Quantum Phase Estimation
And
Shor's Algorithm

Xiaojian Du

Oct. 19th 2023

Quantpostela 2023 IGFAE

Outline

Pre-requisite: Quantum Fourier Transform

In the lecture, we will learn:

1. Quantum Phase Estimation

Quantum kickback
Quantum interference and Phase Estimation

2. RSA Cryptography

How to encrypt and decrypt
How to design public and private keys

3. Basic idea of Shor’s Algorithm

How to hack the RSA cryptography
Factoring problem and period-finding problem
Quantum algorithm for solving period-finding problem

Quantum Phase Estimation

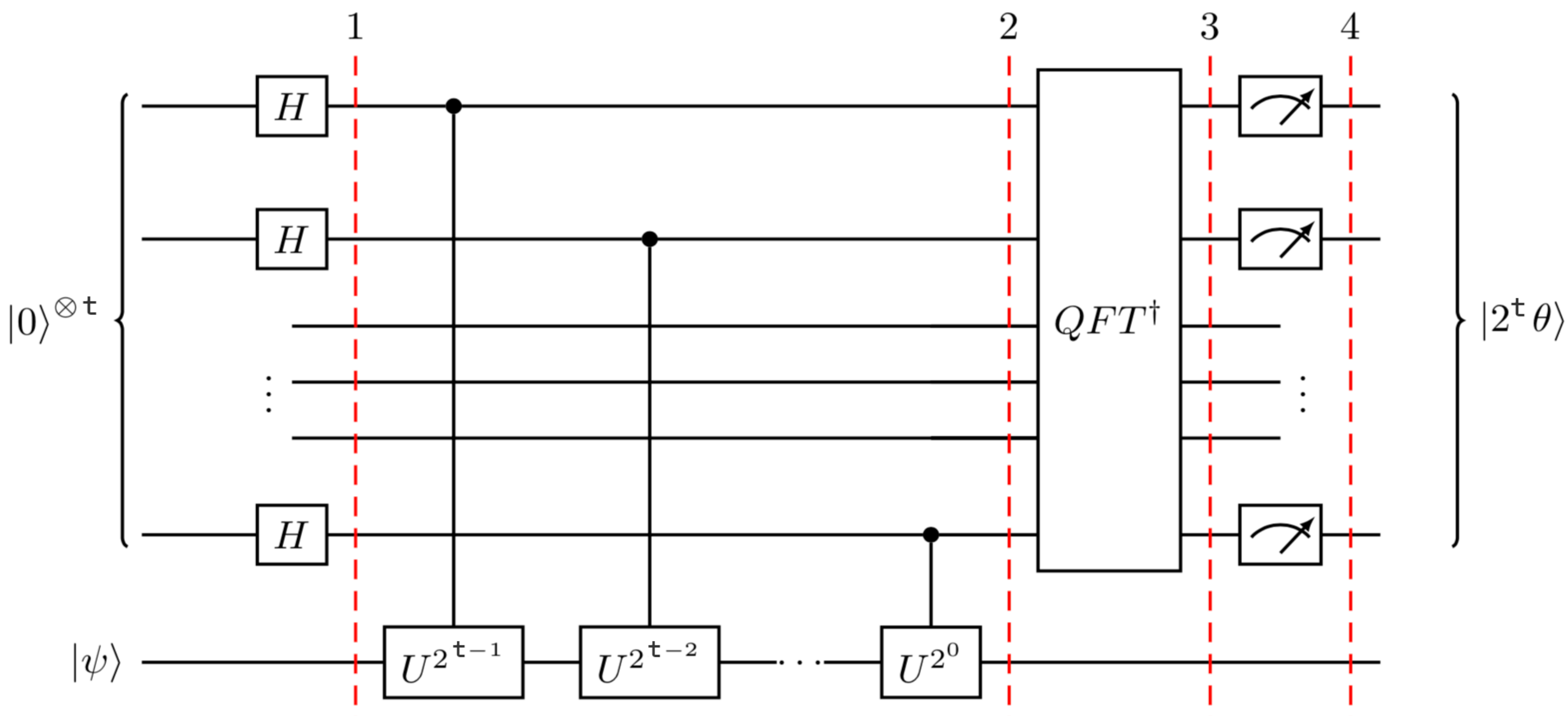
Quantum Phase Estimation

What is QPE:

Given a unitary operator that applies a phase θ to the state

$$U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$$

Quantum Phase Estimation (QPE) algorithm estimates the phase θ with quantum kickback effect



Quantum Kickback

Controlled NOT-gate on $|control, target\rangle$:

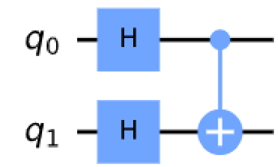
Entangled: $|{+}0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$

$$\boxed{\mathrm{CNOT}|{+}0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)}$$

Unchanged:

$$|{+}\,{+}\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

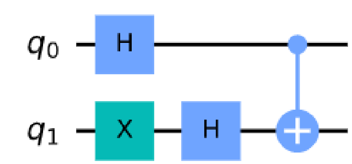
$$\boxed{\mathrm{CNOT}|{+}\,{+}\rangle = \frac{1}{2}(|00\rangle + |11\rangle + |10\rangle + |01\rangle) = |{+}\,{+}\rangle}$$



Kickback:

$$|{+}\,{-}\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

$$\boxed{\mathrm{CNOT}|{+}\,{-}\rangle = \frac{1}{2}(|00\rangle - |01\rangle + |11\rangle - |10\rangle) = |{-}\,{-}\rangle}$$



It affects the state of the control qubit while leaving the state of the target qubit unchanged

Phase Kickback

Controlled T/R/P-gates on $|control, target\rangle$:

T-gate:

$$T|1\rangle = e^{i\pi/4}|1\rangle$$

Controlled T-gate:

$$CT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{\pi i}{4}} \end{pmatrix}$$

$$|1\,+\rangle = |1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)$$

$$\boxed{CT|1\,+\rangle = \frac{1}{\sqrt{2}}(|10\rangle + e^{i\pi/4}|11\rangle) = |1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)}$$

Controlled Rotation-gate (in QFT):

$$CROT_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{2\pi i}{2^k}} \end{pmatrix}$$

Controlled U-gate:

$$CP(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix}$$

Symmetric gate: Applies the phase
iff both control & target bits are $|1\rangle$ (or on state $|11\rangle$)

Controlled Phase Gates

State-vector

```
qc = QuantumCircuit(2)

qc.save_statevector()

simulator = Aer.get_backend('qasm_simulator')
result =simulator.run(qc).result()
print(result.get_statevector(qc))
```

Qiskit simulator can return state vector

Controlled T-gate:

$$CT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{\pi i}{4}} \end{pmatrix}$$

Controlled Phase Gates

State-vector

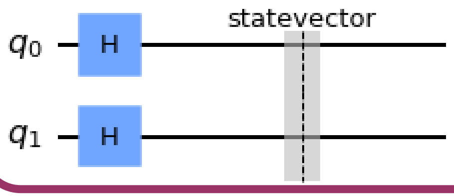
```python
qc = QuantumCircuit(2)

qc.save_statevector()

simulator = Aer.get_backend('qasm_simulator')
result =simulator.run(qc).result()
print(result.get_statevector(qc))
```

Controlled T-gate:

$$CT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{\pi i}{4}} \end{pmatrix}$$

```python
qc.h(0)
qc.h(1)
```

```
Statevector([0.5+0.j, 0.5+0.j, 0.5+0.j, 0.5+0.j],
            dims=(2, 2))
```



$$|state\ vector\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

Controlled Phase Gates

State-vector after Controlled T-gates

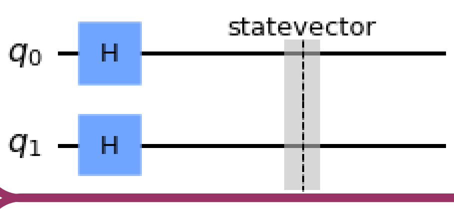
```
qc = QuantumCircuit(2)

qc.save_statevector()

simulator = Aer.get_backend('qasm_simulator')
result =simulator.run(qc).result()
print(result.get_statevector(qc))
```

Controlled T-gate:

$$CT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{\pi i}{4}} \end{pmatrix}$$

```
qc.h(0)
qc.h(1)
```

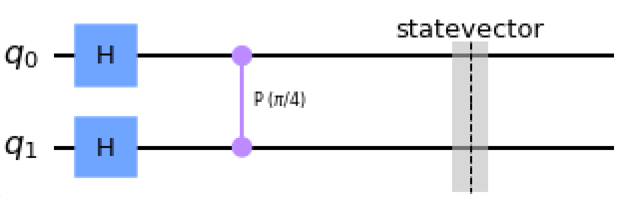
```
Statevector([0.5+0.j, 0.5+0.j, 0.5+0.j, 0.5+0.j],
            dims=(2, 2))
```



$$|state\ vector\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

```
qc.h(0)
qc.h(1)
qc.cp(np.pi/4,0,1)
```

```
Statevector([0.5       +0.j        , 0.5       +0.j        ,
             0.5       +0.j        , 0.35355339+0.35355339j],
            dims=(2, 2))
```



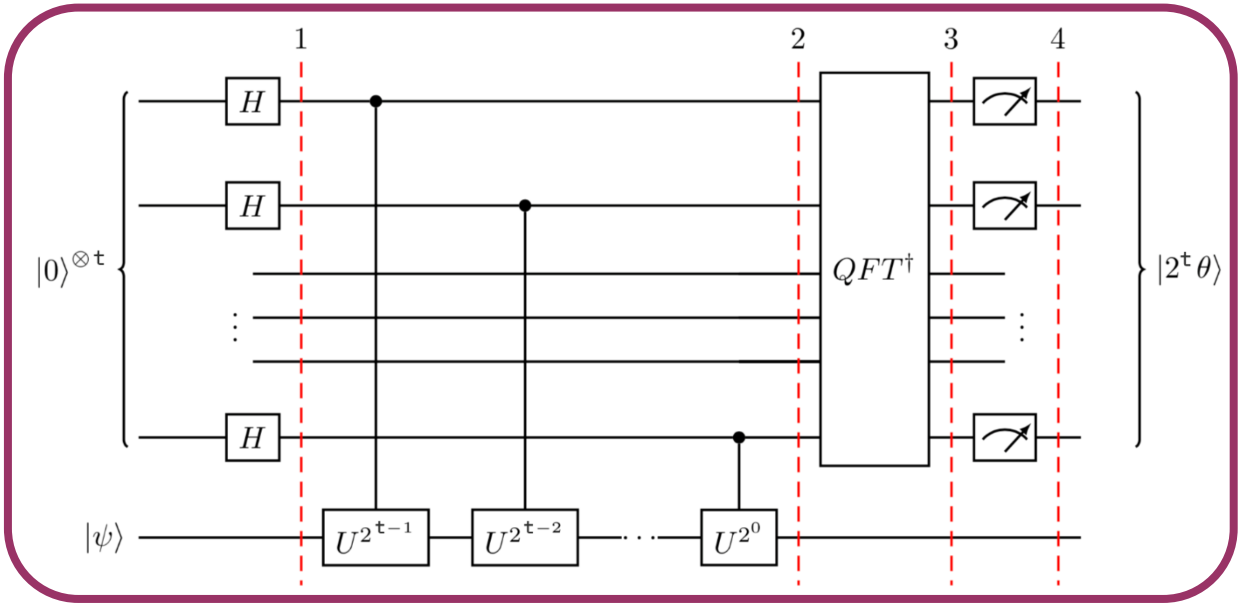
$$|state\ vector\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + e^{\pi i/4}|11\rangle)$$

entangled

QPE Circuit Breakdown

How to construct QPE circuit:

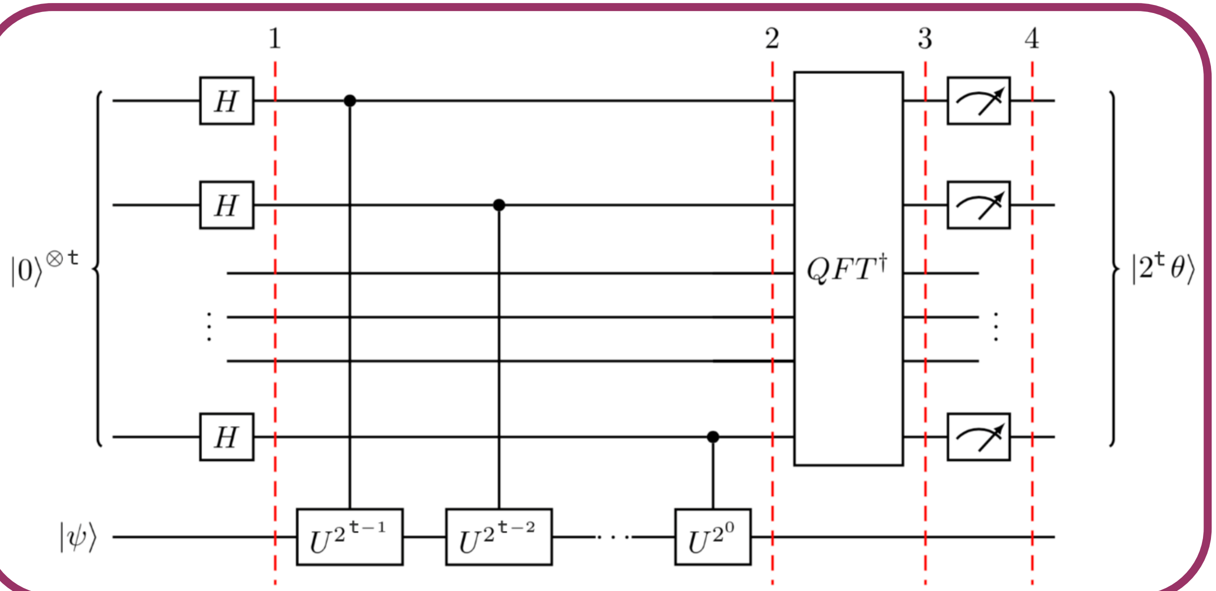
Let's look at the circuit in the previous slide



QPE Circuit Breakdown

Initial

$|\phi_0\rangle = |0\rangle^{\otimes n}|\psi\rangle$



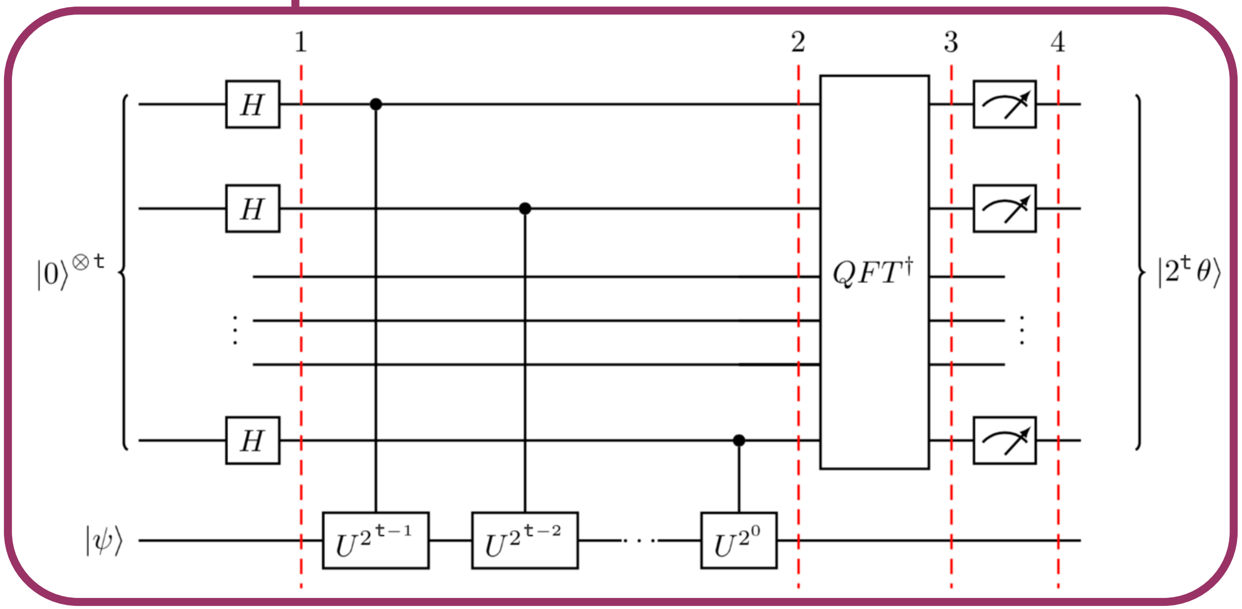
QPE Circuit Breakdown

$$|\phi_1\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle$$

H-gates

Initial

$$|\phi_0\rangle = |0\rangle^{\otimes n}|\psi\rangle$$



QPE Circuit Breakdown

$$|\phi_1\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle$$

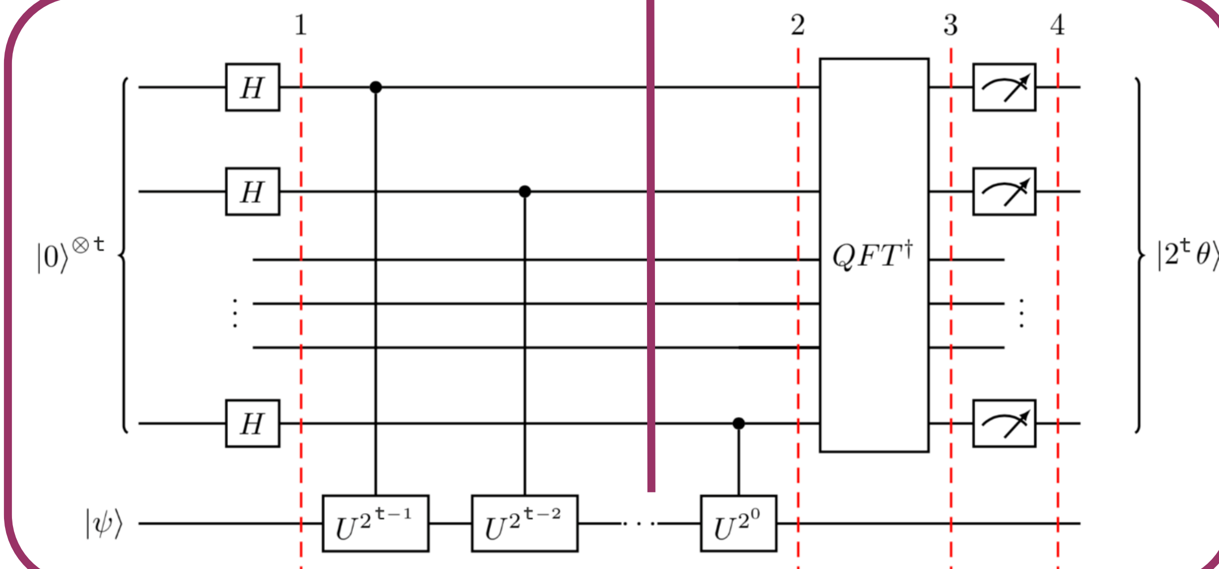
$$CP(2\pi\theta)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\theta}|1\rangle)|\psi\rangle$$

H-gates

Quantum phase kickback
(Remember to initialize $|\psi\rangle=|1\rangle$)

Phase-gates

Initial
$|\phi_0\rangle = |0\rangle^{\otimes n}|\psi\rangle$



QPE Circuit Breakdown

$$|\phi_1\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle$$

$$CP(2\pi\theta)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\theta}|1\rangle)|\psi\rangle$$

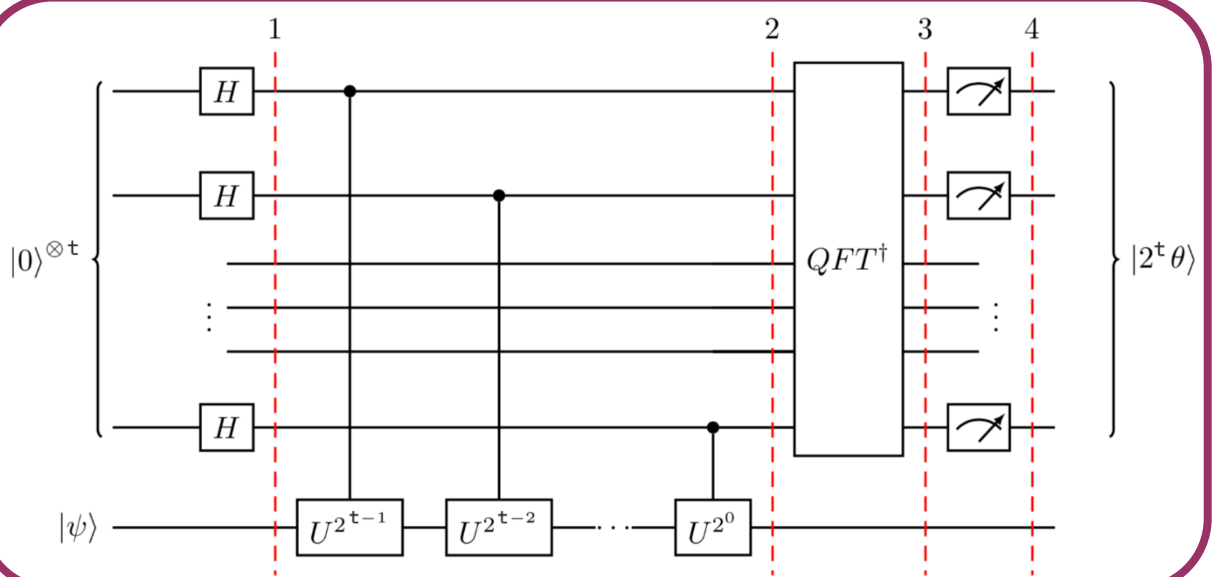
$$|\phi_2\rangle = \frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i\theta 2^{n-1}}|1\rangle\right)\otimes\ldots\otimes\left(|0\rangle + e^{2\pi i\theta 2^{1}}|1\rangle\right)\otimes\left(|0\rangle + e^{2\pi i\theta 2^{0}}|1\rangle\right)|\psi\rangle$$

H-gates

multiple Phase-gates

Initial

$|\phi_0\rangle = |0\rangle^{\otimes n}|\psi\rangle$



QPE Circuit Breakdown

For 1 qubit

$$CP(2\pi\theta)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\theta}|1\rangle)|\psi\rangle$$

For n qubits

$$|\phi_2\rangle = \frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i\theta 2^{n-1}}|1\rangle\right)\otimes\ldots\otimes\left(|0\rangle + e^{2\pi i\theta 2^1}|1\rangle\right)\otimes\left(|0\rangle + e^{2\pi i\theta 2^0}|1\rangle\right)|\psi\rangle$$

Compact form for 1 qubit

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\theta}|1\rangle)|\psi\rangle = \frac{1}{\sqrt{2}}\sum_{x=0}^{1} e^{2\pi i\theta x}|x\rangle$$

Compact form for n qubits

$$\frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i\theta 2^{n-1}}|1\rangle\right)\otimes\ldots\otimes\left(|0\rangle + e^{2\pi i\theta 2^1}|1\rangle\right)\otimes\left(|0\rangle + e^{2\pi i\theta 2^0}|1\rangle\right)|\psi\rangle$$

$$= \frac{1}{\sqrt{2^n}}\left(\sum_{x_0=0}^{1} e^{2\pi i\theta 2^{n-1}x_0}|x_0\rangle\right)\otimes\ldots\otimes\left(\sum_{x_{n-2}=0}^{1} e^{2\pi i\theta 2^1 x_{n-2}}|x_{n-2}\rangle\right)\otimes\left(\sum_{x_{n-1}=0}^{1} e^{2\pi i\theta 2^0 x_{n-1}}|x_{n-1}\rangle\right)|\psi\rangle$$

$$= \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1} e^{2\pi i\theta x}|x\rangle|\psi\rangle$$

Convert to bitstring

$$x = 2^{n-1}x_0 + 2^{n-2}x_1 + \ldots + 2^1 x_{n-2} + 2^0 x_{n-1}, \quad \text{with } x_i = 0 \text{ or } 1$$

For example:

$$(13)_{10} = 2^3 \cdot 1 + 2^2 \cdot 1 + 2^2 \cdot 0 + 2^0 \cdot 1 = (1101)_2$$

QPE Circuit Breakdown

$$|\phi_1\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle$$

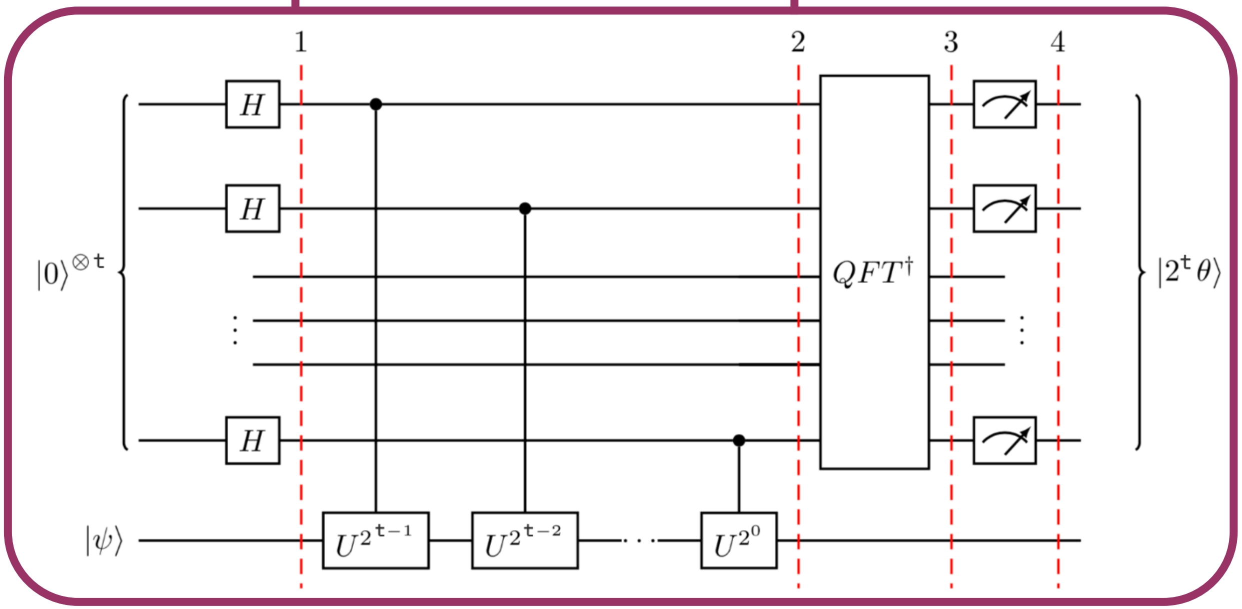
$$CP(2\pi\theta)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\theta}|1\rangle)|\psi\rangle$$

H-gates

$$\begin{aligned}|\phi_2\rangle &= \frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i\theta 2^{n-1}}|1\rangle\right)\otimes\ldots\otimes\left(|0\rangle + e^{2\pi i\theta 2^{1}}|1\rangle\right)\otimes\left(|0\rangle + e^{2\pi i\theta 2^{0}}|1\rangle\right)|\psi\rangle \\ &= \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1} e^{2\pi i\theta x}|x\rangle|\psi\rangle\end{aligned}$$

Initial

$$|\phi_0\rangle = |0\rangle^{\otimes n}|\psi\rangle$$



QPE Circuit Breakdown

$$|\phi_1\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle$$

$$CP(2\pi\theta)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\theta}|1\rangle)|\psi\rangle$$

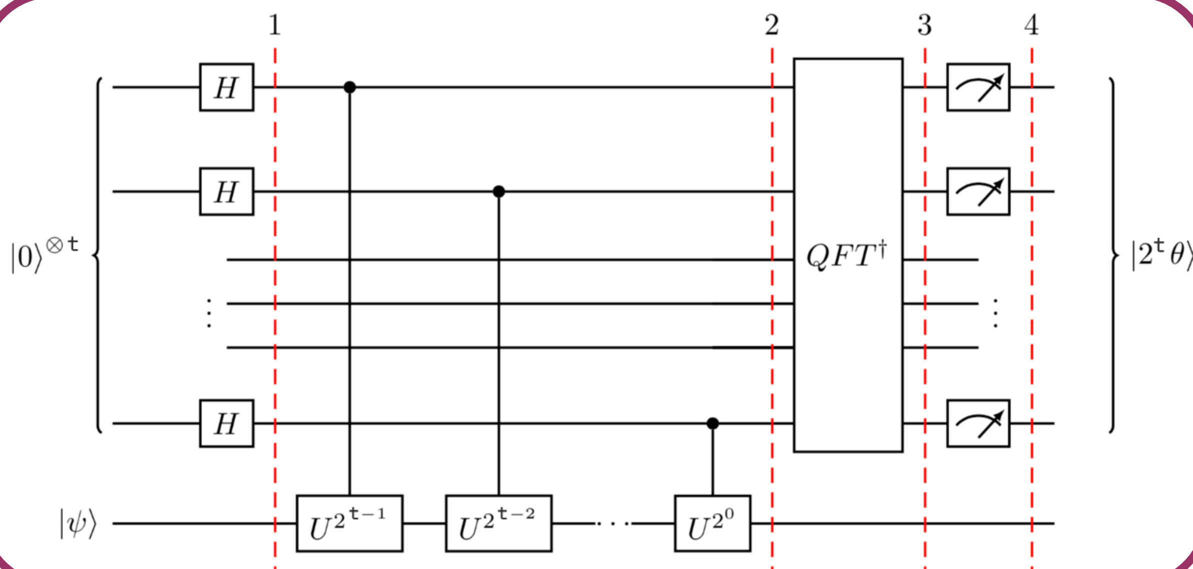
H-gates

$$|\phi_2\rangle = \frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i\theta 2^{n-1}}|1\rangle\right)\otimes\ldots\otimes\left(|0\rangle + e^{2\pi i\theta 2^{1}}|1\rangle\right)\otimes\left(|0\rangle + e^{2\pi i\theta 2^{0}}|1\rangle\right)|\psi\rangle$$

$$= \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1} e^{2\pi i\theta x}|x\rangle|\psi\rangle$$

Initial

$$|\phi_0\rangle = |0\rangle^{\otimes n}|\psi\rangle$$



Recall QFT and invQFT

$$U_{QFT}|x\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{2\pi ixy/2^n}|y\rangle$$

$$U_{QFT^{-1}}|x\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{-2\pi ixy/2^n}|y\rangle$$

QPE Circuit Breakdown

$$|\phi_1\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle$$

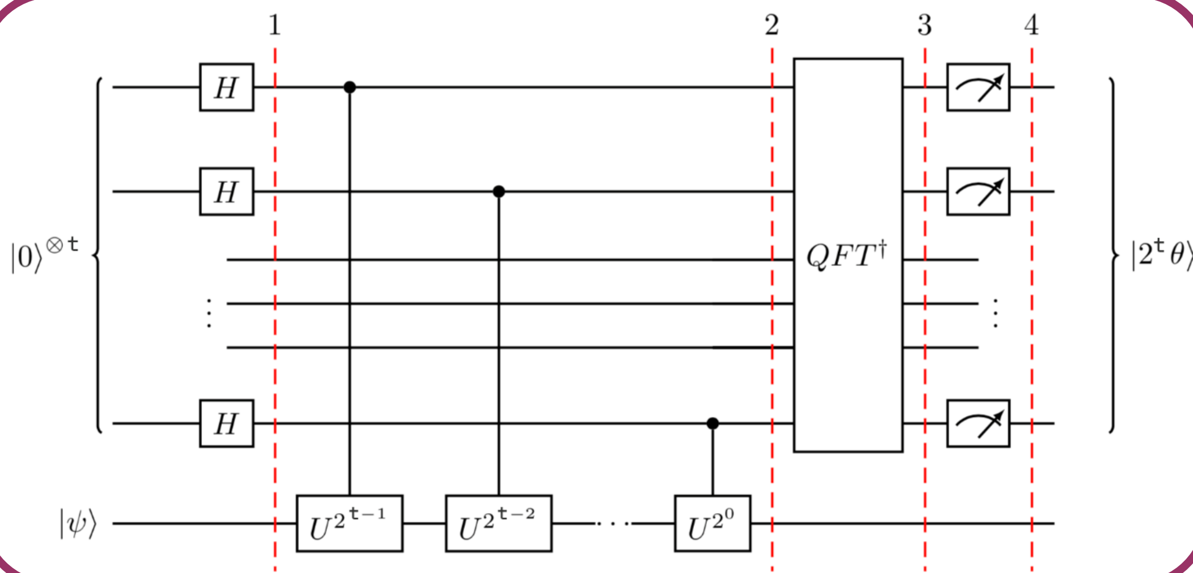
$$CP(2\pi\theta)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\theta}|1\rangle)|\psi\rangle$$

$$|\phi_2\rangle = \frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i\theta 2^{n-1}}|1\rangle\right)\otimes\ldots\otimes\left(|0\rangle + e^{2\pi i\theta 2^{1}}|1\rangle\right)\otimes\left(|0\rangle + e^{2\pi i\theta 2^{0}}|1\rangle\right)|\psi\rangle$$
$$= \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1} e^{2\pi i\theta x}|x\rangle|\psi\rangle$$

H-gates

Initial

$$|\phi_0\rangle = |0\rangle^{\otimes n}|\psi\rangle$$



Recall QFT and invQFT

$$U_{QFT}|x\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{2\pi ixy/2^n}|y\rangle$$

$$U_{QFT^{-1}}|x\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{-2\pi ixy/2^n}|y\rangle$$

$$|\phi_3\rangle = U_{QFT^{-1}}\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1} e^{2\pi i\theta x}|x\rangle|\psi\rangle = \frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1} e^{2\pi i(\theta-\frac{y}{2^n})x}|y\rangle|\psi\rangle$$

QPE Circuit Breakdown

$$|\phi_1\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle$$

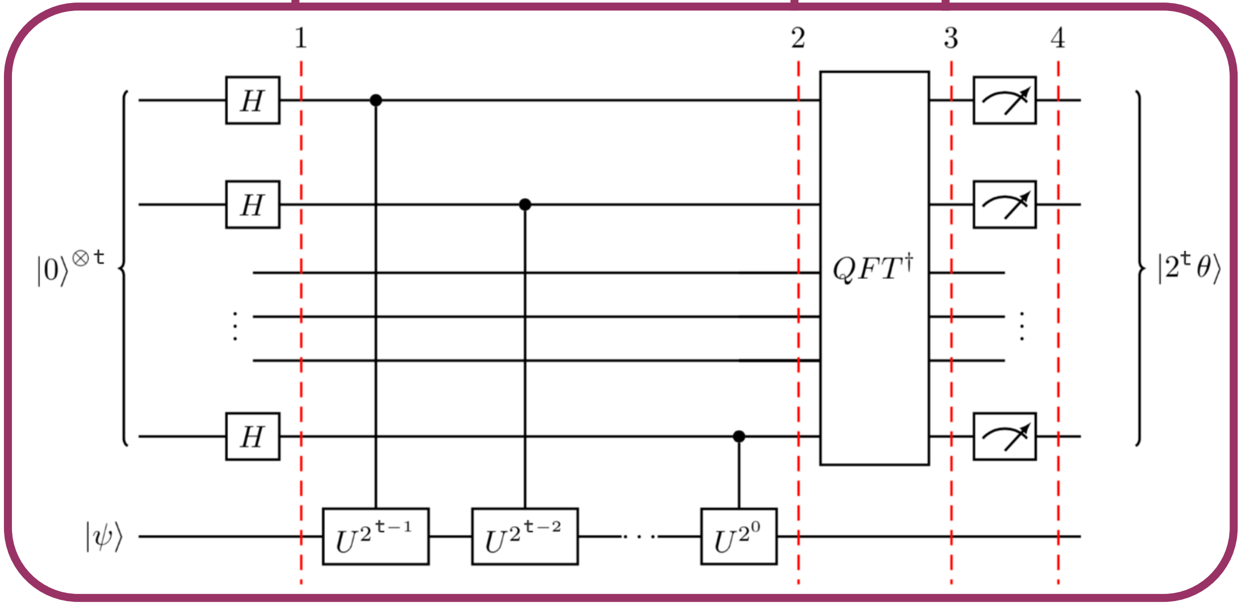
$$CP(2\pi\theta)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\theta}|1\rangle)|\psi\rangle$$

H-gates

$$|\phi_2\rangle = \frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i\theta 2^{n-1}}|1\rangle\right)\otimes\ldots\otimes\left(|0\rangle + e^{2\pi i\theta 2^{1}}|1\rangle\right)\otimes\left(|0\rangle + e^{2\pi i\theta 2^{0}}|1\rangle\right)|\psi\rangle$$
$$= \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1} e^{2\pi i\theta x}|x\rangle|\psi\rangle$$

Initial

$|\phi_0\rangle = |0\rangle^{\otimes n}|\psi\rangle$



Recall QFT and invQFT

$$U_{QFT}|x\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{2\pi i xy/2^n}|y\rangle$$
$$U_{QFT^{-1}}|x\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{-2\pi i xy/2^n}|y\rangle$$

Measurement in high probability when $2^n\theta$ is an integer

It seems to peak near $x=2^n\theta$:

$$|\phi_3\rangle = U_{QFT^{-1}}\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1} e^{2\pi i\theta x}|x\rangle|\psi\rangle = \frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1} e^{2\pi i(\theta-\frac{y}{2^n})x}|y\rangle|\psi\rangle$$

Quantum Interference in the QPE

What should we expect for the measurement?

Let's exchange the sequence of x and y

$$
\begin{aligned}
|\phi_3\rangle &= \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} e^{2\pi i x(\theta-\frac{y}{2^n})}|y\rangle|\psi\rangle \\
&= \frac{1}{N}\sum_{y=0}^{N-1}\left(\sum_{x=0}^{N-1} e^{2\pi i x(\theta-\frac{y}{2^n})}\right)|y\rangle|\psi\rangle \\
&= \frac{1}{N}\sum_{y=0}^{N-1}\boxed{\frac{1-e^{2\pi i(\theta-\frac{y}{2^n})2^n}}{1-e^{2\pi i(\theta-\frac{y}{2^n})}}}|y\rangle|\psi\rangle
\end{aligned}
$$

Strong quantum interference from the phase

The probability of getting result y from the measuring

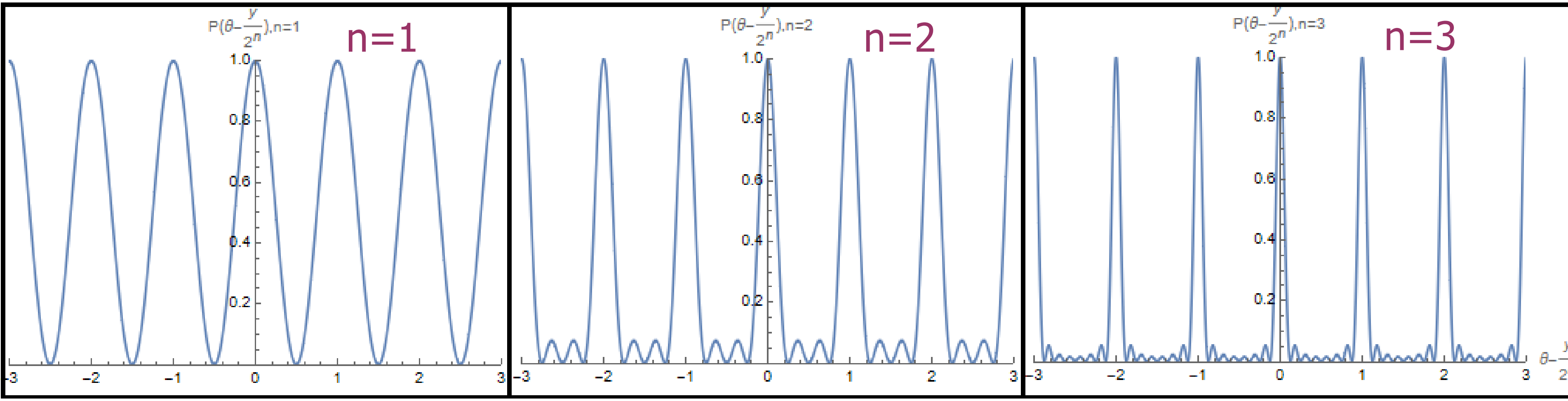
$$
P(y) = \frac{1}{N^2}\left|\frac{1-e^{2\pi i(\theta-\frac{y}{2^n})2^n}}{1-e^{2\pi i(\theta-\frac{y}{2^n})}}\right|^2 = \frac{1}{N^2}\frac{1-\cos[2\pi(\theta-\frac{y}{2^n})2^n]}{1-\cos[2\pi(\theta-\frac{y}{2^n})]}, \quad \text{with} \sum_{y=0}^{N-1} P(y) = 1
$$

Quantum Interference in the QPE

What should we expect for the measurement?

Probability with different number of qubits

$P(\theta - \frac{y}{2^n})$



$\theta - \frac{y}{2^n}$

The probability of getting result y from the measuring

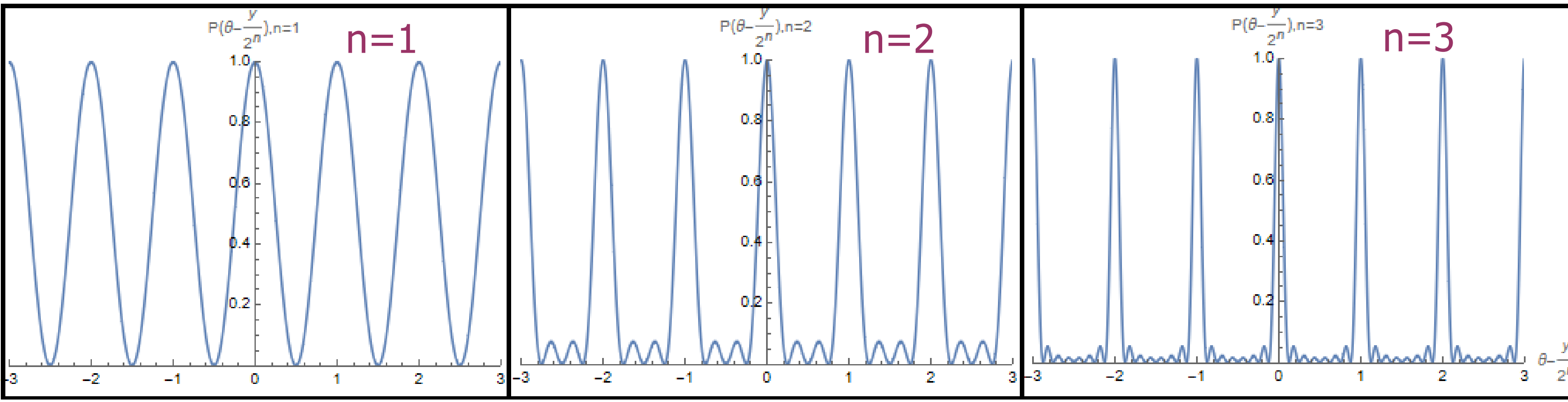
$$P(y) = \frac{1}{N^2}\left|\frac{1 - e^{2\pi i(\theta - \frac{y}{2^n})2^n}}{1 - e^{2\pi i(\theta - \frac{y}{2^n})}}\right|^2 = \frac{1}{N^2}\frac{1 - \cos[2\pi(\theta - \frac{y}{2^n})2^n]}{1 - \cos[2\pi(\theta - \frac{y}{2^n})]}, \quad \text{with} \sum_{y=0}^{N-1} P(y) = 1$$

Quantum Interference in the QPE

What should we expect for the measurement?

Probability with different number of qubits

$P(\theta - \frac{y}{2^n})$



$\theta - \frac{y}{2^n}$

The probability of getting result y from the measuring

$$P(y) = \frac{1}{N^2}\left|\frac{1-e^{2\pi i(\theta-\frac{y}{2^n})2^n}}{1-e^{2\pi i(\theta-\frac{y}{2^n})}}\right|^2 = \frac{1}{N^2}\frac{1-\cos[2\pi(\theta-\frac{y}{2^n})2^n]}{1-\cos[2\pi(\theta-\frac{y}{2^n})]}, \quad \text{with} \sum_{y=0}^{N-1} P(y) = 1$$

The probability peaks when $\theta - \frac{y}{2^n}$ is an integer!

Of course $\theta \pm 1,2,\ldots$ gives the same probability due to the 2π factor

If we restrict the phase $0 < \theta < 1$, the peak value of $\frac{y}{2^n}$ is what we need.

Qiskit Implementation

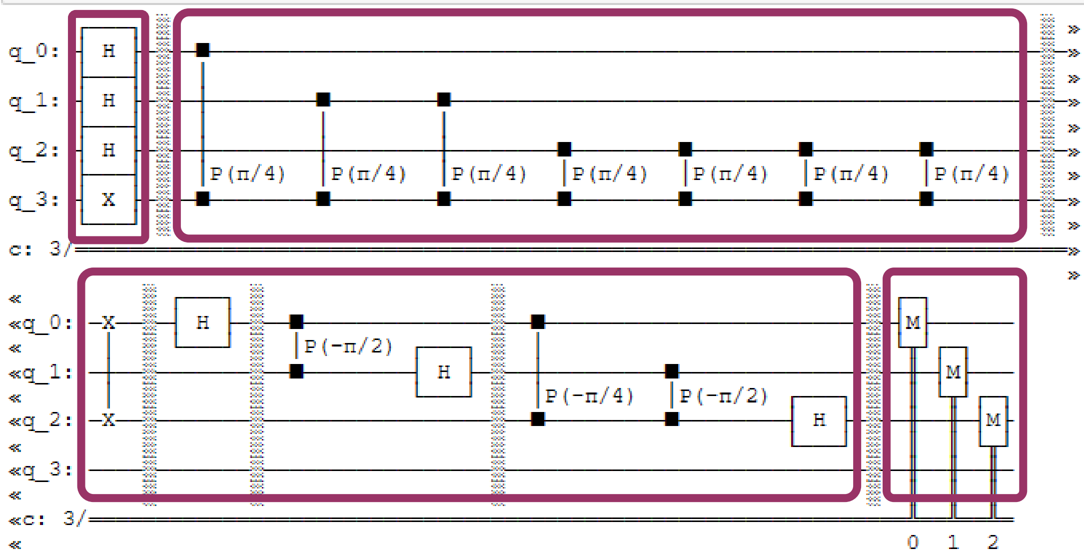
Implementing the QPE

```python
def cp_theta(qc,theta,x,psi):
    qc.cp(2*np.pi*theta,x,psi)
    return qc

def qpe(qc,n,theta):
    for j in range(0,n):
        qc.h(j)
    qc.barrier()
    for j in range(0,n):
        for k in range(0,2**j):
            qc=cp_theta(qc,theta,j,n)
    qc.barrier()
    qc=qft_inv(qc,n)
    for j in range(0,n):
        qc.measure(j,j)
    return qc
```

Qiskit Implementation

Implementing the QPE

```python
def cp_theta(qc,theta,x,psi):
    qc.cp(2*np.pi*theta,x,psi)
    return qc

def qpe(qc,n,theta):
    for j in range(0,n):
        qc.h(j)
    qc.barrier()
    for j in range(0,n):
        for k in range(0,2**j):
            qc=cp_theta(qc,theta,j,n)
    qc.barrier()
    qc=qft_inv(qc,n)
    for j in range(0,n):
        qc.measure(j,j)
    return qc
```



For example:
CT gate or phase $\theta = 1/8$
With 3 qubits

Qiskit Convention $|q_3 \ldots q_0>$

Qiskit Implementation

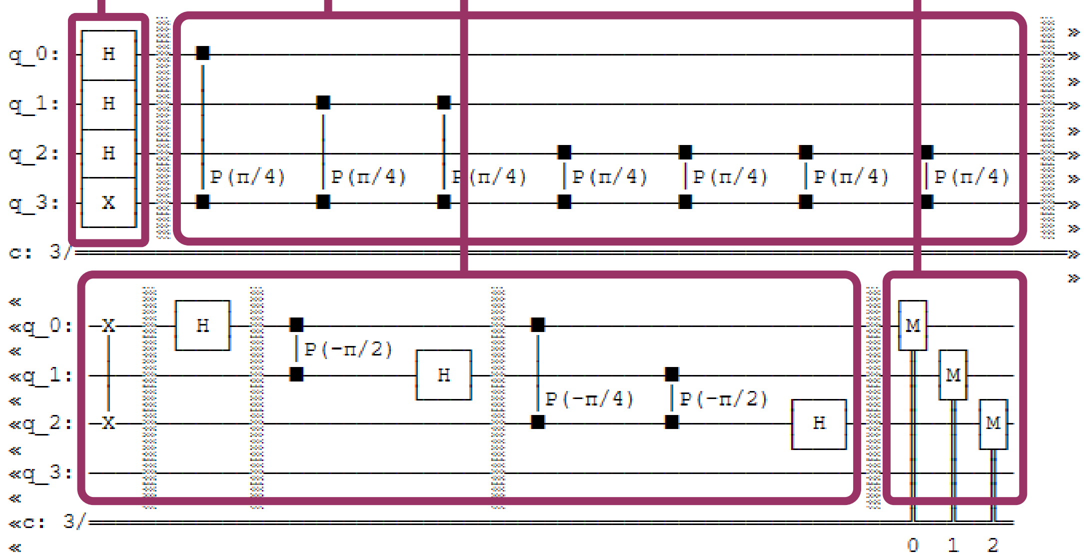
Implementing the QPE

Initialize the ancilla qubit |ψ⟩ with |1⟩ so that the symmetric controlled-phase gate applies

Repeatedly implementing phases (Qiskit uses reversed convention)

Inverse QFT

Only measure the controlling register



For example:
CT gate or phase $\theta = 1/8$
With 3 qubits

Qiskit Convention $|q_3 \ldots q_0\rangle$

Testing the QPE

Testing Controlled-T gate θ=1/8

```
qc=QuantumCircuit(n_qubit+1,n_qubit)
qc.x(n_qubit)
qc=qpe(qc,n_qubit,theta)

simulator = Aer.get_backend('qasm_simulator')
shots=1024
result =simulator.run(qc,shots=shots).result()
counts = result.get_counts(qc)
plot_histogram(counts, figsize=(8,5))
```

Measurement should peak at $\theta \approx \frac{y}{2^n}$

Testing the QPE

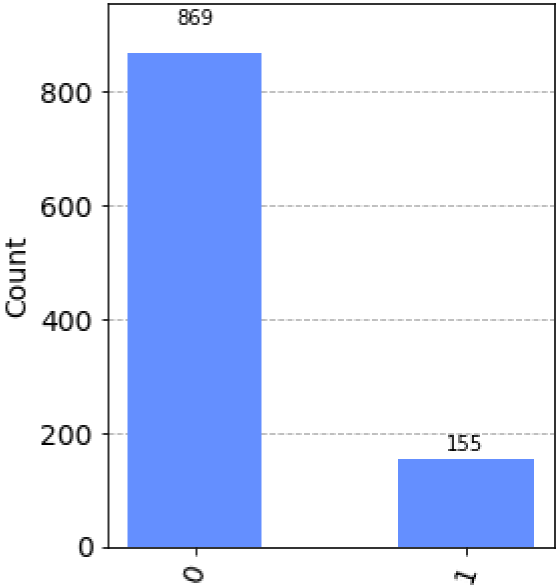
Testing Controlled-T gate θ=1/8

```
qc=QuantumCircuit(n_qubit+1,n_qubit)
qc.x(n_qubit)
qc=qpe(qc,n_qubit,theta)

simulator = Aer.get_backend('qasm_simulator')
shots=1024
result =simulator.run(qc,shots=shots).result()
counts = result.get_counts(qc)
plot_histogram(counts, figsize=(8,5))
```

```
n_qubit=1
theta=1/8
```

1-qubit

Measurement should peak at $\theta \approx \frac{y}{2^n}$



Measurement suggests $\frac{0}{2}$

Testing the QPE

Testing Controlled-T gate θ=1/8

```python
qc=QuantumCircuit(n_qubit+1,n_qubit)
qc.x(n_qubit)
qc=qpe(qc,n_qubit,theta)

simulator = Aer.get_backend('qasm_simulator')
shots=1024
result =simulator.run(qc,shots=shots).result()
counts = result.get_counts(qc)
plot_histogram(counts, figsize=(8,5))
```

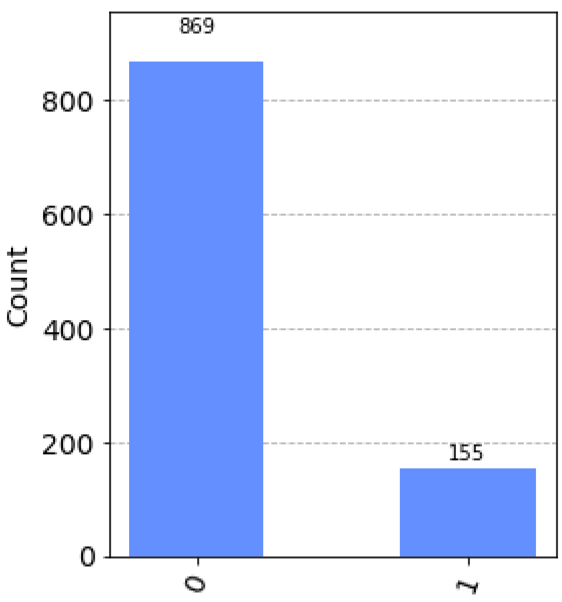
Measurement should peak at $\theta \approx \frac{y}{2^n}$

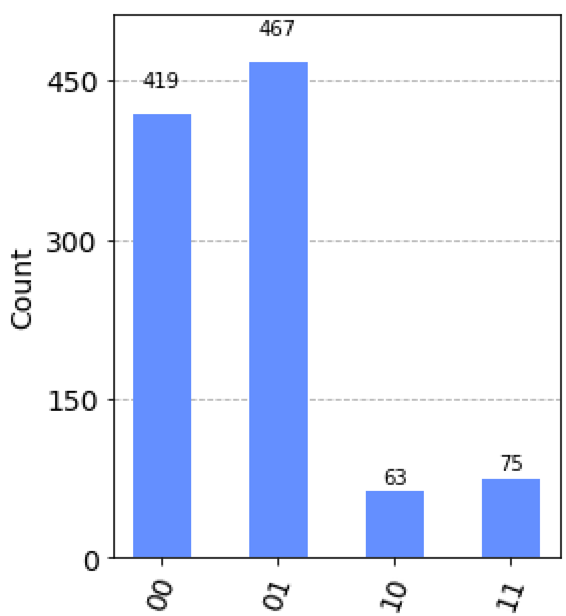
```python
n_qubit=1
theta=1/8
```

1-qubit

```python
n_qubit=2
theta=1/8
```

2-qubits





Measurement suggests $\frac{0}{2}$

Measurement suggests $\frac{0}{4}, \frac{1}{4}$

Testing the QPE

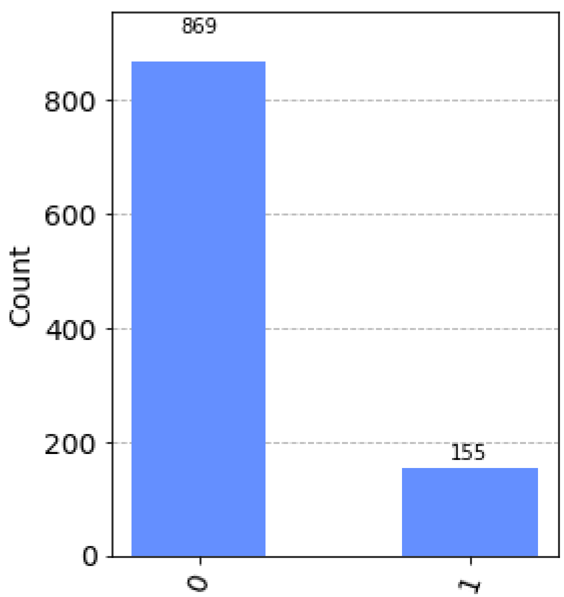
Testing Controlled-T gate θ=1/8

```
qc=QuantumCircuit(n_qubit+1,n_qubit)
qc.x(n_qubit)
qc=qpe(qc,n_qubit,theta)

simulator = Aer.get_backend('qasm_simulator')
shots=1024
result =simulator.run(qc,shots=shots).result()
counts = result.get_counts(qc)
plot_histogram(counts, figsize=(8,5))
```

Measurement should peak at $\theta \approx \frac{y}{2^n}$

```
n_qubit=1
theta=1/8
```

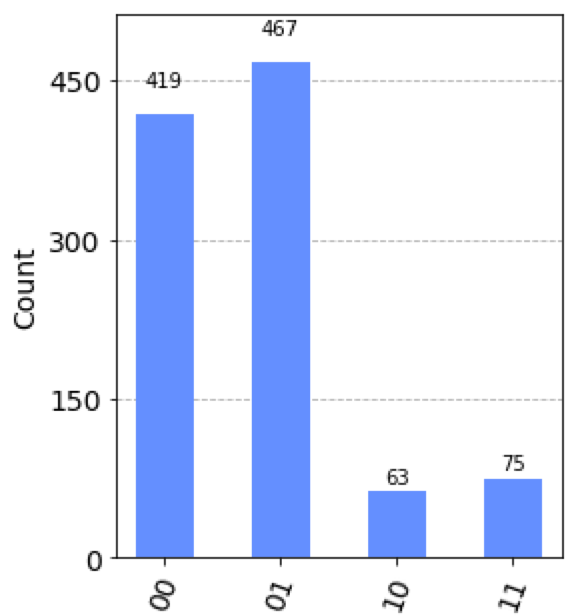
1-qubit



Measurement suggests $\frac{0}{2}$

```
n_qubit=2
theta=1/8
```

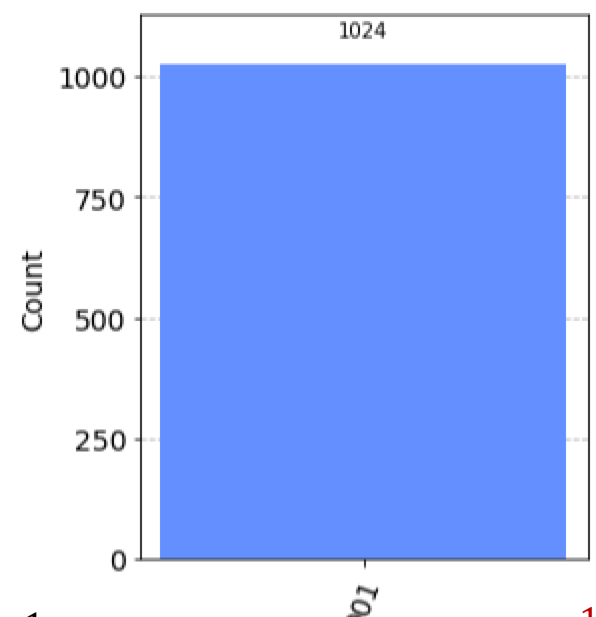
2-qubits



Measurement suggests $\frac{0}{4}, \frac{1}{4}$

```
n_qubit=3
theta=1/8
```

3-qubits



Measurement exactly $\frac{1}{8}$!

Testing the QPE

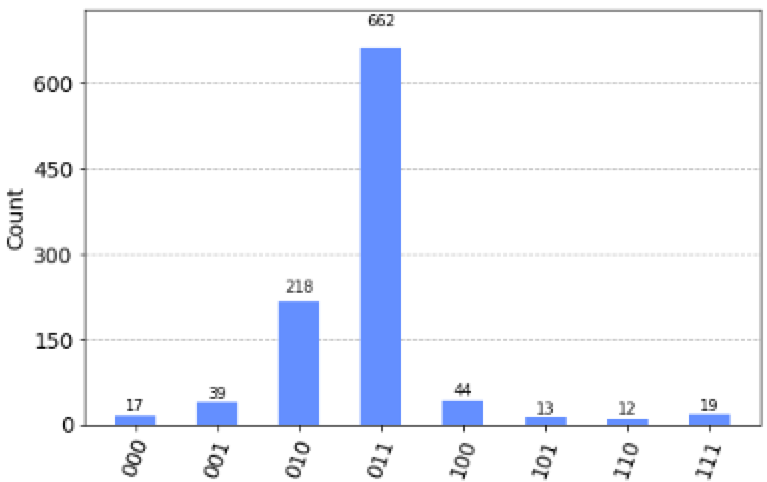
Testing Controlled-T gate θ=1/3

```
qc=QuantumCircuit(n_qubit+1,n_qubit)
qc.x(n_qubit)
qc=qpe(qc,n_qubit,theta)

simulator = Aer.get_backend('qasm_simulator')
shots=1024
result =simulator.run(qc,shots=shots).result()
counts = result.get_counts(qc)
plot_histogram(counts, figsize=(8,5))
```

```
n_qubit=3
theta=1/3
```

3-qubits

Measurement should peak at $\theta \approx \frac{y}{2^n}$



Measurement suggests $\frac{3}{8}$

Testing the QPE

Testing Controlled-T gate θ=1/3

```
qc=QuantumCircuit(n_qubit+1,n_qubit)
qc.x(n_qubit)
qc=qpe(qc,n_qubit,theta)

simulator = Aer.get_backend('qasm_simulator')
shots=1024
result =simulator.run(qc,shots=shots).result()
counts = result.get_counts(qc)
plot_histogram(counts, figsize=(8,5))
```

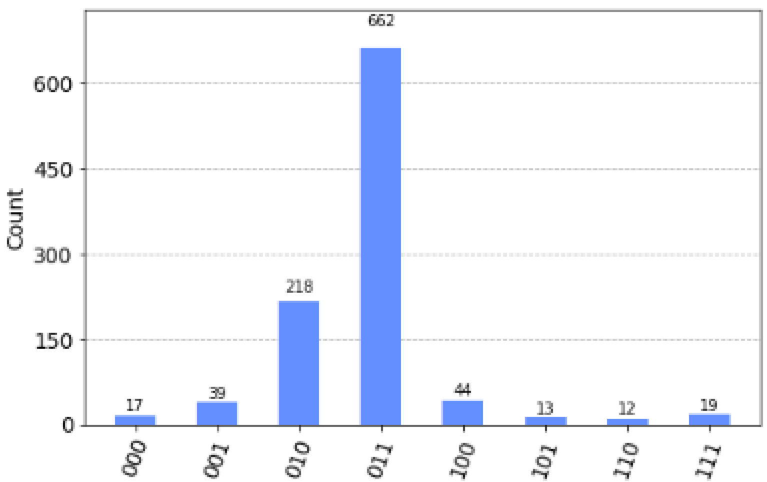
Measurement should peak at $\theta \approx \frac{y}{2^n}$

```
n_qubit=3
theta=1/3
```

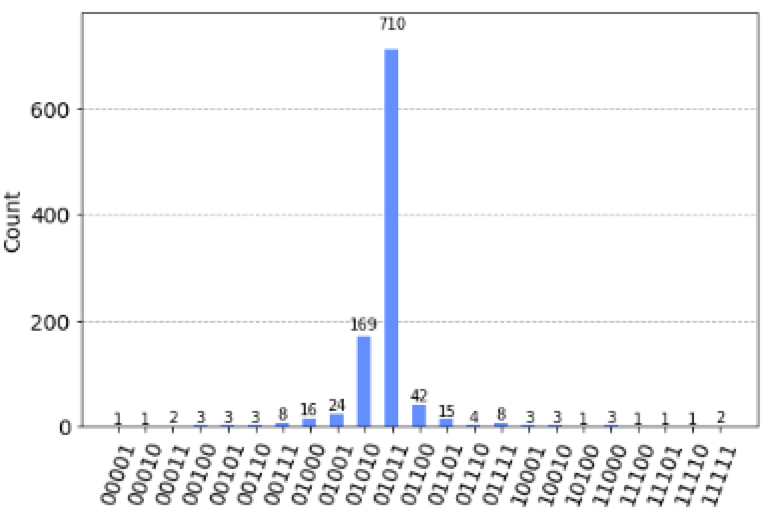
3-qubits

```
n_qubit=5
theta=1/3
```

5-qubits



Measurement suggests $\frac{3}{8}$



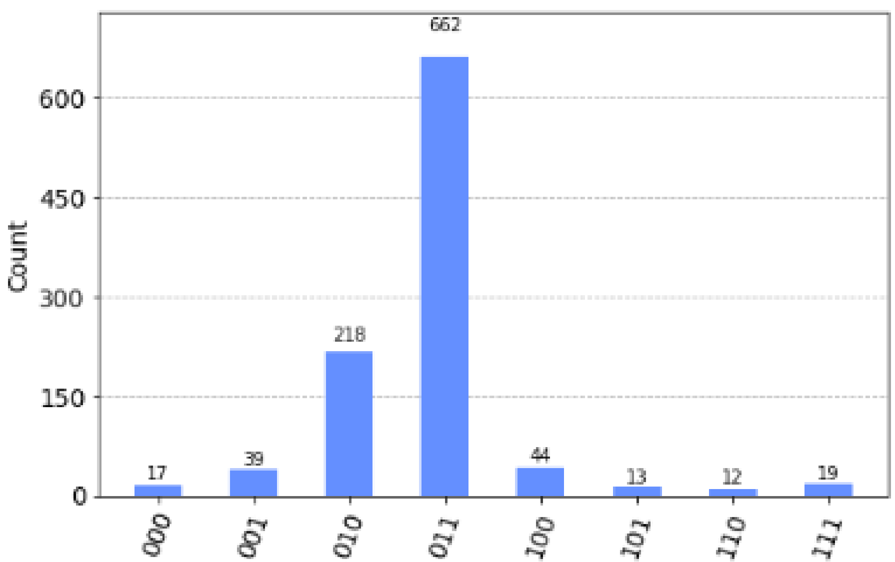
Measurement suggests $\frac{11}{32}$

Testing the QPE

More accurate? More qubits

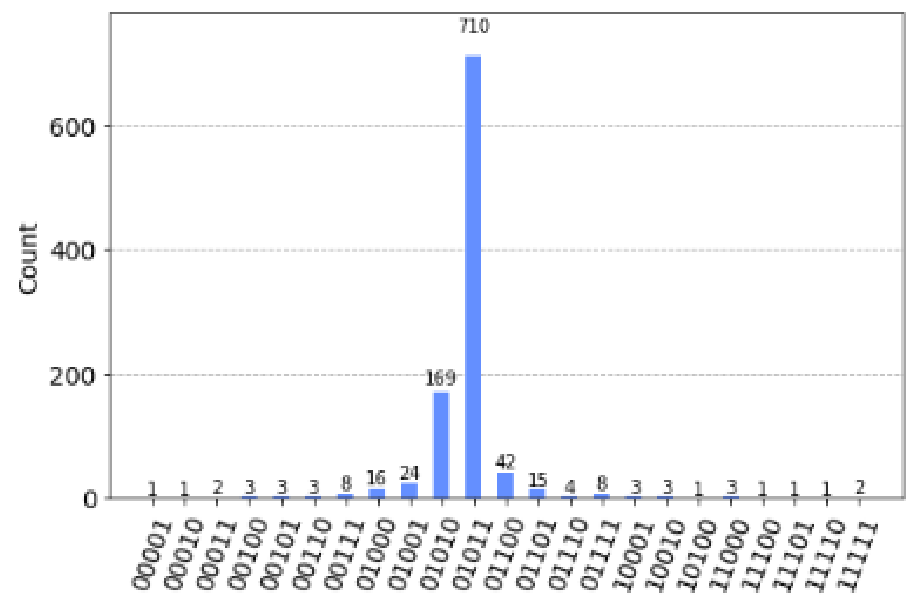
Measurement should peak at $\theta \approx \frac{y}{2^n}$

3-qubits



$(011)_2 = 3 \sim 3/2^3 = 0.375$

5-qubits



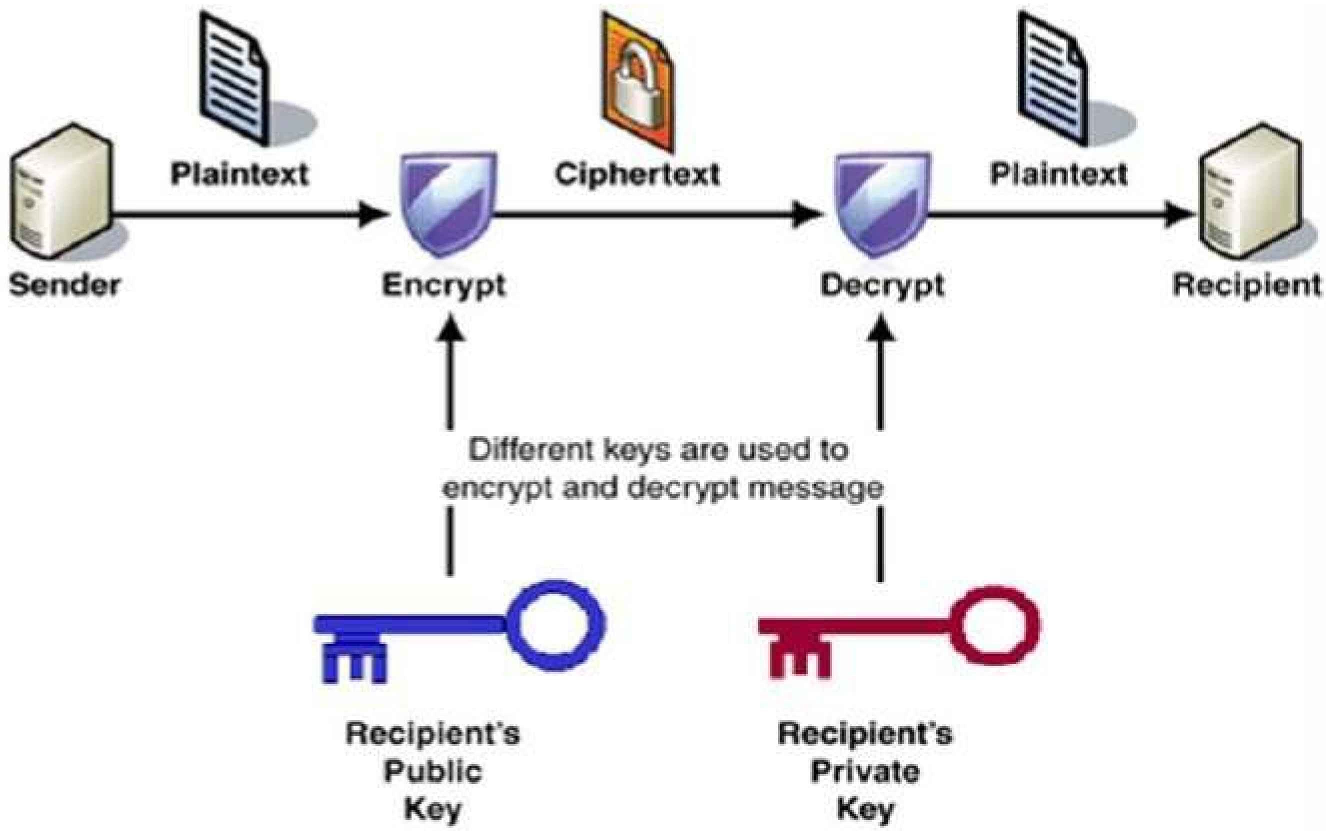
$(01011)_2 = 11 \sim 11/2^5 = 0.34375$

Much better precision

RSA Cryptography

RSA Cryptography

What is RSA cryptography ?



RSA Cryptography

How do the encryption and decryption work?

The Sender starts with the plaintext **P**, it can be digitized in some way
Just for example, translating letters/symbols in **P** with ASCII code

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Then the sender has an integer form for the message **P**
(stored as bitstring in computer)

Encryption and decryption

How do the encryption and decryption work?

The sender encrypt the plaintext **P** with the following to get a cipher text **C**

$$\mathbf{C} = \mathbf{P}^{\mathbf{E}}(\mathbf{mod\ N})$$

The encryption power **E** and the large number **N** are integer numbers

The pair **(E,N)** is called the **public key**

The recipient can distribute the **public key** to “public” so that any sender wants to send recipient a message **P** can encrypt it first them send the ciphertext **C** instead

Encryption and decryption

How do the encryption and decryption work?

The sender encrypt the plaintext **P** with the following to get a cipher text **C**

$$\mathbf{C = P^{E} (mod\ N)}$$

The encryption power **E** and the large number **N** are integer numbers

The pair **(E,N)** is called the **public key**

The recipient can distribute the **public key** to “public” so that any sender wants to send recipient a message **P** can encrypt it first them send the ciphertext **C** instead

Once the recipient receives the ciphertext **C**, the recipient can use the following to get the plaintext **P**

$$\mathbf{P = C^{D} (mod\ N)}$$

The decryption power **D** and the large number **N** are integer numbers

The pair **(D,N)** is called the **private key**

Only the recipient has the **private key**!

Encryption and decryption

How do the encryption and decryption work?

Let’s look at an example with public key **(7,143)** and private key **(43,143)**

Suppose we send a plaintext ‘IGFAE’, the ASCII codes in Decimal formal are ‘I’=73, ‘G’=71, ‘F’=65, ‘A’=70, ‘E’=69. The ciphertexts would be

$$\mathbf{C_1 = 73^7 (mod\ 143) = 83}$$
$$\mathbf{C_2 = 71^7 (mod\ 143) = 124}$$
$$\mathbf{C_3 = 65^7 (mod\ 143) = 65}$$
$$\mathbf{C_4 = 70^7 (mod\ 143) = 60}$$
$$\mathbf{C_5 = 69^7 (mod\ 143) = 108}$$

The corresponding ASCII codes in symbolic form are ‘S’,’|’,’A’,’<’,’l’. So the recipient receives the ‘S|AI’, looks totally different from ‘IGFAE’

Encryption and decryption

How do the encryption and decryption work?

Let’s look at an example with public key **(7,143)** and private key **(43,143)**

Suppose we send a plaintext ‘IGFAE’, the ASCII codes in Decimal formal are ‘I’=73, ‘G’=71, ‘F’=65, ‘A’=70, ‘E’=69. The ciphertexts would be

$$\mathbf{C_1 = 73^7 (mod\ 143) = 83}$$
$$\mathbf{C_2 = 71^7 (mod\ 143) = 124}$$
$$\mathbf{C_3 = 65^7 (mod\ 143) = 65}$$
$$\mathbf{C_4 = 70^7 (mod\ 143) = 60}$$
$$\mathbf{C_5 = 69^7 (mod\ 143) = 108}$$

The corresponding ASCII codes in symbolic form are ‘S’,’|’,’A’,’<’,’l’. So the recipient receives the ‘S|AI’, looks totally different from ‘IGFAE’.
Then the recipient decrypt it with the private key

$$\mathbf{P_1 = 83^{43} (mod\ 143) = 73}$$
$$\mathbf{P_2 = 124^{43} (mod\ 143) = 71}$$
$$\mathbf{P_3 = 65^{43} (mod\ 143) = 65}$$
$$\mathbf{P_4 = 60^{43} (mod\ 143) = 70}$$
$$\mathbf{P_5 = 108^{43} (mod\ 143) = 69}$$

Which is ‘IGFAE’ again

Symmetric Cryptography

It’s a symmetric cryptography

We may notice the symmetric form

$$C = P^E (mod\ N) \qquad\qquad P = C^D (mod\ N)$$

If the recipient wants to send a message, recipient’s private key can be used to encrypt the message, then the sender can use recipient’s public key to decrypt it

$$C = P^D (mod\ N) \qquad\qquad P = C^E (mod\ N)$$

The recipient’s private key is the sender’s public key

The recipient’s public key is the sender’s private key

Symmetric Cryptography

It's a symmetric cryptography

For example, let's use the private key **(43,143)** to encrypt the 'IGFAE'

$$\mathbf{C_1 = 73^{43} (mod\ 143) = 57}$$
$$\mathbf{C_2 = 71^{43} (mod\ 143) = 59}$$
$$\mathbf{C_3 = 65^{43} (mod\ 143) = 65}$$
$$\mathbf{C_4 = 70^{43} (mod\ 143) = 86}$$
$$\mathbf{C_5 = 69^{43} (mod\ 143) = 82}$$

Which means '9;AVR', totally different from 'IGFAE'

Symmetric Cryptography

It’s a symmetric cryptography

For example, let’s use the private key **(43,143)** to encrypt the ‘IGFAE’

$$\mathbf{C_1 = 73^{43}(mod\ 143) = 57}$$
$$\mathbf{C_2 = 71^{43}(mod\ 143) = 59}$$
$$\mathbf{C_3 = 65^{43}(mod\ 143) = 65}$$
$$\mathbf{C_4 = 70^{43}(mod\ 143) = 86}$$
$$\mathbf{C_5 = 69^{43}(mod\ 143) = 82}$$

Which means ‘9;AVR’, totally different from ‘IGFAE’

Then we use the public key **(7,143)** to decrypt it

$$\mathbf{P_1 = 57^{7}(mod\ 143) = 73}$$
$$\mathbf{P_2 = 59^{7}(mod\ 143) = 71}$$
$$\mathbf{P_3 = 65^{7}(mod\ 143) = 65}$$
$$\mathbf{P_4 = 86^{7}(mod\ 143) = 70}$$
$$\mathbf{P_5 = 82^{7}(mod\ 143) = 69}$$

It becomes ‘IGFAE’ again

Hacking

How do hacking work?

Instead of sending the plaintext ‘IGFAE’, we send the ciphertext ‘S|A<I’ instead in the last example.

If an eavesdropper (hacker) get the ciphertext **C**=‘S|A<I’, it needs to be decrypted to reveal the plaintext. But only the public key **(E,N)=(7,143)** is ‘public’, how does the hacker know about the private key **(D,N)=(43,143)** ?

Hacker

Hacking

How do hacking work?

Instead of sending the plaintext ‘IGFAE’, we send the ciphertext ‘S|A<I’ instead in the last example.

If an eavesdropper (hacker) get the ciphertext **C**=‘S|A<I’, it needs to be decrypted to reveal the plaintext. But only the public key **(E,N)=(7,143)** is ‘public’, how does the hacker know about the private key **(D,N)=(43,143)** ?

We have already seen that the pair of keys are symmetric, so they are not arbitrarily but carefully chosen numbers.

Design the keys

Steps to prepare the key pair

(1) Choose two prime numbers, for example **$p=11$** and **$q=13$**

(2) Calculate the large number **$N=pq=143$**

(3) Calculate the Euler totient with **least common multiple $L=\mathrm{lcm}(p-1,q-1)=60$**

(4) Prepare the public key with **greatest common factor** such that **$\gcd(E,L)=1$**

For example $\gcd(7,60)=1$ then choose **$E=7$**

(5) Prepare the private key with rule such that **$ED \pmod{L}=1$**

For example $7 \times 43 \pmod{60}=1$ then choose **$D=43$**

Design the keys

Steps to prepare the key pair

(1) Choose two prime numbers, for example **p=11** and **q=13**
(2) Calculate the large number **N=pq=143**
(3) Calculate the Euler totient with **least common multiple L=lcm(p-1,q-1)=60**
(4) Prepare the public key with **greatest common factor** such that **gcd(E,L)=1**
    For example gcd(7,60)=1 then choose **E=7**
(5) Prepare the private key with rule such that **ED(mod L)=1**
    For example 7x43 (mod 60)=1 then choose **D=43**

This means, if we know **p** and **q**, and the public key **(E,N)** as well, we can cauclate the private key **(D,N)** easily

Design the keys

Steps to prepare the key pair

(1) Choose two prime numbers, for example **p=11** and **q=13**
(2) Calculate the large number **N=pq=143**
(3) Calculate the Euler totient with **least common multiple L=lcm(p-1,q-1)=60**
(4) Prepare the public key with **greatest common factor** such that **gcd(E,L)=1**
For example gcd(7,60)=1 then choose **E=7**
(5) Prepare the private key with rule such that **ED(mod L)=1**
For example 7x43 (mod 60)=1 then choose **D=43**

This means, if we know **p** and **q**, and the public key **(E,N)** as well, we can cauclate the private key **(D,N)** easily

Now the question is, knowing the number **N**, how do we find the prime numbers **p** and **q**?

In reality, **N** can be very large, maybe 256 bits, 512 bits or even 1024 bits, so it is extremely difficult to find **p** and **q**.

Shor’s Factoring Algorithm

Factoring Problem

Factoring a large number

We’ve learned that in order to hack, we need to factorize a large number **N** into two prime numbers **p** and **q**.

One way to factorize is to convert this problem into a **period finding problem**

Define a periodic function with an integer a **coprime to N** (otherwise we already find the factor)

$$f(x) = a^x (mod\ N)$$

Then the smallest non-zero integer r that satisfying

$$f(r) = a^r (mod\ N) = 1$$

is called the **period**.

If it is an even number (if no even r, try different a), then we have

$$p = \gcd(a^{\frac{r}{2}} - 1, N) \qquad \text{and} \qquad q = \gcd(a^{\frac{r}{2}} + 1, N)$$

Since

Factoring Problem

Factoring a large number

For example $\mathbf{N} = 143$, we choose $a = 23$

$$
\begin{aligned}
f(1) &= 23^1 (mod\ 143) = 23 \\
f(2) &= 23^2 (mod\ 143) = 100 \\
f(3) &= 23^3 (mod\ 143) = 12 \\
f(4) &= 23^4 (mod\ 143) = 133 \\
f(5) &= 23^5 (mod\ 143) = 56 \\
f(6) &= 23^6 (mod\ 143) = 1 \\
&\dots
\end{aligned}
$$

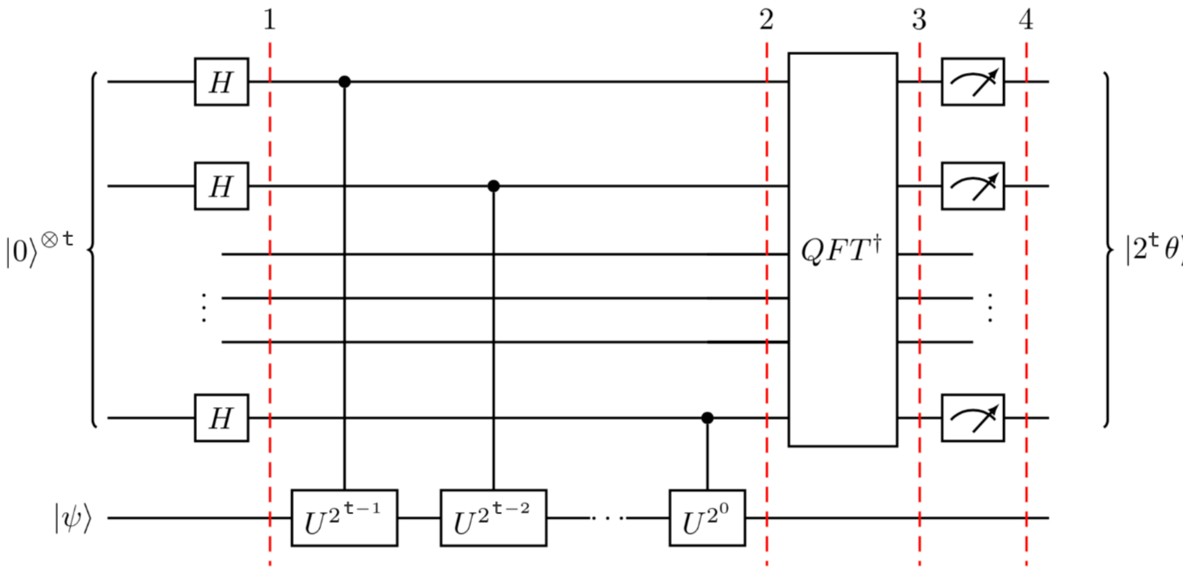
We find that $r = 6$ so that $a^{r/2} = 23^3 = 12167$
We can compute

$$
\mathrm{p} = \gcd(23^3 - 1, 143) = 11
$$
$$
\mathrm{q} = \gcd(23^3 + 1, 143) = 13
$$

Now the **factoring problem** is converted to a **period finding problem**

Period Finding Problem

Quantum Algorithm for Period Finding Problem



The idea is very similar to the QPE but different by the following:

(1) Instead of controlled phase gate, one considers a controlled module

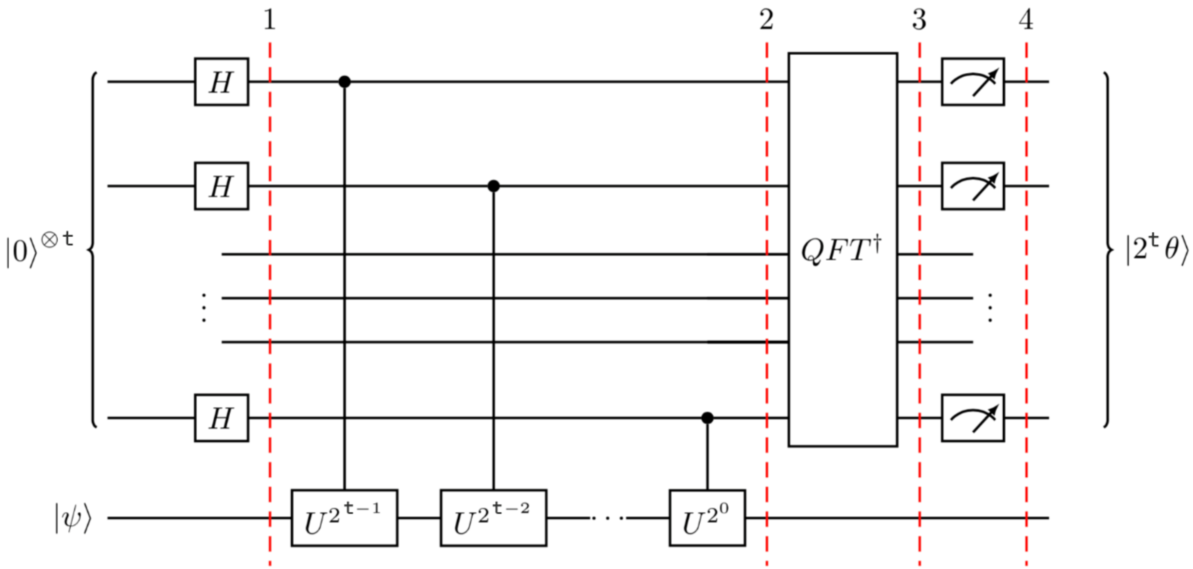
$U|\psi\rangle = |a\psi(\mathrm{mod}\ N)\rangle$ and $U^k|\psi\rangle = |a^k\psi(\mathrm{mod}\ \mathbf{N})\rangle$

(2) Instead of 1 ancilla qubit in the QPE, Shor's algorithm requires n ancilla qubits such that $2^n > \mathbf{N}$

The measurements will result in multiple peaks at $\frac{k}{r}$ with $k \leq r$

Period Finding Problem

Quantum Algorithm for Period Finding Problem



It will result in a final state as

$$|\phi\rangle = \left[\frac{1}{2^L}\left(\sum_{x=0}^{2^L-1}\sum_{y=0}^{2^L-1} e^{-2\pi i x \frac{y}{2^L}} |y\rangle \otimes |a^{x(mod r)}\psi(mod N)\rangle\right)\right]$$

And the probability

$$P(y) = \frac{1}{2^{2L}}\left[r\frac{1-\cos(2\pi(2^L//r)r\frac{y}{2^L})}{1-\cos(2\pi r\frac{y}{2^L})} + 2^L(mod r)\right]$$

The measurements will result in periodically multiple peaks at $\frac{k}{r}$ with $k < r$

(see discussions in the jupyter notebook as well as probability analysis in the QPE)