Michał Kałmucki 151944

E91 Quantum Key Distribution Project Report

This report summarizes the results of simulating the E91 quantum key distribution protocol using Qiskit. The experiment involves generating singlet states, performing measurements in different bases, generating a secret key, and computing the CHSH correlation to verify quantum entanglement.

Task 1: Singlet State Preparation

Charlie prepares singlet states (|\psi_s\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)) which are then distributed between Alice and Bob.

Code Snippet:

```
def singlet():
    qr = QuantumRegister(2, 'qr')
    cr = ClassicalRegister(2, 'cr')
    qc = QuantumCircuit(qr, cr)

    # |ψ_s> = (|01> - |10>)/√2
    qc.x(1)
    qc.h(0)
    qc.cx(0,1)
    qc.z(0)
    return qc
```

Result: Singlet states prepared for all 1024 trials.

Task 2: Measurement by Alice and Bob

Alice and Bob randomly choose measurement bases:

- 1 → X basis
- 2 → W = (X+Z)/√2
- 3 → Z basis

Measurements are applied to the singlet qubits and results converted to ±1.

Code Snippet:

```
def measure_a(qc, basis, target_qubit=0):
    if basis == 1: qc.h(target_qubit)
    elif basis == 2: qc.s(target_qubit); qc.h(target_qubit); qc.t(target_qubit);
qc.h(target_qubit)
    elif basis == 3: pass
    return qc

def measure_b(qc, basis, target_qubit=1):
    if basis == 1: qc.s(target_qubit); qc.h(target_qubit); qc.t(target_qubit);
qc.h(target_qubit)
    elif basis == 2: pass
    elif basis == 3: qc.s(target_qubit); qc.h(target_qubit); qc.tdg(target_qubit);
qc.h(target_qubit)
    return qc
```

Result: Measurement outcomes recorded for 1024 singlet pairs.

Task 3: Recording the Results

The measurement results are stored as classical bits and converted to ±1 for computation.

Sample result snippet (first few measurements):

```
{
    "alice_bit": 0,
    "bob_bit": 1,
    "alice_basis": 1,
    "bob_basis": 3,
    "a": 1,
    "a_prime": -1
}
```

Task 4: Key Generation from Compatible Bases

Alice and Bob extract the secret key from compatible measurement bases:

- Alice basis 2 & Bob basis 1
- Alice basis 3 & Bob basis 2

Code Snippet:

```
alice_key = []
bob_key = []
mismatches = 0

for r in results:
    if (r["alice_basis"]==2 and r["bob_basis"]==1) or (r["alice_basis"]==3 and
r["bob_basis"]==2):
        alice_key.append(r["alice_bit"])
        bob_corrected = 1 - r["bob_bit"]
        bob_key.append(bob_corrected)
        if r["alice_bit"] != bob_corrected:
            mismatches += 1

print(f"Number of mismatched key bits: {mismatches}")
```

Result:

- Number of mismatched key bits: 0
- Total key bits generated: 249

Task 5: CHSH Correlation Table

Results are grouped by measurement type and outcome to calculate probabilities and expectation values.

Code Snippet:

```
def countGroup(group):
    counts = {(1,1):0,(1,-1):0,(-1,1):0,(-1,-1):0}
    for a,a_p in group:
        counts[(a,a_p)] +=1
    return counts

def calculateExpectation(counts):
    total = sum(counts.values())
    return sum(a*a_p*(count/total) for (a,a_p), count in counts.items())
```

CHSH Table Results:

Measurement	(b,b')	(a,a')	n_ij	p_ij	p·(a·a')	Expectation
X ⊗ W	(1, 1)	(1, 1)	10	0.0862	0.0862	-0.6897
X ⊗ W	(1, 1)	(1, -1)	40	0.3448	-0.3448	-0.6897
X ⊗ W	(1, 1)	(-1, 1)	58	0.5000	-0.5000	-0.6897
X ⊗ W	(1, 1)	(-1, -1)	8	0.0690	0.0690	-0.6897
X ⊗ V	(1, 3)	(1, 1)	42	0.3717	0.3717	0.6283
X ⊗ V	(1, 3)	(1, -1)	14	0.1239	-0.1239	0.6283
X ⊗ V	(1, 3)	(-1, 1)	7	0.0619	-0.0619	0.6283
X ⊗ V	(1, 3)	(-1, -1)	50	0.4425	0.4425	0.6283
Z ⊗ W	(3, 1)	(1, 1)	10	0.1000	0.1000	-0.7400
Z ⊗ W	(3, 1)	(1, -1)	46	0.4600	-0.4600	-0.7400
Z ⊗ W	(3, 1)	(-1, 1)	41	0.4100	-0.4100	-0.7400
Z ⊗ W	(3, 1)	(-1, -1)	3	0.0300	0.0300	-0.7400

Measurement	(b,b')	(a,a')	n_ij	p_ij	p·(a·a')	Expectation
Z ⊗ V	(3, 3)	(1, 1)	5	0.0431	0.0431	-0.6552
Z ⊗ V	(3, 3)	(1, -1)	48	0.4138	-0.4138	-0.6552
Z ⊗ V	(3, 3)	(-1, 1)	48	0.4138	-0.4138	-0.6552
Z ⊗ V	(3, 3)	(-1, -1)	15	0.1293	0.1293	-0.6552

Task 6: CHSH Correlation Value

Code Snippet:

```
E_XW = calculateExpectation(countGroup(grouped[(1,1)]))
E_XV = calculateExpectation(countGroup(grouped[(1,3)]))
E_ZW = calculateExpectation(countGroup(grouped[(3,1)]))
E_ZV = calculateExpectation(countGroup(grouped[(3,3)]))
S = E_XW - E_XV + ZW + ZV
print(f"CHSH correlation value: S = {S}")
```

Result:

- CHSH correlation value: S = -2.7131
- Total key bits: 249

Conclusion:

The simulation successfully generates secret key bits using compatible measurement bases and demonstrates a CHSH correlation value exceeding classical limits, indicating quantum entanglement between Alice's and Bob's qubits.