Discrete log, factoring, and proofs of quantumness in 2026

Greg Meyer

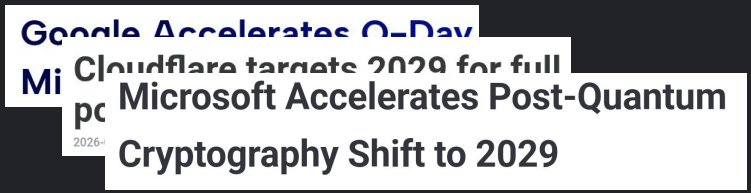
a.k.a. Greg Kahanamoku-Meyer

August 31, 2026

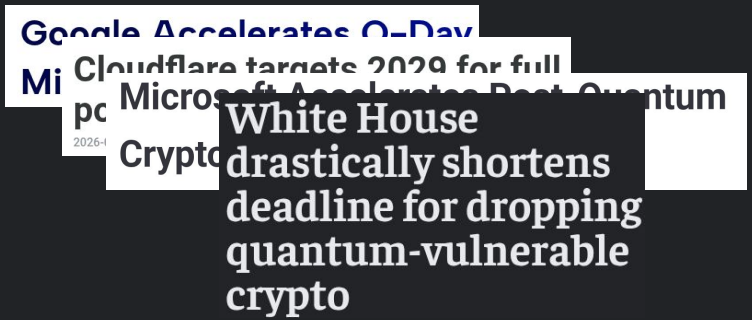
Post-quantum cryptography migration timelines are shifting

Post-quantum cryptography migration timelines are shifting

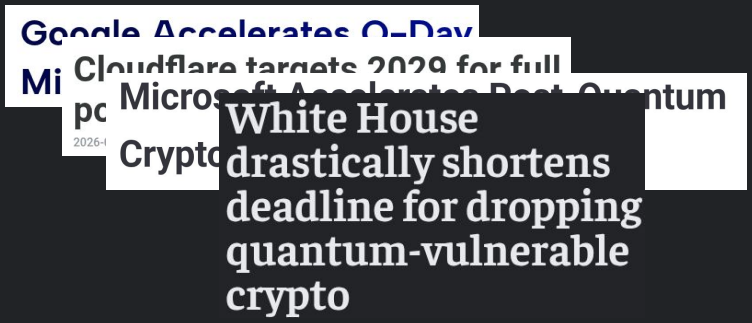
Post-quantum cryptography migration timelines are shifting



Post-quantum cryptography migration timelines are shifting



Post-quantum cryptography migration timelines are shifting



How did this happen?

Structure of this talk

Structure of this talk

Structure of this talk

In this talk:

- Heuristics and assumptions that seemed reasonable, but have not held up

Structure of this talk

In this talk:

- Heuristics and assumptions that seemed reasonable, but have not held up
- **Setting:** algorithms for cryptography

Structure of this talk

In this talk:

- Heuristics and assumptions that seemed reasonable, but have not held up
- **Setting:** algorithms for cryptography
  - Factoring and discrete log

Structure of this talk

In this talk:

- Heuristics and assumptions that seemed reasonable, but have not held up
- **Setting:** algorithms for cryptography
  - Factoring and discrete log
  - Proofs of quantumness

Structure of this talk

In this talk:

- Heuristics and assumptions that seemed reasonable, but have not held up
- **Setting:** algorithms for cryptography
  - Factoring and discrete log
  - Proofs of quantumness
- How should this inform our work moving forward?

Overview

Assumption: factoring is the right thing to focus on

Overview

Assumption: factoring is the right thing to focus on

Assumption: input length is a lower bound on qubit count

Overview

Assumption: factoring is the right thing to focus on

Assumption: input length is a lower bound on qubit count

Assumption: logical qubit count is what matters

Overview

Assumption: factoring is the right thing to focus on

Assumption: input length is a lower bound on qubit count

Assumption: logical qubit count is what matters

Assumption: quantum circuits must be (mostly) reversible

Overview

Assumption: factoring is the right thing to focus on

Assumption: input length is a lower bound on qubit count

Assumption: logical qubit count is what matters

Assumption: quantum circuits must be (mostly) reversible

Assumption: breaking cryptography is a long way off

Overview

Assumption: factoring is the right thing to focus on

Assumption: input length is a lower bound on qubit count

Assumption: logical qubit count is what matters

Assumption: quantum circuits must be (mostly) reversible

Assumption: breaking cryptography is a long way off

Comparing factoring and discrete log

Observation: factoring is often used as a stand-in for “breaking cryptography”

Comparing factoring and discrete log

Observation: factoring is often used as a stand-in for “breaking cryptography”

Is the focus on factoring appropriate?

Comparison: use in modern classical cryptography

Example: digital signature schemes

Data: Cloudflare blog (blog.cloudflare.com/ml-dsa-will-have-to-do/)

Scheme	Ed25519	RSA-2048
Vuln. problem	EC discrete log	factoring
Public key size (bytes)	32	272
Signature size (bytes)	64	256
Time to sign (arb. units)	0.15	80
Time to verify (arb. units)	1.3	0.4

Comparison: use in modern classical cryptography

Example: digital signature schemes

Data: Cloudflare blog (blog.cloudflare.com/ml-dsa-will-have-to-do/)

Scheme	Ed25519	RSA-2048
Vuln. problem	EC discrete log	factoring
Public key size (bytes)	32	272
Signature size (bytes)	64	256
Time to sign (arb. units)	0.15	80
Time to verify (arb. units)	1.3	0.4

Schemes relying on the hardness of **elliptic curve discrete log** are used more widely on the internet.

Comparison: quantum cost to break cryptographic problem sizes

Note: these are **logical** costs, in the abstract circuit model.

Comparison: quantum cost to break cryptographic problem sizes

Note: these are **logical** costs, in the abstract circuit model.

Elliptic curve discrete log

Given $x = [d]g$, find d.

For curve `secp256k1`:

- Qubits: ~ 1200
- Toffolis: $\sim 2^{26}$

Babbush et al. ’26 (arXiv:2603.28846)

Schrottenloher ’26 (arXiv:2606.02235)

Comparison: quantum cost to break cryptographic problem sizes

Note: these are **logical** costs, in the abstract circuit model.

Elliptic curve discrete log

Given $x = [d]g$, find d.

For curve `secp256k1`:

- Qubits: ~ 1200
- Toffolis: $\sim 2^{26}$

Babbush et al. '26 (arXiv:2603.28846)
Schrottenloher '26 (arXiv:2606.02235)

RSA integer factorization

Given $N = pq$, find p and q.

For RSA-2048:

- Qubits: ~ 1400
- Toffolis: $\sim 2^{33}$

Gidney '25 (arXiv:2505.15917)

Comparison: quantum cost to break cryptographic problem sizes

Note: these are **logical** costs, in the abstract circuit model.

Elliptic curve discrete log

Given $x = [d]g$, find d.

For curve `secp256k1`:

- Qubits: ~ 1200
- Toffolis: $\sim 2^{26}$

Babbush et al. '26 (arXiv:2603.28846)
Schrottenloher '26 (arXiv:2606.02235)

RSA integer factorization

Given $N = pq$, find p and q.

For RSA-2048:

- Qubits: ~ 1400
- Toffolis: $\sim 2^{33}$

Gidney '25 (arXiv:2505.15917)

Elliptic curve discrete log has cheaper quantum circuits (as of now).

Comparing factoring and discrete log

Most attention given	factoring
Most widely used	discrete log
Cheapest quantum cost	discrete log

Comparing factoring and discrete log

Most attention given	factoring
Most widely used	discrete log
Cheapest quantum cost	discrete log

We're giving attention to the problem that is **less widely used** and **more expensive**.

Comparing factoring and discrete log

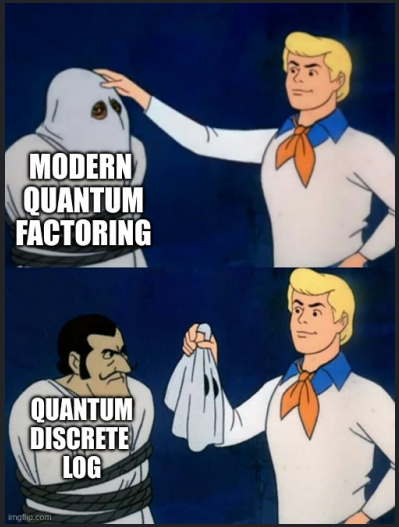
Most attention given	factoring
Most widely used	discrete log
Cheapest quantum cost	discrete log

We're giving attention to the problem that is
less widely used and **more expensive**.

... but there's even another reason to focus on discrete log...

Let’s see who you really are...

Let’s see who you really are...



Factoring RSA integers via discrete logarithm

Foundational paper: Ekerå and Håstad ’17 (arXiv:1702.00249)

Observation

Consider an integer $N = pq$.

For the vast majority of positive integers $g < N$:

$$g^{N+1} \equiv g^{p+q} \pmod{N}$$

Factoring RSA integers via discrete logarithm

Foundational paper: Ekerå and Håstad ’17 (arXiv:1702.00249)

Observation

Consider an integer $N = pq$.

For the vast majority of positive integers $g < N$:

$$g^{N+1} \equiv g^{p+q} \pmod{N}$$

🌟 Knowing $p + q$ is sufficient to factor N.

Factoring RSA integers via discrete logarithm

Foundational paper: Ekerå and Håstad ’17 (arXiv:1702.00249)

Observation

Consider an integer $N = pq$.

For the vast majority of positive integers $g < N$:

$$g^{N+1} \equiv g^{p+q} \pmod{N}$$

🌟 Knowing $p + q$ is sufficient to factor N.

Plan

🖥️ = classical step, ⚛️ = quantum step

1. 🖥️ Pick an integer $g < N$

Factoring RSA integers via discrete logarithm

Foundational paper: Ekerå and Håstad ’17 (arXiv:1702.00249)

Observation

Consider an integer $N = pq$.

For the vast majority of positive integers $g < N$:

$$g^{N+1} \equiv g^{p+q} \pmod{N}$$

🌟 Knowing $p + q$ is sufficient to factor N.

Plan

🖥️ = classical step, ⚛️ = quantum step

1. 🖥️ Pick an integer $g < N$
2. 🖥️ Compute $x = g^{N+1} \bmod N$

Factoring RSA integers via discrete logarithm

Foundational paper: Ekerå and Håstad ’17 (arXiv:1702.00249)

Observation

Consider an integer $N = pq$.

For the vast majority of positive integers $g < N$:

$$g^{N+1} \equiv g^{p+q} \pmod{N}$$

🌟 Knowing $p + q$ is sufficient to factor N.

Plan

🖥️ = classical step, ⚛️ = quantum step

1. 🖥️ Pick an integer $g < N$
2. 🖥️ Compute $x = g^{N+1} \bmod N$
3. ⚛️ Compute $\text{dlog}_g(x) = p + q$

Factoring RSA integers via discrete logarithm

Foundational paper: Ekerå and Håstad ’17 (arXiv:1702.00249)

Observation

Consider an integer $N = pq$.

For the vast majority of positive integers $g < N$:

$$g^{N+1} \equiv g^{p+q} \pmod{N}$$

🌟 Knowing $p + q$ is sufficient to factor N.

Plan

🖥️ = classical step, ⚛️ = quantum step

1. 🖥️ Pick an integer $g < N$
2. 🖥️ Compute $x = g^{N+1} \bmod N$
3. ⚛️ Compute $\text{dlog}_g(x) = p + q$
4. 🖥️ Factor N using $p + q$

Factoring RSA integers via discrete logarithm

Foundational paper: Ekerå and Håstad ’17 (arXiv:1702.00249)

Observation

Consider an integer $N = pq$.

For the vast majority of positive integers $g < N$:

$$g^{N+1} \equiv g^{p+q} \pmod{N}$$

🌟 Knowing $p + q$ is sufficient to factor N.

Plan

🖥️ = classical step, ⚛️ = quantum step

1. 🖥️ Pick an integer $g < N$
2. 🖥️ Compute $x = g^{N+1} \bmod N$
3. ⚛️ Compute $\text{dlog}_g(x) = p + q$
4. 🖥️ Factor N using $p + q$

Factoring RSA integers via discrete logarithm

Foundational paper: Ekerå and Håstad ’17 (arXiv:1702.00249)

Observation

Consider an integer $N = pq$.

For the vast majority of positive integers $g < N$:

$$g^{N+1} \equiv g^{p+q} \pmod{N}$$

🌟 Knowing $p + q$ is sufficient to factor N.

Plan

🖥️ = classical step, ⚛️ = quantum step

1. 🖥️ Pick an integer $g < N$
2. 🖥️ Compute $x = g^{N+1} \bmod N$
3. ⚛️ Compute $\text{dlog}_g(x) = p + q$
4. 🖥️ Factor N using $p + q$

So, we can factor via a discrete log. **Is this helpful?**

Cost depends on the period

Foundational paper: Ekerå and Håstad ’17 (arXiv:1702.00249)

In **quantum period finding**, circuit size increases with the size of the period.

Cost depends on the period

Foundational paper: Ekerå and Håstad ’17 (arXiv:1702.00249)

In **quantum period finding**, circuit size increases with the size of the period.

Shor’s original idea

Factor by finding period of $f(x) = a^x \bmod N$.

Fact: period has size $O(N)$.

Cost depends on the period

Foundational paper: Ekerå and Håstad ’17 (arXiv:1702.00249)

In **quantum period finding**, circuit size increases with the size of the period.

Shor’s original idea

Factor by finding period of $f(x) = a^x \bmod N$.

Fact: period has size $O(N)$.

Ekerå-Håstad ’17

Factor by taking the discrete log of $g^{N+1} \equiv g^{p+q} \pmod{N}$

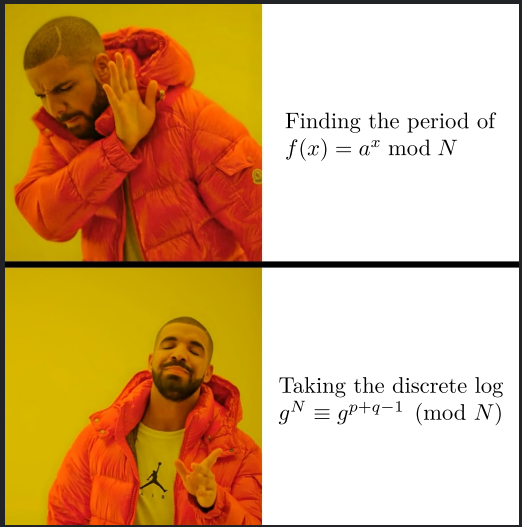
Fact: period has size $O(\sqrt{N})$.

Factoring RSA integers via discrete logarithm

Summary: nowadays, all RSA factoring circuits...

Factoring RSA integers via discrete logarithm

Summary: nowadays, all RSA factoring circuits...



Summary

Assumption: factoring is the thing to focus on

Reality:

- Elliptic curve discrete log is **more widely relied upon** than factoring

Summary

Assumption: factoring is the thing to focus on

Reality:

- Elliptic curve discrete log is **more widely relied upon** than factoring
- Elliptic curve discrete log has **cheaper quantum circuits** than factoring

Summary

Assumption: factoring is the thing to focus on

Reality:

- Elliptic curve discrete log is **more widely relied upon** than factoring
- Elliptic curve discrete log has **cheaper quantum circuits** than factoring
- Modern circuits for RSA factoring are **discrete log** circuits anyway

Overview

Assumption: factoring is the right thing to focus on

Assumption: input length is a lower bound on qubit count

Assumption: logical qubit count is what matters

Assumption: quantum circuits must be (mostly) reversible

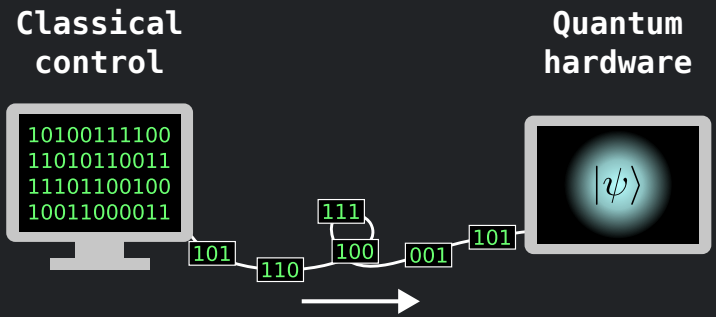
Assumption: breaking cryptography is a long way off

How do we put classical input into a quantum computer?

How do we put classical input into a quantum computer?

If we input an n-bit value, we must need n qubits, right?

How do we put classical input into a quantum computer?



If we input an n-bit value, we must need n qubits, right? Maybe not!

Streaming algorithms

Typical setting

Streaming algorithms

Typical setting

Streaming as a limitation

Streaming algorithms

Typical setting

Streaming as a limitation

"You only get to see m bits of the n-bit input at a time"

This setting

Streaming algorithms

Typical setting

Streaming as a limitation

"You only get to see m bits of the n-bit input at a time"

This setting

Streaming as a tool

Streaming algorithms

Typical setting

Streaming as a limitation

"You only get to see m bits of the n-bit input at a time"

This setting

Streaming as a tool

"Find an algorithm that only needs to see m bits of the n-bit input at a time!"

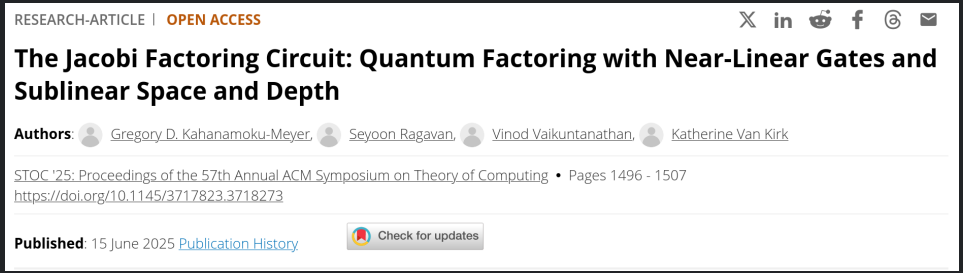
Streaming algorithms

Typical setting	**This setting**
Streaming as a limitation	Streaming as a tool
"You only get to see m bits of the n-bit input at a time"	"Find an algorithm that only needs to see m bits of the n-bit input at a time!"

Two quantum algorithms that do this:

1. Jacobi factoring
2. New discrete log/RSA factoring algorithms

Jacobi factoring



RESEARCH-ARTICLE | OPEN ACCESS

The Jacobi Factoring Circuit: Quantum Factoring with Near-Linear Gates and Sublinear Space and Depth

Authors: Gregory D. Kahanamoku-Meyer, Seyoon Ragavan, Vinod Vaikuntanathan, Katherine Van Kirk

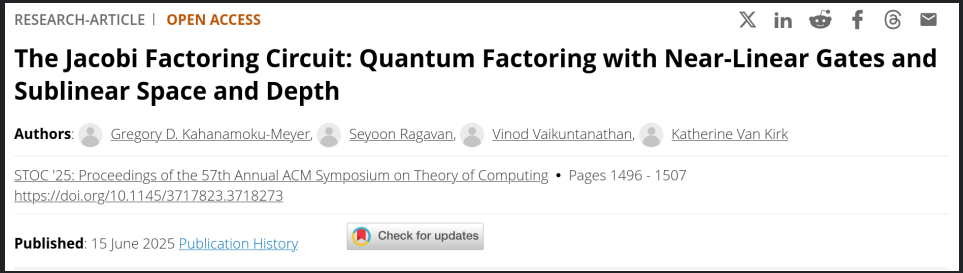
STOC '25: Proceedings of the 57th Annual ACM Symposium on Theory of Computing • Pages 1496 - 1507
https://doi.org/10.1145/3717823.3718273

Published: 15 June 2025 Publication History

Check for updates

Quantum algorithm for factoring integers of the **special form** $N = P^2Q$.

Jacobi factoring



RESEARCH-ARTICLE | OPEN ACCESS

The Jacobi Factoring Circuit: Quantum Factoring with Near-Linear Gates and Sublinear Space and Depth

Authors: Gregory D. Kahanamoku-Meyer, Seyoon Ragavan, Vinod Vaikuntanathan, Katherine Van Kirk

STOC '25: Proceedings of the 57th Annual ACM Symposium on Theory of Computing • Pages 1496 - 1507
https://doi.org/10.1145/3717823.3718273

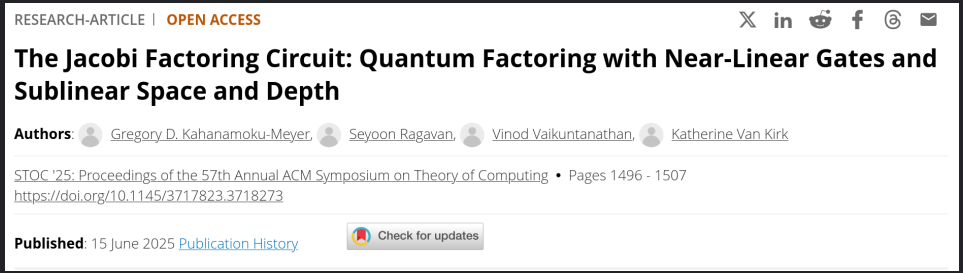
Published: 15 June 2025 Publication History

Check for updates

Quantum algorithm for factoring integers of the **special form** $N = P^2Q$.

Let $n = \lceil \log N \rceil$ and $m = \lceil \log Q \rceil$.

Jacobi factoring



RESEARCH-ARTICLE | OPEN ACCESS

The Jacobi Factoring Circuit: Quantum Factoring with Near-Linear Gates and Sublinear Space and Depth

Authors: Gregory D. Kahanamoku-Meyer, Seyoon Ragavan, Vinod Vaikuntanathan, Katherine Van Kirk

STOC '25: Proceedings of the 57th Annual ACM Symposium on Theory of Computing • Pages 1496 - 1507
https://doi.org/10.1145/3717823.3718273

Published: 15 June 2025 Publication History

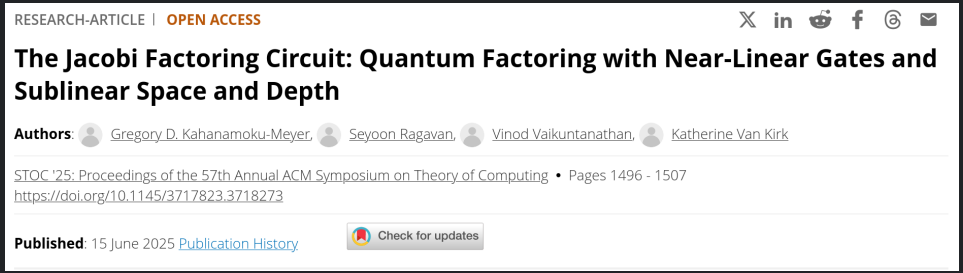
Check for updates

Quantum algorithm for factoring integers of the **special form** $N = P^2Q$.

Let $n = \lceil \log N \rceil$ and $m = \lceil \log Q \rceil$.

- Gates: $\widetilde{O}(n)$
- Qubits: $\widetilde{O}(m)$
- Depth: $\widetilde{O}(m)$

Jacobi factoring



RESEARCH-ARTICLE | OPEN ACCESS

The Jacobi Factoring Circuit: Quantum Factoring with Near-Linear Gates and Sublinear Space and Depth

Authors: Gregory D. Kahanamoku-Meyer, Seyoon Ragavan, Vinod Vaikuntanathan, Katherine Van Kirk

STOC '25: Proceedings of the 57th Annual ACM Symposium on Theory of Computing • Pages 1496 - 1507
https://doi.org/10.1145/3717823.3718273

Published: 15 June 2025 Publication History

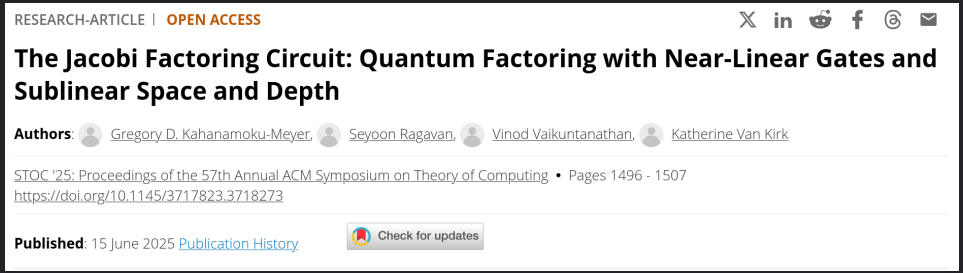
Check for updates

Quantum algorithm for factoring integers of the **special form** $N = P^2Q$.

Let $n = \lceil \log N \rceil$ and $m = \lceil \log Q \rceil$.

- Gates: $\widetilde{O}(n)$
- Qubits: $\widetilde{O}(m)$
- Depth: $\widetilde{O}(m)$

Jacobi factoring



RESEARCH-ARTICLE | OPEN ACCESS

The Jacobi Factoring Circuit: Quantum Factoring with Near-Linear Gates and Sublinear Space and Depth

Authors: Gregory D. Kahanamoku-Meyer, Seyoon Ragavan, Vinod Vaikuntanathan, Katherine Van Kirk

STOC '25: Proceedings of the 57th Annual ACM Symposium on Theory of Computing • Pages 1496 - 1507
https://doi.org/10.1145/3717823.3718273

Published: 15 June 2025 Publication History

Check for updates

Quantum algorithm for factoring integers of the **special form** $N = P^2Q$.

Let $n = \lceil \log N \rceil$ and $m = \lceil \log Q \rceil = O(n^{2/3})$.
(smallest m that maintains classical hardness)

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

Jacobi factoring: how does it work?

Observation

The *Jacobi symbol*

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

is periodic with period Q
when $N = P^2Q$.

Jacobi factoring: how does it work?

Observation

The *Jacobi symbol*

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

is periodic with period Q when $N = P^2Q$.

Plan

Use quantum period finding to find Q!

Jacobi factoring: how does it work?

Observation

The *Jacobi symbol*

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

is periodic with period Q when $N = P^2Q$.

Plan

Use quantum period finding to find Q!

This requires creating a state

$$\sum_x |x\rangle\,|f(x)\rangle$$

Jacobi factoring: how does it work?

Observation

The *Jacobi symbol*

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

is periodic with period Q when $N = P^2 Q$.

Plan

Use quantum period finding to find Q!

This requires creating a state

$$\sum_x |x\rangle\,|f(x)\rangle$$

Note: input register $|x\rangle$ only needs $m = O(\log Q)$ qubits!

Jacobi factoring: how does it work?

Observation

The *Jacobi symbol*

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

is periodic with period Q when $N = P^2 Q$.

Plan

Use quantum period finding to find Q!

This requires creating a state

$$\sum_x |x\rangle\,|f(x)\rangle$$

Note: input register $|x\rangle$ only needs $m = O(\log Q)$ qubits!

But how do we compute the Jacobi symbol?

Computing the Jacobi symbol in low space

Goal: circuit to implement $\sum_x |x\rangle |0\rangle \to \sum_x |x\rangle |\left(\frac{x}{N}\right)\rangle$

Computing the Jacobi symbol in low space

Goal: circuit to implement $\sum_x |x\rangle |0\rangle \to \sum_x |x\rangle |\left(\frac{x}{N}\right)\rangle$

Inputs:

- N, a big (n-bit) classical number
- $|x\rangle$, a small (m-bit) quantum number

Computing the Jacobi symbol in low space

Goal: circuit to implement $\sum_x |x\rangle|0\rangle \to \sum_x |x\rangle|\left(\frac{x}{N}\right)\rangle$

Inputs:

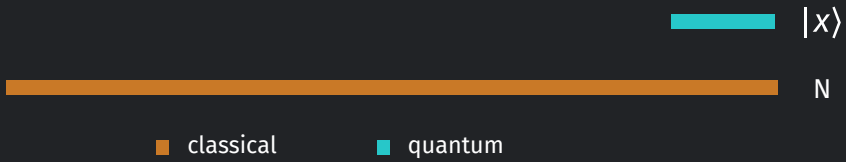
- N, a big (n-bit) classical number
- $|x\rangle$, a small (m-bit) quantum number

Existing circuits

- Take two quantum inputs
- Both **gate** and **qubit** count linear in the larger input
- Using them directly would yield $O(n)$ qubit count 😢

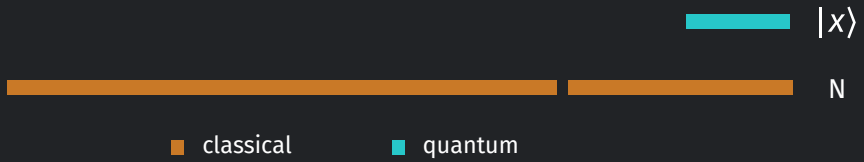
Streaming classical bits to a quantum circuit

Key idea: using properties of the Jacobi symbol, find an "equivalent" m-bit value $|N'\rangle$ such that we only need to compute $\left(\frac{x}{N'}\right)$



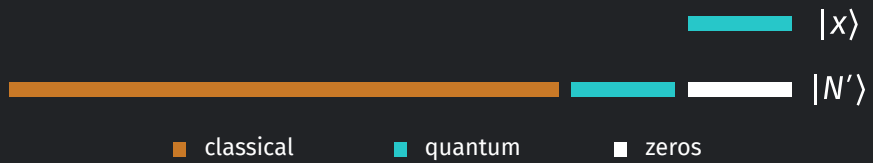
Streaming classical bits to a quantum circuit

Key idea: using properties of the Jacobi symbol, find an "equivalent" m-bit value $|N'\rangle$ such that we only need to compute $\left(\frac{x}{N'}\right)$



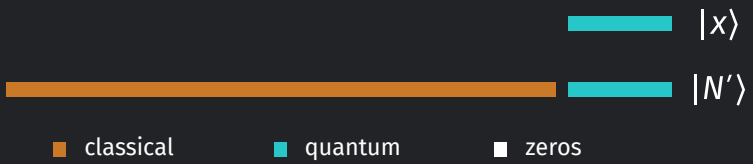
Streaming classical bits to a quantum circuit

Key idea: using properties of the Jacobi symbol, find an "equivalent" m-bit value $|N'\rangle$ such that we only need to compute $\left(\frac{x}{N'}\right)$



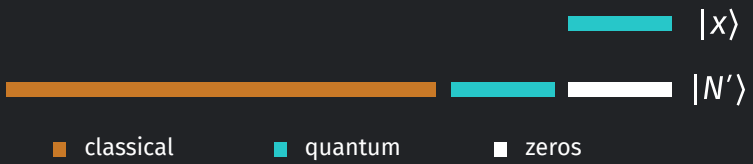
Streaming classical bits to a quantum circuit

Key idea: using properties of the Jacobi symbol, find an "equivalent" m-bit value $|N'\rangle$ such that we only need to compute $\left(\frac{x}{N'}\right)$



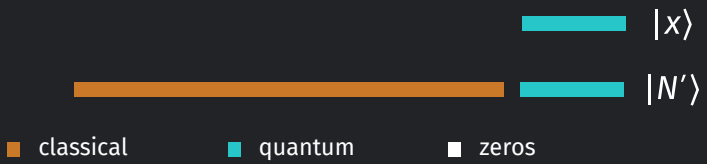
Streaming classical bits to a quantum circuit

Key idea: using properties of the Jacobi symbol, find an "equivalent" m-bit value $|N'\rangle$ such that we only need to compute $\left(\frac{x}{N'}\right)$



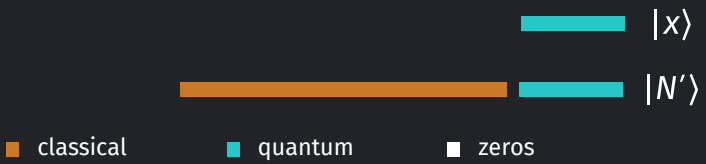
Streaming classical bits to a quantum circuit

Key idea: using properties of the Jacobi symbol, find an "equivalent" m-bit value $|N'\rangle$ such that we only need to compute $\left(\frac{x}{N'}\right)$



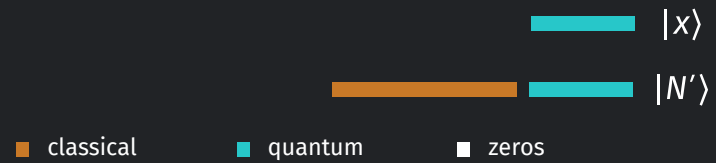
Streaming classical bits to a quantum circuit

Key idea: using properties of the Jacobi symbol, find an "equivalent" m-bit value $|N'\rangle$ such that we only need to compute $\left(\frac{x}{N'}\right)$



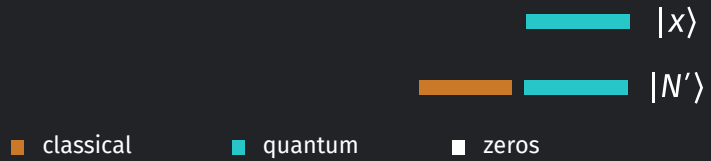
Streaming classical bits to a quantum circuit

Key idea: using properties of the Jacobi symbol, find an "equivalent" m-bit value $|N'\rangle$ such that we only need to compute $\left(\frac{x}{N'}\right)$



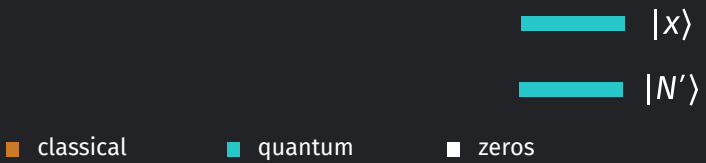
Streaming classical bits to a quantum circuit

Key idea: using properties of the Jacobi symbol, find an "equivalent" m-bit value $|N'\rangle$ such that we only need to compute $\left(\frac{x}{N'}\right)$



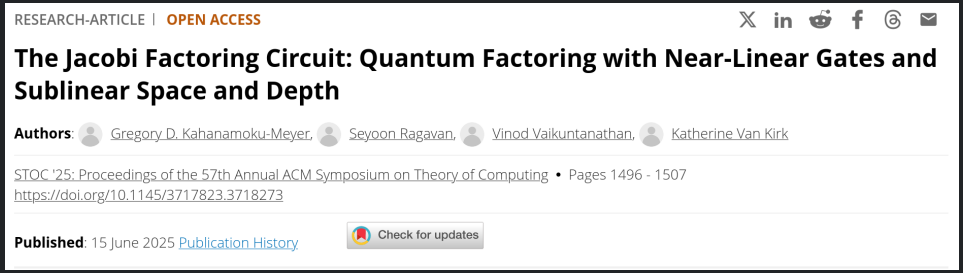
Streaming classical bits to a quantum circuit

Key idea: using properties of the Jacobi symbol, find an "equivalent" m-bit value $|N'\rangle$ such that we only need to compute $\left(\frac{x}{N'}\right)$



Now with two length-m quantum inputs,
standard circuits for $\left(\frac{x}{N'}\right)$ have depth and qubits $\widetilde{O}(m)$

Summary: Jacobi factoring



RESEARCH-ARTICLE | OPEN ACCESS

The Jacobi Factoring Circuit: Quantum Factoring with Near-Linear Gates and Sublinear Space and Depth

Authors: Gregory D. Kahanamoku-Meyer, Seyoon Ragavan, Vinod Vaikuntanathan, Katherine Van Kirk

STOC '25: Proceedings of the 57th Annual ACM Symposium on Theory of Computing • Pages 1496 - 1507
https://doi.org/10.1145/3717823.3718273

Published: 15 June 2025 Publication History

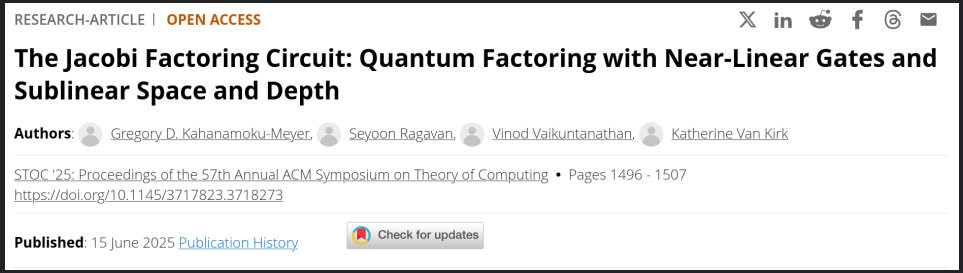
Check for updates

Quantum algorithm for factoring integers of the **special form** $N = P^2Q$.

Key: For $N = P^2Q$, $\left(\frac{x}{N}\right)$ has period Q

Qubit count: $\widetilde{O}(\log Q)$

Summary: Jacobi factoring



RESEARCH-ARTICLE | OPEN ACCESS

The Jacobi Factoring Circuit: Quantum Factoring with Near-Linear Gates and Sublinear Space and Depth

Authors: Gregory D. Kahanamoku-Meyer, Seyoon Ragavan, Vinod Vaikuntanathan, Katherine Van Kirk

STOC '25: Proceedings of the 57th Annual ACM Symposium on Theory of Computing • Pages 1496 - 1507
https://doi.org/10.1145/3717823.3718273

Published: 15 June 2025 Publication History

Check for updates

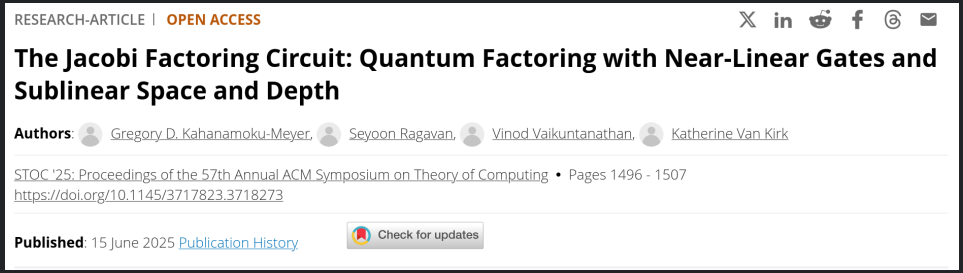
Quantum algorithm for factoring integers of the **special form** $N = P^2Q$.

Key: For $N = P^2Q$, $\left(\frac{x}{N}\right)$ has period Q

Qubit count: $\widetilde{O}(\log Q)$

Does this work for RSA integers?

Summary: Jacobi factoring



RESEARCH-ARTICLE | OPEN ACCESS

The Jacobi Factoring Circuit: Quantum Factoring with Near-Linear Gates and Sublinear Space and Depth

Authors: Gregory D. Kahanamoku-Meyer, Seyoon Ragavan, Vinod Vaikuntanathan, Katherine Van Kirk

STOC '25: Proceedings of the 57th Annual ACM Symposium on Theory of Computing • Pages 1496 - 1507
https://doi.org/10.1145/3717823.3718273

Published: 15 June 2025 Publication History

Check for updates

Quantum algorithm for factoring integers of the **special form** $N = P^2Q$.

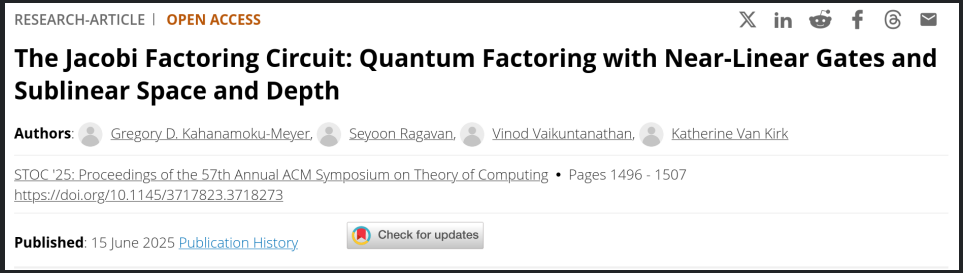
Key: For $N = P^2Q$, $\left(\frac{x}{N}\right)$ has period Q

Qubit count: $\widetilde{O}(\log Q)$

Does this work for RSA integers?

❌ For $N = PQ$, $\left(\frac{x}{N}\right)$ has period N

Summary: Jacobi factoring



RESEARCH-ARTICLE | OPEN ACCESS

The Jacobi Factoring Circuit: Quantum Factoring with Near-Linear Gates and Sublinear Space and Depth

Authors: Gregory D. Kahanamoku-Meyer, Seyoon Ragavan, Vinod Vaikuntanathan, Katherine Van Kirk

STOC '25: Proceedings of the 57th Annual ACM Symposium on Theory of Computing • Pages 1496 - 1507
https://doi.org/10.1145/3717823.3718273

Published: 15 June 2025 Publication History

Check for updates

Quantum algorithm for factoring integers of the **special form** $N = P^2Q$.

Key: For $N = P^2Q$, $\left(\frac{x}{N}\right)$ has period Q

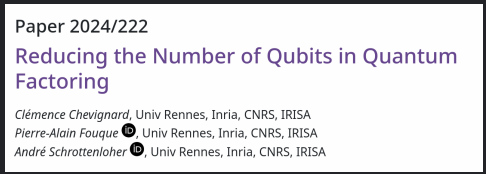
Qubit count: $\widetilde{O}(\log Q)$

Does this work for RSA integers?

❌ For $N = PQ$, $\left(\frac{x}{N}\right)$ has period N

What can we do instead?

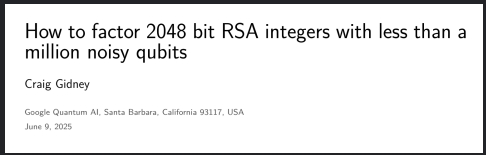
Very space-efficient discrete log modulo N (and thus RSA factoring)


Paper 2024/222

Reducing the Number of Qubits in Quantum Factoring

Clémence Chevignard, Univ Rennes, Inria, CNRS, IRISA
Pierre-Alain Fouque, Univ Rennes, Inria, CNRS, IRISA
André Schrottenloher, Univ Rennes, Inria, CNRS, IRISA

eprint:2024/222

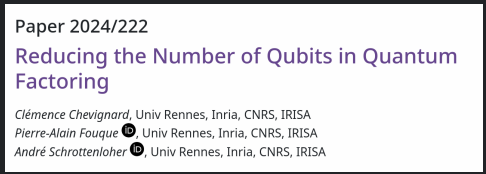

How to factor 2048 bit RSA integers with less than a million noisy qubits

Craig Gidney

Google Quantum AI, Santa Barbara, California 93117, USA
June 9, 2025

arXiv:2505.15917

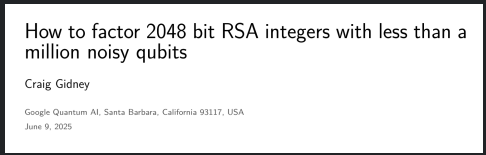
Very space-efficient discrete log modulo N (and thus RSA factoring)


Paper 2024/222

Reducing the Number of Qubits in Quantum Factoring

Clémence Chevignard, Univ Rennes, Inria, CNRS, IRISA
Pierre-Alain Fouque, Univ Rennes, Inria, CNRS, IRISA
André Schrottenloher, Univ Rennes, Inria, CNRS, IRISA

eprint:2024/222


How to factor 2048 bit RSA integers with less than a million noisy qubits

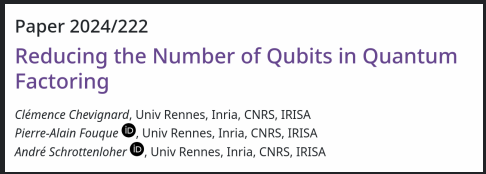
Craig Gidney

Google Quantum AI, Santa Barbara, California 93117, USA
June 9, 2025

arXiv:2505.15917

Recall the *discrete log modulo N:*
$\Rightarrow$ given g, N and $x = g^d \bmod N$, find d.

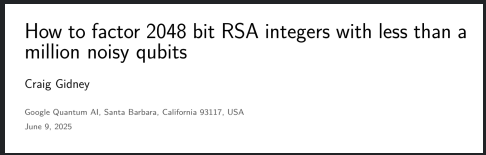
Very space-efficient discrete log modulo N (and thus RSA factoring)


Paper 2024/222

Reducing the Number of Qubits in Quantum Factoring

Clémence Chevignard, Univ Rennes, Inria, CNRS, IRISA
Pierre-Alain Fouque, Univ Rennes, Inria, CNRS, IRISA
André Schrottenloher, Univ Rennes, Inria, CNRS, IRISA

eprint:2024/222


How to factor 2048 bit RSA integers with less than a million noisy qubits

Craig Gidney

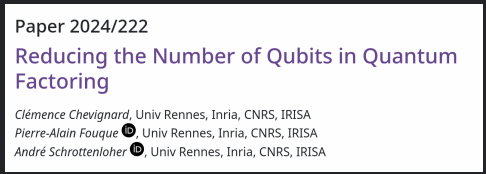
Google Quantum AI, Santa Barbara, California 93117, USA
June 9, 2025

arXiv:2505.15917

Recall the *discrete log modulo N:*
$\Rightarrow$ given g, N and $x = g^d \bmod N$, find d.

Result: quantum circuit to find $\mathrm{dlog}_g(x) = d$ using $\sim \log d$ qubits.

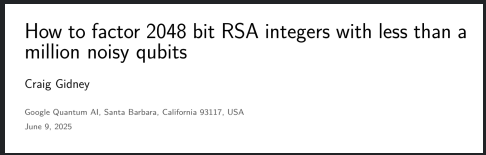
Very space-efficient discrete log modulo N (and thus RSA factoring)


Paper 2024/222

Reducing the Number of Qubits in Quantum Factoring

Clémence Chevignard, Univ Rennes, Inria, CNRS, IRISA
Pierre-Alain Fouque, Univ Rennes, Inria, CNRS, IRISA
André Schrottenloher, Univ Rennes, Inria, CNRS, IRISA

eprint:2024/222


How to factor 2048 bit RSA integers with less than a million noisy qubits

Craig Gidney

Google Quantum AI, Santa Barbara, California 93117, USA
June 9, 2025

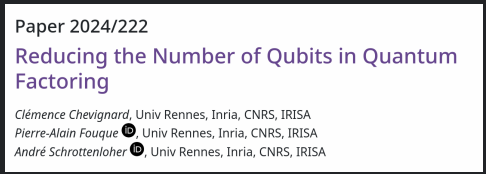
arXiv:2505.15917

Recall the *discrete log modulo N:*
$\Rightarrow$ given g, N and $x = g^d \bmod N$, find d.

Result: quantum circuit to find $\mathrm{dlog}_g(x) = d$ using $\sim \log d$ qubits.

Recall: To factor n-bit RSA integers, take a discrete log where $\log d \sim n/2$.

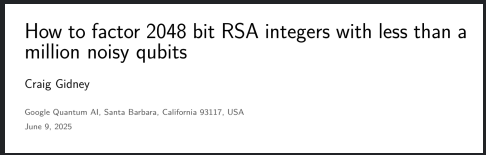
Very space-efficient discrete log modulo N (and thus RSA factoring)


Paper 2024/222

Reducing the Number of Qubits in Quantum Factoring

Clémence Chevignard, Univ Rennes, Inria, CNRS, IRISA
Pierre-Alain Fouque, Univ Rennes, Inria, CNRS, IRISA
André Schrottenloher, Univ Rennes, Inria, CNRS, IRISA

eprint:2024/222


How to factor 2048 bit RSA integers with less than a million noisy qubits

Craig Gidney

Google Quantum AI, Santa Barbara, California 93117, USA
June 9, 2025

arXiv:2505.15917

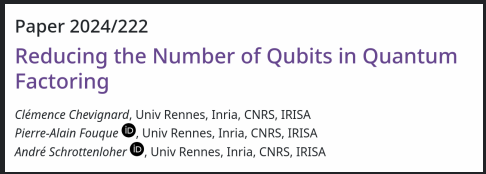
Recall the *discrete log modulo N:*
$\Rightarrow$ given g, N and $x = g^d \bmod N$, find d.

Result: quantum circuit to find $\mathrm{dlog}_g(x) = d$ using $\sim \log d$ qubits.

Recall: To factor n-bit RSA integers, take a discrete log where $\log d \sim n/2$.

Corollary: Factoring n-bit RSA integers with $\sim n/2$ qubits.

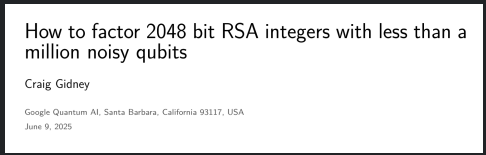
Public service announcement: beware of overly focusing on factoring!


Paper 2024/222

Reducing the Number of Qubits in Quantum Factoring

Clémence Chevignard, Univ Rennes, Inria, CNRS, IRISA
Pierre-Alain Fouque, Univ Rennes, Inria, CNRS, IRISA
André Schrottenloher, Univ Rennes, Inria, CNRS, IRISA

eprint:2024/222


How to factor 2048 bit RSA integers with less than a million noisy qubits

Craig Gidney

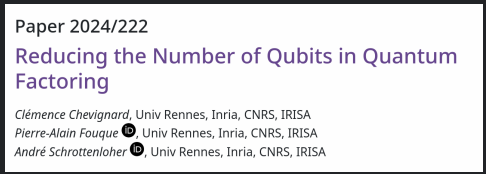
Google Quantum AI, Santa Barbara, California 93117, USA
June 9, 2025

arXiv:2505.15917

Recall the *discrete log modulo N:*
$\Rightarrow$ given g, N and $x = g^d \bmod N$, find d.

Result: quantum circuit to find $\mathrm{dlog}_g(x) = d$ using $\sim \log d$ qubits.

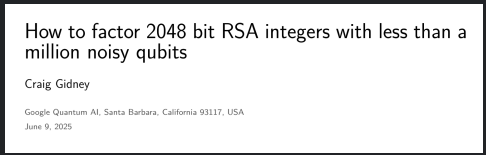
Public service announcement: beware of overly focusing on factoring!


Paper 2024/222

Reducing the Number of Qubits in Quantum Factoring

Clémence Chevignard, Univ Rennes, Inria, CNRS, IRISA
Pierre-Alain Fouque, Univ Rennes, Inria, CNRS, IRISA
André Schrottenloher, Univ Rennes, Inria, CNRS, IRISA

eprint:2024/222


How to factor 2048 bit RSA integers with less than a million noisy qubits

Craig Gidney

Google Quantum AI, Santa Barbara, California 93117, USA
June 9, 2025

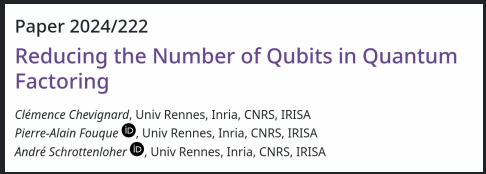
arXiv:2505.15917

Recall the *discrete log modulo N:*
$\Rightarrow$ given g, N and $x = g^d \bmod N$, find d.

Result: quantum circuit to find $\mathrm{dlog}_g(x) = d$ using $\sim \log d$ qubits.

Proof of quantumness: forget about factoring...

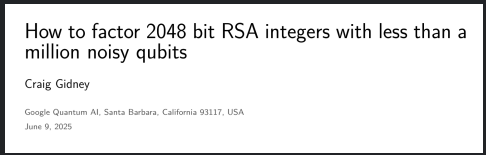
Public service announcement: beware of overly focusing on factoring!


Paper 2024/222

Reducing the Number of Qubits in Quantum Factoring

Clémence Chevignard, Univ Rennes, Inria, CNRS, IRISA
Pierre-Alain Fouque, Univ Rennes, Inria, CNRS, IRISA
André Schrottenloher, Univ Rennes, Inria, CNRS, IRISA

eprint:2024/222


How to factor 2048 bit RSA integers with less than a million noisy qubits

Craig Gidney

Google Quantum AI, Santa Barbara, California 93117, USA
June 9, 2025

arXiv:2505.15917

Recall the *discrete log modulo N:*
$\Rightarrow$ given g, N and $x = g^d \bmod N$, find d.

Result: quantum circuit to find $\mathrm{dlog}_g(x) = d$ using $\sim \log d$ qubits.

Proof of quantumness: forget about factoring...

What if we set up a discrete log problem in which $\log d$ is as small as possible?

Recall: costs from beginning of talk

Note: these are **logical** costs, in the abstract circuit model.

Elliptic curve discrete log

Given $x = [d]g$, find d.

For curve `secp256k1`:

- Qubits: ~ 1200
- Toffolis: $\sim 2^{26}$

Babbush et al. '26 (arXiv:2603.28846)
Schrottenloher '26 (arXiv:2606.02235)

RSA integer factorization

Given $N = pq$, find p and q.

For RSA-2048:

- Qubits: ~ 1400
- Toffolis: $\sim 2^{33}$

Gidney '25 (arXiv:2505.15917)

Recall: costs from beginning of talk

Note: these are **logical** costs, in the abstract circuit model.

Elliptic curve discrete log

Given $x = [d]g$, find d.

For curve `secp256k1`:

- Qubits: ~ 1200
- Toffolis: $\sim 2^{26}$

Babbush et al. ’26 (arXiv:2603.28846)
Schrottenloher ’26 (arXiv:2606.02235)

RSA integer factorization

Given $N = pq$, find p and q.

For RSA-2048:

- Qubits: ~ 1400
- Toffolis: $\sim 2^{33}$

Gidney ’25 (arXiv:2505.15917)

Finite field discrete log

Given $x \equiv g^d \pmod N$ find d.

For 2048-bit N and 225-bit d:

Recall: costs from beginning of talk

Note: these are **logical** costs, in the abstract circuit model.

Elliptic curve discrete log

Given $x = [d]g$, find d.

For curve `secp256k1`:

- Qubits: ~ 1200
- Toffolis: $\sim 2^{26}$

Babbush et al. '26 (arXiv:2603.28846)
Schrottenloher '26 (arXiv:2606.02235)

RSA integer factorization

Given $N = pq$, find p and q.

For RSA-2048:

- Qubits: ~ 1400
- Toffolis: $\sim 2^{33}$

Gidney '25 (arXiv:2505.15917)

Finite field discrete log

Given $x \equiv g^d \pmod N$ find d.

For 2048-bit N and 225-bit d:

- Qubits: ~ 350
- Toffolis: $\sim 2^{26}$

(note: preliminary numbers!)

Recall: costs from beginning of talk

Note: these are **logical** costs, in the abstract circuit model.

Elliptic curve discrete log

Given $x = [d]g$, find d.

For curve `secp256k1`:

- Qubits: ~ 1200
- Toffolis: $\sim 2^{26}$

Babbush et al. '26 (arXiv:2603.28846)
Schrottenloher '26 (arXiv:2606.02235)

RSA integer factorization

Given $N = pq$, find p and q.

For RSA-2048:

- Qubits: ~ 1400
- Toffolis: $\sim 2^{33}$

Gidney '25 (arXiv:2505.15917)

Finite field discrete log

Given $x \equiv g^d \pmod N$ find d.

For 2048-bit N and 225-bit d:

- Qubits: ~ 350
- Toffolis: $\sim 2^{26}$

(note: preliminary numbers!)

Good thing finite field discrete log isn't really used in practice, right?

Finite-field discrete log is used in standardized cryptography

Finite-field discrete log is used in standardized cryptography

RFC **7919:** Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)

D. Gillmor

PROPOSED STANDARD

Internet Engineering Task Force (IETF) — D. Gillmor
Request for Comments: 7919 — ACLU
Updates: 2246, 4346, 4492, 5246 — August 2016
Category: Standards Track
ISSN: 2070-1721

Finite-field discrete log is used in standardized cryptography

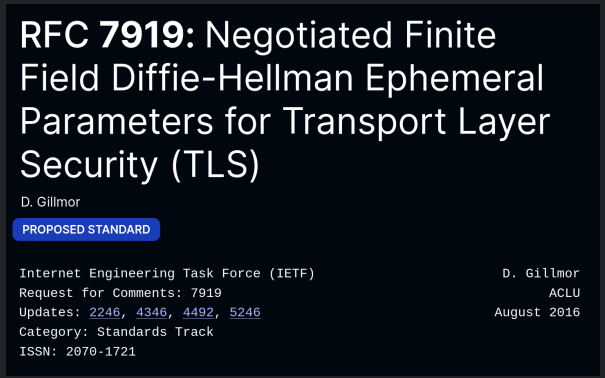
RFC **7919:** Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)

D. Gillmor

PROPOSED STANDARD

Internet Engineering Task Force (IETF) — D. Gillmor
Request for Comments: 7919 — ACLU
Updates: 2246, 4346, 4492, 5246 — August 2016
Category: Standards Track
ISSN: 2070-1721

Finite-field discrete log is used in standardized cryptography



OK, but surely the parameters aren’t set so the exponent is small, right?

The parameters are absolutely set like that

RFC 7919:

“Peers using [a 2048-bit modulus] that want to optimize their key exchange with a short exponent (Section 5.2) should choose a secret key of at least 225 bits.”

The parameters are absolutely set like that

RFC 7919:

“Peers using [a 2048-bit modulus] that want to optimize their key exchange with a short exponent (Section 5.2) should choose a secret key of at least 225 bits.”

Comparing costs

Note: these are **logical** costs, in the abstract circuit model.

Elliptic curve discrete log

Given $x = [d]g$, find d.

For curve `secp256k1`:

- Qubits: ~ 1200
- Toffolis: $\sim 2^{26}$

Babbush et al. '26 (arXiv:2603.28846)
Schrottenloher '26 (arXiv:2606.02235)

RSA integer factorization

Given $N = pq$, find p and q.

For RSA-2048:

- Qubits: ~ 1400
- Toffolis: $\sim 2^{33}$

Gidney '25 (arXiv:2505.15917)

Finite field discrete log

Given $x \equiv g^d \pmod N$ find d.

For 2048-bit N and 225-bit d:

- Qubits: ~ 350
- Toffolis: $\sim 2^{26}$

(note: preliminary!)

All of these are relevant for real-world cryptography!!

Summary

Assumption: input length is a lower bound on qubit count

Summary

Assumption: input length is a lower bound on qubit count

Two number-theory counterexamples:

Jacobi factoring

Can factor n-bit numbers $N = P^2Q$
with only $\widetilde{O}(\log Q)$ qubits!

Summary

Assumption: input length is a lower bound on qubit count

Two number-theory counterexamples:

Jacobi factoring

Can factor n-bit numbers $N = P^2Q$ with only $\widetilde{O}(\log Q)$ qubits!

We may set $\log Q$ as low as $O(n^{2/3})$ without affecting classical hardness

Residue number system arithmetic

Can take the discrete logarithm

$$\text{dlog}(g^d \bmod N) \Rightarrow d$$

using only $\widetilde{O}(\log d) < \log N$ qubits!

Summary

Assumption: input length is a lower bound on qubit count

Two number-theory counterexamples:

Jacobi factoring

Can factor n-bit numbers $N = P^2Q$ with only $\widetilde{O}(\log Q)$ qubits!

We may set $\log Q$ as low as $O(n^{2/3})$ without affecting classical hardness

Residue number system arithmetic

Can take the discrete logarithm

$\text{dlog}(g^d \bmod N) \Rightarrow d$

using only $\widetilde{O}(\log d) < \log N$ qubits!

Key insight: *streaming*—the qubits never need to hold the entire input at once

Overview

Assumption: factoring is the right thing to focus on

Assumption: input length is a lower bound on qubit count

Assumption: logical qubit count is what matters

Assumption: quantum circuits must be (mostly) reversible

Assumption: breaking cryptography is a long way off

Metrics for cost

How should we balance logical *qubits* vs *Toffolis*?

Metrics for cost

How should we balance logical *qubits* vs *Toffolis*? (not to mention depth, other gates, ...)

Metrics for cost

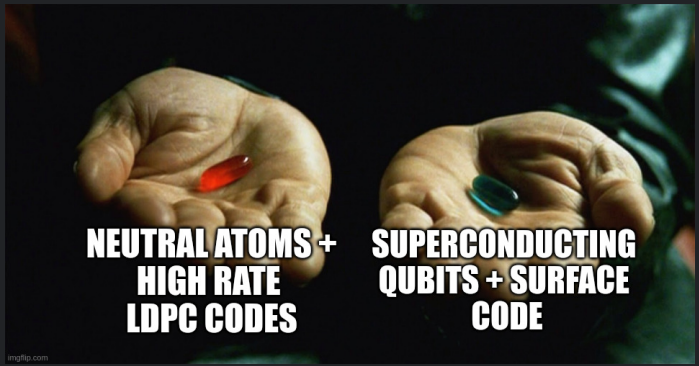
How should we balance logical *qubits* vs *Toffolis*? (not to mention depth, other gates, ...)

Depends strongly on architecture!

Metrics for cost

How should we balance logical *qubits* vs *Toffolis*? (not to mention depth, other gates, ...)

Depends strongly on architecture!



Metrics for cost

How should we balance logical *qubits* vs *Toffolis*? (not to mention depth, other gates, ...)

Depends strongly on architecture!

Surface code
+ superconducting

High rate LDPC
+ neutral atoms

Metrics for cost

How should we balance logical *qubits* vs *Toffolis*? (not to mention depth, other gates, ...)

Depends strongly on architecture!

Surface code
+ superconducting

- Fast cycle time

High rate LDPC
+ neutral atoms

- Slow cycle time

Metrics for cost

How should we balance logical *qubits* vs *Toffolis*? (not to mention depth, other gates, ...)

Depends strongly on architecture!

Surface code
+ superconducting

- Fast cycle time
- Many physical qubits per logical

High rate LDPC
+ neutral atoms

- Slow cycle time
- Fewer physical qubits per logical

Metrics for cost

How should we balance logical *qubits* vs *Toffolis*? (not to mention depth, other gates, ...)

Depends strongly on architecture!

Surface code
+ superconducting

- Fast cycle time
- Many physical qubits per logical
- Parallelism has big space footprint

High rate LDPC
+ neutral atoms

- Slow cycle time
- Fewer physical qubits per logical
- Parallelism more achievable (?)

Metrics for cost

How should we balance logical *qubits* vs *Toffolis*? (not to mention depth, other gates, ...)

Depends strongly on architecture!

Surface code + superconducting

- Fast cycle time
- Many physical qubits per logical
- Parallelism has big space footprint
- **Minimize logical qubits at (almost) all costs**

High rate LDPC + neutral atoms

- Slow cycle time
- Fewer physical qubits per logical
- Parallelism more achievable (?)

Metrics for cost

How should we balance logical *qubits* vs *Toffolis*? (not to mention depth, other gates, ...)

Depends strongly on architecture!

Surface code + superconducting

- Fast cycle time
- Many physical qubits per logical
- Parallelism has big space footprint
- **Minimize logical qubits at (almost) all costs**

High rate LDPC + neutral atoms

- Slow cycle time
- Fewer physical qubits per logical
- Parallelism more achievable (?)
- **Balance depth and logical qubit count?**

Metrics for cost

How should we balance logical *qubits* vs *Toffolis*? (not to mention depth, other gates, ...)

Depends strongly on architecture!

Surface code + superconducting

- Fast cycle time
- Many physical qubits per logical
- Parallelism has big space footprint
- **Minimize logical qubits at (almost) all costs**

High rate LDPC + neutral atoms

- Slow cycle time
- Fewer physical qubits per logical
- Parallelism more achievable (?)
- **Balance depth and logical qubit count?**

Many algorithmic decisions made with **surface code in mind.**

Metrics for cost

How should we balance logical *qubits* vs *Toffolis*? (not to mention depth, other gates, ...)

Depends strongly on architecture!

Surface code
+ superconducting

- Fast cycle time
- Many physical qubits per logical
- Parallelism has big space footprint
- **Minimize logical qubits at (almost) all costs**

High rate LDPC
+ neutral atoms

- Slow cycle time
- Fewer physical qubits per logical
- Parallelism more achievable (?)
- **Balance depth and logical qubit count?**

Many algorithmic decisions made with **surface code in mind.**
Take a second look at old constructions under new heuristics?

Overview

Assumption: factoring is the right thing to focus on

Assumption: input length is a lower bound on qubit count

Assumption: logical qubit count is what matters

Assumption: quantum circuits must be (mostly) reversible

Assumption: breaking cryptography is a long way off

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$x$$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_1(x)$$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_2(f_1(x))$$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_{k-1}(\cdots f_2(f_1(x))\cdots)$$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $x_i = f_i(\cdots)$. One quantum way to do it:

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$x$$

Let $x_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle$$

Using measurements to uncompute

How to do $|x\rangle\,|0\rangle \to |x\rangle\,|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_1(x)$$

Let $x_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\,|x_1\rangle$$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_2(f_1(x))$$

Let $x_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\ |x_1\rangle\ |x_2\rangle$$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_{k-1}(\cdots f_2(f_1(x))\cdots)$$

Let $x_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\,|x_1\rangle\,|x_2\rangle\,\cdots\,|x_{k-1}\rangle$$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $x_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\ |x_1\rangle\ |x_2\rangle\ \cdots\ |x_{k-1}\rangle\ |f(x)\rangle$$

Using measurements to uncompute

How to do $|x\rangle\,|0\rangle \to |x\rangle\,|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $x_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\,|x_1\rangle\,|x_2\rangle\,\cdots \qquad\qquad |f(x)\rangle$$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $x_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\,|x_1\rangle\,|x_2\rangle \qquad\qquad |f(x)\rangle$$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $x_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\,|x_1\rangle \qquad\qquad |f(x)\rangle$$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

😃 Total quantum steps: $2k-1$ (optimal)

Let $x_i = f_i(\cdots)$. One quantum way to do it:

$|x\rangle$ $|f(x)\rangle$

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $x_i = f_i(\cdots)$. One quantum way to do it:

$|x\rangle$ $|f(x)\rangle$

😃 Total quantum steps: $2k-1$ (optimal)

😢 Requires $O(k)$ ancilla registers

Using measurements to uncompute

How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently
when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $x_i = f_i(\cdots)$. One quantum way to do it:

$|x\rangle$ $|f(x)\rangle$

😃 Total quantum steps: $2k-1$ (optimal)

😢 Requires $O(k)$ ancilla registers

Is it possible to do better?

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$

x x_1 x_2 x_3 $f(x)$

State: $|x\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle$

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$

x x_1 x_2 x_3 $f(x)$

State: $|x\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle$

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$

x x_1 x_2 x_3 $f(x)$

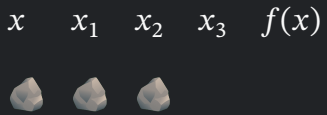
State: $|x\rangle\,|x_1\rangle\,|0\rangle\,|0\rangle\,|0\rangle$

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



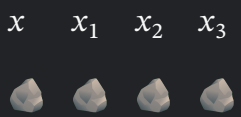
State: $|x\rangle\,|x_1\rangle\,|x_2\rangle\,|0\rangle\,|0\rangle$

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



State: $|x\rangle\,|x_1\rangle\,|x_2\rangle\,|x_3\rangle\,|0\rangle$

Pebble games for reversible computation

[Bennett '89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$

x x_1 x_2 x_3 $f(x)$

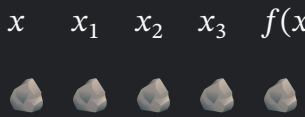
State: $|x\rangle\,|x_1\rangle\,|x_2\rangle\,|x_3\rangle\,|f(x)\rangle$

Pebble games for reversible computation

[Bennett '89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)
- Removing a pebble → uncomputing that value (*frees space*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



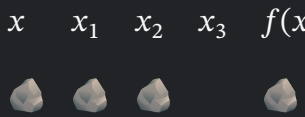
State: $|x\rangle\,|x_1\rangle\,|x_2\rangle\,|x_3\rangle\,|f(x)\rangle$

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)
- Removing a pebble → uncomputing that value (*frees space*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



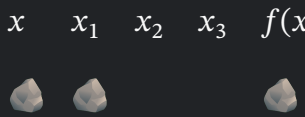
State: $|x\rangle\,|x_1\rangle\,|x_2\rangle\,|0\rangle\,|f(x)\rangle$

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)
- Removing a pebble → uncomputing that value (*frees space*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



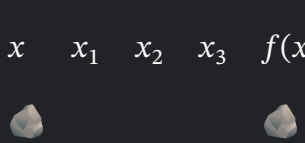
State: $|x\rangle\,|x_1\rangle\,|0\rangle\,|0\rangle\,|f(x)\rangle$

Pebble games for reversible computation

[Bennett '89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)
- Removing a pebble → uncomputing that value (*frees space*)

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



State: $|x\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|f(x)\rangle$

Pebble games for reversible computation

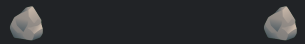
[Bennett '89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)
- Removing a pebble → uncomputing that value (*frees space*)

Game rules

- **Start:** pebble on x
- **Goal:** pebble on $f(x)$; no pebbles on x_i
- Can only place/remove if value to the left has a pebble

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



State: $|x\rangle\,|0\rangle\,|0\rangle\,|0\rangle\,|f(x)\rangle$

Pebble games for reversible computation

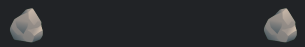
[Bennett '89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)
- Removing a pebble → uncomputing that value (*frees space*)

Game rules

- **Start:** pebble on x
- **Goal:** pebble on $f(x)$; no pebbles on x_i
- Can only place/remove if value to the left has a pebble

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$



Naive strategy

Time cost (# steps): $2k - 1$ (optimal) 😃

Pebble games for reversible computation

[Bennett ’89] introduced pebble games:

- Placing a pebble → computing a value (*uses new space*)
- Removing a pebble → uncomputing that value (*frees space*)

Game rules

- **Start:** pebble on x
- **Goal:** pebble on $f(x)$; no pebbles on x_i
- Can only place/remove if value to the left has a pebble

Let $f(x) = f_k(\cdots f_2(f_1(x))\cdots)$
and $x_i = f_i(\cdots)$

$x \quad x_1 \quad x_2 \quad x_3 \quad f(x)$


Naive strategy

Time cost (# steps): $2k - 1$ (optimal) 😃
Space (# pebbles): $O(k)$ 😭

Using less space: a recursive strategy

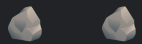
1. Pebble game from start to $k/2$

x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$

Using less space: a recursive strategy

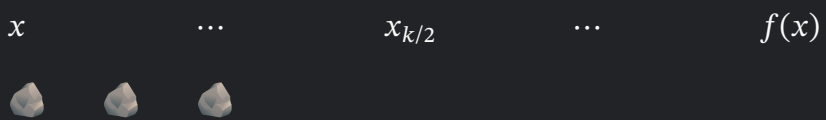
1. Pebble game from start to $k/2$

x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$



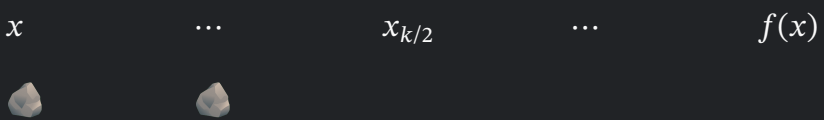
Using less space: a recursive strategy

1. Pebble game from start to $k/2$



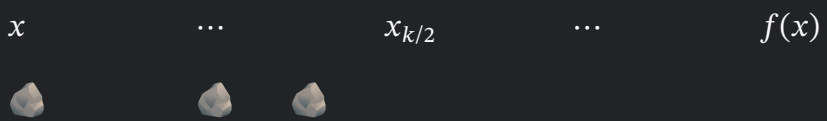
Using less space: a recursive strategy

1. Pebble game from start to $k/2$



Using less space: a recursive strategy

1. Pebble game from start to $k/2$



Using less space: a recursive strategy

1. Pebble game from start to $k/2$

Using less space: a recursive strategy

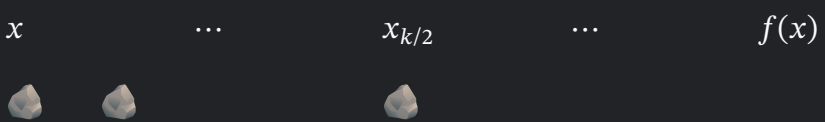
1. Pebble game from start to $k/2$

Using less space: a recursive strategy

1. Pebble game from start to $k/2$

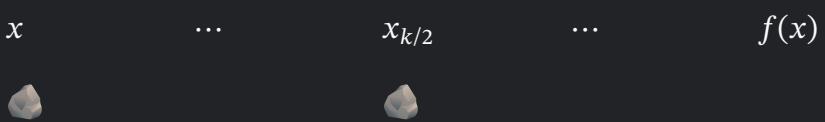
Using less space: a recursive strategy

1. Pebble game from start to $k/2$



Using less space: a recursive strategy

1. Pebble game from start to $k/2$



Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$

Using less space: a recursive strategy

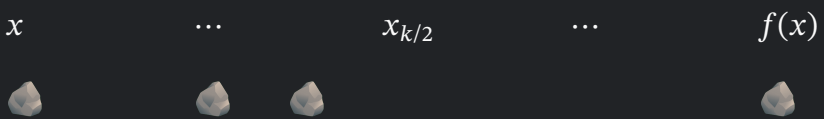
1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$

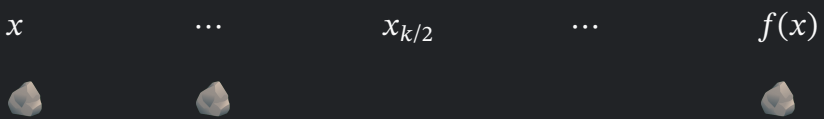
Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$



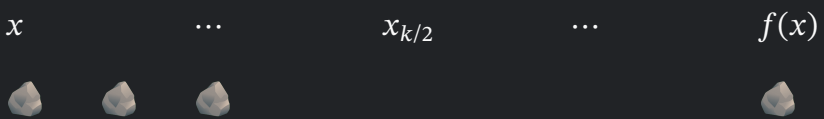
Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$



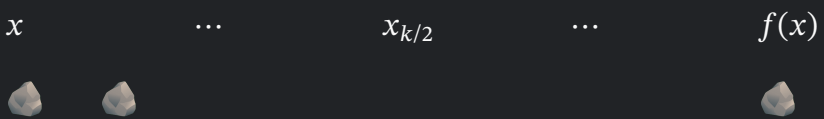
Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$



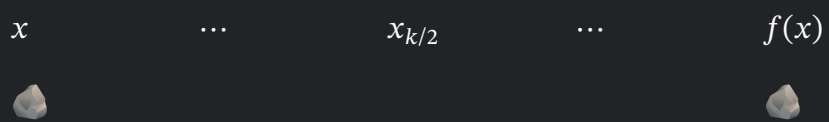
Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$



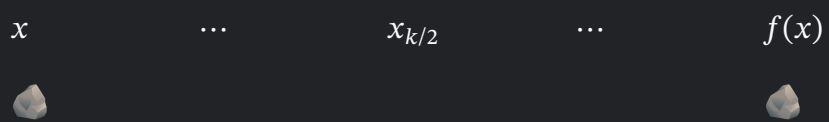
Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$



Using less space: a recursive strategy

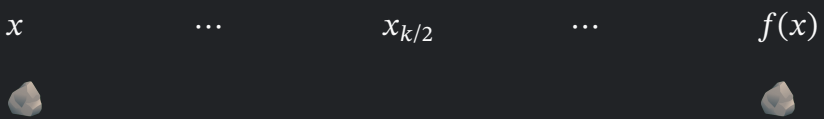
1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$



Space usage (# pebbles): $O(\log k)$ 😃

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$

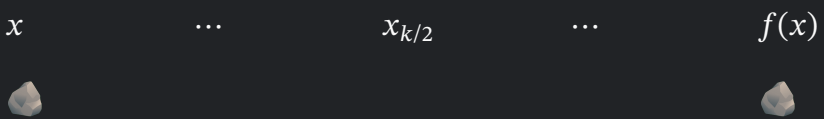


Space usage (# pebbles): $O(\log k)$ 😃

Time cost (# steps): $T(k) = 3T(k/2)$

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$



Space usage (# pebbles): $O(\log k)$ 😃

Time cost (# steps): $O(k^{\log_2 3}) \approx O(k^{1.58\cdots})$ steps 😢

Using less space: a recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$

x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$

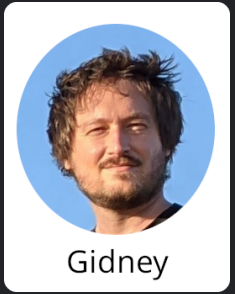

Space usage (# pebbles): $O(\log k)$ 😃

Time cost (# steps): $O(k^{\log_2 3}) \approx O(k^{1.58\cdots})$ steps 😢
Generalization: space $O(2^{1/\epsilon} \log k)$, time $O(k^{1+\epsilon})$ 😐

Spookiness: a quantum resource

“Spooky Pebble Games and Irreversible Uncomputation”
algassert.com/post/1905

Spookiness: a quantum resource



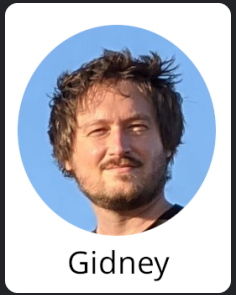
"Spooky Pebble Games and Irreversible Uncomputation"
algassert.com/post/1905

Using measurement to uncompute

Given an intermediate value

$$|x_i\rangle$$

Spookiness: a quantum resource



"Spooky Pebble Games and Irreversible Uncomputation"
algassert.com/post/1905

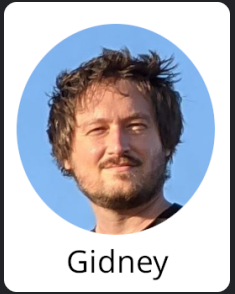
Using measurement to uncompute

Given an intermediate value

$$|x_i\rangle$$

what if we apply $H^{\otimes n}$ and then measure it?

Spookiness: a quantum resource



"Spooky Pebble Games and Irreversible Uncomputation"
algassert.com/post/1905

Using measurement to uncompute

Given an intermediate value

$$|x_i\rangle$$

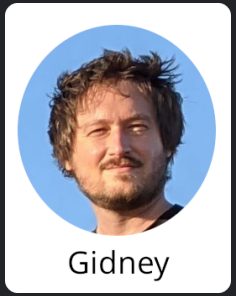
what if we apply $H^{\otimes n}$ and then measure it?

Get phase

$$(-1)^{d \cdot x_i}$$

where d is *classically-known*.

Spookiness: a quantum resource



"Spooky Pebble Games and Irreversible Uncomputation"
algassert.com/post/1905

Using measurement to uncompute

Given an intermediate value

$$|x_i\rangle$$

what if we apply $H^{\otimes n}$ and then measure it?

Get phase

$$(-1)^{d \cdot x_i}$$

where d is *classically-known*.

We've turned $|x_i\rangle$ into a ghost!

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost
- Need x_i again eventually to fix phase

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost
- Need x_i again eventually to fix phase

New rules:

- can only place or remove a pebble if there is a pebble to its left

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost
- Need x_i again eventually to fix phase

New rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost
- Need x_i again eventually to fix phase

New rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later

Spookiness: a quantum resource

Ghosting: replacing $|x_i\rangle$ with phase error $(-1)^{d \cdot x_i}$ for *classically-known* d

Features:

- Storing ghosts doesn't require qubits
- Don't need x_{i-1} to create a ghost
- Need x_i again eventually to fix phase

New rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later
- exorcism is free when x_i has a pebble

The power of spooky pebbling

Recursive strategy

1. Pebble game from start to $k/2$
2. Pebble game from $k/2$ to k
3. Pebble game from start to $k/2$

x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$

The power of spooky pebbling

Spooky recursive strategy

1. Spooky pebble game from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

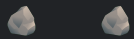
x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

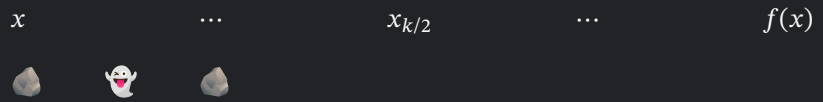
x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$



The power of spooky pebbling

Spooky recursive strategy

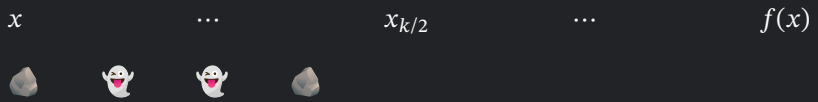
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

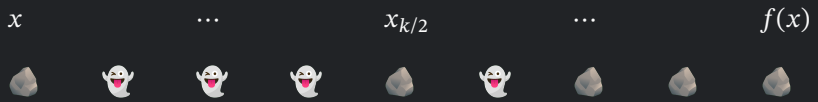
Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

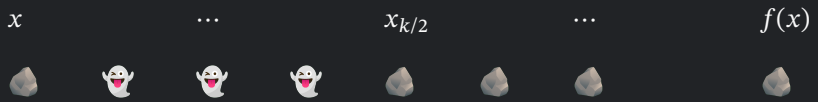
Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

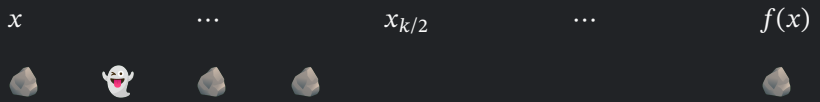
Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

The power of spooky pebbling

Spooky recursive strategy

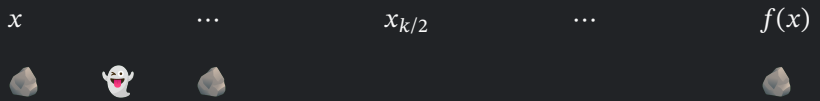
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

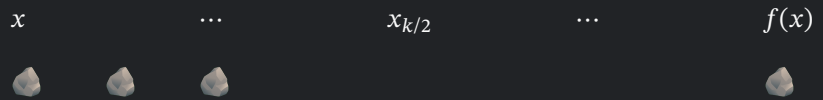
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

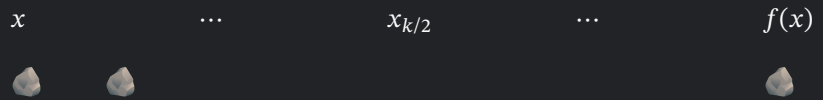
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

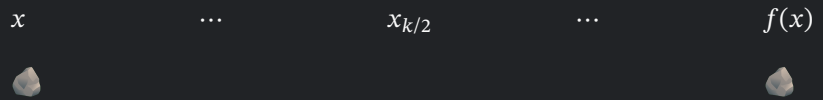
1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$



The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k
3. Spooky pebble game from start to $k/2$

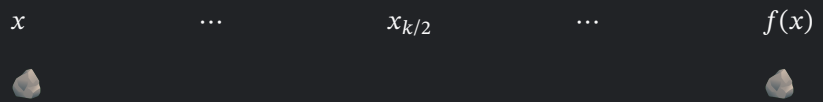
x $\cdots$ $x_{k/2}$ $\cdots$ $f(x)$


Space: $O(\log k)$ pebbles 😃 **Time cost (# steps):**

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$
2. Spooky pebble game from $k/2$ to k : $T(k/2)$ steps
3. Spooky pebble game from start to $k/2$: $T(k/2)$ steps

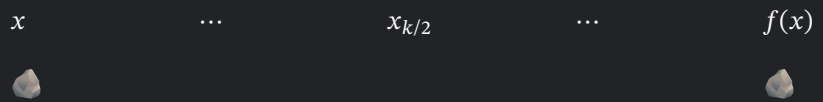


Space: $O(\log k)$ pebbles 😃 **Time cost (# steps):**

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$: $O(k)$ steps (compared to $T(k/2)$ without spookiness)
2. Spooky pebble game from $k/2$ to k : $T(k/2)$ steps
3. Spooky pebble game from start to $k/2$: $T(k/2)$ steps

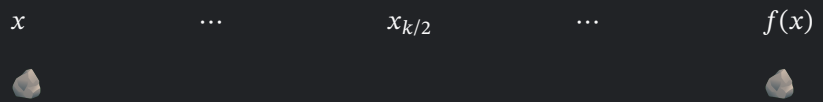


Space: $O(\log k)$ pebbles 😃 **Time cost (# steps):**

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$: $O(k)$ steps (compared to $T(k/2)$ without spookiness)
2. Spooky pebble game from $k/2$ to k : $T(k/2)$ steps
3. Spooky pebble game from start to $k/2$: $T(k/2)$ steps



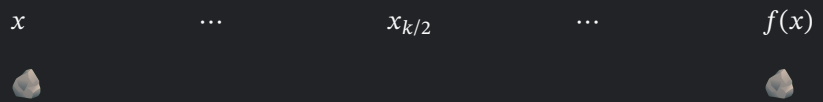
Space: $O(\log k)$ pebbles 😃

Time cost (# steps): $T(k) = O(k) + 2T(k/2)$

The power of spooky pebbling

Spooky recursive strategy

1. *Blast* from start to $k/2$: $O(k)$ steps (compared to $T(k/2)$ without spookiness)
2. Spooky pebble game from $k/2$ to k : $T(k/2)$ steps
3. Spooky pebble game from start to $k/2$: $T(k/2)$ steps



Space: $O(\log k)$ pebbles 😃

Time cost (# steps): $O(k \log k)$ steps 🤩

The power of spooky pebbling

Algorithm	Parallel	Spooky	Space	Depth
Trivial	N	N	k	$2k-1$
Bennett ’89	N	N	$O(2^{1/\epsilon} \cdot \log k)$	$O(k^{1+\epsilon})$
Gidney ’19 KSS ’25	N	Y	$O(\log k)$	$O(k \log k)$

The power of spooky pebbling

Algorithm	Parallel	Spooky	Space	Depth
Trivial	N	N	k	$2k-1$
Bennett ’89	N	N	$O(2^{1/\epsilon} \cdot \log k)$	$O(k^{1+\epsilon})$
Gidney ’19 KSS ’25	N	Y	$O(\log k)$	$O(k \log k)$

Is the best of both worlds possible?

The power of spooky pebbling

Algorithm	Parallel	Spooky	Space	Depth
Trivial	N	N	k	$2k-1$
Bennett '89	N	N	$O(2^{1/\epsilon} \cdot \log k)$	$O(k^{1+\epsilon})$
Gidney '19 KSS '25	N	Y	$O(\log k)$	$O(k \log k)$
KMRV '25	Y	Y	$O(\log k)$	$2k-1$

Is the best of both worlds possible?
Yes! Kahanamoku-Meyer, Ragavan, Van Kirk (arXiv:2510.08432)

Example: an optimal parallel spooky pebble game for $k = 12$

x x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} $f(x)$

Step number: 0

Max. pebble count: 1

Example: an optimal parallel spooky pebble game for $k = 12$

x x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} $f(x)$

Step number: 1

Max. pebble count: 2

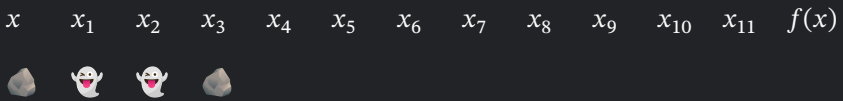
Example: an optimal parallel spooky pebble game for $k = 12$

x x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} $f(x)$

Step number: 2

Max. pebble count: 2

Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 3

Max. pebble count: 2

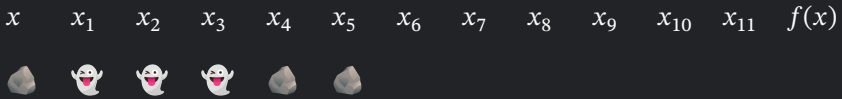
Example: an optimal parallel spooky pebble game for $k = 12$

x	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	$f(x)$
🪨	👻	👻	👻	🪨								

Step number: 4

Max. pebble count: 2

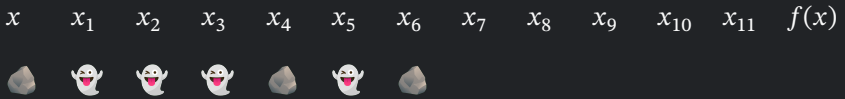
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 5

Max. pebble count: 3

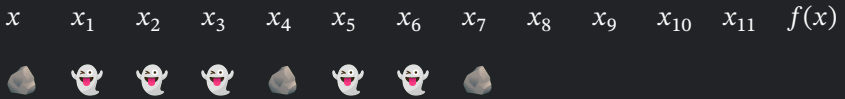
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 6

Max. pebble count: 3

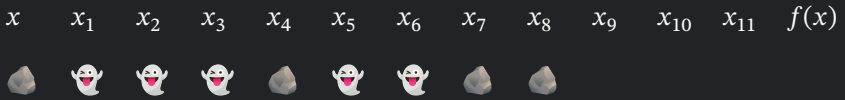
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 7

Max. pebble count: 3

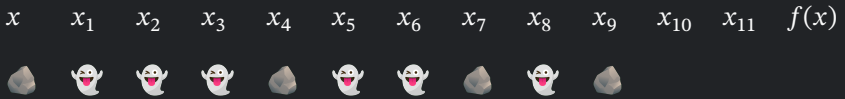
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 8

Max. pebble count: 4

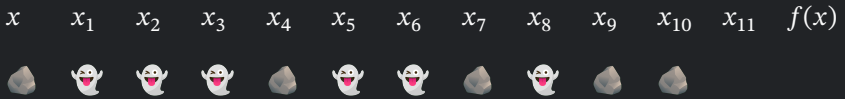
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 9

Max. pebble count: 4

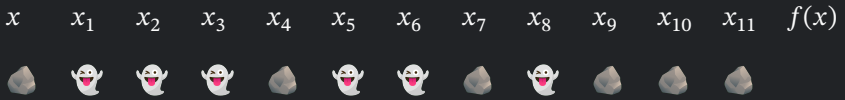
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 10

Max. pebble count: 5

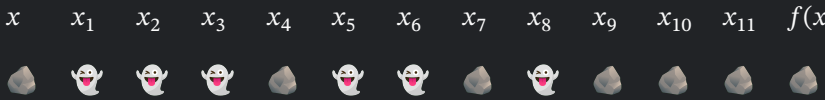
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 11

Max. pebble count: 6

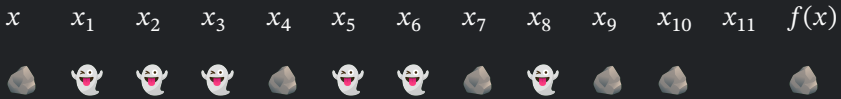
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 12

Max. pebble count: 7

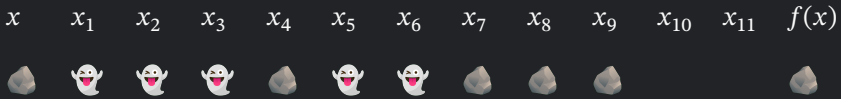
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 13

Max. pebble count: 7

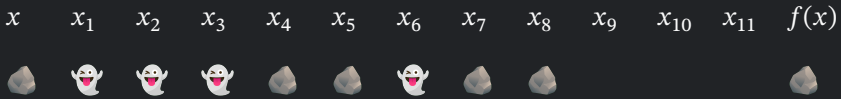
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 14

Max. pebble count: 7

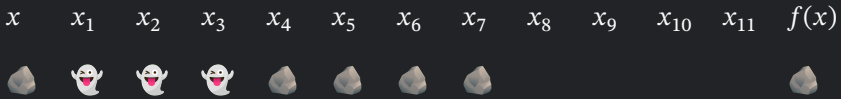
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 15

Max. pebble count: 7

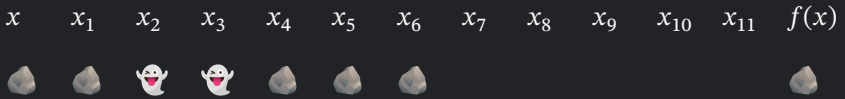
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 16

Max. pebble count: 7

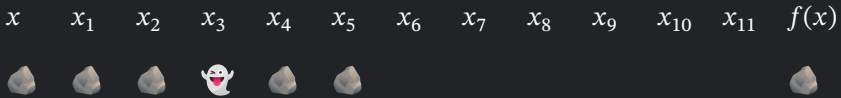
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 17

Max. pebble count: 7

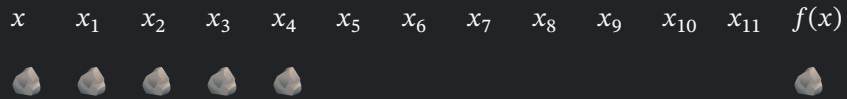
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 18

Max. pebble count: 7

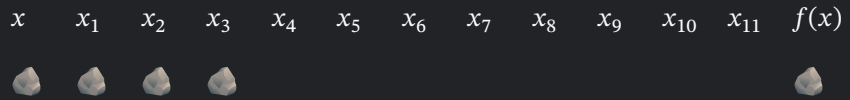
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 19

Max. pebble count: 7

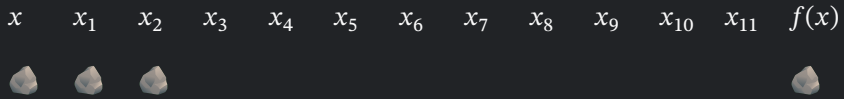
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 20

Max. pebble count: 7

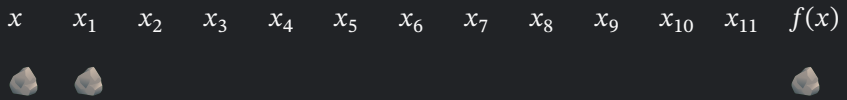
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 21

Max. pebble count: 7

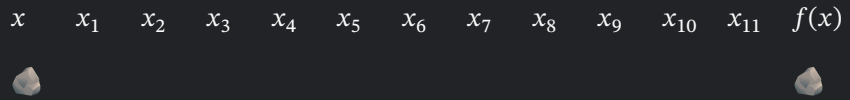
Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 22

Max. pebble count: 7

Example: an optimal parallel spooky pebble game for $k = 12$



Step number: 23

Max. pebble count: 7

Example: an optimal parallel spooky pebble game for $k = 12$

x x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} $f(x)$

Step number: $23 = 2k - 1$, which is optimal 🤩

Max. pebble count: 7

Summary

Assumption: quantum circuits must be (mostly) reversible

Reality:

- **Measurement** is useful at the logical level for much more than just readout

Summary

Assumption: quantum circuits must be (mostly) reversible

Reality:

- **Measurement** is useful at the logical level for much more than just readout
- Judicious use of intermediate measurements can **dramatically lower circuit sizes**

Summary

Assumption: quantum circuits must be (mostly) reversible

Reality:

- **Measurement** is useful at the logical level for much more than just readout
- Judicious use of intermediate measurements can **dramatically lower circuit sizes**

Summary

Assumption: quantum circuits must be (mostly) reversible

Reality:

- **Measurement** is useful at the logical level for much more than just readout
- Judicious use of intermediate measurements can **dramatically lower circuit sizes**

Some reading:

- Measurement-based uncomputation makes addition much cheaper
  - Reducing T count to $4n$: Gidney ’17 (arXiv:1709.06648)
  - Constant number of ancillas for classical-quantum addition: Gidney ’25 (arXiv:2507.23079)
- Superposition masking: Jaques + Gidney ’20 (arXiv:2008.04577)

Overview

Assumption: factoring is the right thing to focus on

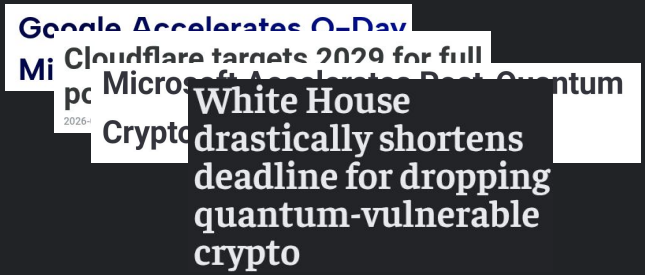
Assumption: input length is a lower bound on qubit count

Assumption: logical qubit count is what matters

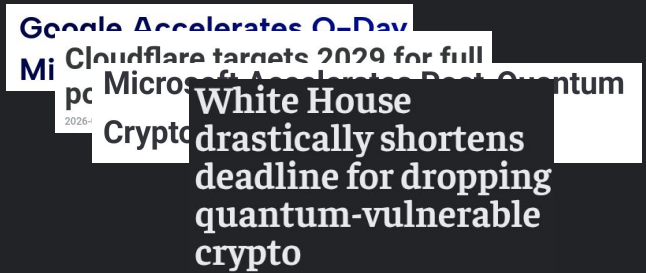
Assumption: quantum circuits must be (mostly) reversible

Assumption: breaking cryptography is a long way off

Quantum-vulnerable cryptography is closer than ever to being broken

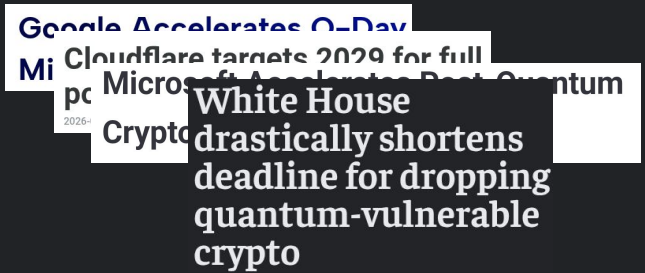


Quantum-vulnerable cryptography is closer than ever to being broken



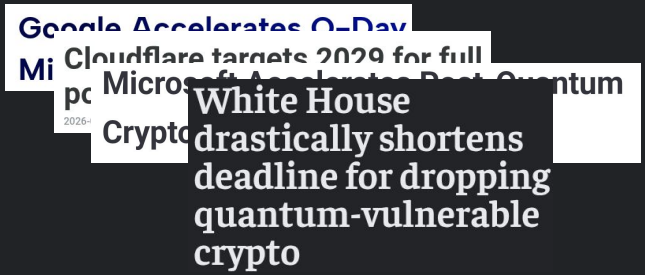
- Sudden algorithmic improvements over the past few years have exceeded all expectations

Quantum-vulnerable cryptography is closer than ever to being broken



- Sudden algorithmic improvements over the past few years have exceeded all expectations
- Rapidly evolving changes in expected *architecture* and thus in logical circuit *cost metrics*

Quantum-vulnerable cryptography is closer than ever to being broken



- Sudden algorithmic improvements over the past few years have exceeded all expectations
- Rapidly evolving changes in expected *architecture* and thus in logical circuit *cost metrics*
- These changes have not only shifted the timeline, but also increased the **uncertainty**

Looking forward

What assumptions are about to be broken?
Let’s think about it this week!

INTELLIGENCE COMMUNITY
POSTDOCTORAL RESEARCH
FELLOWSHIP PROGRAM

SIMONS INSTITUTE
for the Theory of Computing

Thank you!

Email: gkm@mit.edu
Website: https://gmeyer.net/

GDKM was supported by an appointment to the Intelligence Community Postdoctoral Research Fellowship Program at MIT administered by Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy and the Office of the Director of National Intelligence (ODNI).

Backup