The Jacobi Factoring Circuit

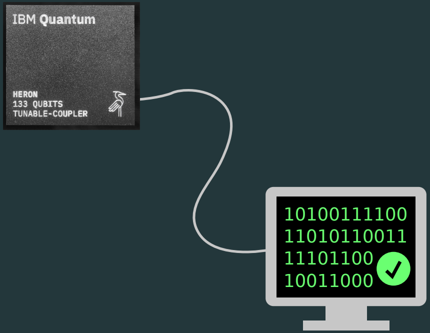
Classically-hard factoring in sublinear quantum space and depth

Gregory D. Kahanamoku-Meyer*, Seyoon Ragavan*, Vinod Vaikuntanathan*, Katherine van Kirk†

*MIT, †Harvard

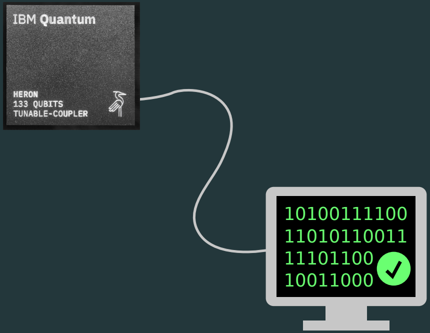
November 19, 2024

Verifiable quantum advantage



How can a single **black-box** device prove its quantum capability to a skeptical **classical** verifier?

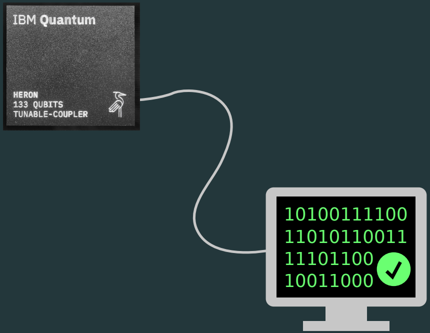
Verifiable quantum advantage



Ideal protocol:

How can a single **black-box** device prove its quantum capability to a skeptical **classical** verifier?

Verifiable quantum advantage

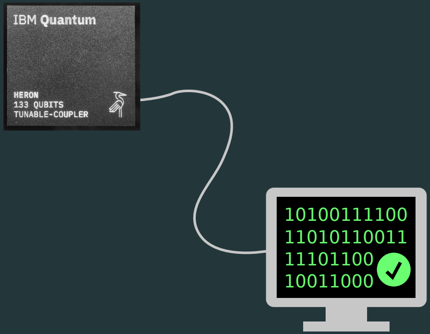


Ideal protocol:

- **Provably classically hard**, reducible to an *established* problem

How can a single **black-box** device prove its quantum capability to a skeptical **classical** verifier?

Verifiable quantum advantage

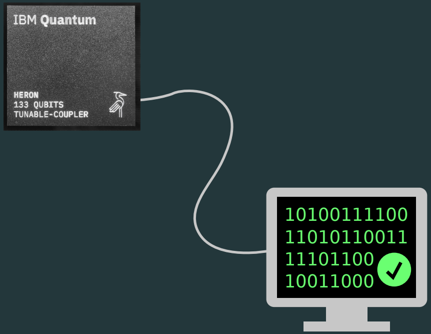


Ideal protocol:

- **Provably classically hard**, reducible to an *established* problem
- **Polynomial-time** classical verification

How can a single **black-box** device prove its quantum capability to a skeptical **classical** verifier?

Verifiable quantum advantage



Ideal protocol:

- **Provably classically hard**, reducible to an *established* problem
- **Polynomial-time** classical verification
- **Small circuits** in terms of qubits, gates, and depth

How can a single **black-box** device prove its quantum capability to a skeptical **classical** verifier?

Verifiable quantum advantage via factoring

Algorithms for Quantum Computation:
Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

Protocol: Pick primes p, q, ask the quantum device to factor n-bit $N = pq$.

Verifiable quantum advantage via factoring

Algorithms for Quantum Computation: Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

Protocol: Pick primes p, q, ask the quantum device to factor n-bit $N = pq$.

Gates: $\widetilde{\mathcal{O}}(n^2)$ Depth: $\widetilde{\mathcal{O}}(n)$ Qubits: $\widetilde{\mathcal{O}}(n)$

Verifiable quantum advantage via factoring

An Efficient Quantum Factoring Algorithm

Oded Regev*

Protocol: Pick primes p, q, ask the quantum device to factor n-bit $N = pq$.

Gates: $\widetilde{\mathcal{O}}(n^{3/2})$ **Depth:** $\widetilde{\mathcal{O}}(n^{1/2})$ **Qubits:** $\widetilde{\mathcal{O}}(n)^*$

* with the optimizations of Ragavan and Vaikuntanathan [arXiv:2310.00899]

Verifiable quantum advantage via factoring

Article | Open access | Published: 01 August 2022

Classically verifiable quantum advantage from a computational Bell test

Gregory D. Kahanamoku-Meyer, Soonwon Choi, Umesh V. Vazirani & Norman Y. Yao

Protocol:

3-round interactive protocol; quantum device evaluates $x^2 \bmod N$ for n-bit $N = pq$

Gates: $\widetilde{\mathcal{O}}(n)$ **Depth:** $\mathcal{O}(\mathrm{polylog}\, n)$ **Qubits:** $\widetilde{\mathcal{O}}(n)$

Verifiable quantum advantage via factoring

Article | Open access | Published: 01 August 2022

Classically verifiable quantum advantage from a computational Bell test

Gregory D. Kahanamoku-Meyer, Soonwon Choi, Umesh V. Vazirani & Norman Y. Yao

Protocol:

3-round interactive protocol; quantum device evaluates $x^2 \bmod N$ for n-bit $N = pq$

Gates: $\widetilde{\mathcal{O}}(n)$ **Depth:** $\mathcal{O}(\mathrm{polylog}\, n)$ **Qubits:** $\widetilde{\mathcal{O}}(n)$

… note it doesn't actually factor the number!

Verifiable quantum advantage via factoring

Algorithm	Gates	Depth	Qubits
Shor	$\widetilde{\mathcal{O}}(n^2)$	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n)$
Regev + RV23	$\widetilde{\mathcal{O}}(n^{3/2})$	$\widetilde{\mathcal{O}}(n^{1/2})$	$\widetilde{\mathcal{O}}(n)$
$x^2 \bmod N$	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n^0)$	$\widetilde{\mathcal{O}}(n)$

All algorithms implemented with fast, low-depth multipliers.
Tildes indicate omitted polylog factors.

Verifiable quantum advantage via factoring

Algorithm	Gates	Depth	Qubits
Shor	$\widetilde{\mathcal{O}}(n^2)$	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n)$
Regev + RV23	$\widetilde{\mathcal{O}}(n^{3/2})$	$\widetilde{\mathcal{O}}(n^{1/2})$	$\widetilde{\mathcal{O}}(n)$
$x^2 \bmod N$	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n^0)$	$\widetilde{\mathcal{O}}(n)$

All algorithms implemented with fast, low-depth multipliers.
Tildes indicate omitted polylog factors.

"Factoring numbers of practical significance requires far more qubits than available in the near future." –Wikipedia: Shor's algorithm

Verifiable quantum advantage via factoring

Algorithm	Gates	Depth	Qubits
Shor	$\widetilde{\mathcal{O}}(n^2)$	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n)$
Regev + RV23	$\widetilde{\mathcal{O}}(n^{3/2})$	$\widetilde{\mathcal{O}}(n^{1/2})$	$\widetilde{\mathcal{O}}(n)$
$x^2 \bmod N$	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n^0)$	$\widetilde{\mathcal{O}}(n)$

All algorithms implemented with fast, low-depth multipliers.
Tildes indicate omitted polylog factors.

"Factoring numbers of practical significance requires far more qubits than available in the near future." –Wikipedia: Shor's algorithm

"Cool but that's still too many qubits" –every experimentalist when I talk about $x^2 \bmod N$

Verifiable quantum advantage via factoring

For n-bit numbers of the form $N = pq$:

Algorithm	Gates	Depth	Qubits
Shor	$\widetilde{\mathcal{O}}(n^2)$	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n)$
Regev + RV23	$\widetilde{\mathcal{O}}(n^{3/2})$	$\widetilde{\mathcal{O}}(n^{1/2})$	$\widetilde{\mathcal{O}}(n)$
$x^2 \bmod N$	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n^0)$	$\widetilde{\mathcal{O}}(n)$

For n-bit numbers of the form $N = p^2 q$, with $q < 2^m$:

Algorithm	Gates	Depth	Qubits
This work	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n/m + m)$	$\widetilde{\mathcal{O}}(m)$

Verifiable quantum advantage via factoring

For n-bit numbers of the form $N = pq$:

Algorithm	Gates	Depth	Qubits
Shor	$\widetilde{\mathcal{O}}(n^2)$	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n)$
Regev + RV23	$\widetilde{\mathcal{O}}(n^{3/2})$	$\widetilde{\mathcal{O}}(n^{1/2})$	$\widetilde{\mathcal{O}}(n)$
$x^2 \bmod N$	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n^0)$	$\widetilde{\mathcal{O}}(n)$

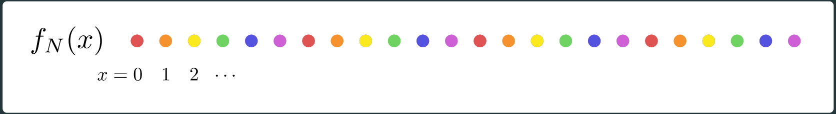
For n-bit numbers of the form $N = p^2 q$, with $q < 2^m$:

Algorithm	Gates	Depth	Qubits
This work	$\widetilde{\mathcal{O}}(n)$	$\widetilde{\mathcal{O}}(n/m + m)$	$\widetilde{\mathcal{O}}(m)$

Space and depth proportional to the **length of the factor!**

Why do we need $\mathcal{O}(n)$ qubits?

Find some $f_N(x)$ w/ period *P*, where *P* can be used to find factors



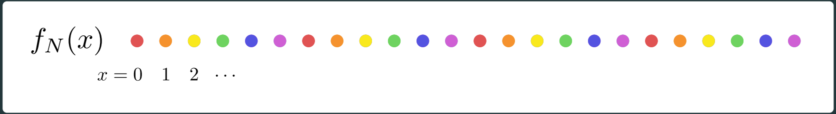
Why do we need $\mathcal{O}(n)$ qubits?

Find some $f_N(x)$ w/ period *P*, where *P* can be used to find factors

→

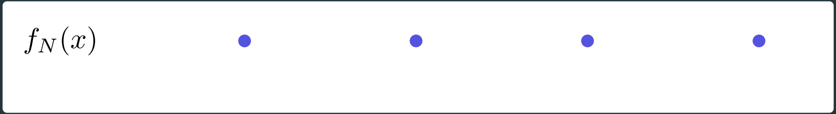
Generate

$$\sum_{x=0}^{\text{poly}(P)} |x\rangle\, |f_N(x)\rangle$$



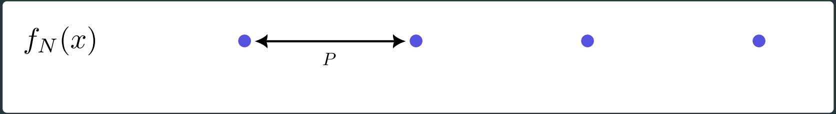
Why do we need $\mathcal{O}(n)$ qubits?

Find some $f_N(x)$ w/ period *P*, where *P* can be used to find factors → **Generate** $\sum_{x=0}^{\text{poly}(P)} |x\rangle |f_N(x)\rangle$ → **Measure, yielding superposition over coset**



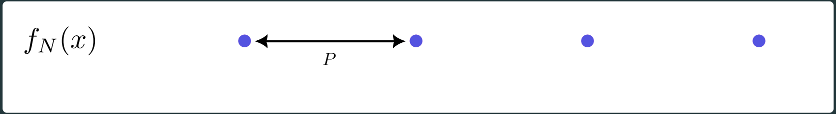
Why do we need $\mathcal{O}(n)$ qubits?

Find some $f_N(x)$ w/ period *P*, where *P* can be used to find factors → **Generate** $\sum_{x=0}^{\mathrm{poly}(P)} |x\rangle |f_N(x)\rangle$ → **Measure, yielding superposition over coset** → **QFT on x register, measure, postprocess to find *P***



Why do we need $\mathcal{O}(n)$ qubits?

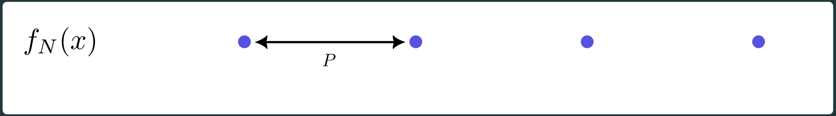
Find some $f_N(x)$ w/ period *P*, where *P* can be used to find factors → **Generate** $\sum_{x=0}^{\text{poly}(P)} |x\rangle |f_N(x)\rangle$ → **Measure, yielding superposition over coset** → **QFT on x register, measure, postprocess to find *P***



<u>Shor's algorithm:</u> Function: $f_N(x) = a^x \bmod N$ Period: $P = \text{ord}_N(a) = \mathcal{O}(N)$

Why do we need $\mathcal{O}(n)$ qubits?

Find some $f_N(x)$ w/ period *P*, where *P* can be used to find factors → **Generate** $\sum_{x=0}^{\mathrm{poly}(P)} |x\rangle |f_N(x)\rangle$ → **Measure, yielding superposition over coset** → **QFT on x register, measure, postprocess to find *P***



Shor's algorithm: Function: $f_N(x) = a^x \bmod N$ Period: $P = \mathrm{ord}_N(a) = \mathcal{O}(N)$

Could we find a function with smaller period?

Some number theory

Legendre symbol

Legendre symbol

For a prime p:

$$\left(\frac{x}{p}\right) = \begin{cases} 0 & \text{if } x \equiv 0 \pmod{p} \\ 1 & \text{if } \exists\, w \text{ s.t. } w^2 \equiv x \pmod{p} \\ -1 & \text{otherwise} \end{cases}$$

Legendre symbol

For a prime p:

$$\left(\frac{x}{p}\right) = \begin{cases} 0 & \text{if } x \equiv 0 \pmod{p} \\ 1 & \text{if } \exists\, w \text{ s.t. } w^2 \equiv x \pmod{p} \\ -1 & \text{otherwise} \end{cases}$$

Legendre symbol is 1) **efficient to compute** given x and p, 2) **periodic** with period p

Jacobi symbol

For a composite number $N = \prod_i p_i$:

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

Jacobi symbol is 1) **efficient to compute** given x and N, 2) **periodic** with period...?

Some number theory

Jacobi symbol is 1) **efficient to compute** given x and N, 2) **periodic** with period...?

For $N = pq$:

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

Some number theory

Jacobi symbol is 1) **efficient to compute** given x and N, 2) **periodic** with period...?

For $N = pq$:

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

Period is N—not helpful for factoring!

Some number theory

Jacobi symbol is 1) **efficient to compute** given x and N, 2) **periodic** with period...?

For $N = p^2q$:

$$\left(\frac{x}{N}\right) = \left(\frac{x}{p}\right)^2 \left(\frac{x}{q}\right)$$

Some number theory

Jacobi symbol is 1) **efficient to compute** given x and N, 2) **periodic** with period...?

For $N = p^2q$:

$$\left(\frac{x}{N}\right) = \left(\frac{x}{p}\right)^2 \left(\frac{x}{q}\right) = \left(\frac{x}{q}\right)$$

Some number theory

Jacobi symbol is 1) **efficient to compute** given x and N, 2) **periodic** with period...?

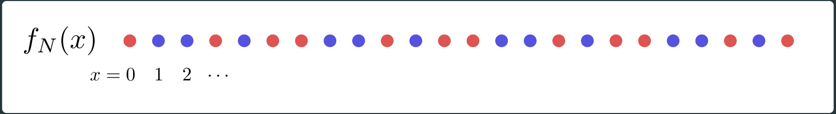
For $N = p^2 q$:

$$\left(\frac{x}{N}\right) = \left(\frac{x}{p}\right)^2 \left(\frac{x}{q}\right) = \left(\frac{x}{q}\right)$$

Period is q—exactly what we need!!

Factoring with the Jacobi symbol

Find some $f_N(x)$ w/ period *P*, where *P* can be used to find factors



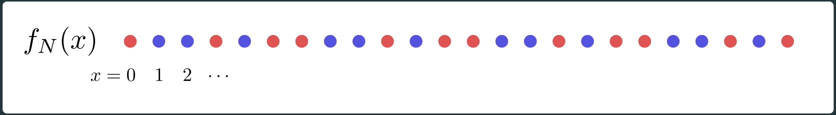
Jacobi factoring, for $N = p^2 q$:

Function: $f_N(x) = \left(\frac{x}{N}\right)$

Period: $P = q$

Factoring with the Jacobi symbol

Find some $f_N(x)$ w/ period P, where P can be used to find factors → **Generate** $\sum_{x=0}^{\mathrm{poly}(P)} |x\rangle\,|f_N(x)\rangle$



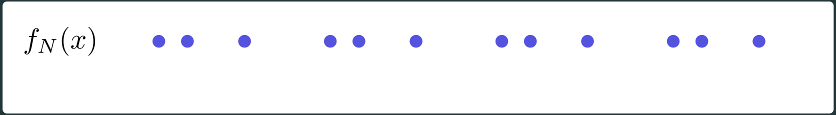
Jacobi factoring, for $N = p^2 q$:

Function: $f_N(x) = \left(\frac{x}{N}\right)$

Period: $P = q$

Factoring with the Jacobi symbol

Find some $f_N(x)$ w/ period P, where P can be used to find factors → **Generate** $\sum_{x=0}^{\mathrm{poly}(P)} |x\rangle |f_N(x)\rangle$ → **Measure, yielding superposition over coset**



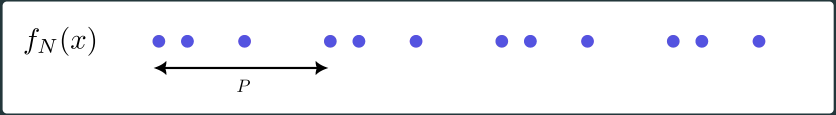
Jacobi factoring, for $N = p^2 q$:

Function: $f_N(x) = \left(\frac{x}{N}\right)$

Period: $P = q$

Factoring with the Jacobi symbol

Find some $f_N(x)$ w/ period *P*, where *P* can be used to find factors → **Generate** $\sum_{x=0}^{\text{poly}(P)} |x\rangle |f_N(x)\rangle$ → **Measure, yielding superposition over coset** → **QFT on x register, measure, postprocess to find *P***



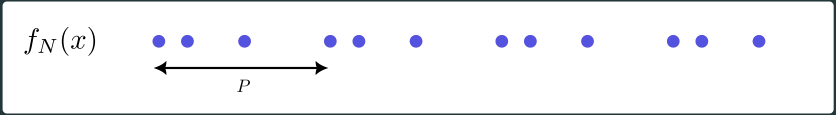
Jacobi factoring, for $N = p^2 q$:

Function: $f_N(x) = \left(\frac{x}{N}\right)$

Period: $P = q$

Factoring with the Jacobi symbol

Find some $f_N(x)$ w/ period *P*, where *P* can be used to find factors → **Generate** $\sum_{x=0}^{\mathrm{poly}(P)} |x\rangle\,|f_N(x)\rangle$ → **Measure, yielding superposition over coset** → **QFT on x register, measure, postprocess to find *P***



Jacobi factoring, for $N = p^2q$: Function: $f_N(x) = \left(\frac{x}{N}\right)$ Period: $P = q$

Is this going to actually work?

Li, Peng, Du, Suter ’12

LDPS. “An Efficient Exact Quantum Algorithm for the Integer Square-free Decomposition Problem.” Nature Scientific Reports, 2012.

Quantum squarefree decomposition $N \to P^2 Q$ via Jacobi symbol was known in the literature a decade ago!

Li, Peng, Du, Suter ’12

LDPS. “An Efficient Exact Quantum Algorithm for the Integer Square-free Decomposition Problem.” Nature Scientific Reports, 2012.

Quantum squarefree decomposition $N \to P^2 Q$ via Jacobi symbol was known in the literature a decade ago!

Their results:

- Period finding yields Q *exactly* if we take a superposition $x \in [0, N-1]$
- Jacobi symbol can be computed efficiently via standard circuits

Li, Peng, Du, Suter ’12

LDPS. “An Efficient Exact Quantum Algorithm for the Integer Square-free Decomposition Problem.” Nature Scientific Reports, 2012.

Quantum squarefree decomposition $N \to P^2 Q$ via Jacobi symbol was known in the literature a decade ago!

Our contributions:

- Period finding yields Q *exactly* if we take a superposition $x \in [0, N-1]$
  - With superposition only to $\mathrm{poly}(Q)$, algorithm still succeeds w.h.p.
- Jacobi symbol can be computed efficiently via standard circuits

Li, Peng, Du, Suter ’12

LDPS. “An Efficient Exact Quantum Algorithm for the Integer Square-free Decomposition Problem.” Nature Scientific Reports, 2012.

Quantum squarefree decomposition $N \to P^2 Q$ via Jacobi symbol was known in the literature a decade ago!

Our contributions:

- Period finding yields Q *exactly* if we take a superposition $x \in [0, N-1]$
  - With superposition only to $\mathrm{poly}(Q)$, algorithm still succeeds w.h.p.
- Jacobi symbol can be computed efficiently via standard circuits
  - When quantum input is small, **extremely** efficient quantum circuits exist!

Computing the Jacobi symbol

Goal: Compute $\left(\frac{x}{N}\right)$

Computing the Jacobi symbol

Goal: Compute $\left(\frac{x}{N}\right)$

$$\left(\frac{a}{b}\right) \in \{-1, 0, 1\} \tag{1}$$

Computing the Jacobi symbol

Goal: Compute $|x\rangle \to \left(\frac{x}{N}\right)|x\rangle$

$$\left(\frac{a}{b}\right) \widetilde{\in} \{-1, 1\} \quad (1)$$

Computing the Jacobi symbol

Goal: Compute $|x\rangle \to \left(\frac{x}{N}\right)|x\rangle$

$$\left(\frac{a}{b}\right) \widetilde{\in} \{-1, 1\} \tag{1}$$

Recall: N is classical, n bits; $|x\rangle$ is quantum, m qubits—and potentially $m \ll n$.

Goal: Compute $|x\rangle \rightarrow \left(\frac{x}{N}\right)|x\rangle$

$$\left(\frac{a}{b}\right) \widetilde{\in} \{-1, 1\} \tag{1}$$

Recall: N is classical, n bits; $|x\rangle$ is quantum, m qubits—and potentially $m \ll n$.

The "big" input is entirely classical.
Can we implement this circuit using only $O(m)$ qubits?

Computing the Jacobi symbol

Goal: Compute $|x\rangle \to \left(\frac{x}{N}\right)|x\rangle$

$$\left(\frac{a}{b}\right) \tilde{\in} \{-1, 1\} \tag{1}$$

Recall: N is classical, n bits; $|x\rangle$ is quantum, m qubits—and potentially $m \ll n$.

The "big" input is entirely classical.
Can we implement this circuit using only $O(m)$ qubits?

Yes!

Computing the Jacobi symbol

Goal: Compute $|x\rangle \to \left(\frac{x}{N}\right)|x\rangle$; N classical

Computing the Jacobi symbol

Goal: Compute $|x\rangle \to \left(\frac{x}{N}\right)|x\rangle$; N classical

Some identities:

$$\left(\frac{a}{b}\right) = \left(\frac{a \bmod b}{b}\right)$$

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

Computing the Jacobi symbol

Goal: Compute $|x\rangle \to \left(\frac{x}{N}\right)|x\rangle$; N classical

Some identities:

$$\left(\frac{a}{b}\right) = \left(\frac{a \bmod b}{b}\right)$$

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

Jacobi symbol $\Leftrightarrow$ Greatest common divisor

Computing the Jacobi symbol

Jacobi symbol $\Leftrightarrow$ Greatest common divisor

Classic GCD algorithms: (let $b < a$)

Computing the Jacobi symbol

Jacobi symbol $\Leftrightarrow$ Greatest common divisor

Classic GCD algorithms: (let $b < a$)

Euclidean algorithm

Euclid, Greece, 2000 years ago

Iterate:

$$\gcd(a, b) \to \gcd(b, a \bmod b)$$

Computing the Jacobi symbol

Jacobi symbol $\Leftrightarrow$ Greatest common divisor

Classic GCD algorithms: (let $b < a$)

Euclidean algorithm

Euclid, Greece, 2000 years ago

Iterate:

$$\gcd(a, b) \to \gcd(b, a \bmod b)$$

Binary GCD

Fangtian, China, 2000 years ago

Suppose a, b odd

Iterate:

$$\gcd(a, b) \to \gcd(b, (a - b)/2^k)$$

Computing the Jacobi symbol

Jacobi symbol $\Leftrightarrow$ Greatest common divisor

Classic GCD algorithms: (let $b < a$)

Euclidean algorithm

Euclid, Greece, 2000 years ago

Iterate:

$$\gcd(a, b) \to \gcd(b, a \bmod b)$$

Binary GCD

Fangtian, China, 2000 years ago

Suppose a, b odd

Iterate:

$$\gcd(a, b) \to \gcd(b, (a - b)/2^k)$$

Both seem to require at least $\mathcal{O}(n)$ qubits, and not reversible...

Computing the Jacobi symbol

Result: Quantum circuit for $|x\rangle \to \left(\frac{x}{N}\right)|x\rangle$, with qubit count indepedent of N

Computing the Jacobi symbol

Result: Quantum circuit for $|x\rangle \to \left(\frac{x}{N}\right)|x\rangle$, with qubit count indepedent of N

Key identities:

$$\left(\frac{a}{b}\right) = \left(\frac{a - kb}{b}\right) \quad \forall\, k$$

$$\left(\frac{a}{b}\right) = (-1)^{f(\ell)} \left(\frac{a'}{b}\right) \quad \text{for } \ell \text{ s.t. } a' = a/2^{\ell} \text{ is odd}$$

Computing the Jacobi symbol

Result: Quantum circuit for $|x\rangle \rightarrow \left(\frac{x}{N}\right)|x\rangle$, with qubit count independent of N

Key identities:

$$\left(\frac{a}{b}\right) = \left(\frac{a-kb}{b}\right) \quad \forall\, k$$

$$\left(\frac{a}{b}\right) = (-1)^{f(\ell)}\left(\frac{a'}{b}\right) \quad \text{for } \ell \text{ s.t. } a' = a/2^{\ell} \text{ is odd}$$

Idea: For n-bit N and m-bit x,
find $N' = kx$ s.t. only leading m bits of $N - N'$ are nonzero

Computing the Jacobi symbol

Idea: For n-bit N and m-bit x,
find $N' = kx$ s.t. only leading m bits of $N - N'$ are nonzero

Overall plan:

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

Computing the Jacobi symbol

Idea: For n-bit N and m-bit x,
find $N' = kx$ s.t. only leading m bits of $N - N'$ are nonzero

Overall plan:

$$\left(\frac{x}{N}\right) \to \left(\frac{N}{x}\right)$$

Computing the Jacobi symbol

Idea: For n-bit N and m-bit x,
find $N' = kx$ s.t. only leading m bits of $N - N'$ are nonzero

Overall plan:

$$\left(\frac{x}{N}\right) \to \left(\frac{N}{x}\right) \to \left(\frac{N - kx}{x}\right)$$

Computing the Jacobi symbol

Idea: For n-bit N and m-bit x,
find $N' = kx$ s.t. only leading m bits of $N - N'$ are nonzero

Overall plan:

$$\left(\frac{x}{N}\right) \to \left(\frac{N}{x}\right) \to \left(\frac{N - kx}{x}\right) \to \left(\frac{(N - kx)/2^{n-m}}{x}\right)$$

Computing the Jacobi symbol

Idea: For n-bit N and m-bit x,
find $N' = kx$ s.t. only leading m bits of $N - N'$ are nonzero

Overall plan:

$$\left(\frac{x}{N}\right) \to \left(\frac{N}{x}\right) \to \left(\frac{N-kx}{x}\right) \to \left(\frac{(N-kx)/2^{n-m}}{x}\right)$$

Last value has two m-bit inputs; cost is independent of N with standard circuits.

Computing the Jacobi symbol

Idea: For n-bit N and m-bit x,
find $N' = kx$ s.t. only leading m bits of $N - N'$ are nonzero

Overall plan:

$$\left(\frac{x}{N}\right) \to \left(\frac{N}{x}\right) \to \left(\frac{N-kx}{x}\right) \to \left(\frac{(N-kx)/2^{n-m}}{x}\right)$$

Last value has two m-bit inputs; cost is independent of N with standard circuits.

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle |0^m\rangle \rightarrow |x\rangle |N'\rangle$

```
|x⟩ =                 |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=                 |0 0 0 0 0 0 0⟩
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩  =                |1 0 0 1 0 1 1⟩
 N   = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=                 |0 0 0 0 0 0 0⟩
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =                 |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=                 |1 0 0 1 0 1 1⟩
                              |c⟩=|1⟩
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =                   |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=                   |1 0 0 1 0 1 1⟩
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =               |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=               |0 1 0 0 1 0 1⟩1
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =             |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=             |0 1 0 0 1 0 1⟩1
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩  =             |1 0 0 1 0 1 1⟩
 N   = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=              |0 1 0 0 1 0 1⟩1
                           |c⟩=|0⟩
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =             |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=             |0 1 0 0 1 0 1⟩1
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =           |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=           |0 0 1 0 0 1 0⟩1 1
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =           |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=           |0 0 0 1 0 0 1⟩0 1 1
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =       |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=       |1 0 1 0 1 0 0⟩0 1 1
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle |0^m\rangle \to |x\rangle |N'\rangle$

```
|x⟩ =       |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=       |0 1 0 1 0 1 0⟩0 0 1 1
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =       |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=       |1 1 1 0 1 0 1⟩0 0 1 1
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =     |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=     |0 1 1 1 0 1 0⟩1 0 0 1 1
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =  |1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=  |0 0 1 1 1 0 1⟩0 1 0 0 1 1
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =|1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=|0 0 0 1 1 1 0⟩1 0 1 0 0 1 1
```

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =|1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=|0 0 0 1 1 1 0⟩1 0 1 0 0 1 1
```

Gate count: $\mathcal{O}(nm)$.

Computing the Jacobi symbol

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =|1 0 0 1 0 1 1⟩
 N  = 1 1 0 0 1 1 0 1 0 1 0 0 1 1
|N'⟩=|0 0 0 1 1 1 0⟩1 0 1 0 0 1 1
```

Gate count: $\mathcal{O}(nm)$. **We can do better!**

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =                                                    |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=                                        |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =                                                     |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=                                     |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0⟩


|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =                                               |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=                               |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0⟩
```

```
|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =                                                    |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=                                        |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0⟩
                                                         |c⟩ = |0 0 1 1 1 0 1 1⟩


|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =                                                         |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=                                         |0 1 1 1 0 0 0 0 1 1 1 1 0 1 0 1⟩
                                                             |c⟩ = |0 0 1 1 1 0 1 1⟩


|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =                                                    |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=                                    |0 1 1 1 0 0 0 0 1 1 1 1 0 1 0 1⟩
```

```
|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =                       |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=               |0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0⟩1 1 1 1 0 1 0 1


|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =                               |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=                |0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0⟩1 1 1 1 0 1 0 1

|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =                                |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=                |0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0⟩1 1 1 1 0 1 0 1
                               |c⟩ = |1 0 0 0 1 0 1 1⟩


|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =                                |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=                |0 1 1 1 0 1 0 0 0 1 0 1 1 0 0 1⟩1 1 1 1 0 1 0 1
                               |c⟩ = |1 0 0 0 1 0 1 1⟩


|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =                                |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=                |0 1 1 1 0 1 0 0 0 1 0 1 1 0 0 1⟩1 1 1 1 0 1 0 1
```

```
|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =                |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=|0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0⟩0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1


|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =                |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=|0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0⟩0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
               |c⟩ = |1 0 1 0 1 0 0 1⟩


|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =                 |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=|1 0 0 1 0 0 0 1 1 1 1 0 0 1 0 1⟩0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
              |c⟩ = |1 0 1 0 1 0 0 1⟩


|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

```
|x⟩ =                 |1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=|1 0 0 1 0 0 0 1 1 1 1 0 0 1 0 1⟩0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1


|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =|1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=|1 0 0 1 0 0 0 1⟩1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
```

```
|x_modinv⟩ = |1 0 0 1 0 1 0 1⟩
|1/x⟩ = |1 0 1 0 1 1 0 1 0⟩
```

Computing the Jacobi symbol, fast!

Goal #2: Circuit for $|x\rangle\,|0^m\rangle \to |x\rangle\,|N'\rangle$

```
|x⟩ =|1 0 1 1 1 1 0 1⟩
 N  = 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
|N'⟩=|1 0 0 1 0 0 0 1⟩1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1
```

Computing the Jacobi symbol, fast!

Result: Fast circuit for $|x\rangle\,|0^m\rangle \rightarrow |x\rangle\,|N'\rangle$

Suppose t-bit multiplication costs $G_M(t)$ gates, $D_M(t)$ depth, $S_M(t)$ qubits.

Circuit cost:

Gates: $\mathcal{O}(\frac{n}{m} \cdot G_M(m))$

Computing the Jacobi symbol, fast!

Result: Fast circuit for $|x\rangle |0^m\rangle \rightarrow |x\rangle |N'\rangle$

Suppose t-bit multiplication costs $G_M(t)$ gates, $D_M(t)$ depth, $S_M(t)$ qubits.

Circuit cost:

Gates: $\mathcal{O}(\frac{n}{m} \cdot G_M(m))$

Depth: $\mathcal{O}(\frac{n}{m} \cdot D_M(m))$

Computing the Jacobi symbol, fast!

Result: Fast circuit for $|x\rangle |0^m\rangle \to |x\rangle |N'\rangle$

Suppose t-bit multiplication costs $G_M(t)$ gates, $D_M(t)$ depth, $S_M(t)$ qubits.

<u>Circuit cost:</u>

Gates: $\mathcal{O}(\frac{n}{m} \cdot G_M(m))$

Depth: $\mathcal{O}(\frac{n}{m} \cdot D_M(m))$

Space: $\mathcal{O}(S_M(m))$

Computing the Jacobi symbol

Overall plan:

$$\left(\frac{x}{N}\right) \to \left(\frac{N}{x}\right) \to \left(\frac{N - kx}{x}\right) \to \left(\frac{(N - kx)/2^{n-m}}{x}\right)$$

Putting it all together: asymptotic costs

Main result: Circuit for factoring n-bit integers $N = p^2 q$, with $q < 2^m$

Putting it all together: asymptotic costs

Main result: Circuit for factoring n-bit integers $N = p^2 q$, with $q < 2^m$

Schoolbook mult. + standard GCD:

Gates: $\mathcal{O}(nm)$
Depth: $\mathcal{O}(n)$
Space: $\mathcal{O}(m)$

Putting it all together: asymptotic costs

Main result: Circuit for factoring n-bit integers $N = p^2 q$, with $q < 2^m$

Schoolbook mult. + standard GCD:

Gates: $\mathcal{O}(nm)$
Depth: $\mathcal{O}(n)$
Space: $\mathcal{O}(m)$

Fast mult. + fast GCD:

Gates: $\mathcal{O}(n \log m)$
Depth: $\widetilde{\mathcal{O}}(n/m + m)$
Space: $\widetilde{\mathcal{O}}(m)$

Aside: fast multiplication in low space

[GDKM, Yao; arXiv:2403.18006]

New quantum multiplication circuit:

- Gates: $\mathcal{O}_\epsilon(t^{1+\epsilon})$
- Ancillas: zero!

Aside: fast multiplication in low space

[GDKM, Yao; arXiv:2403.18006]

New quantum multiplication circuit:

- **Gates:** $\mathcal{O}_\epsilon(t^{1+\epsilon})$
- **Ancillas:** zero!

[GDKM, Gidney, Chuang; in prep.]

Parallel version of that circuit:

- **Depth:** $\mathcal{O}_\epsilon(t^\epsilon)$
- **Ancillas:** $o(t)$

Aside: fast multiplication in low space

[GDKM, Yao; arXiv:2403.18006]

New quantum multiplication circuit:

- Gates: $\mathcal{O}_\epsilon(t^{1+\epsilon})$
- Ancillas: zero!

[GDKM, Gidney, Chuang; in prep.]

Parallel version of that circuit:

- Depth: $\mathcal{O}_\epsilon(t^\epsilon)$
- Ancillas: $o(t)$

<u>This mult. + standard GCD:</u>

Gates: $\mathcal{O}_\epsilon(nm^\epsilon + m^2)$

Depth: $\mathcal{O}_\epsilon((n/m)^{1+\epsilon} + m)$

Space: $\mathcal{O}(m)$

Putting it all together: asymptotic costs

Main result: Circuit for factoring n-bit integers $N = p^2 q$, with $q < 2^m$

Schoolbook mult. + standard GCD:

Gates: $\mathcal{O}(nm)$
Depth: $\mathcal{O}(n + m)$
Space: $\mathcal{O}(m)$

Fast mult. + fast GCD:

Gates: $\mathcal{O}(n \log m)$
Depth: $\widetilde{\mathcal{O}}(n/m + m)$
Space: $\widetilde{\mathcal{O}}(m)$

What should we set m to?

What integers should we apply it to?

Classical factoring: for integers $N = p^2 q$, with $n = \log N$ and $m = \log q$

General Number Field Sieve:

Used for RSA integers

Costs roughly $\exp\left(\mathcal{O}(\sqrt[3]{n})\right)$

Lenstra ECM/Mulder:

Used for integers with small factors

Costs roughly $\exp\left(\mathcal{O}(\sqrt{m})\right)$

What integers should we apply it to?

Classical factoring: for integers $N = p^2 q$, with $n = \log N$ and $m = \log q$

General Number Field Sieve:

Used for RSA integers

Costs roughly $\exp\left(\mathcal{O}(\sqrt[3]{n})\right)$

Lenstra ECM/Mulder:

Used for integers with small factors

Costs roughly $\exp\left(\mathcal{O}(\sqrt{m})\right)$

Set $m = \mathcal{O}(n^{2/3})$ for the *cheapest* quantum circuit classically as hard as RSA

Putting it all together: asymptotic costs

Main result: Circuit for factoring n-bit integers $N = p^2q$, with $\log q = m = O(n^{2/3})$

Schoolbook mult. + standard GCD:

Gates: $\mathcal{O}(n^{5/3})$
Depth: $\mathcal{O}(n)$
Space: $\mathcal{O}(n^{2/3})$

Fast mult. + fast GCD:

Gates: $\widetilde{\mathcal{O}}(n)$
Depth: $\widetilde{\mathcal{O}}(n^{2/3})$
Space: $\widetilde{\mathcal{O}}(n^{2/3})$

Putting it all together: asymptotic costs

Main result: Circuit for factoring n-bit integers $N = p^2 q$, with $\log q = m = O(n^{2/3})$

<u>Schoolbook mult. + standard GCD:</u>

Gates: $\mathcal{O}(n^{5/3})$
Depth: $\mathcal{O}(n)$
Space: $\mathcal{O}(n^{2/3})$

<u>Fast mult. + fast GCD:</u>

Gates: $\widetilde{\mathcal{O}}(n)$
Depth: $\widetilde{\mathcal{O}}(n^{2/3})$
Space: $\widetilde{\mathcal{O}}(n^{2/3})$

Space $2m + o(m)$ seems achievable.
Classically-hard factoring with a few hundred qubits?

Summary and open questions

Factoring certain n-bit integers $N = p^2 q$ in:

- **Gates:** $\widetilde{\mathcal{O}}(n)$
- **Space** and **depth:** $\widetilde{\mathcal{O}}(n^{2/3})$

Summary and open questions

Factoring certain n-bit integers $N = p^2 q$ in:

- **Gates:** $\widetilde{\mathcal{O}}(n)$
- **Space** and **depth:** $\widetilde{\mathcal{O}}(n^{2/3})$

Open questions/directions:

Summary and open questions

Factoring certain n-bit integers $N = p^2 q$ in:

- **Gates:** $\widetilde{\mathcal{O}}(n)$
- **Space** and **depth:** $\widetilde{\mathcal{O}}(n^{2/3})$

Open questions/directions:

- Practical classical hardness—what should m be, concretely?

Summary and open questions

Factoring certain n-bit integers $N = p^2 q$ in:

- **Gates:** $\widetilde{\mathcal{O}}(n)$
- **Space** and **depth:** $\widetilde{\mathcal{O}}(n^{2/3})$

Open questions/directions:

- Practical classical hardness—what should m be, concretely?
- Optimization of concrete circuits

Summary and open questions

Factoring certain n-bit integers $N = p^2 q$ in:

- **Gates:** $\widetilde{\mathcal{O}}(n)$
- **Space** and **depth:** $\widetilde{\mathcal{O}}(n^{2/3})$

Open questions/directions:

- Practical classical hardness—what should m be, concretely?
- Optimization of concrete circuits
- Can this be generalized?

Summary and open questions

Factoring certain n-bit integers $N = p^2 q$ in:

- **Gates:** $\widetilde{\mathcal{O}}(n)$
- **Space** and **depth:** $\widetilde{\mathcal{O}}(n^{2/3})$

Open questions/directions:

- Practical classical hardness—what should m be, concretely?
- Optimization of concrete circuits
- Can this be generalized?
  - Currently: completely factor any integer with **distinct exponents** in prime factorization

Summary and open questions

Factoring certain n-bit integers $N = p^2 q$ in:

- **Gates:** $\widetilde{\mathcal{O}}(n)$
- **Space** and **depth:** $\widetilde{\mathcal{O}}(n^{2/3})$

Open questions/directions:

- Practical classical hardness—what should m be, concretely?
- Optimization of concrete circuits
- Can this be generalized?
  - Currently: completely factor any integer with **distinct exponents** in prime factorization
  - Further generalizations? RSA??

Questions?

Greg
Kahanamoku-Meyer

Seyoon
Ragavan

Vinod
Vaikuntanathan

Katherine
van Kirk

gkm@mit.edu
https://gregkm.me/