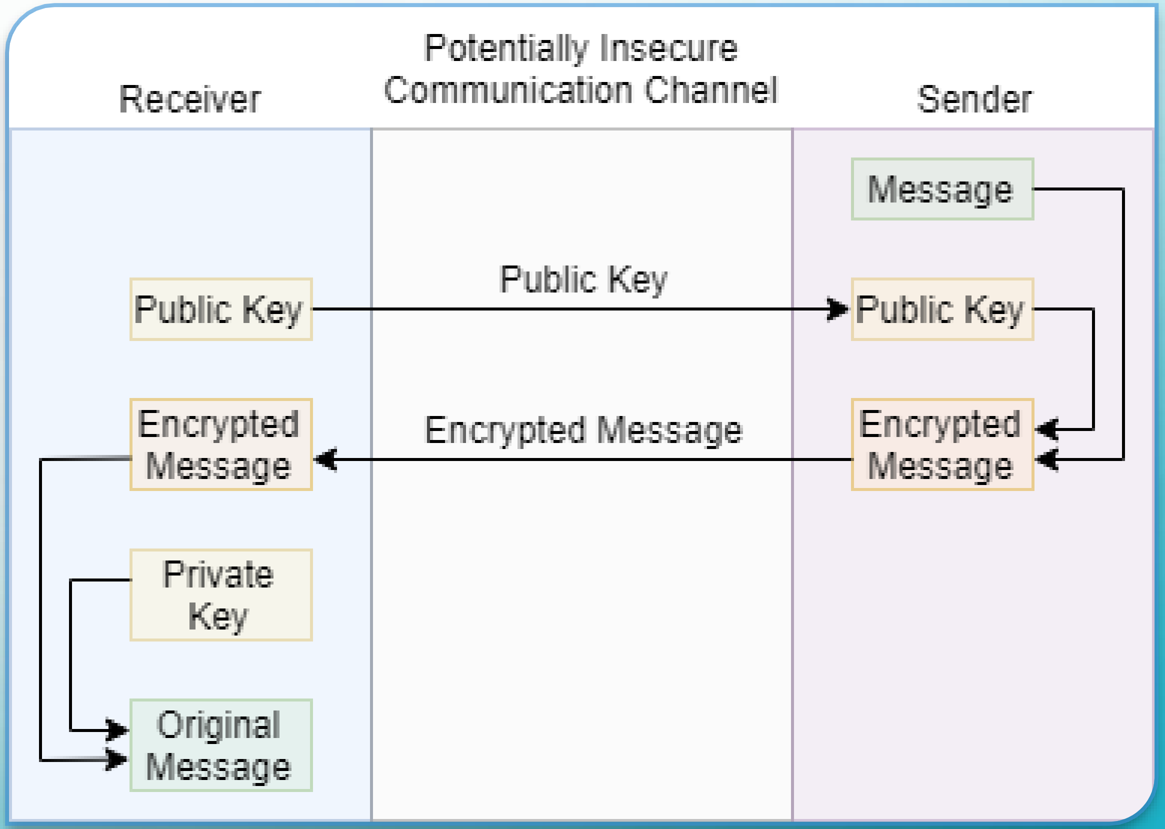
# SHOR'S ALGORITHM

An Efficient Solution to the Factoring Problem

*Adam Klein*

# BACKGROUND

- Finding the factors of a large number is a famously difficult problem
- The best classical algorithms run in super polynomial time (VERY slow)
- RSA Encryption relies on this difficulty to encrypt messages

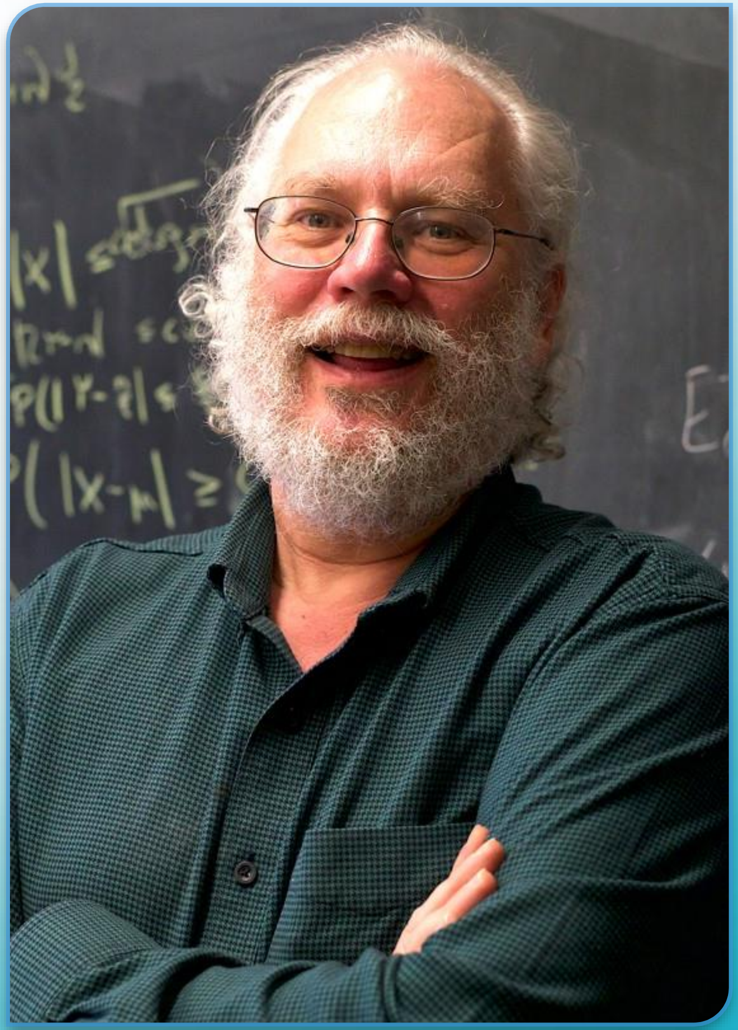



Source

# PETER SHOR

- Math Professor at MIT
- In 1994 he developed Shor's Algorithm: a way to factor numbers in polynomial time
  - Won the Nevanlinna and Gödel Prizes
- Warns that his algorithm could be a threat to international security
- Won a silver medal at the International Math Olympiad in Yugoslavia (if anyone was wondering)

Wikipedia



Source

# SHOR'S SOLUTION:

PERIOD FINDING

# PERIOD FINDING

$$\mathbf{f(x) = } \boldsymbol{a^x mod\ N}$$

- $f$ is periodic (it repeats)
- The period $r$ can be used to find factors of $N$ using Euclid's Algorithm for common divisors:

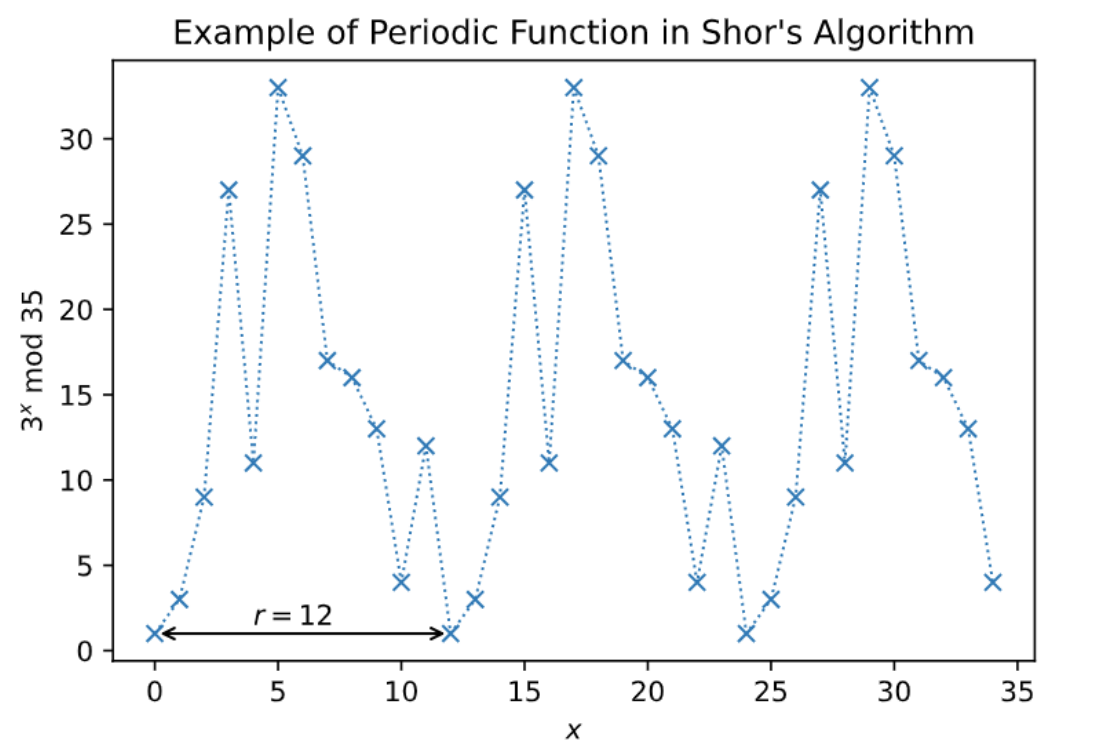
$$F_1 = \gcd\left(a^{\frac{r}{2}} - 1, N\right)$$

$$F_2 = \gcd(a^{\frac{r}{2}} + 1, N)$$

Example:

$$\gcd\left(3^{\frac{12}{2}} - 1, 35\right) = 7$$

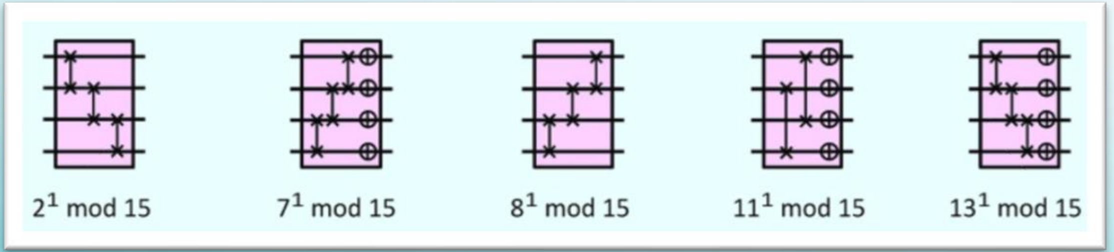
$$\gcd\left(3^{\frac{12}{2}} + 1, 35\right) = 5$$



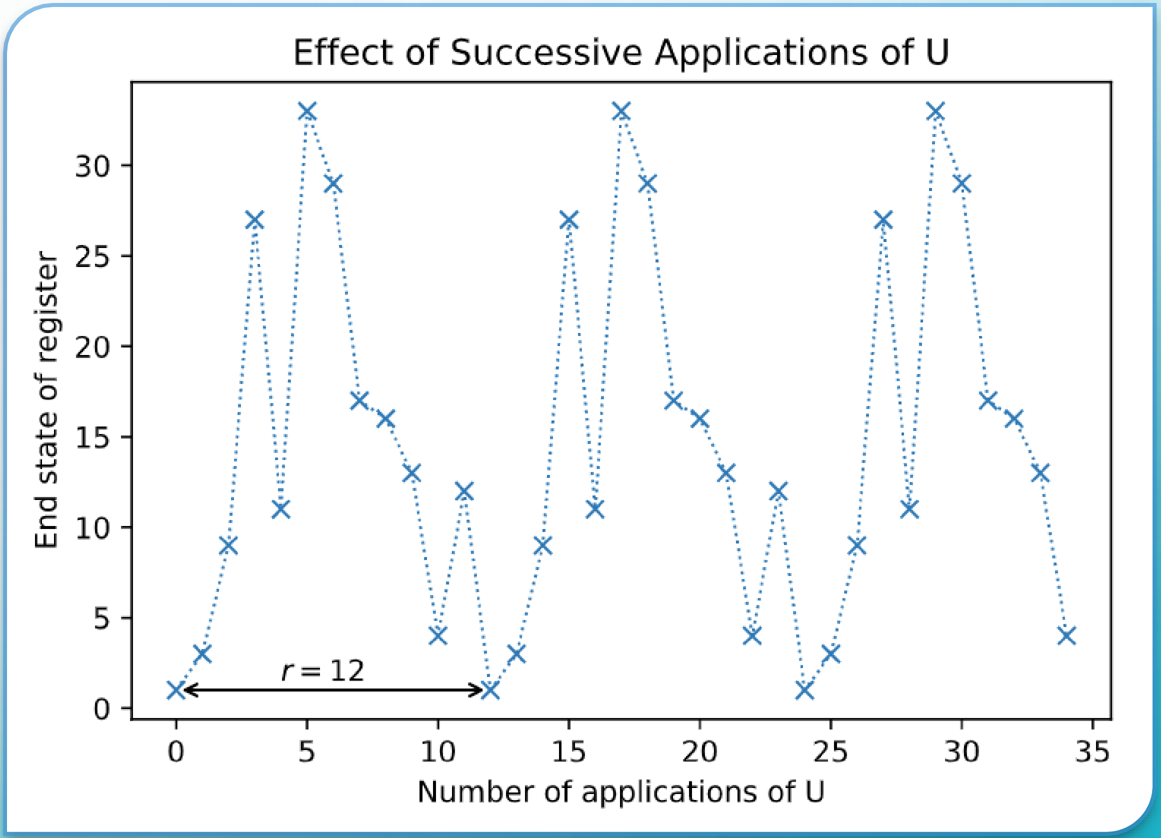

a mod b = remainder of a/b

# CALCULATING THE PERIOD

$$U|y\rangle = |ay \bmod N\rangle$$




$3y \bmod 35$



# CALCULATING THE PERIOD

Superposition of all states in the cycle:

$$|u_0\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |a^k \; mod \; N\rangle$$

Is an eigenstate with eigenvalue 1:

$$U|u_0\rangle = |u_0\rangle$$

(not useful)

$$\mathbf{3y \; mod \; 35}$$

$$
\begin{aligned}
|u_0\rangle &= \frac{1}{\sqrt{12}}(|1\rangle + |3\rangle + |9\rangle \cdots + |4\rangle + |12\rangle) \\
U|u_0\rangle &= \frac{1}{\sqrt{12}}(U|1\rangle + U|3\rangle + U|9\rangle \cdots + U|4\rangle + U|12\rangle) \\
&= \frac{1}{\sqrt{12}}(|3\rangle + |9\rangle + |27\rangle \cdots + |12\rangle + |1\rangle) \\
&= |u_0\rangle
\end{aligned}
$$

# CALCULATING THE PERIOD

Superposition of all states in the cycle:

$$|u_1\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{-\frac{2\pi ik}{r}}|a^k \; mod \; N\rangle$$

Is an eigenstate with eigenvalue $e^{\frac{2\pi i}{r}}$:

$$U|u_1\rangle = e^{\frac{2\pi i}{r}}|u_1\rangle$$

**$3y \; mod \; 35$**

$$|u_1\rangle = \frac{1}{\sqrt{12}}(|1\rangle + e^{-\frac{2\pi i}{12}}|3\rangle + e^{-\frac{4\pi i}{12}}|9\rangle \cdots + e^{-\frac{20\pi i}{12}}|4\rangle + e^{-\frac{22\pi i}{12}}|12\rangle)$$

$$U|u_1\rangle = \frac{1}{\sqrt{12}}(|3\rangle + e^{-\frac{2\pi i}{12}}|9\rangle + e^{-\frac{4\pi i}{12}}|27\rangle \cdots + e^{-\frac{20\pi i}{12}}|12\rangle + e^{-\frac{22\pi i}{12}}|1\rangle)$$

$$U|u_1\rangle = e^{\frac{2\pi i}{12}} \cdot \frac{1}{\sqrt{12}}(e^{\frac{-2\pi i}{12}}|3\rangle + e^{-\frac{4\pi i}{12}}|9\rangle + e^{-\frac{6\pi i}{12}}|27\rangle \cdots + e^{-\frac{22\pi i}{12}}|12\rangle + e^{-\frac{24\pi i}{12}}|1\rangle)$$

$$U|u_1\rangle = e^{\frac{2\pi i}{12}}|u_1\rangle$$

# CALCULATING THE PERIOD

Superposition of all states in the cycle:

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi iks}{r}} |a^k \ mod \ N\rangle$$

Is an eigenstate with eigenvalue $e^{\frac{2\pi is}{r}}$:

$$U|u_s\rangle = e^{\frac{2\pi is}{r}} |u_s\rangle$$

**$3y \ mod \ 35$**

$$|u_s\rangle = \frac{1}{\sqrt{12}}(|1\rangle + e^{-\frac{2\pi is}{12}}|3\rangle + e^{-\frac{4\pi is}{12}}|9\rangle \cdots + e^{-\frac{20\pi is}{12}}|4\rangle + e^{-\frac{22\pi is}{12}}|12\rangle)$$

$$U|u_s\rangle = \frac{1}{\sqrt{12}}(|3\rangle + e^{-\frac{2\pi is}{12}}|9\rangle + e^{-\frac{4\pi is}{12}}|27\rangle \cdots + e^{-\frac{20\pi is}{12}}|12\rangle + e^{-\frac{22\pi is}{12}}|1\rangle)$$

$$U|u_s\rangle = e^{\frac{2\pi is}{12}} \cdot \frac{1}{\sqrt{12}}(e^{-\frac{2\pi is}{12}}|3\rangle + e^{-\frac{4\pi is}{12}}|9\rangle + e^{-\frac{6\pi is}{12}}|27\rangle \cdots + e^{-\frac{22\pi is}{12}}|12\rangle + e^{-\frac{24\pi is}{12}}|1\rangle)$$

$$U|u_s\rangle = e^{\frac{2\pi is}{12}}|u_s\rangle$$

# CALCULATING THE PERIOD

$|u_s\rangle$ gives us eigenstates for every value of s:

$$0 \leq s \leq r-1$$

The superposition of all these eigenstates gives:

$$\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|u_s\rangle = |1\rangle$$

$$\boldsymbol{7y\ mod\ 15}$$

$$\frac{1}{2}\Big( \quad |u_0\rangle = \frac{1}{2}(|1\rangle + |7\rangle + |4\rangle + |13\rangle)\ldots$$

$$+|u_1\rangle = \frac{1}{2}(|1\rangle + e^{-\frac{2\pi i}{4}}|7\rangle + e^{-\frac{4\pi i}{4}}|4\rangle + e^{-\frac{6\pi i}{4}}|13\rangle)\ldots$$

$$+|u_2\rangle = \frac{1}{2}(|1\rangle + e^{-\frac{4\pi i}{4}}|7\rangle + e^{-\frac{8\pi i}{4}}|4\rangle + e^{-\frac{12\pi i}{4}}|13\rangle)\ldots$$

$$+|u_3\rangle = \frac{1}{2}(|1\rangle + e^{-\frac{6\pi i}{4}}|7\rangle + e^{-\frac{12\pi i}{4}}|4\rangle + e^{-\frac{18\pi i}{4}}|13\rangle) \quad \Big) = |1\rangle$$

# CALCULATING THE PERIOD

$$U|u_s\rangle = e^{\frac{2\pi i s}{r}}|u_s\rangle$$

is of the form:

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

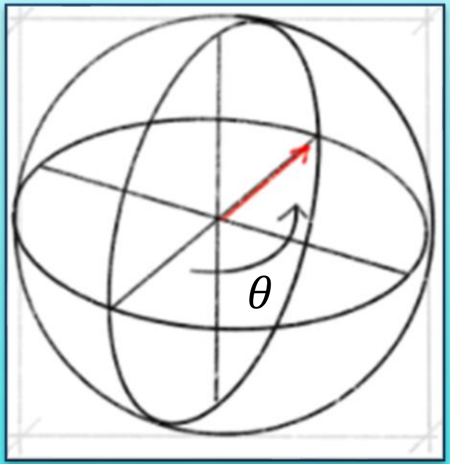
$$\theta = \frac{s}{r}$$

We can use Quantum Phase Estimation to find $\theta$

# PHASE KICKBACK

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

Acting on $|\psi\rangle$ with a controlled U gate will phase shift the control qubit by $e^{2\pi i\theta}$




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

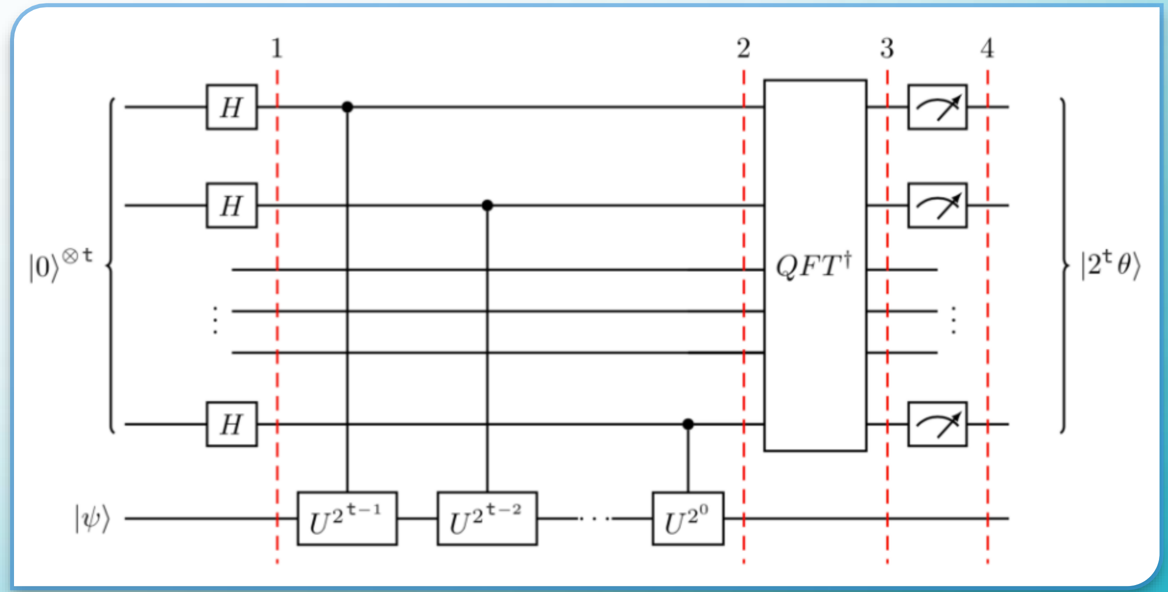
$$\begin{aligned}|1+\rangle &= |1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\\ &= \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)\end{aligned}$$

$$\begin{aligned}\text{Controlled-T}|1+\rangle &= \frac{1}{\sqrt{2}}(|10\rangle + e^{i\pi/4}|11\rangle)\\ &= |1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)\end{aligned}$$

# QUANTUM PHASE ESTIMATION (QPE)

1. Initialize $n$ “counting qubits” in the Hadamard basis
2. For each counting qubit $q_t$, use it as a control qubit for $CU$ repeated for $2^t$ times
3. Each qubit is now in the state:

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



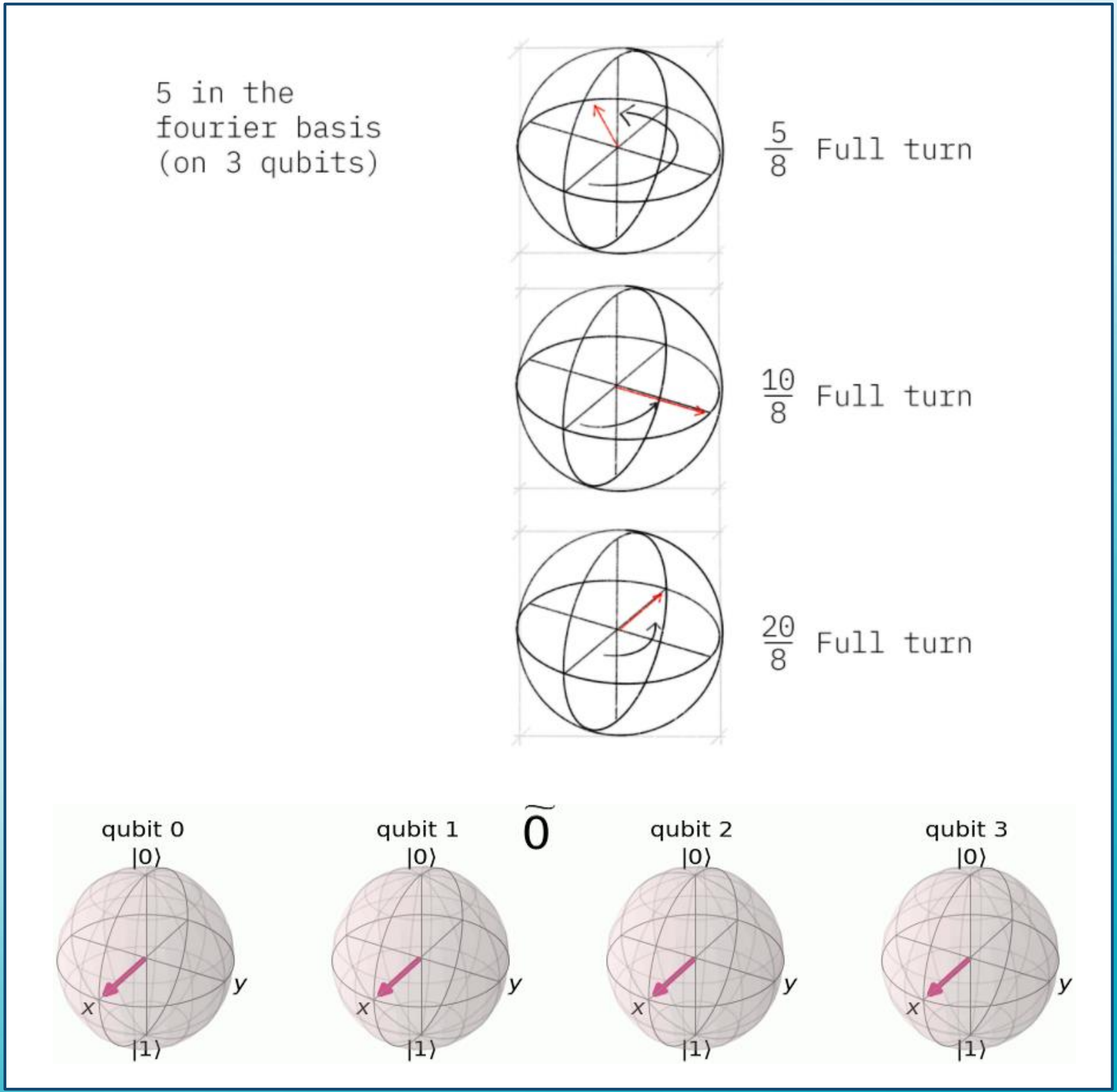
# FOURIER BASIS

- A system of storing number on qubits
- Can be converted to and from binary basis
- To represent the number *k* on *n* qubits, each qubit $q_t$ is in the state:

$$|q_t\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{\frac{2\pi i 2^t k}{2^n}}|1\rangle)$$

- When used in QPE:

$$\theta = \frac{k}{2^n}$$



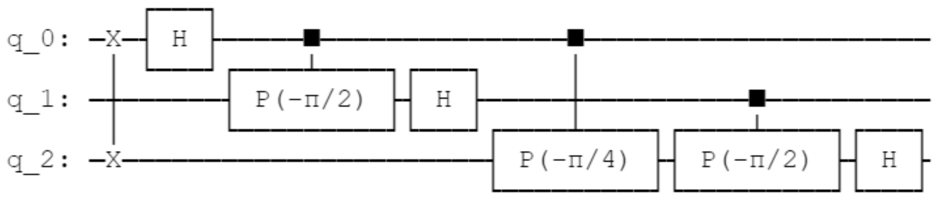
# QFT INVERSION

- Numbers in the Fourier bases can be converted into binary number *k*

$$k = 2^n\theta$$

```
qftInvert = QuantumCircuit(n)

# Invert ordering of qubits
for q in range(n//2):
    qftInvert.swap(q, n-qubit-1)

# Convert phase to standard binary
for q in range(n):
    for m in range(q):
        QFT.cp(-np.pi/float(2**(q-m)), m, q)
    QFT.h(j)
```

QFT inversion For n = 3:



# PHASE ESTIMATION

- When QPE is done on $U|1\rangle$ (remember $|1\rangle$ is the superposition of all $|u_s\rangle$):

$$k = \frac{2^n s}{r}$$

for some random *s*.

- *K* and *n* are known, so we can solve for $\frac{s}{r} = \frac{k}{2^n}$
- Repeated applications will yield $\frac{s_1}{r}, \frac{s_2}{r}, \ldots$
- "Continued Fractions" can be used to solve for *r*.

# SHOR'S ALGORITHM

To find the factors of *N*:

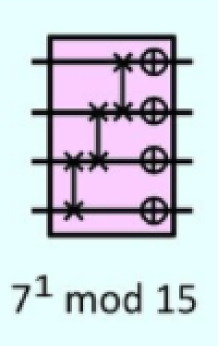
1. Choose some random *a* (some guesses are better than others)
2. Create the operator $U|y\rangle = |ay \bmod N\rangle$
3. Use QPE to find the period
4. Use Euclid's algorithm to find the factors of *N*
5. Repeat with different *a* until suitable factors are found
    - If the period is odd, it can't be used in Euclid's Algorithm

# EXAMPLE:

FINDING FACTORS FOR 15

# FINDING FACTORS OF 15

1. Try a = 7
2. Create operator: $U|y\rangle = |7y\ mod\ 15\rangle$:




```python
def 7_amod15():
    """
    Return a controlled gate that does 7 mod15
    multiplication.
    """
    U = QuantumCircuit(4)
    U.swap(2,3)
    U.swap(1,2)
    U.swap(0,1)
    for q in range(4):
        U.x(q)
    U = U.to_gate()
    U.name = "7 mod 15"
    c_U = U.control(1)
    return c_U
```

# FINDING FACTORS FOR 15

- We want to be able to be able to repeat U, so we'll make a gate that does $U^p$:

```
def 7_amod15(power):
    """
    Return a controlled gate that does 7 mod15
    multiplication.
    Repeated power times.
    """
    U = QuantumCircuit(4)
    for i in range(power):
        U.swap(2,3)
        U.swap(1,2)
        U.swap(0,1)
        for q in range(4):
            U.x(q)
    U = U.to_gate()
    U.name = "%i^%i mod 15" % (7, power)
    c_U = U.control(1)
    return c_U
```

# FINDING FACTORS FOR 15

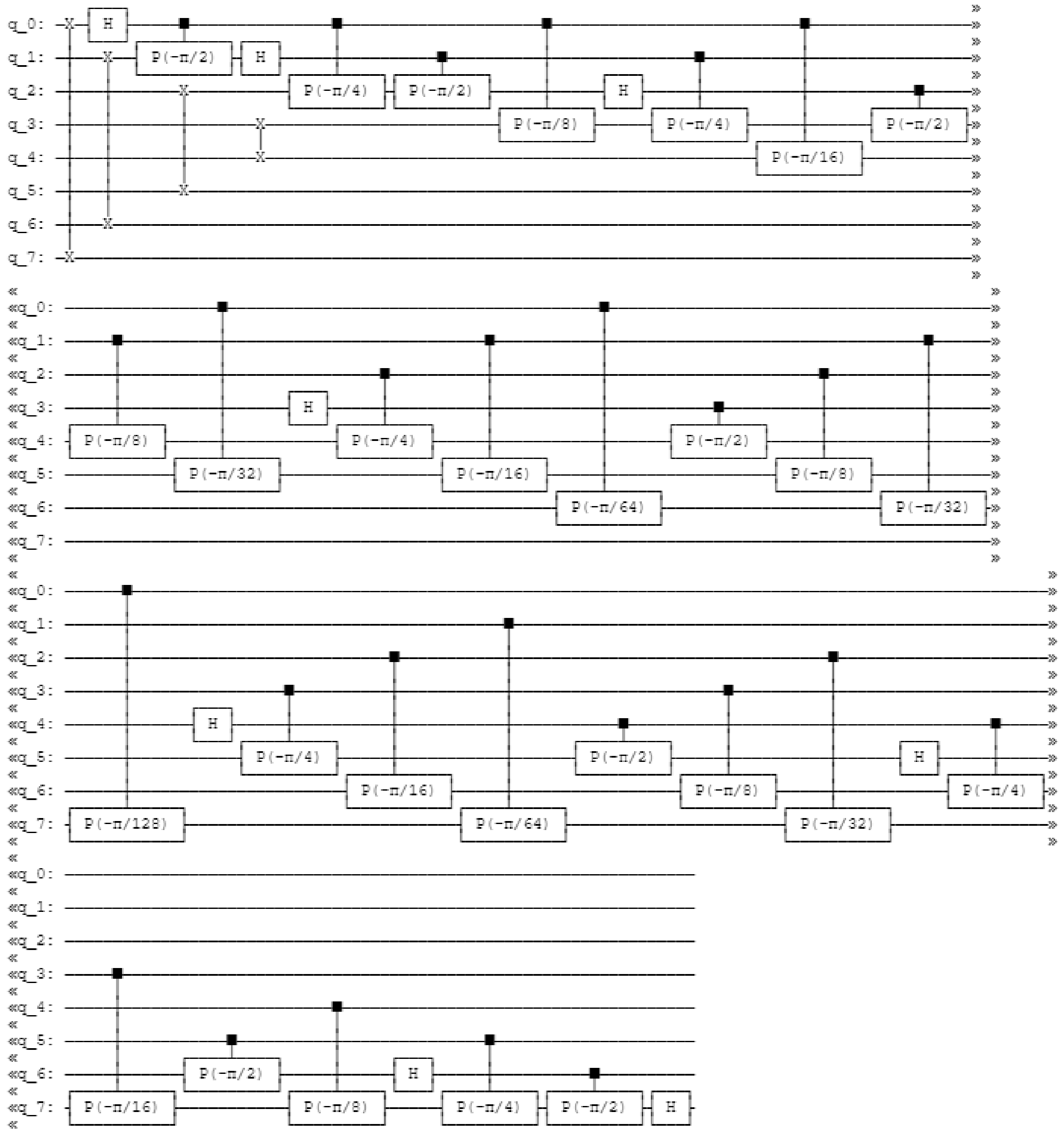
- For QPE, we need a QFT inversion gate that converts from Fourier basis to binary:

(We'll use 8 counting qubits)

```
qftInvert = QuantumCircuit(n)

# Invert ordering of qubits
for q in range(n//2):
    qftInvert.swap(q, n-qubit-1)

# Convert phase to standard binary
for q in range(n):
    for m in range(q):
        QFT.cp(-np.pi/float(2**(q-m)), m, q)
    QFT.h(j)
```

# QFT Inversion of n=8

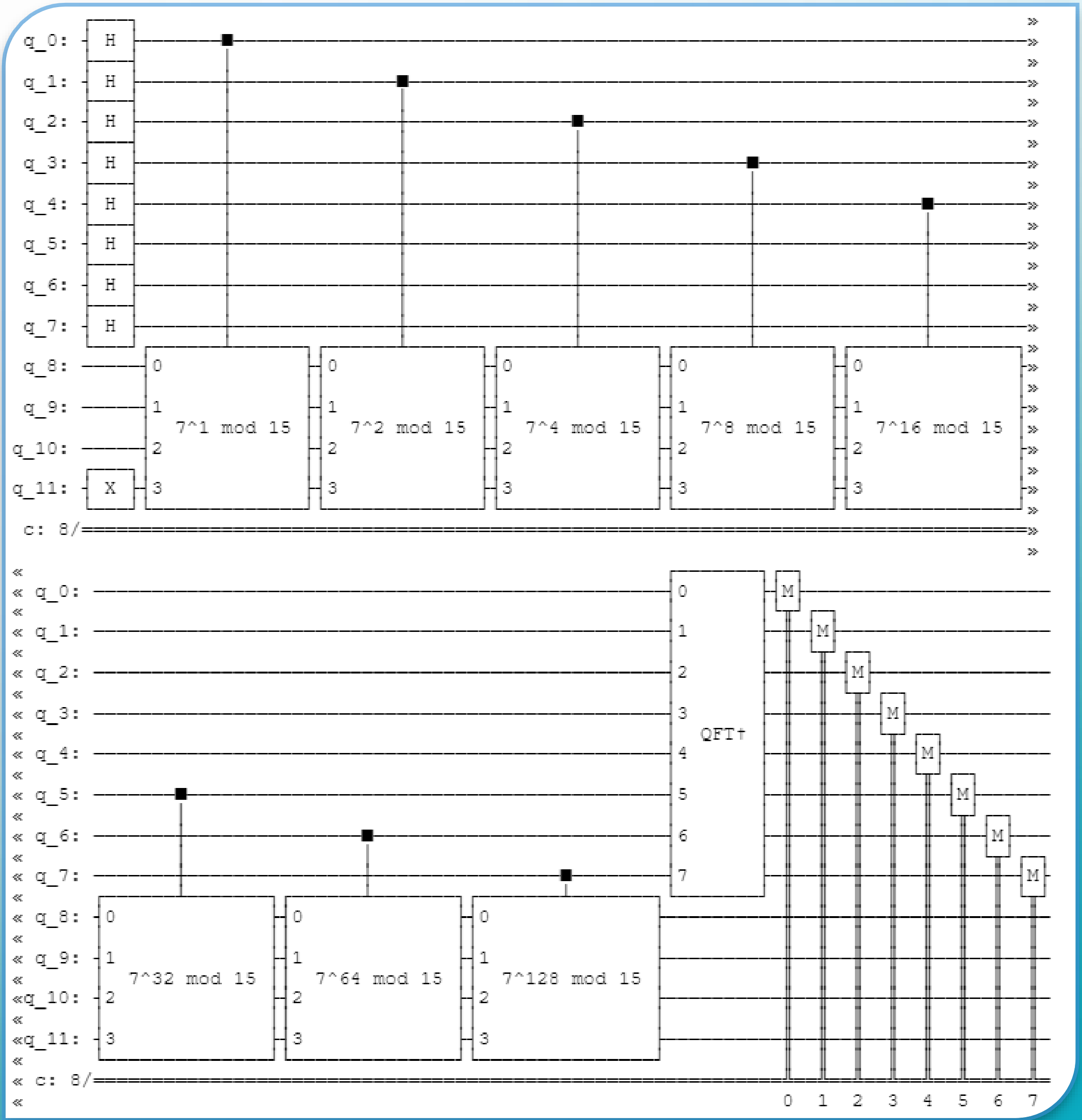
Out[50]:



# FINDING FACTORS OF 15

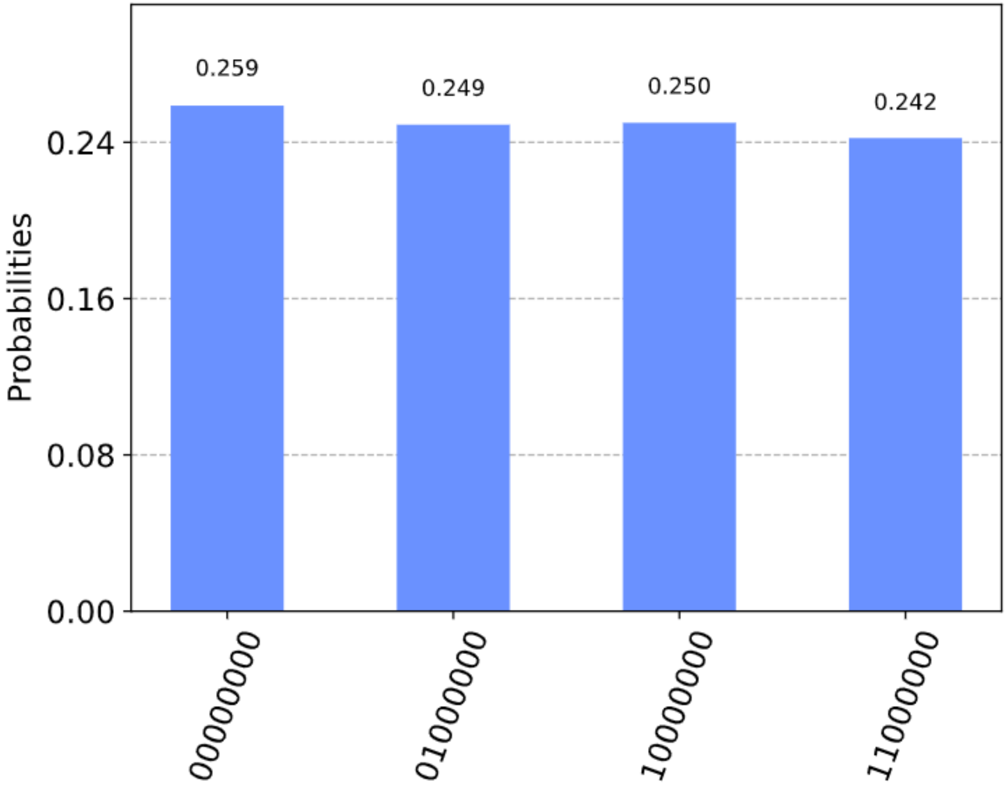
- Put it together to create a QPE circuit:
  - $q_0 - q_7$ are counting qubits
  - $q_8 - q_{11}$ are in $|1\rangle$ state

```python
#N_COUNT+4 qubits, N_COUNT bits
circuit = QuantumCircuit(N_COUNT+4, N_COUNT)

#Initialize counting qubits
circuit.h(range(N_COUNT))
#Initialize 1 state
circuit.x(N_COUNT+3)

#Phase kickback on counting qubits
for q in range(N_COUNT):
    circuit.append(c_amod15(A, 2**q), [q]+[i+N_COUNT for i in range(4)])

#Convert result to binary
circuit.append(qft_dagger(N_COUNT), range(N_COUNT))

#Measure resulting k
circuit.measure(range(N_COUNT), range(N_COUNT))
```



# FINDING FACTORS OF 15

- After repeated simulation, you get 4 different results:
  - Phase $= \frac{k}{2^n}, n = 8$




| | Register Output | | Phase |
|---|---|---|---|
| 0 | 00000000(bin) | = 0(dec) | 0/256 = 0.00 |
| 1 | 01000000(bin) | = 64(dec) | 64/256 = 0.25 |
| 2 | 10000000(bin) | = 128(dec) | 128/256 = 0.50 |
| 3 | 11000000(bin) | = 192(dec) | 192/256 = 0.75 |

# FINDING FACTORS FOR 15

- Using the results, we can calculate the period r

| | Phase | Fraction |
|---|---|---|
| 0 | 0.00 | 0/1 |
| 1 | 0.25 | 1/4 |
| 2 | 0.50 | 1/2 |
| 3 | 0.75 | 3/4 |

$$\frac{s_1}{r} = \frac{0}{1}, \quad \frac{s_2}{r} = \frac{1}{4}, \quad \frac{s_3}{r} = \frac{1}{2}, \quad \frac{s_4}{r} = \frac{3}{4}$$

Continued fractions gives

$$r = 4$$

# FINDING FACTORS FOR 15

Since *r* is even, we can use Euclid's algorithm on *r* to find the factors:

```
guesses = [gcd(A**(r//2)-1, 15), gcd(A**(r//2)+1, 15)]
print(guesses)
```

```
[3, 5]
```

3 and 5 are indeed the factors of 15, so the algorithm was successful.

# QUESTIONS?

Sources:

Shor's Algorithm (qiskit.org)

Quantum Fourier Transform (qiskit.org)

Quantum Phase Estimation (qiskit.org)

Phase Kickback (qiskit.org)

Realization of a scalable Shor algorithm | Science (sciencemag.org)