The Jacobi Factoring Circuit

a talk based on joint work by…

Greg Kahanamoku-Meyer
MIT

Seyoon Ragavan
MIT

Vinod Vaikuntanathan
MIT

Katherine Van Kirk
Harvard

QIP 2026 | January 29, 2026

A super compact quantum factoring circuit

a talk based on joint work by…

Greg Kahanamoku-Meyer
MIT

Seyoon Ragavan
MIT

Vinod Vaikuntanathan
MIT

Katherine Van Kirk
Harvard

QIP 2026 | January 29, 2026

Background: Classical Factoring

Integer Factoring Problem

Given an n-bit integer $N < 2^n$, find its prime factorization in poly(n) time.

A "crash course"

general integers

special-form integers

Background: Classical Factoring

Integer Factoring Problem

Given an n-bit integer $N < 2^n$, find its prime factorization in poly(n) time.

A "crash course"

general integers

- Brute force: exp($O(n)$)

special-form integers

Background: Classical Factoring

Integer Factoring Problem

Given an n-bit integer $N < 2^n$, find its prime factorization in poly(n) time.

A “crash course”

general integers

- Brute force: exp($O(n)$)
- Quadratic sieve (many works from Fermat to Pomerance ‘82): exp($\tilde{O}(n^{1/2})$)

special-form integers

Background: Classical Factoring

Integer Factoring Problem

Given an n-bit integer $N < 2^n$, find its prime factorization in poly(n) time.

A “crash course”

general integers

- Brute force: exp($O(n)$)
- Quadratic sieve (many works from Fermat to Pomerance ‘82): exp($\tilde{O}(n^{1/2})$)
- General number field sieve (Pollard ’88; Buhler-Lenstra-Pomerance ‘93): exp($\tilde{O}(n^{1/3})$)

special-form integers

Background: Classical Factoring

Integer Factoring Problem

Given an n-bit integer $N < 2^n$, find its prime factorization in poly(n) time.

A "crash course"

general integers

- Brute force: exp($O(n)$)
- Quadratic sieve (many works from Fermat to Pomerance '82): exp($\tilde{O}(n^{1/2})$)
- General number field sieve (Pollard '88; Buhler-Lenstra-Pomerance '93): exp($\tilde{O}(n^{1/3})$)

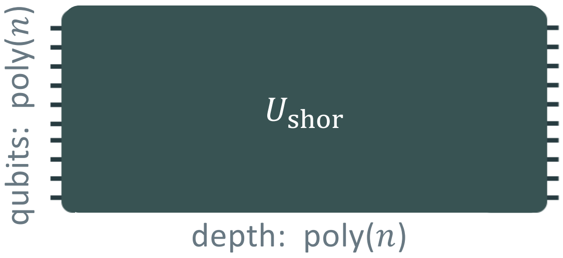
special-form integers

- Lenstra ECM ('87): exp($\tilde{O}((\log P)^{1/2})$) where P is smallest prime factor of N

$n = \log N$

Shor's algorithm can factor *any* n-bit number using $O(n^2)$ gates, $O(n)$ qubits

$$N = P * Q$$



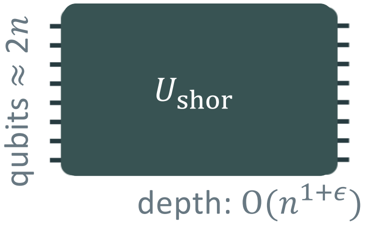
Shor

Shor '95

$n = \log N$

Shor's algorithm can factor *any* n-bit number using $O(n^2)$ gates, $O(n)$ qubits

$$N = P * Q$$



Shor

+

Kahanamoku-Meyer *et al.*

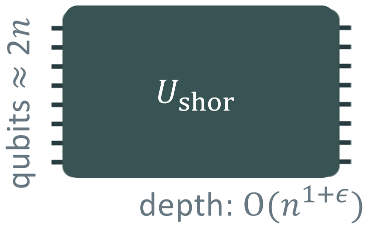
G. Kahanamoku-Meyer, N. Yao. arXiv:2403.18006
G. Kahanamoku-Meyer, J. Blue, T. Bergamaschi, C. Gidney, I. Chuang. arXiv:2505.00701

$n = \log N$

$m = \log Q$

Shor's algorithm can factor *any* n-bit number using $O(n^2)$ gates, $O(n)$ qubits

$$N = P * Q$$



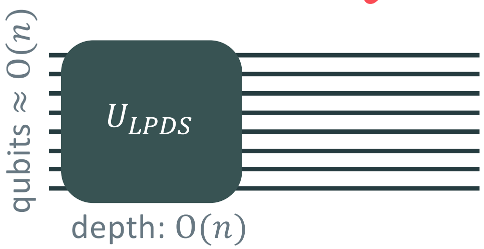
Shor

+

Kahanamoku-Meyer *et al.*

Jacobi algorithm can factor *some* n-bit numbers using only $O(n)$ gates

$$N = P^2 * Q$$



Li

Peng

Du

Suter

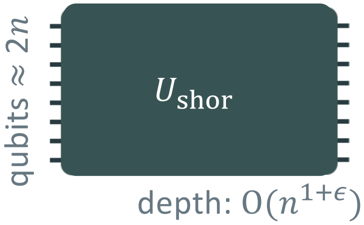
Li, Peng, Du, Suter, *Nat. Comm.* 2012

$n = \log N$

$m = \log Q$

Shor's algorithm can factor *any* n-bit number using $O(n^2)$ gates, $O(n)$ qubits

$$N = P * Q$$



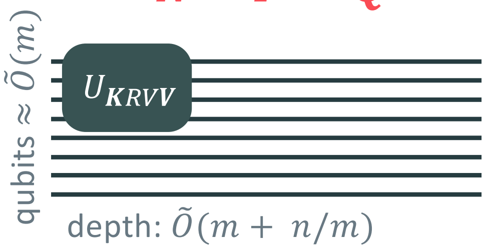
Shor

+

Kahanamoku-Meyer *et al.*

Jacobi algorithm can factor *some* n-bit numbers using only $O(n)$ gates, $O(m)$ qubits

$$N = P^2 * Q$$



Li

Peng

Du

Suter

Kahanamoku-Meyer

SR

Vaikuntanathan

KVK

$n = \log N$
$m = \log Q$

Aside: how to set $m = \log Q$ relative to n?

$$N = P^2 * Q$$

$n = \log N$
$m = \log Q$

Aside: how to set $m = \log Q$ relative to n?

$$N = P^2 * Q$$

$n = \log N$
$m = \log Q$

Aside: how to set $m = \log Q$ relative to n?

$$N = P^2 * Q$$

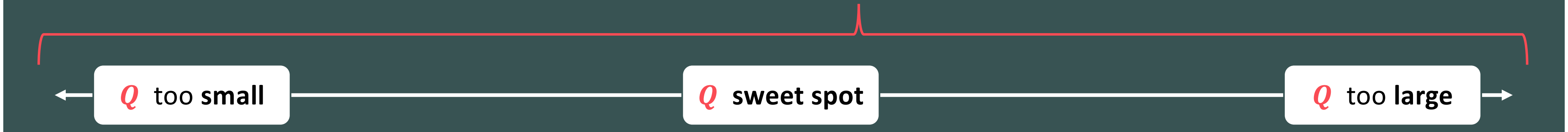
Q too large

our circuit is no better than LPDS’12

$\boldsymbol{n} = \log \boldsymbol{N}$
$\boldsymbol{m} = \log \boldsymbol{Q}$

Aside: how to set $m = \log Q$ relative to n?

$$N = P^2 * Q$$



Q too small

Q sweet spot

Q too large

classical algorithms could exploit this structure to run faster than general NFS

- NFS: $\exp(\tilde{O}(n^{1/3}))$
- Lenstra ECM: $\exp(\tilde{O}(m^{1/2}))$

sweet spot: set $m = \tilde{O}(n^{2/3})$

our circuit is no better than LPDS’12

$n = \log N$
$m = \log Q$

Aside: how to set $m = \log Q$ relative to n?

$$N = P^2 * \underbrace{Q}$$

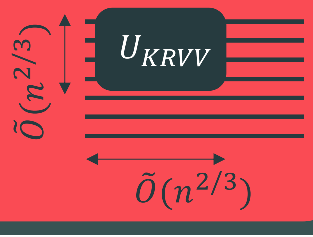
Q too small ← — Q sweet spot — Q too large →

classical algorithms could exploit this structure to run faster than general NFS

- NFS: $\exp(\tilde{O}(n^{1/3}))$
- Lenstra ECM: $\exp(\tilde{O}(m^{1/2}))$

sweet spot: set $m = \tilde{O}(n^{2/3})$

our result →



our circuit is no better than LPDS'12

Outline

$n = \log N$

$m = \log Q$

1

Shor's algorithm can factor *any* n-bit number using $O(n^2)$ gates, $O(n)$ qubits

$$N = P * Q$$

Shor

2

Jacobi algorithm can factor *some* n-bit numbers using only $O(n)$ gates, $O(m)$ qubits

$$N = P^2 * Q$$

Li

Peng

Du

Suter

Kahanamoku-Meyer

SR

Vaikuntanathan

KVK

Outline

$n = \log N$

$m = \log Q$

1

Shor's algorithm can factor *<u>any</u>* n-bit number using $O(n^2)$ gates, $O(n)$ qubits

$$N = P * Q$$

Shor

2

Jacobi algorithm can factor *<u>some</u>* n-bit numbers using only $O(n)$ gates, $O(m)$ qubits

$$N = P^2 * Q$$

Li

Peng

Du

Suter

Kahanamoku
-Meyer

SR

Vaikuntanathan

KVK

Preliminary: Quantum Period Finding

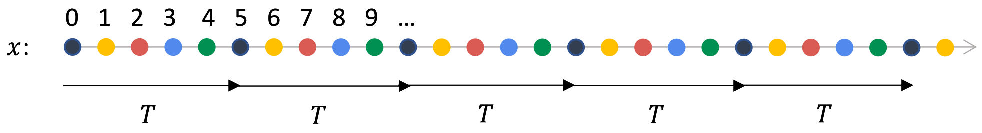
Setup

Given periodic function $f\colon \mathbb{Z} \to \mathbb{Z}$ with unknown period T

$$f(x+T) = f(x)$$

Informal Theorem Statement

For "reasonable" f, one can quantumly recover T using only the gates/ space needed to compute $f(x)$ for $|x| \le \text{poly}(T)$



color denotes value of $f(x)$

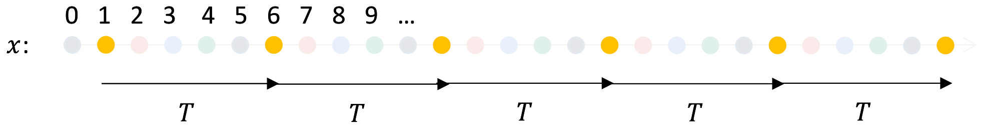
Preliminary: Quantum Period Finding

Setup

Given periodic function $f\colon \mathbb{Z} \to \mathbb{Z}$ with unknown period T $\qquad f(x+T) = f(x)$

Informal Theorem Statement

For "reasonable" f, one can quantumly recover T using only the gates/ space needed to compute $f(x)$ for $|x| \le \mathrm{poly}(T)$



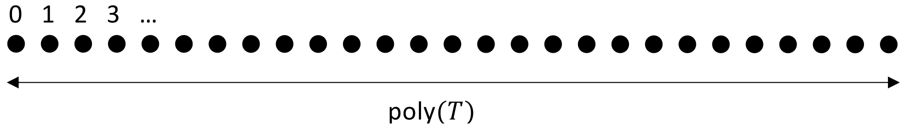
color denotes value of $f(x)$

Bare minimum: Need $O(\log T)$ qubits for the superposition

Preliminary: Quantum Period Finding

Algorithm

1 $$\sum_{x=0}^{poly(T)} | x > \quad =$$

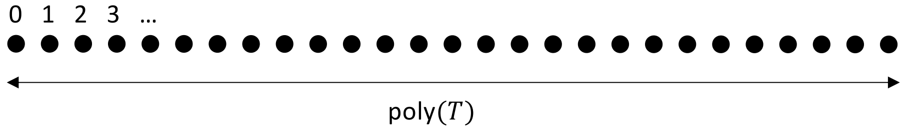


Preliminary: Quantum Period Finding

Algorithm

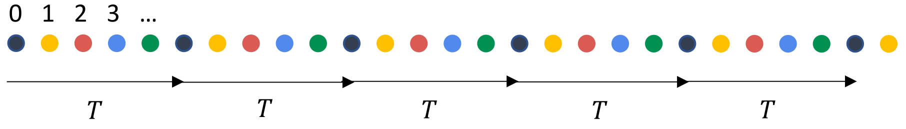
1

$$\sum_{x=0}^{poly(T)} | x > \quad =$$



2

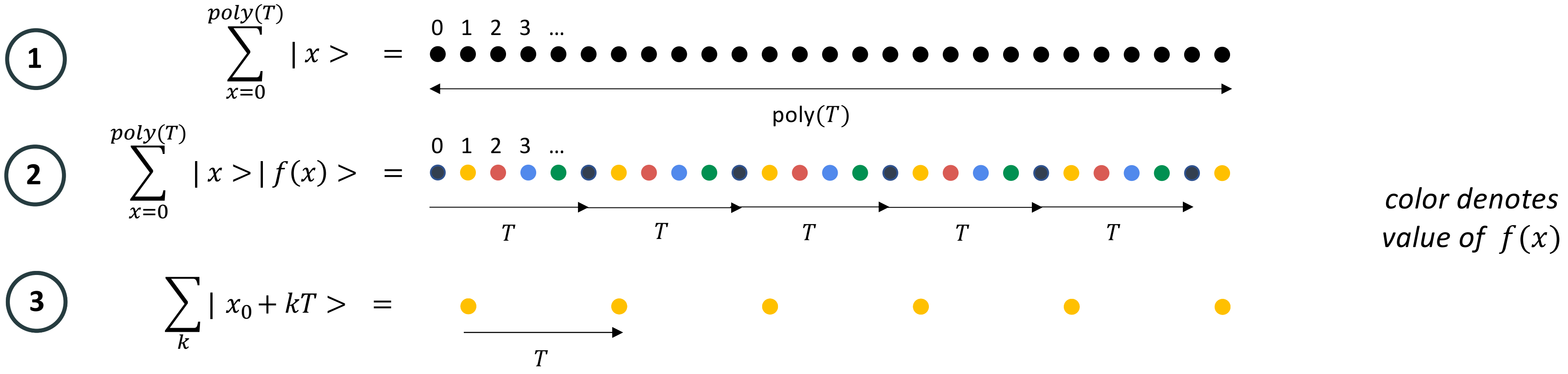
$$\sum_{x=0}^{poly(T)} | x > | f(x) > \quad =$$



color denotes value of $f(x)$

Preliminary: Quantum Period Finding

Algorithm



1 $\sum_{x=0}^{poly(T)} |\, x > \; =$

2 $\sum_{x=0}^{poly(T)} |\, x > |\, f(x) > \; =$

3 $\sum_{k} |\, x_0 + kT > \; =$

color denotes value of $f(x)$

Preliminary: Quantum Period Finding

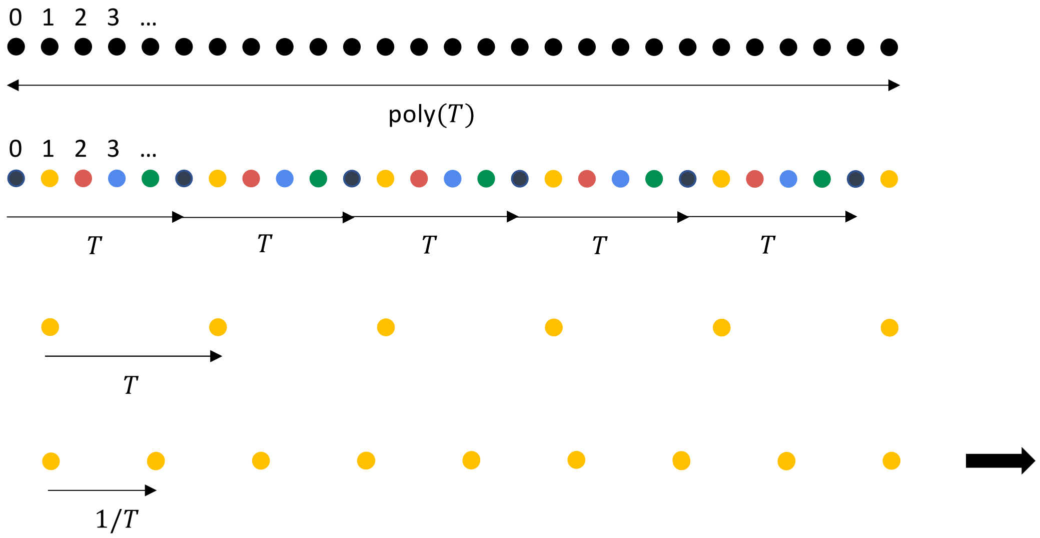
Algorithm

1. $\sum_{x=0}^{poly(T)} |\, x > \;=$

2. $\sum_{x=0}^{poly(T)} |\, x > |\, f(x) > \;=$

3. $\sum_{k} |\, x_0 + kT > \;=$

4. $QFT\,[\sum_{k} |\, x_0 + kT >] \;=$



color denotes value of $f(x)$

outcome is a random multiple of $1/T$

Shor’s factoring algorithm

The high-level idea

Goal: Find a nontrivial factor (not 1 or N) of the number N

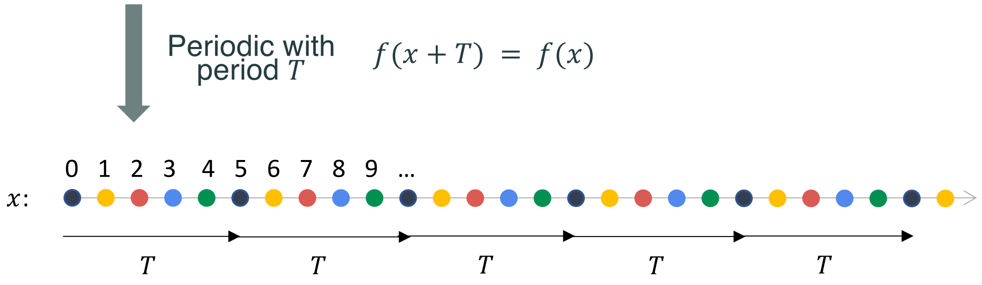
Shor's factoring algorithm

The high-level idea

Goal: Find a nontrivial factor (not 1 or N) of the number N

Peter Shor

$$f(x) = b^{2x} \bmod N$$



color denotes value of $f(x)$

Shor’s factoring algorithm

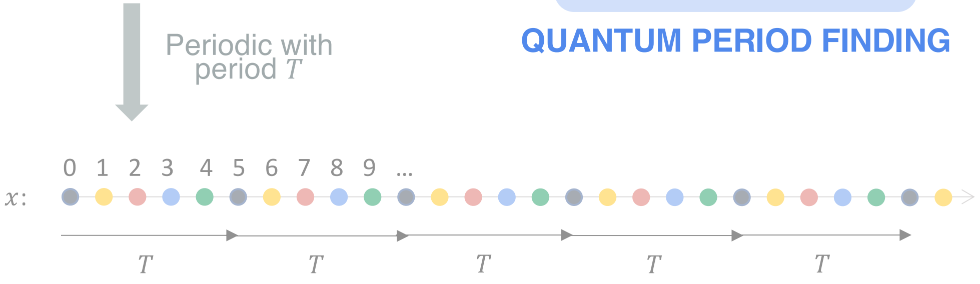
The high-level idea

Goal: Find a nontrivial factor (not 1 or N) of the number N

Peter Shor

$f(x) = b^{2x} \bmod N$ → Find the period T → Know $b^{2T} - 1 = (b^T - 1)(b^T + 1)$ is divisible by $N \rightarrow$ hope $\gcd(b^T - 1, N)$ is a nontrivial factor

QUANTUM PERIOD FINDING



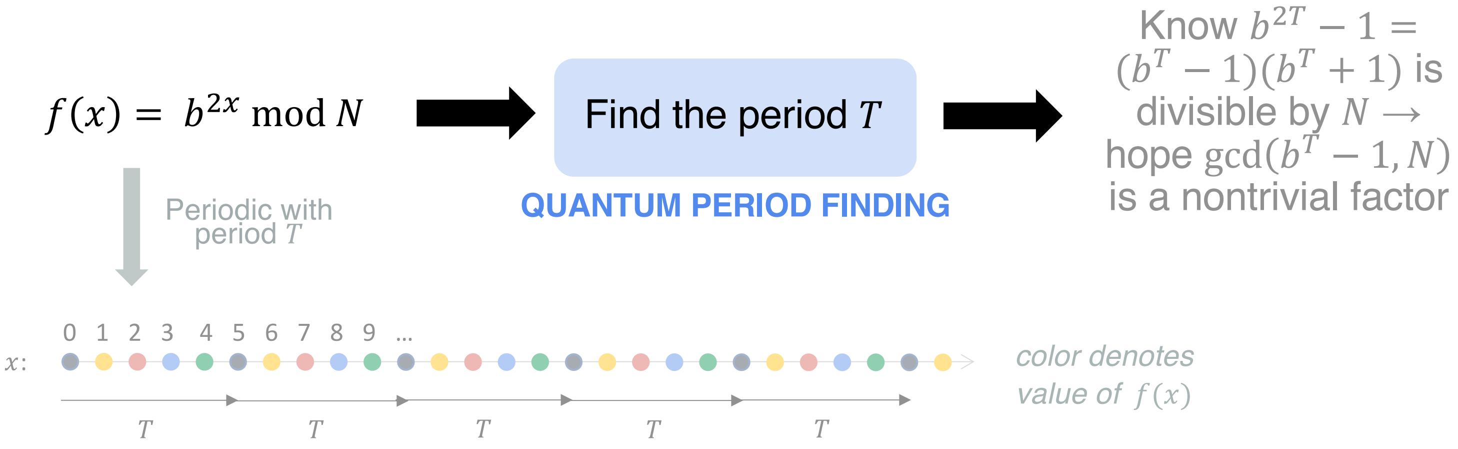
color denotes value of $f(x)$

Shor’s factoring algorithm

Costs

- Period of $f(x)$ is $O(N)$ → Bare minimum **qubit count**: $O(\log N) = \boldsymbol{O(n)}$
- Turns out that computing f requires $\boldsymbol{\tilde{O}(n^2)}$ **gates**

Peter Shor



$f(x) = b^{2x} \bmod N$

Periodic with period T

Find the period T

QUANTUM PERIOD FINDING

Know $b^{2T} - 1 = (b^T - 1)(b^T + 1)$ is divisible by N → hope $\gcd(b^T - 1, N)$ is a nontrivial factor

color denotes value of $f(x)$

Outline

$n = \log N$
$m = \log Q$

1

Shor's algorithm can factor _any_ n-bit number using $O(n^2)$ gates, $O(n)$ qubits

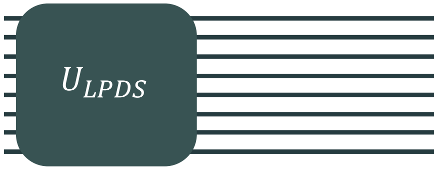
$$N = P * Q$$

Shor

2a

Jacobi algorithm can factor _some_ n-bit numbers using only $O(n)$ gates

$$N = P^2 * Q$$



Li

Peng

Du

Suter

Recall:

Cost of Shor's factoring algorithm

- Period of $f(x)$: $O(N)$
- Gate count to compute f: $O(n^2)$

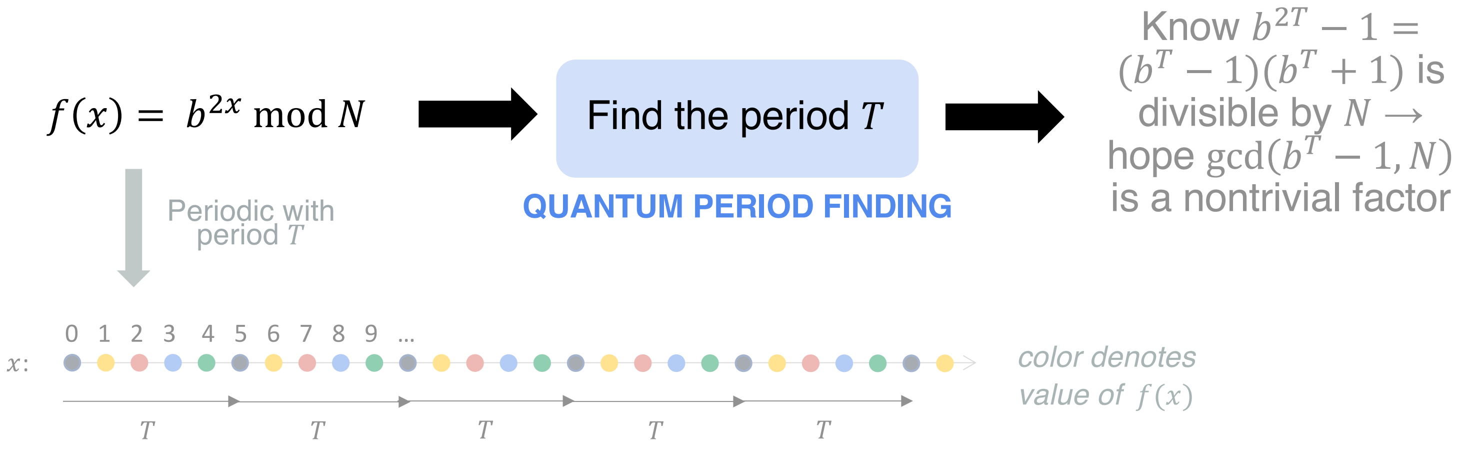
What if we could find a different function $j(x)$ which:

a) Also has periodicity that enables us to factor N;

b) Has a shorter period; and

c) Is easier to implement?

Peter Shor



$f(x) = b^{2x} \bmod N$

Periodic with period T

Find the period T

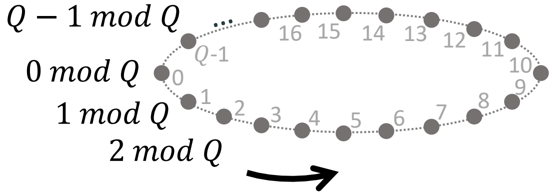
QUANTUM PERIOD FINDING

Know $b^{2T} - 1 = (b^T - 1)(b^T + 1)$ is divisible by $N \rightarrow$ hope $\gcd(b^T - 1, N)$ is a nontrivial factor

color denotes value of $f(x)$

Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.



$Q - 1\ mod\ Q$

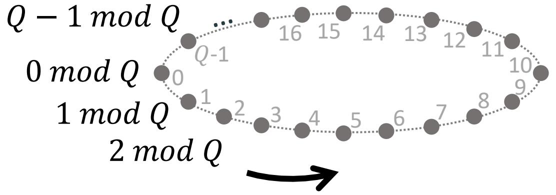
$0\ mod\ Q$

$1\ mod\ Q$

$2\ mod\ Q$

Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.



$Q-1\ mod\ Q$

$0\ mod\ Q$

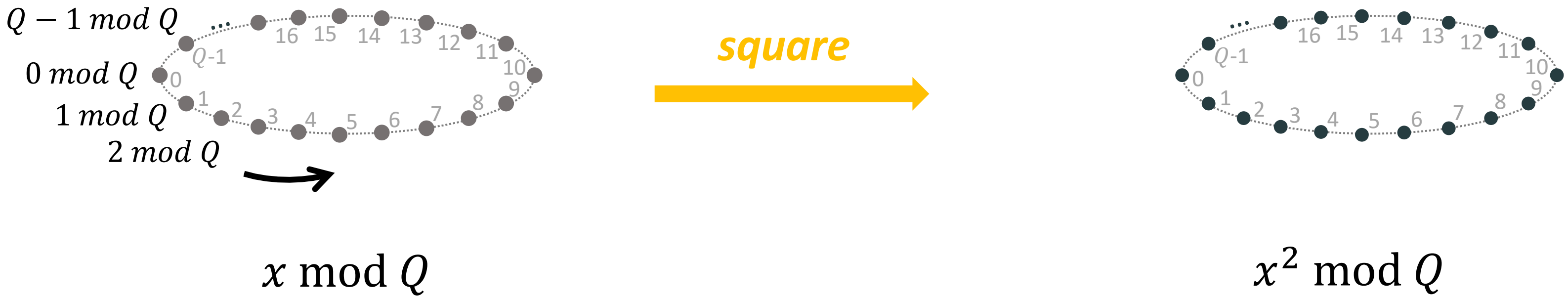
$1\ mod\ Q$

$2\ mod\ Q$

$x \bmod Q$

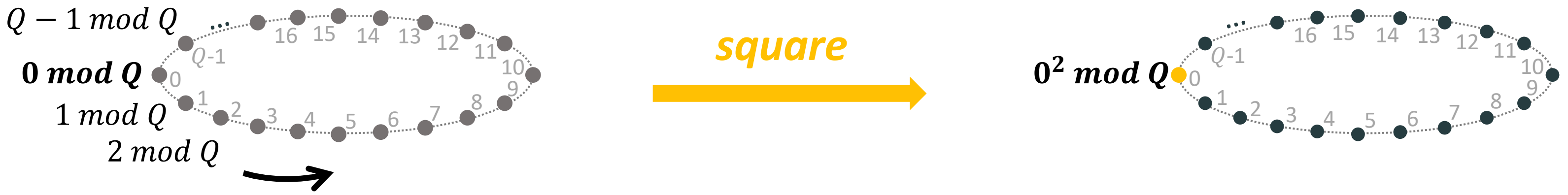
Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.



Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.

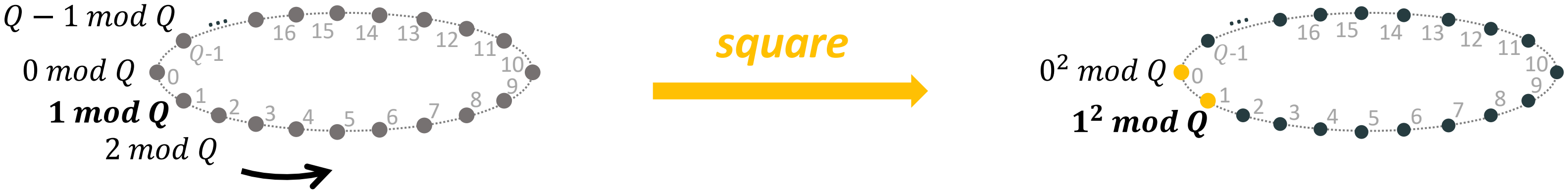


$x \bmod Q$

$x^2 \bmod Q$

Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.

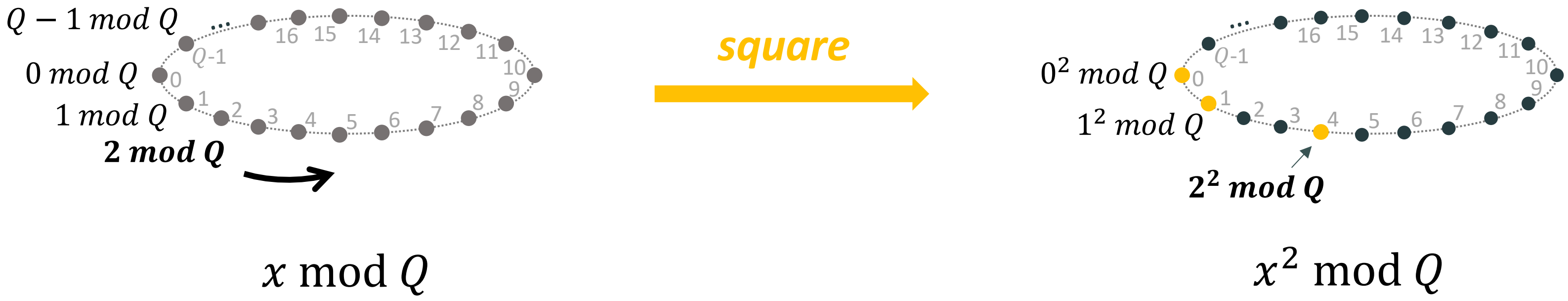


$x \bmod Q$

$x^2 \bmod Q$

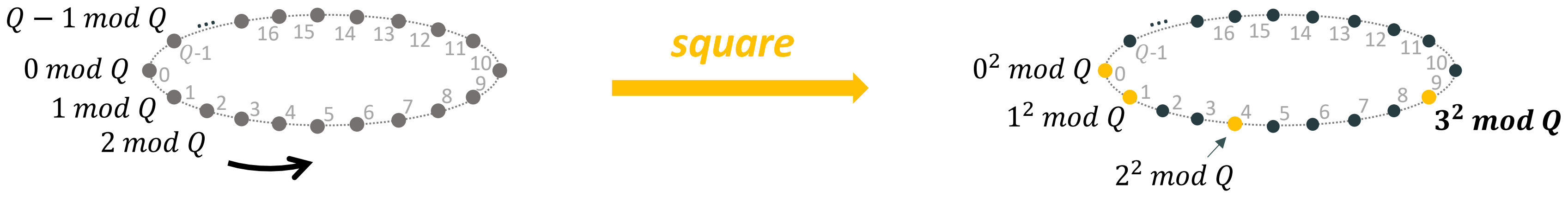
Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.



Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.

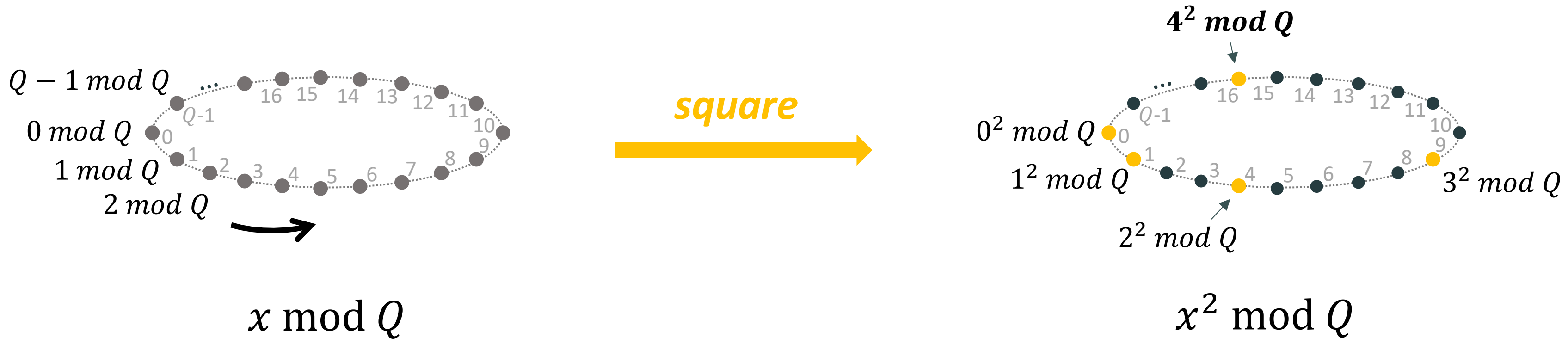


x mod Q

x^2 mod Q

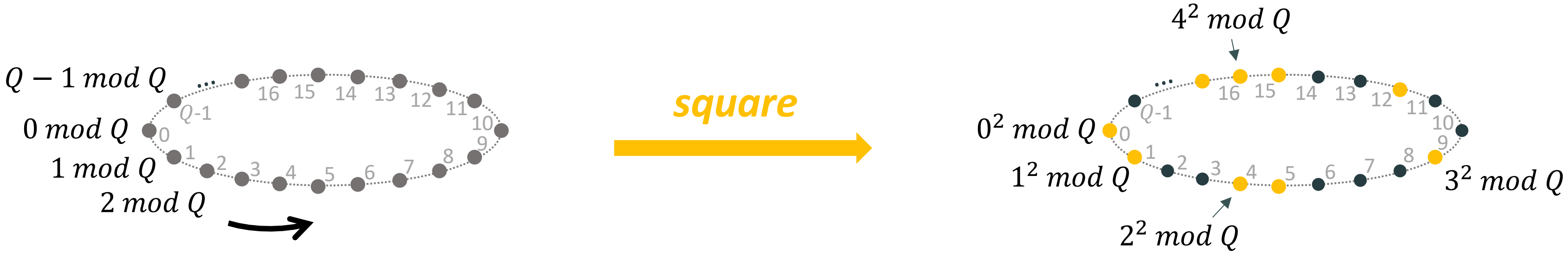
Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.



Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.

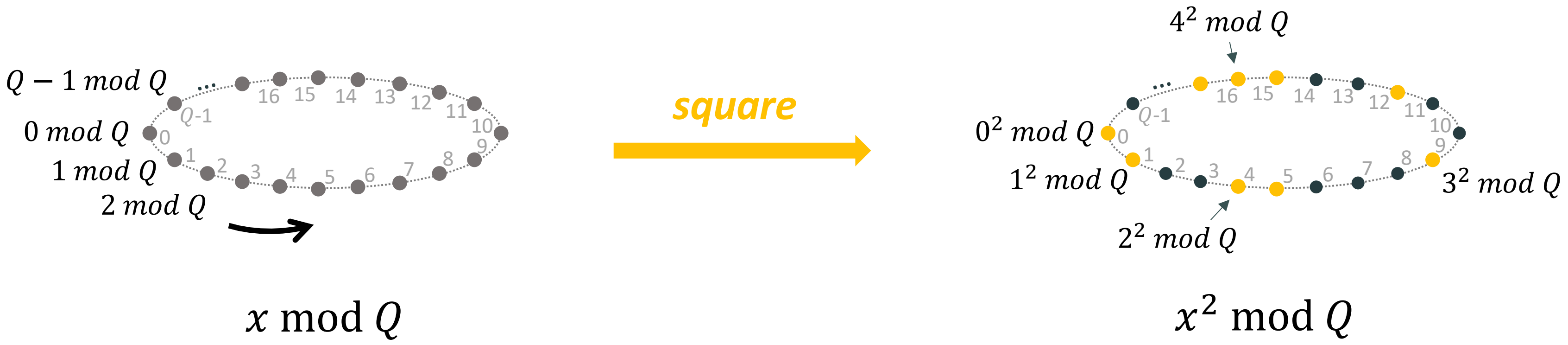


x mod Q

x^2 mod Q

Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.



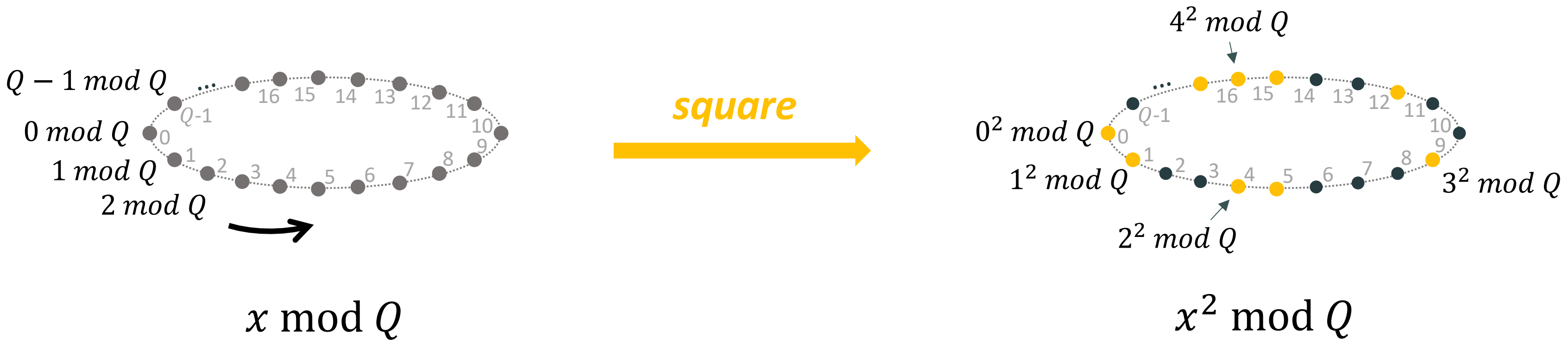
The color on each site is a flag for whether it is a quadratic residue.

● -> *yes, quadratic residue*

● -> *no, not quadratic residue*

Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.



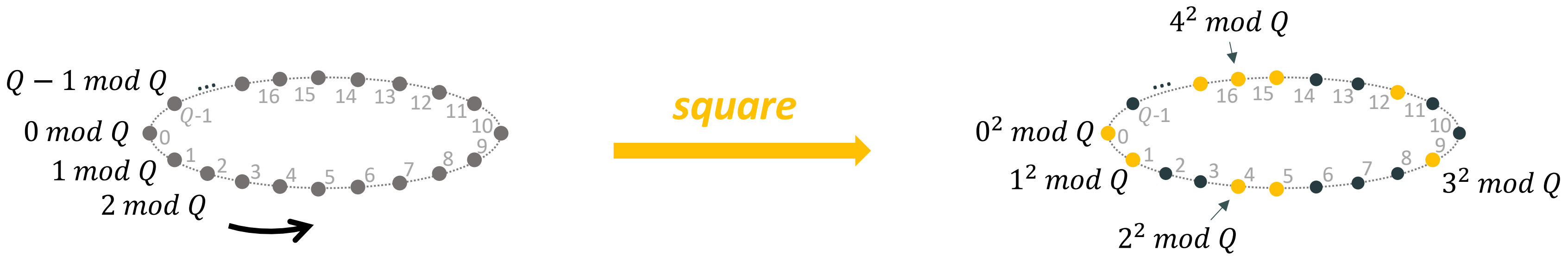
The Legendre Symbol

$$\left(\frac{a}{Q}\right) = \begin{cases} +1, & a \text{ is nonzero quadratic residue mod } Q \\ 0\ , & a \text{ is } 0 \text{ mod } Q \\ -1, & \text{otherwise} \end{cases}$$

a **is a quadratic residue mod** Q **if exists** x **such that** $x^2 \equiv a \pmod{Q}$

Tool: The Legendre Symbol

Consider the ring modulo Q, where Q is prime.



x mod Q

x^2 mod Q

The Legendre Symbol

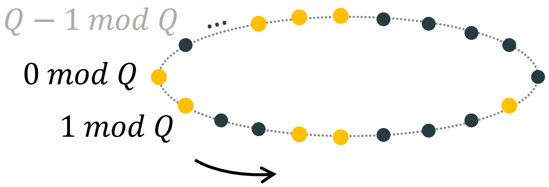
$$\left(\frac{a}{Q}\right) = \begin{cases} +1, & a \text{ is nonzero quadratic residue mod } Q \\ 0\ , & a \text{ is } 0 \bmod Q \\ -1, & \text{otherwise} \end{cases}$$

a **is a quadratic residue mod** Q **if exists** x **such that** $x^2 \equiv a \pmod{Q}$

Tools:

The Legendre Symbol

For prime Q, the Legendre Symbol $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



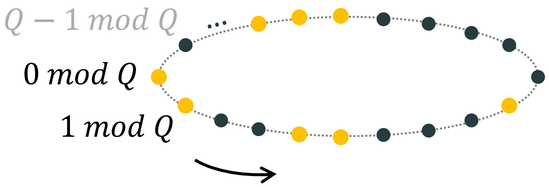
$$\left(\frac{x}{Q}\right) = \begin{cases} \text{🟡} & , \quad +1 \\ \text{●} & , \quad -1 \end{cases}$$

note: ignoring 0 mod Q

Tools:

The Legendre Symbol

For prime Q, the Legendre Symbol $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



$$\left(\frac{x}{Q}\right) = \begin{cases} 🟡 & , +1 \\ ● & , -1 \end{cases}$$

note: ignoring 0 mod Q

The Jacobi Symbol

The Jacobi Symbol $\left(\frac{x}{N}\right)$ generalizes the Legendre Symbol to composite moduli:

(non prime N) $N = P_1 P_2 \dots P_r$ → $\left(\frac{x}{N}\right) = \left(\frac{x}{P_1}\right)\left(\frac{x}{P_2}\right)\dots\left(\frac{x}{P_r}\right)$

This helpful function is the Jacobi symbol!

Cost of Shor's factoring algorithm

- Period of $f(x)$: $O(N)$
- Gate count to compute f: $O(n^2)$

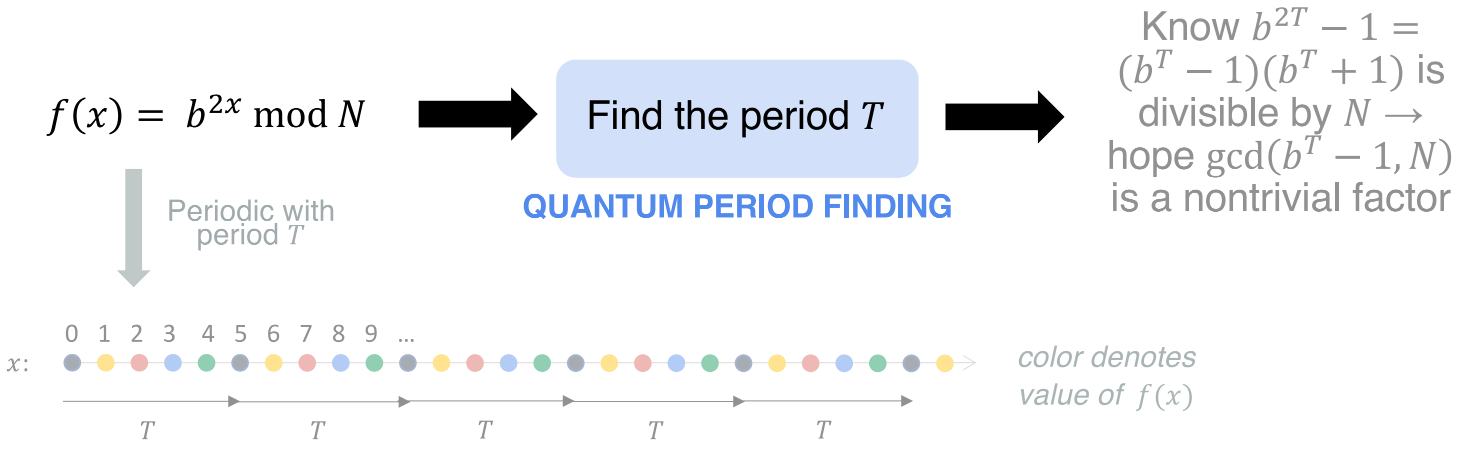
What if we could find a different function $j(x)$ which:

a) Also has periodicity that enables us to factor N;

b) Has a shorter period; and

c) Is easier to implement?

Peter Shor



$f(x) = b^{2x} \bmod N$ → Find the period T → Know $b^{2T} - 1 = (b^T - 1)(b^T + 1)$ is divisible by $N \rightarrow$ hope $\gcd(b^T - 1, N)$ is a nontrivial factor

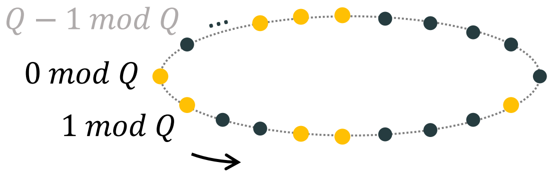
QUANTUM PERIOD FINDING

Periodic with period T

color denotes value of $f(x)$

Efficiency of Computing Jacobi

For prime Q, the Legendre Symbol $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



$\left(\frac{x}{Q}\right) =$ 🟡 , $+1$; ● , -1

note: ignoring $0 \bmod Q$

The Jacobi Symbol $\left(\frac{x}{N}\right)$ generalizes the Legendre Symbol to composite moduli:

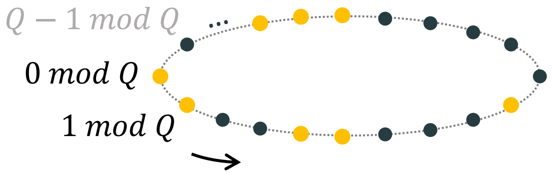
(non prime N) $N = P_1 P_2 \ldots P_r$ → $\left(\frac{x}{N}\right) = \left(\frac{x}{P_1}\right)\left(\frac{x}{P_2}\right) \cdots \left(\frac{x}{P_r}\right)$

Jacobi Symbol is Very Efficiently Computable: We can compute $\left(\frac{x}{N}\right)$ in time $\tilde{O}(\log \mathrm{N})$, *without* knowing the factorization of N. *Stay tuned!*

Li, Peng, Du, Suter, *Scientific Reports* (2012)

Jacobi Symbol Periodicity

For prime Q, the **Legendre Symbol** $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



$\left(\frac{x}{Q}\right) = \begin{cases} ● , & +1 \\ ● , & -1 \end{cases}$

note: ignoring $0 \bmod Q$

The **Jacobi Symbol** $\left(\frac{x}{N}\right)$ generalizes the Legendre Symbol to composite moduli:

(non prime N) $N = P_1 P_2 \ldots P_r \rightarrow \left(\frac{x}{N}\right) = \left(\frac{x}{P_1}\right)\left(\frac{x}{P_2}\right) \cdots \left(\frac{x}{P_r}\right)$

Jacobi Symbol on the ring mod N

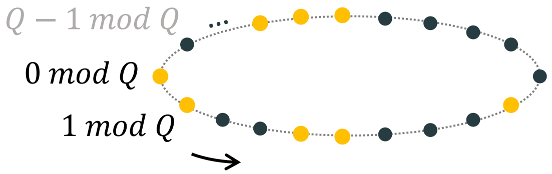
$j(x) = \left(\frac{x}{N}\right)$

N is not prime

Li, Peng, Du, Suter, *Scientific Reports* (2012)

Jacobi Symbol Periodicity

For prime Q, the Legendre Symbol $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



$$\left(\frac{x}{Q}\right) = \begin{cases} 🟡 \ , & +1 \\ ● \ , & -1 \end{cases}$$

note: ignoring $0 \bmod Q$

The Jacobi Symbol $\left(\frac{x}{N}\right)$ generalizes the Legendre Symbol to composite moduli:

(non prime N) $N = P_1 P_2 \ldots P_r$ → $\left(\frac{x}{N}\right) = \left(\frac{x}{P_1}\right)\left(\frac{x}{P_2}\right)\cdots\left(\frac{x}{P_r}\right)$

Jacobi Symbol on the ring mod N

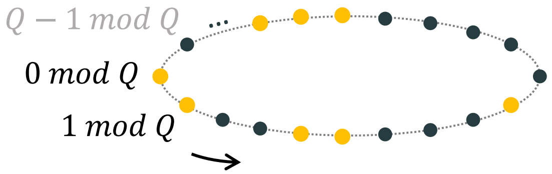
$$j(x) = \left(\frac{x}{N}\right) = \left(\frac{x}{P^2 Q}\right)$$

N is not prime

Li, Peng, Du, Suter, *Scientific Reports* (2012)

Jacobi Symbol Periodicity

For prime Q, the Legendre Symbol $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



$\left(\frac{x}{Q}\right) =$ { 🟡 , $+1$; ⚫ , -1 }

note: ignoring $0 \bmod Q$

The Jacobi Symbol $\left(\frac{x}{N}\right)$ generalizes the Legendre Symbol to composite moduli:

(non prime N) $N = P_1 P_2 \dots P_r$ → $\left(\frac{x}{N}\right) = \left(\frac{x}{P_1}\right)\left(\frac{x}{P_2}\right) \cdots \left(\frac{x}{P_r}\right)$

Jacobi Symbol on the ring mod N

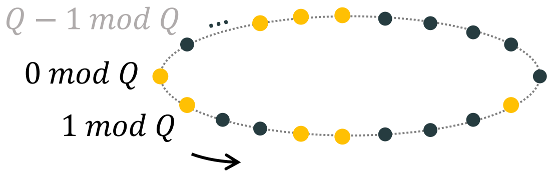
$$j(x) = \left(\frac{x}{N}\right) = \left(\frac{x}{P^2 Q}\right) = \left(\frac{x}{P}\right)\left(\frac{x}{P}\right)\left(\frac{x}{Q}\right)$$

N is not prime

Li, Peng, Du, Suter, *Scientific Reports* (2012)

Jacobi Symbol Periodicity

For prime Q, the Legendre Symbol $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



$$\left(\frac{x}{Q}\right) = \begin{cases} 🟡\ , & +1 \\ ●\ , & -1 \end{cases}$$

note: ignoring $0 \bmod Q$

The Jacobi Symbol $\left(\frac{x}{N}\right)$ generalizes the Legendre Symbol to composite moduli:

(non prime N) $N = P_1 P_2 \ldots P_r \;\rightarrow\; \left(\frac{x}{N}\right) = \left(\frac{x}{P_1}\right)\left(\frac{x}{P_2}\right)\cdots\left(\frac{x}{P_r}\right)$

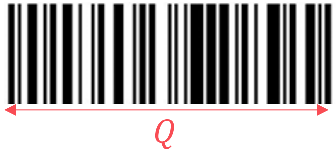
Jacobi Symbol on the ring mod N

N is not prime

$$j(x) = \left(\frac{x}{N}\right) = \left(\frac{x}{P^2 Q}\right) = \left(\frac{x}{P}\right)\left(\frac{x}{P}\right)\left(\frac{x}{Q}\right)$$

$$= \left(\frac{x}{Q}\right)$$

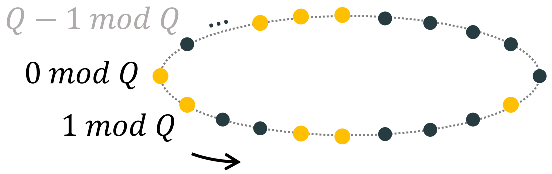
Recall: Jacobi symbol is +1/-1



Li, Peng, Du, Suter, *Scientific Reports* (2012)

Jacobi Symbol Periodicity

For prime Q, the Legendre Symbol $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



$$\left(\frac{x}{Q}\right) = \begin{cases} 🟡 , & +1 \\ ⚫ , & -1 \end{cases}$$

note: ignoring $0 \bmod Q$

The Jacobi Symbol $\left(\frac{x}{N}\right)$ generalizes the Legendre Symbol to composite moduli:

(non prime N) $N = P_1 P_2 \dots P_r$ → $\left(\frac{x}{N}\right) = \left(\frac{x}{P_1}\right)\left(\frac{x}{P_2}\right) \cdots \left(\frac{x}{P_r}\right)$

$$\sum_{x=0}^{N-1} |\, x >|\, j(x) > \quad =$$

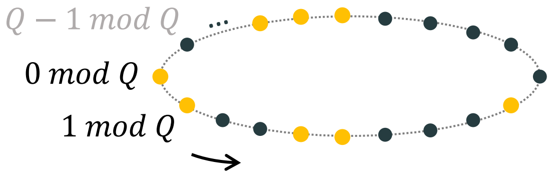
⬇

$$\left(\frac{x}{N}\right) = \left(\frac{x}{Q}\right)$$

Li, Peng, Du, Suter, *Scientific Reports* (2012)

Jacobi Symbol Periodicity

For prime Q, the Legendre Symbol $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



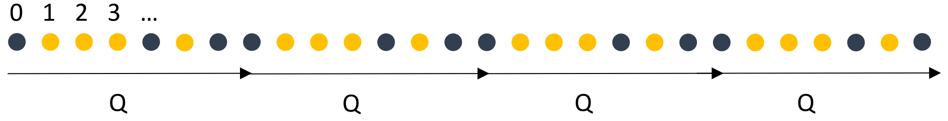
$$\left(\frac{x}{Q}\right) = \begin{cases} 🟡 & , \; +1 \\ ● & , \; -1 \end{cases}$$

note: ignoring $0 \bmod Q$

The Jacobi Symbol $\left(\frac{x}{N}\right)$ generalizes the Legendre Symbol to composite moduli:

(non prime N) $N = P_1 P_2 \ldots P_r$ → $\left(\frac{x}{N}\right) = \left(\frac{x}{P_1}\right)\left(\frac{x}{P_2}\right) \cdots \left(\frac{x}{P_r}\right)$

$$\sum_{x=0}^{N-1} |\, x > |\, j(x) > \;\; =$$



color denotes value:

🟡 -> $j(x) = +1$

● -> $j(x) = -1$

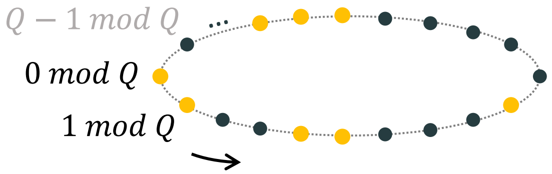
⬇

$$\left(\frac{x}{N}\right) = \left(\frac{x}{Q}\right)$$

Li, Peng, Du, Suter, *Scientific Reports* (2012)

Jacobi Symbol Periodicity

For prime Q, the Legendre Symbol $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



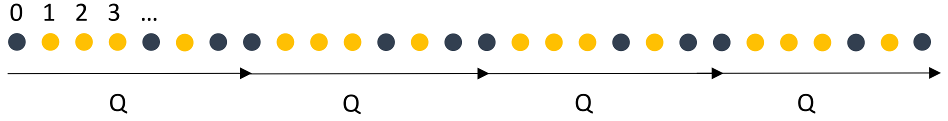
$$\left(\frac{x}{Q}\right) = \begin{cases} 🟡 \ , & +1 \\ ⚫ \ , & -1 \end{cases}$$

note: ignoring 0 mod Q

The Jacobi Symbol $\left(\frac{x}{N}\right)$ generalizes the Legendre Symbol to composite moduli:

(non prime N) $N = P_1 P_2 \ldots P_r$ → $\left(\frac{x}{N}\right) = \left(\frac{x}{P_1}\right)\left(\frac{x}{P_2}\right)\cdots\left(\frac{x}{P_r}\right)$

$$\sum_{x=0}^{N-1} |\, x > |\, j(x) > \; =$$

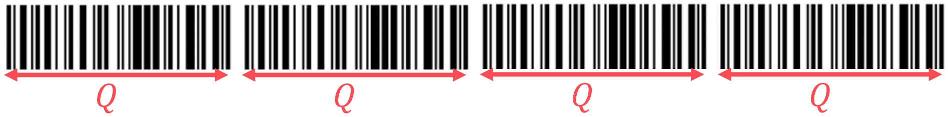


color denotes value:

🟡 -> $j(x) = +1$

⚫ -> $j(x) = -1$

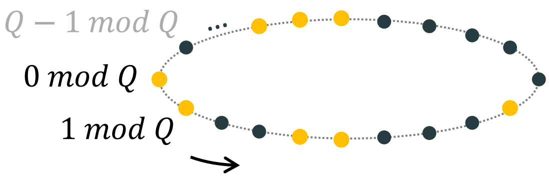
$$\left(\frac{x}{N}\right) = \left(\frac{x}{Q}\right)$$



Now we can find Q with quantum period finding!

Factoring with the Jacobi Symbol

For prime Q, the Legendre Symbol $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



$$\left(\frac{x}{Q}\right) = \begin{cases} 🟡 & , +1 \\ ● & , -1 \end{cases}$$

note: ignoring $0 \bmod Q$

The Jacobi Symbol $\left(\frac{x}{N}\right)$ generalizes the Legendre Symbol to composite moduli:

(non prime N) $N = P_1 P_2 \dots P_r$ → $\left(\frac{x}{N}\right) = \left(\frac{x}{P_1}\right)\left(\frac{x}{P_2}\right)\cdots\left(\frac{x}{P_r}\right)$

Cost of this Jacobi factoring algorithm (“LPDS”)

- Period of $j(x)$ is Q

Li

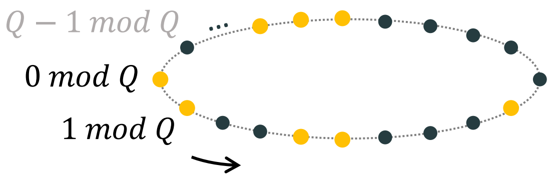
Peng

Du

Suter

Factoring with the Jacobi Symbol

For prime Q, the Legendre Symbol $\left(\frac{x}{Q}\right)$ flags whether x is a quadratic residue mod Q.



$$\left(\frac{x}{Q}\right) = \begin{cases} \text{🟡} \; , & +1 \\ \text{●} \; , & -1 \end{cases}$$

note: ignoring $0 \bmod Q$

The Jacobi Symbol $\left(\frac{x}{N}\right)$ generalizes the Legendre Symbol to composite moduli:

(non prime N) $N = P_1 P_2 \dots P_r$ → $\left(\frac{x}{N}\right) = \left(\frac{x}{P_1}\right)\left(\frac{x}{P_2}\right) \cdots \left(\frac{x}{P_r}\right)$

Cost of this Jacobi factoring algorithm (“LPDS”)

- Period of $j(x)$ is Q
- **Gates/space/depth to compute Jacobi: $\tilde{O}(\log N)$**

Li

Peng

Du

Suter

Outline

$n = \log N$

$m = \log Q$

1

Shor's algorithm can factor *any* n-bit number using $O(n^2)$ gates, $O(n)$ qubits

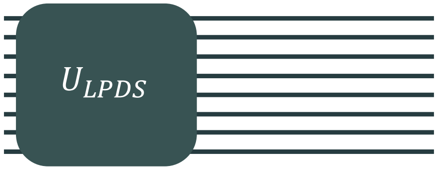
$N = P * Q$

Shor

2a

Jacobi algorithm can factor *some* n-bit numbers using only $O(n)$ gates

$N = P^2 * Q$



Li

Peng

Du

Suter

Outline

$n = \log N$

$m = \log Q$

1

Shor's algorithm can factor *any* n-bit number using $O(n^2)$ gates, $O(n)$ qubits

$$N = P * Q$$

Shor

2b

Jacobi algorithm can factor *some* n-bit numbers using only $O(n)$ gates, $\boxed{O(m)\text{ qubits}}$

$$N = P^2 * Q$$

Li

Peng

Du

Suter

Kahanamoku-Meyer

SR

Vaikuntanathan

KVK

Idea 1: Shortening the Superposition

Period of the Jacobi symbol is Q rather than $O(N)$ as in Shor
$\longrightarrow$ the “bare minimum” qubit count is now just $O(\log Q)$!

Remaining challenge:

Can we actually compute the Jacobi symbol using this bare minimum number of qubits? Not even enough to write down N!

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

Goal: Compute a function with **small quantum** input and **big classical** input.

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

Goal: Compute a function with **small quantum** input and **big classical** input.

quantum

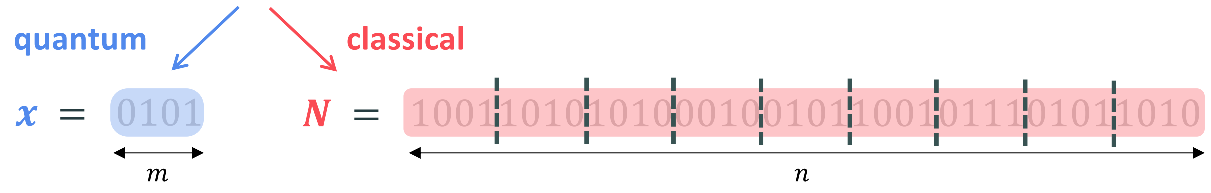
$$x = \underbrace{\boxed{0101}}_{m}$$

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

Goal: Compute a function with **small quantum** input and **big classical** input.



quantum

classical

$x = 0101$

$N = 1001\,1010\,1010\,0010\,0101\,1001\,0111\,0101\,1010$

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

Goal: Compute a function with **small quantum** input and **big classical** input.

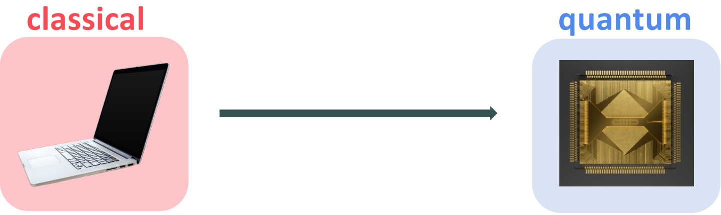
quantum

classical

$$x = \underbrace{0101}_{m} \qquad N = 1001\,1010\,1010\,0010\,0101\,1001\,0111\,0101\,\underbrace{1010}_{m}$$

Solution: “streaming”

Feeds m bits of N at a time
to the quantum computer



Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

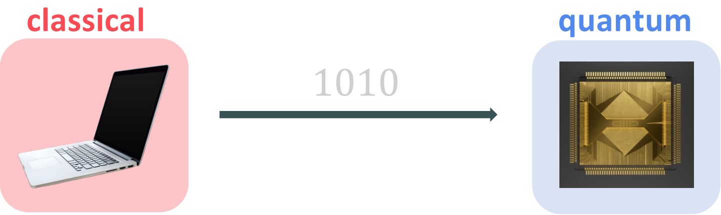
Goal: Compute a function with **small quantum** input and **big classical** input.

quantum classical

$$x = \underbrace{0101}_{m} \qquad N = 1001\,1010\,1010\,0010\,0101\,1001\,0111\,0101\,\underbrace{1010}_{m}$$

Solution: “streaming”

Feeds m bits of N at a time to the quantum computer



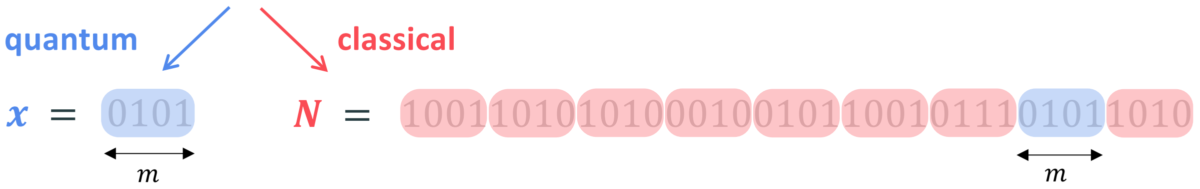
Do some arithmetic that frees up m qubits!

Idea 2: “Quantum Streaming”

30,000 Foot View

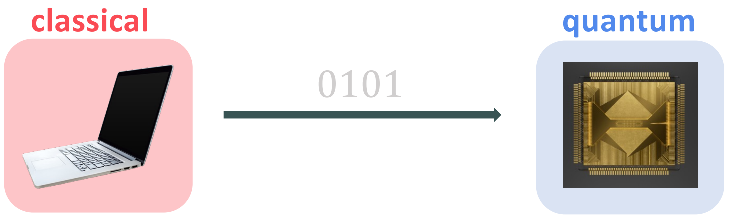
$n = \log N$
$m = \log Q$

Goal: Compute a function with **small quantum** input and **big classical** input.



Solution: “streaming”

Feeds m bits of N at a time to the quantum computer



Do some arithmetic that frees up m qubits!

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

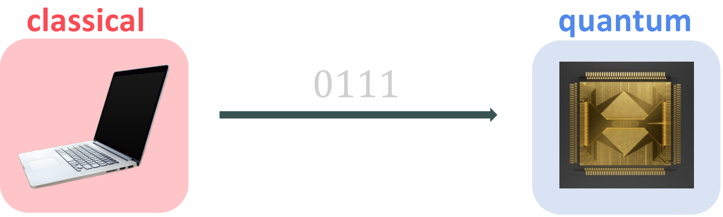
Goal: Compute a function with **small quantum** input and **big classical** input.

quantum   classical

$x = \underbrace{0101}_{m}$ $N = 100110101010001001011001\underbrace{0111}_{m}01011010$

Solution: “streaming”

Feeds m bits of N at a time to the quantum computer



Do some arithmetic that frees up m qubits!

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

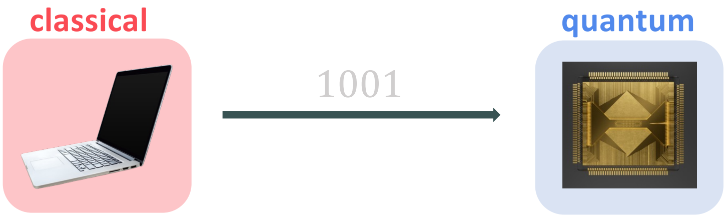
Goal: Compute a function with **small quantum** input and **big classical** input.

quantum classical

$$x = \underset{\overset{\longleftrightarrow}{m}}{0101} \qquad N = 1001\,1010\,1010\,0010\,0101\,\underset{\overset{\longleftrightarrow}{m}}{1001}\,0111\,0101\,1010$$

Solution: “streaming”

Feeds m bits of N at a time to the quantum computer



Do some arithmetic that frees up m qubits!

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

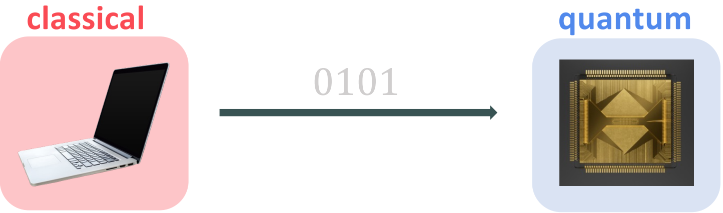
Goal: Compute a function with **small quantum** input and **big classical** input.

quantum classical

$x = \underbrace{0101}_{m}$ $N = 1001\,1010\,1010\,0010\,\underbrace{0101}_{m}\,1001\,0111\,0101\,1010$

Solution: “streaming”

Feeds m bits of N at a time to the quantum computer



Do some arithmetic that frees up m qubits!

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

Goal: Compute a function with **small quantum** input and **big classical** input.

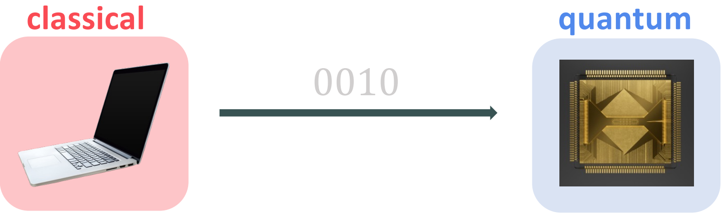
quantum

classical

$x = \underbrace{0101}_{m}$ $\quad N = 1001101010100\underbrace{0010}_{m}0101100101110101 1010$

Solution: “streaming”

Feeds m bits of N at a time to the quantum computer



Do some arithmetic that frees up m qubits!

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

Goal: Compute a function with **small quantum** input and **big classical** input.

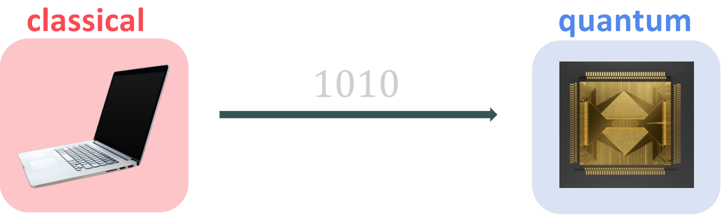
quantum classical

$x = 0101$ $\qquad N = 100110101010001001011001011101011010$

m $\qquad m$

Solution: “streaming”

Feeds m bits of N at a time to the quantum computer



Do some arithmetic that frees up m qubits!

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

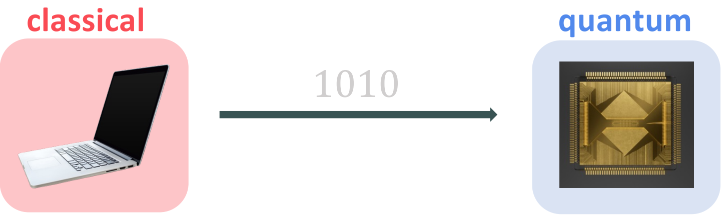
Goal: Compute a function with **small quantum** input and **big classical** input.

quantum   classical

$x = \underbrace{0101}_{m}$ $N = 1001\underbrace{1010}_{m}1010\,0010\,0101\,1001\,0111\,0101\,1010$

Solution: “streaming”

Feeds m bits of N at a time to the quantum computer



Do some arithmetic that frees up m qubits!

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

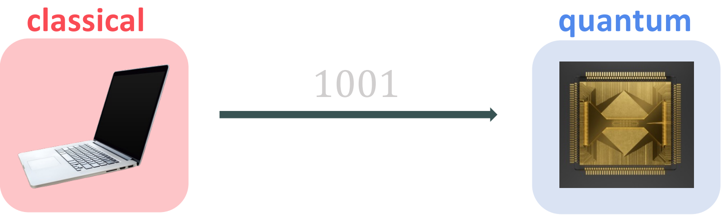
Goal: Compute a function with **small quantum** input and **big classical** input.

quantum ↙ ↘ classical

$$x = \underbrace{0101}_{m} \qquad N = \underbrace{1001}_{m}101010100010010110010111010110 10$$

Solution: “streaming”

Feeds m bits of N at a time to the quantum computer



Do some arithmetic that frees up m qubits!

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

Goal: Compute a function with **small quantum** input and **big classical** input.

quantum classical

$$x = \underbrace{0101}_{m} \qquad N = \underbrace{1001}_{m}1010\,1010\,0010\,0101\,1001\,0111\,0101\,1010$$

Solution: “streaming”

Feeds m bits of N at a time to the quantum computer

classical
quantum

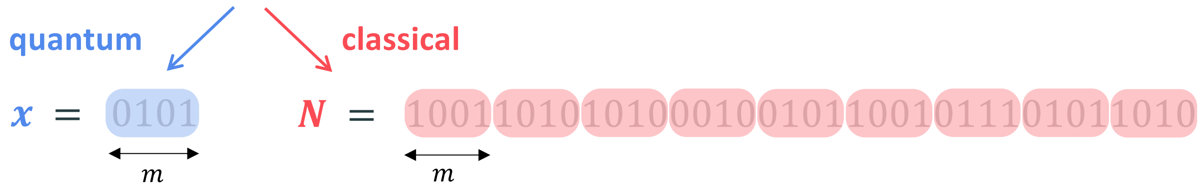
While it processed $O(n)$ bits, the quantum computer only needed $O(m)$ space!

Idea 2: “Quantum Streaming”

30,000 Foot View

$n = \log N$
$m = \log Q$

Goal: Compute a function with **small quantum** input and **big classical** input.



$x = \underbrace{0101}_{m}$ $\quad N = \underbrace{1001}_{m}10101010001001011001011101011010$

But quantum streaming is just a hope.

Why does the Jacobi symbol lend itself to streaming?

Aside: Computing Jacobi

Jacobi Symbol: $\left(\frac{a}{b}\right)$

Properties

(1) **periodicity** : $\left(\frac{a}{b}\right) = \left(\frac{a \bmod b}{b}\right)$

Aside: Computing Jacobi

Jacobi Symbol: $\left(\frac{a}{b}\right)$

Properties

(1) **periodicity** : $\left(\frac{a}{b}\right) = \left(\frac{a \bmod b}{b}\right)$

(2) **reciprocity** : $\left(\frac{a}{b}\right) = (-1)^{f(a,b)} \left(\frac{b}{a}\right)$

Aside: Computing Jacobi

Euclidean Algorithm

Extended Euclidean algorithm can compute *any* function with these two properties!

Jacobi Symbol: $\left(\frac{a}{b}\right)$

Properties

(1) **periodicity** : $\left(\frac{a}{b}\right) = \left(\frac{\boldsymbol{a} \bmod \boldsymbol{b}}{\boldsymbol{b}}\right)$

(2) **reciprocity** : $\left(\frac{a}{b}\right) = (-1)^{f(\boldsymbol{a},\boldsymbol{b})} \left(\frac{b}{a}\right)$

Greatest Common Divisor: $GCD(a, b)$

Properties

(1) **periodicity**: $GCD(\boldsymbol{a}, \boldsymbol{b}) = GCD(\boldsymbol{a} \bmod \boldsymbol{b}, \boldsymbol{b})$

(2) **reciprocity** : $GCD(\boldsymbol{a}, \boldsymbol{b}) = GCD(\boldsymbol{b}, \boldsymbol{a})$

Aside: Computing Jacobi

Euclidean Algorithm

Extended Euclidean algorithm can compute *any* function with these two properties!

Jacobi Symbol: $\left(\frac{a}{b}\right)$

Properties

(1) **periodicity** : $\left(\frac{a}{b}\right) = \left(\frac{a \bmod b}{b}\right)$

(2) **reciprocity** : $\left(\frac{a}{b}\right) = (-1)^{f(a,b)} \left(\frac{b}{a}\right)$

Euclidean Algorithm for $\left(\frac{a}{b}\right)$

If $a < b$: **swap** $a \leftrightarrow b$

Else : **take mod** $a \leftarrow a \bmod b$

Streaming for Jacobi

Example

Euclidean Algorithm for $\left(\frac{a}{b}\right)$

```
If $\boldsymbol{a} < \boldsymbol{b}$ :  swap  $\boldsymbol{a} \leftrightarrow \boldsymbol{b}$
Else :  take mod  $\boldsymbol{a} \leftarrow \boldsymbol{a} \bmod \boldsymbol{b}$
```

Streaming for Jacobi

$n = \log(N)$
$m = \log(Q)$

Example

Euclidean Algorithm for $\left(\frac{a}{b}\right)$

If $\boldsymbol{a} < \boldsymbol{b}$: **swap** $\boldsymbol{a} \leftrightarrow \boldsymbol{b}$
Else : **take mod** $\boldsymbol{a} \leftarrow \boldsymbol{a} \bmod \boldsymbol{b}$

$$\left(\frac{\mathbf{x}}{\boldsymbol{N}}\right) = \left(\frac{\overset{\overset{\boldsymbol{m}}{\longleftrightarrow}}{0101}}{\underset{\boldsymbol{n}}{\underleftrightarrow{10011010101010001001011001011101011010}}}\right)$$

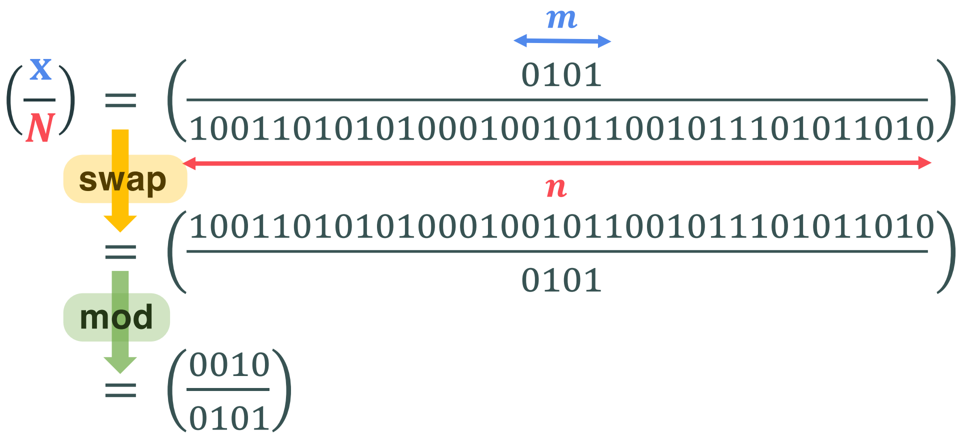
Streaming for Jacobi

$n = \log(N)$
$m = \log(Q)$

Example

Euclidean Algorithm for $\left(\frac{a}{b}\right)$

If $\boldsymbol{a} < \boldsymbol{b}$: **swap** $\boldsymbol{a} \leftrightarrow \boldsymbol{b}$
Else : **take mod** $\boldsymbol{a} \leftarrow \boldsymbol{a} \bmod \boldsymbol{b}$



$$\left(\frac{\mathbf{X}}{\boldsymbol{N}}\right) = \left(\frac{\overset{\boldsymbol{m}}{\longleftrightarrow}\\0101}{\underset{\boldsymbol{n}}{\underleftrightarrow{100110101010001001011001011101011010}}}\right)$$

$$\xrightarrow{\text{swap}} = \left(\frac{100110101010001001011001011101011010}{0101}\right)$$

$$\xrightarrow{\text{mod}} = \left(\frac{0010}{0101}\right)$$

Streaming for Jacobi

$n = \log(N)$
$m = \log(Q)$

Example

Euclidean Algorithm for $\left(\frac{a}{b}\right)$

If $\boldsymbol{a} < \boldsymbol{b}$: **swap** $\boldsymbol{a} \leftrightarrow \boldsymbol{b}$
Else : **take mod** $\boldsymbol{a} \leftarrow \boldsymbol{a} \bmod \boldsymbol{b}$

$$\left(\frac{x}{N}\right) = \left(\frac{\overset{\overset{m}{\longleftrightarrow}}{0101}}{\underset{\underset{n}{\longleftrightarrow}}{100110101010001001011001011101011010}}\right)$$

$$\Big\downarrow \text{swap}$$

$$= \left(\frac{100110101010001001011001011101011010}{0101}\right)$$

$$\Big\downarrow \text{mod}$$

$$= \left(\frac{0010}{0101}\right)$$

Crucial observation:

- Because $N > x$, we always compute $N \; mod \; x$.

Streaming for Jacobi

$n = \log(N)$

$m = \log(Q)$

Example

Euclidean Algorithm for $\left(\frac{a}{b}\right)$

If $a < b$: swap $a \leftrightarrow b$

Else : take mod $a \leftarrow a \bmod b$

$$\left(\frac{x}{N}\right) = \left(\frac{\overset{m}{\overleftrightarrow{0101}}}{\underset{n}{\underleftrightarrow{100110101010001001011001011101011010}}}\right)$$

$$\Big\downarrow \text{swap}$$

$$= \left(\frac{100110101010001001011001011101011010}{0101}\right)$$

$$\Big\downarrow \text{mod}$$

$$= \underset{m}{\underleftrightarrow{\left(\frac{0010}{0101}\right)}}$$

$$\dots$$

Crucial observation:

- Because $N > x$, we always compute $N \bmod x$.
- Then rest of the computation is always $\tilde{O}(m)$.

Streaming for Jacobi

$n = \log(N)$
$m = \log(Q)$

Example

Euclidean Algorithm for $\left(\frac{a}{b}\right)$

If $a < b$: **swap** $a \leftrightarrow b$
Else : **take mod** $a \leftarrow a \bmod b$

$$\left(\frac{x}{N}\right) = \left(\frac{\overset{m}{\overleftrightarrow{0101}}}{\underset{n}{\underleftrightarrow{100110101010001001011001011101011010}}}\right)$$

swap

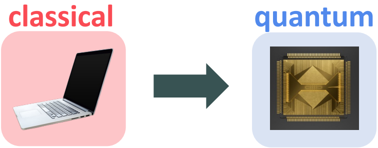
$$= \left(\frac{100110101010001001011001011101011010}{0101}\right)$$

mod

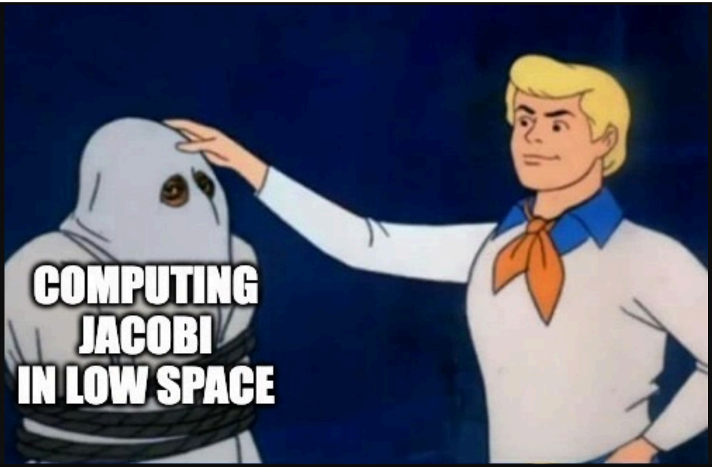
$$= \underset{m}{\underleftrightarrow{\left(\frac{0010}{0101}\right)}}$$

…

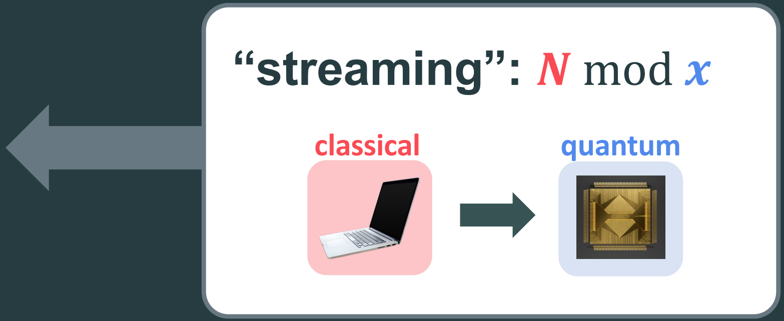
"streaming": $N \bmod x$



Streaming for Jacobi



Streaming for Jacobi



“streaming”: $N \bmod x$

Costs of Our Algorithm

$\boldsymbol{n} = \log \boldsymbol{N}$
$\boldsymbol{m} = \log \boldsymbol{Q}$

Main Result: **Circuit for factoring $\boldsymbol{N = P^2 * Q}$**

Gates $= \widetilde{\boldsymbol{O}}(\boldsymbol{n})$

Depth $= \widetilde{\boldsymbol{O}}(\boldsymbol{n/m + m})$

Space $= \widetilde{\boldsymbol{O}}(\boldsymbol{m})$

Rough workload:

1. "Streaming": n/m multiplications of m-bit numbers
2. Jacobi symbol with two m-bit inputs

Costs of Our Algorithm

$\boldsymbol{n = \log N}$
$\boldsymbol{m = \log Q}$

Main Result: **Circuit for factoring $\boldsymbol{N = P^2 * Q}$**

Gates $= \boldsymbol{\tilde{O}(n)}$

Depth $= \boldsymbol{\tilde{O}(n/m + m)} \;\; = \boldsymbol{\tilde{O}(n^{2/3})}$

Space $= \boldsymbol{\tilde{O}(m)} \;\; = \boldsymbol{\tilde{O}(n^{2/3})}$

Rough workload:

1. "Streaming": n/m multiplications of m-bit numbers
2. Jacobi symbol with two m-bit inputs

Recall: can set $m = \log Q$ as low as $\tilde{O}(n^{2/3})$ while preserving the classical cost of factoring

This could be a great candidate for an efficiently-verifiable proof of quantumness!

Conclusion

Compact quantum circuit for classically hard factoring instance

U_{KRVV}

$N = P^2 Q$ with Q small

However… This is not all numbers!

For cryptographic relevance, we want $N = PQ$ and both P and Q to be large like N.

Stay tuned for the next talk!

Thank you!

arXiv:2412.12558

Greg Kahanamoku-Meyer

Seyoon Ragavan

Vinod Vaikuntanathan

Katherine Van Kirk

HARVARD UNIVERSITY

Hertz Foundation