714286
Min Error rate: 0.05555555555555555
Average of error :  0.18739516508943982


######With Eve and Allignment Error######
Max Error rate: 0.6060606060606061
Min Error rate: 0.19230769230769232
Average of error :  0.4060310755883736
Eve'sMax Error rate: 0.53125
Eve's Min Error rate: 0.0
Eve's Average of error :  0.3375617109035062
```

2.2.2 4-dimensional BB84

```
######Without Eve and CrossTalk######
Max Error rate: 0.0
Min Error rate: 0.0
Average of error :  0.0


######With Eve Without CrossTalk######
Max Error rate: 0.6285714285714286
Min Error rate: 0.14285714285714285
Average of error :  0.3809400164147009
Eve's Max Error rate: 0.5714285714285714
Eve's Min Error rate: 0.25
Eve's Average of error :  0.39109044719379293


#####Without Eve With CrossTalk#######
Max Error rate: 0.3142857142857143
Min Error rate: 0.0
Average of error :  0.14534804086920894


######With Eve and CrossTalk######
```

```
Max Error rate: 0.6551724137931034
Min Error rate: 0.27586206896551724
Average of error :  0.45933256076776713
Eve'sMax Error rate: 0.6285714285714286
Eve's Min Error rate: 0.0
Eve's Average of error :  0.43038109557166243
```

2.3 Discussion

At first glance, the results indicate that the error introduced by Eve to Bob's key is greater in the 4-dimensional case, suggesting that our system has more tolerance for Quantum Bit Error Rate (QBER) and can accommodate a higher level of error. Additionally, the error introduced by Cross-Talk is lower than the error caused by Alignment. However, it's essential to note that this comparison may not be entirely reliable because the Cross-Talk matrix selected is not based on a real-world scenario; instead, it is a simplified unitary matrix. Specifically, the Cross-Talk matrix is as follows:

$$\text{Gate} = \begin{bmatrix} 0.9 & \frac{\sqrt{19}}{10}i & 0 & 0 \\ \frac{\sqrt{19}}{10}i & 0.9 & 0 & 0 \\ 0 & 0 & 0.9 & \frac{\sqrt{19}}{10}i \\ 0 & 0 & \frac{\sqrt{19}}{10}i & 0.9 \end{bmatrix} \tag{2.1}$$

Wherein, it induces interference and Cross-Talk only between neighboring modes with a probability of 0.19. But it also exclude $l = \mp 1$ for $l = \pm 1$. Returning to our results, in both cases, we observe that Eve can obtain a key with no errors. However, even in the scenario with minimal error, Eve still introduces non-zero errors into Bob's key. So we can almost always tell Eve's presence.