Week 9: Phase Estimation and and Shor's Factoring Algorithm

COMS 4281 (Fall 2025)

Last time

- Phase Estimation: A unitary U induces phases on its eigenvectors. The goal is to estimate the phase corresponding to an eigenvector.
- This will be useful for many problems in quantum computing, because sometimes we can construct a unitary U whose phases encode the solutions we want. Factoring and physics simulations are two major examples.

RSA and the Factoring problem

RSA Cryptosystem

- Invented by Rivest, Shamir, and Adleman in 1977
- Most widely deployed public-key cryptosystem
- Enables public-key encryption as well as digital signatures

RSA Cryptosystem

- Invented by Rivest, Shamir, and Adleman in 1977
- Most widely deployed public-key cryptosystem
- Enables public-key encryption as well as digital signatures

Its security assumes that the Factoring Problem is hard for computers!

We won't cover how RSA cryptosystem works here, but you can look it up on Wikipedia.

Factoring problem

Input: Positive integer N written in binary.

Output: Prime factorization of N as $p_1^{a_1} p_2^{a_2} \cdots$.

Factoring problem

Input: Positive integer N written in binary.

Output: Prime factorization of N as $p_1^{a_1} p_2^{a_2} \cdots$.

The prime factorization of N is unique by the **Fundamental Theorem of Arithmetic**.

To find a factorization of N, it suffices to be able to find *some* nontrivial divisor of N.

Factoring problem

Input: Positive integer N written in binary.

Output: Prime factorization of N as $p_1^{a_1} p_2^{a_2} \cdots$.

The prime factorization of N is unique by the **Fundamental Theorem of Arithmetic**.

To find a factorization of N, it suffices to be able to find *some* nontrivial divisor of N.

Example: What is a nontrivial factor of
215274110271888970189601520131282542925777358884567598017049767677813314521885913567301105977349105960249790711158521430207931466520284014061994699492757040775 3?

Factoring problem

A simple way to factor N is to iterate from $2, 3, \ldots$ and try to find a number that divides N.

Factoring problem

A simple way to factor N is to iterate from $2, 3, \ldots$ and try to find a number that divides N.

In the worst case, if N is not prime, then one only has to iterate up to $\sqrt{N}$. (Why?)

Factoring problem

A simple way to factor N is to iterate from $2, 3, \ldots$ and try to find a number that divides N.

In the worst case, if N is not prime, then one only has to iterate up to $\sqrt{N}$. (Why?)

The number of iterations is

$$\sqrt{N} = \sqrt{2^{\log N}} = 2^{(\log N)/2}$$

which is **exponential** in the binary representation of N. For a typical RSA key with 2048 bits, this requires

$$2^{1024}$$

iterations.

Factoring problem

It is widely believed that Factoring is hard for classical computers. The best classical algorithm, known as the **General Number Field Sieve**, takes time roughly

$$\exp\left(O(\log N)^{1/3}\right) .$$

This is still **exponential** in the number of digits of N (albeit an improved exponential). Using GNFS, RSA-2048 keys can be factored in time

$$2^{112}$$

which is large, but much smaller than before.

Shor's algorithm

A quantum algorithm to solve Factoring in $\mathrm{poly}(\log N)$ steps.

Discovered by Peter Shor in 1993. He was inspired by Simon's Algorithm.

Shor's algorithm

A quantum algorithm to solve Factoring in $\mathrm{poly}(\log N)$ steps.

Discovered by Peter Shor in 1993. He was inspired by Simon's Algorithm.

Shor's Algorithm is also a hybrid classical-quantum algorithm.

1. **Classical part**: reduce the factoring problem to **order finding**. This means that if you can solve order finding, you can factor numbers.

Shor's algorithm

A quantum algorithm to solve Factoring in $\mathrm{poly}(\log N)$ steps.

Discovered by Peter Shor in 1993. He was inspired by Simon's Algorithm.

Shor's Algorithm is also a hybrid classical-quantum algorithm.

1. **Classical part**: reduce the factoring problem to **order finding**. This means that if you can solve order finding, you can factor numbers.
2. **Quantum part**: solve order finding.

Order Finding

Input: given positive integers N, x such that

1. $1 \leq x < N$
2. $\gcd(N, x) = 1$ (i.e. they do not have any nontrivial factors in common; we also say that x is **coprime** with N)

Order Finding

Input: given positive integers N, x such that

1. $1 \leq x < N$
2. $\gcd(N, x) = 1$ (i.e. they do not have any nontrivial factors in common; we also say that x is **coprime** with N)

Output: find smallest integer r such that $x^r = 1 \bmod N$ (called the **order** of x mod N).

Order finding example

Find order of $x = 7 \bmod N = 15$.

Order finding example

Find order of $x = 7 \bmod N = 15$.

$$7^1 = 7, \qquad 7^2 = 4, \qquad 7^3 = 13, \qquad 7^4 = 1 \pmod{15}\ .$$

Therefore order of $x \bmod N$ is $r = 4$.

Order finding example

Find order of $x = 7 \mod N = 15$.

$$7^1 = 7, \qquad 7^2 = 4, \qquad 7^3 = 13, \qquad 7^4 = 1 \qquad \pmod{15} .$$

Therefore order of $x \mod N$ is $r = 4$.

Note that the function $f(k) = x^k \mod N$ is **periodic**.

Order finding example

Find order of $x = 7 \mod N = 15$.

$$7^1 = 7, \qquad 7^2 = 4, \qquad 7^3 = 13, \qquad 7^4 = 1 \qquad \pmod{15}.$$

Therefore order of $x \mod N$ is $r = 4$.

Note that the function $f(k) = x^k \mod N$ is **periodic**. This is because

$$x^{k+r} = x^k x^r = x^k \mod N$$

whenever r is the order of $x \mod N$. So $f(k) = f(k + r)$.

Order finding example

The condition that $\gcd(x, N) = 1$ is necessary for the order finding problem to be well-defined.

Order finding example

The condition that $\gcd(x, N) = 1$ is necessary for the order finding problem to be well-defined.

Consider $x = 5, N = 15$. This does **not** satisfy $\gcd(x, N) = 1$. It also does not have an order:

$$5^1 = 5, \qquad 5^2 = 10, \qquad 5^3 = 5, \qquad \cdots \qquad \pmod{15} .$$

Complexity of order finding

If you can efficiently solve order finding, then you can efficiently factor. (We won't cover the reduction here).

Thus order finding must be **at least** as hard as factoring.

Complexity of order finding

If you can efficiently solve order finding, then you can efficiently factor. (We won't cover the reduction here).

Thus order finding must be **at least** as hard as factoring.

The converse is also true: if you can factor numbers efficiently, you can also solve order finding efficiently.

Quantum algorithm for Order Finding

Quantum Algorithm for Order Finding

The Order Finding problem

Input: Integers N and x such that $\gcd(x, N) = 1$.

Goal: Find smallest integer r such that $x^r = 1 \mod N$.

Quantum Algorithm for Order Finding

The Order Finding problem

Input: Integers N and x such that $\gcd(x, N) = 1$.

Goal: Find smallest integer r such that $x^r = 1 \mod N$.

To efficiently solve this on a quantum computer, we will use the Phase Estimation Algorithm with a carefully constructed unitary that depends on x, N. The phases will yield valuable information about the order of x (analogous to the quantum subroutine in Simon's algorithm).

Modular multiplication unitary

Fix x, N.

Consider the Hilbert space $\mathbb{C}^N$ with standard basis $\{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$. Define the **modular multiplication unitary** on this space

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

Modular multiplication unitary

Fix x, N.

Consider the Hilbert space $\mathbb{C}^N$ with standard basis $\{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$. Define the **modular multiplication unitary** on this space

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

Example: $x = 7, N = 15$

$$U|1\rangle = |7\rangle, \qquad , U|2\rangle = |14\rangle, \qquad U|3\rangle = |6\rangle, \qquad \cdots$$

$$U|13\rangle = |1\rangle, \qquad U|14\rangle = |8\rangle$$

Modular multiplication unitary

Fix x, N. What are some eigenvectors and eigenvalues of U?

Modular multiplication unitary

Fix x, N. What are some eigenvectors and eigenvalues of U?

Let $r = \mathrm{ord}(x)$ (i.e. $x^r = 1 \bmod N$). Then for all $0 \leq s < r$, the vector

$$|v_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-2\pi i \frac{k}{r} s\Big) \, |x^k \bmod N\rangle$$

Modular multiplication unitary

Fix x, N. What are some eigenvectors and eigenvalues of U?

Let $r = \mathrm{ord}(x)$ (i.e. $x^r = 1 \bmod N$). Then for all $0 \leq s < r$, the vector

$$|v_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-2\pi i \frac{k}{r} s\Big) |x^k \bmod N\rangle$$

is an eigenvector satisfying

$$U |v_s\rangle = \exp\left(2\pi i \frac{s}{r}\right) |v_s\rangle$$

Modular multiplication unitary

Fix x, N. What are some eigenvectors and eigenvalues of U?

Let $r = \mathrm{ord}(x)$ (i.e. $x^r = 1 \bmod N$). Then for all $0 \leq s < r$, the vector

$$|v_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-2\pi i \frac{k}{r} s \Big) |x^k \bmod N\rangle$$

is an eigenvector satisfying

$$U |v_s\rangle = \exp\left(2\pi i \frac{s}{r} \right) |v_s\rangle$$

The amplitudes of $|v_s\rangle$ and its eigenvalue encodes the answer we're looking for!

Modular multiplication unitary

Consider the example $x = 7, N = 15, \mathrm{ord}(x, N) = 4$ again. One of the eigenvectors is

$$|v_1\rangle = \frac{1}{\sqrt{4}} \sum_{k=0}^{3} \exp\Big(-2\pi i \frac{k}{4}\Big) \, |7^k \bmod 15\rangle$$

Modular multiplication unitary

Consider the example $x = 7, N = 15, \mathrm{ord}(x, N) = 4$ again. One of the eigenvectors is

$$|v_1\rangle = \frac{1}{\sqrt{4}} \sum_{k=0}^{3} \exp\Big(-2\pi i \frac{k}{4}\Big) \; |7^k \bmod 15\rangle$$

Applying the modular multiplication unitary in superposition:

$$U\,|v_1\rangle = \frac{1}{\sqrt{4}}\Big(U\,|7^0\rangle + e^{-2\pi i \frac{1}{4}}\, U\,|7^1\rangle + e^{-2\pi i \frac{2}{4}}\, U\,|7^2\rangle + e^{-2\pi i \frac{3}{4}}\, U\,|7^3\rangle \Big)$$

.

Modular multiplication unitary

Consider the example $x = 7, N = 15, \mathrm{ord}(x, N) = 4$ again. One of the eigenvectors is

$$|v_1\rangle = \frac{1}{\sqrt{4}} \sum_{k=0}^{3} \exp\left(-2\pi i \frac{k}{4}\right) |7^k \bmod 15\rangle$$

Applying the modular multiplication unitary in superposition:

$$\begin{aligned} U|v_1\rangle &= \frac{1}{\sqrt{4}}\Big(U|7^0\rangle + e^{-2\pi i \frac{1}{4}} U|7^1\rangle + e^{-2\pi i \frac{2}{4}} U|7^2\rangle + e^{-2\pi i \frac{3}{4}} U|7^3\rangle \Big) \\ &= \frac{1}{\sqrt{4}}\Big(|7^1\rangle + e^{-2\pi i \frac{1}{4}} |7^2\rangle + e^{-2\pi i \frac{2}{4}} |7^3\rangle + e^{-2\pi i \frac{3}{4}} |7^4\rangle \Big) \end{aligned}$$

.

Modular multiplication unitary

Consider the example $x = 7, N = 15, \mathrm{ord}(x, N) = 4$ again. One of the eigenvectors is

$$|v_1\rangle = \frac{1}{\sqrt{4}} \sum_{k=0}^{3} \exp\Big(-2\pi i \frac{k}{4} \Big) \; |7^k \bmod 15\rangle$$

Applying the modular multiplication unitary in superposition:

$$\begin{aligned} U\,|v_1\rangle &= \frac{1}{\sqrt{4}}\Big(U\,|7^0\rangle + e^{-2\pi i \frac{1}{4}} U\,|7^1\rangle + e^{-2\pi i \frac{2}{4}} U\,|7^2\rangle + e^{-2\pi i \frac{3}{4}} U\,|7^3\rangle \Big) \\ &= \frac{1}{\sqrt{4}}\Big(|7^1\rangle + e^{-2\pi i \frac{1}{4}} |7^2\rangle + e^{-2\pi i \frac{2}{4}} |7^3\rangle + e^{-2\pi i \frac{3}{4}} |7^4\rangle \Big) \\ &= e^{2\pi i \frac{1}{4}} \frac{1}{\sqrt{4}}\Big(|7^0\rangle + e^{-2\pi i \frac{1}{4}} |7^1\rangle + e^{-2\pi i \frac{2}{4}} |7^2\rangle + e^{-2\pi i \frac{3}{4}} |7^3\rangle \Big)\;. \end{aligned}$$

Warmup to quantum order finding

Imagine that we had

1. The ability to query controlled U^k for $1 \leq k \leq N^3$, and

Warmup to quantum order finding

Imagine that we had

1. The ability to query controlled U^k for $1 \leq k \leq N^3$, and
2. Access to the eigenvector $|v_s\rangle$ (where s is coprime with r) whose eigenvalue is $\exp(2\pi i \frac{s}{r})$.

Warmup to quantum order finding

Imagine that we had

1. The ability to query controlled U^k for $1 \leq k \leq N^3$, and
2. Access to the eigenvector $|v_s\rangle$ (where s is coprime with r) whose eigenvalue is $\exp(2\pi i \frac{s}{r})$.

Set $\epsilon \sim 1/N^3$. By running phase estimation with $O(\log 1/\epsilon) = O(\log N)$ queries to controlled U (and its powers), we can obtain an estimate of

$$\widetilde{\theta} = \frac{s}{r} \pm \epsilon.$$

From this, we will try to recover the value of r, which solves order finding problem.

Issue 1: The algorithm outputs something that looks like

$$\widetilde{\theta} = 0.011110110011\ldots$$

We don't know s, r, and this estimate is also noisy. How do find the s/r fraction corresponding to this?

Issue 1: The algorithm outputs something that looks like

$$\widetilde{\theta} = 0.011110110011\ldots$$

We don't know s, r, and this estimate is also noisy. How do find the s/r fraction corresponding to this?

Issue 2: How do we get our hands on the eigenvector $|v_s\rangle$?

Issue 1: The algorithm outputs something that looks like

$$\widetilde{\theta} = 0.011110110011\ldots$$

We don't know s, r, and this estimate is also noisy. How do find the s/r fraction corresponding to this?

Issue 2: How do we get our hands on the eigenvector $|v_s\rangle$?

Issue 3: How can we implement controlled U^k for $1 \leq k \leq N^3$ in an efficient way?

Getting the eigenvectors

We can solve Issue 2 by running Phase Estimation on the superposition

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |v_s\rangle$$

Getting the eigenvectors

We can solve Issue 2 by running Phase Estimation on the superposition

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|v_s\rangle$$

While it's hard to construct $|v_s\rangle$ individually, this superposition is easy to prepare, because this is equal to the standard basis state $|1\rangle$. **(exercise!)**

After running PEA on input $|1\rangle$, the output will be approximately

$$\approx \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |v_s\rangle \otimes |\widetilde{\theta}_s\rangle$$

where $|\widetilde{\theta}_s - \frac{s}{r}| \leq 1/N^3$.

After running PEA on input $|1\rangle$, the output will be approximately

$$\approx \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |v_s\rangle \otimes |\widetilde{\theta}_s\rangle$$

where $|\widetilde{\theta}_s - \frac{s}{r}| \leq 1/N^3$.

Measuring the second register yields $|\widetilde{\theta}_s\rangle$ for a uniformly random $0 \leq s < r$.

Imagine that $\widetilde{\theta}_s$ was actually s/r *exactly*, and furthermore s was coprime to r.

Then we can recover s, r from $\widetilde{\theta}_s$.

Imagine that $\widetilde{\theta}_s$ was actually s/r *exactly*, and furthermore s was coprime to r.

Then we can recover s, r from $\widetilde{\theta}_s$.

Example: Suppose $\widetilde{\theta}_s = 0.358$. Then clearly

$$\widetilde{\theta}_s = \frac{358}{1000} = \frac{179}{500}.$$

Imagine that $\widetilde{\theta}_s$ was actually s/r *exactly*, and furthermore s was coprime to r.

Then we can recover s, r from $\widetilde{\theta}_s$.

Example: Suppose $\widetilde{\theta}_s = 0.358$. Then clearly

$$\widetilde{\theta}_s = \frac{358}{1000} = \frac{179}{500}.$$

This is equal to s/r. But since s/r is already in most reduced form, it must be $s = 179$ and $r = 500$.

However $\widetilde{\theta}_s$ is not s/r exactly.

We solve **Issue 1** by the **Continued Fractions Algorithm**, which is a fast classical algorithm for finding “nearby” rational expansions of decimal numbers. (We won’t cover Continued Fractions here).

Modular multiplication unitary

Claim: The modular multiplication unitary $U|y\rangle = |xy \bmod N\rangle$ can be implemented by a quantum circuit with $\mathrm{poly}(\log N)$ gates, using ancilla.

Modular multiplication unitary

Claim: The modular multiplication unitary $U|y\rangle = |xy \text{ mod } N\rangle$ can be implemented by a quantum circuit with $\mathrm{poly}(\log N)$ gates, using ancilla.

Proof: Suppose input is $|y\rangle$.

1. Using reversible circuit for computing $y \mapsto xy \bmod N$, compute $|y\rangle|0\rangle \mapsto |y\rangle|xy \text{ mod } N\rangle$.

Modular multiplication unitary

Claim: The modular multiplication unitary $U|y\rangle = |xy \text{ mod } N\rangle$ can be implemented by a quantum circuit with $\mathrm{poly}(\log N)$ gates, using ancilla.

Proof: Suppose input is $|y\rangle$.

1. Using reversible circuit for computing $y \mapsto xy \bmod N$, compute $|y\rangle|0\rangle \mapsto |y\rangle|xy \text{ mod } N\rangle$.
2. Swap registers: $|y\rangle|xy \text{ mod } N\rangle \mapsto |xy \text{ mod } N\rangle|y\rangle$.

Modular multiplication unitary

Claim: The modular multiplication unitary $U|y\rangle = |xy \text{ mod } N\rangle$ can be implemented by a quantum circuit with $\mathrm{poly}(\log N)$ gates, using ancilla.

Proof: Suppose input is $|y\rangle$.

1. Using reversible circuit for computing $y \mapsto xy \bmod N$, compute $|y\rangle|0\rangle \mapsto |y\rangle|xy \text{ mod } N\rangle$.
2. Swap registers: $|y\rangle|xy \text{ mod } N\rangle \mapsto |xy \text{ mod } N\rangle|y\rangle$.
3. Using reversible circuit for computing $z \mapsto x^{-1}z \bmod N$, compute $|xy \text{ mod } N\rangle|y\rangle \mapsto |xy \text{ mod } N\rangle|0\rangle$.

Modular multiplication unitary

Claim: The modular multiplication unitary $U|y\rangle = |xy \text{ mod } N\rangle$ can be implemented by a quantum circuit with $\mathrm{poly}(\log N)$ gates, using ancilla.

Proof: Suppose input is $|y\rangle$.

1. Using reversible circuit for computing $y \mapsto xy \bmod N$, compute $|y\rangle|0\rangle \mapsto |y\rangle|xy \text{ mod } N\rangle$.
2. Swap registers: $|y\rangle|xy \text{ mod } N\rangle \mapsto |xy \text{ mod } N\rangle|y\rangle$.
3. Using reversible circuit for computing $z \mapsto x^{-1}z \bmod N$, compute $|xy \text{ mod } N\rangle|y\rangle \mapsto |xy \text{ mod } N\rangle|0\rangle$.

Because the ancilla register starts and ends in $|0\rangle$, we can discard it from the system.

Modular multiplication unitary

Claim: The modular multiplication unitary $U|y\rangle = |xy \text{ mod } N\rangle$ can be implemented by a quantum circuit with $\mathrm{poly}(\log N)$ gates, using ancilla.

Proof continued: The complexity of this implementation of U is dominated by the implementation of the classical reversible circuits for the functions $y \mapsto xy \bmod N$, and for computing $z \mapsto x^{-1}z \bmod N$. These can be done using $\mathrm{poly}(\log N)$ gates because this is basic modular arithmetic which can be done efficiently on a classical computer.

Modular multiplication unitary, repeated

Consider the powered modular multiplication unitary

$$U^k |y\rangle = |x^k y \bmod N\rangle$$

Fact: This map is unitary, and is computable by a quantum circuit with $\mathrm{poly}(n, \log k)$ gates.

Modular multiplication unitary, repeated

Consider the powered modular multiplication unitary

$$U^k |y\rangle = |x^k y \bmod N\rangle$$

Fact: This map is unitary, and is computable by a quantum circuit with $\mathrm{poly}(n, \log k)$ gates.

This uses fact that one can "shortcut" computing x^k modulo N without doing k multiplications, by repeatedly squaring, reducing mod N, squaring, reducing mod N, etc...

$$x \to x^2 \to x^2 \text{ mod } N \to (x^2 \text{ mod } N)^2 \to x^4 \text{ mod } N \to \cdots$$

Summary of Order Finding algorithm

1. Run Phase Estimation with the unitary controlled-U (and its powers) and input state $|1\rangle$ (which is uniform superposition of eigenvectors).

Summary of Order Finding algorithm

1. Run Phase Estimation with the unitary controlled-U (and its powers) and input state $|1\rangle$ (which is uniform superposition of eigenvectors).
2. Get random estimate $\widetilde{\theta}$.

Summary of Order Finding algorithm

1. Run Phase Estimation with the unitary controlled-U (and its powers) and input state $|1\rangle$ (which is uniform superposition of eigenvectors).
2. Get random estimate $\widetilde{\theta}$.
3. Use Continued Fractions algorithm on $\widetilde{\theta}$ to obtain reduced form a/b of s/r.

Summary of Order Finding algorithm

1. Run Phase Estimation with the unitary controlled-U (and its powers) and input state $|1\rangle$ (which is uniform superposition of eigenvectors).
2. Get random estimate $\widetilde{\theta}$.
3. Use Continued Fractions algorithm on $\widetilde{\theta}$ to obtain reduced form a/b of s/r.
4. Check whether $x^b = 1 \bmod N$. If so, then $b = r$. Otherwise, go back to Step 1.

Summary of Order Finding algorithm

1. Run Phase Estimation with the unitary controlled-U (and its powers) and input state $|1\rangle$ (which is uniform superposition of eigenvectors).
2. Get random estimate $\widetilde{\theta}$.
3. Use Continued Fractions algorithm on $\widetilde{\theta}$ to obtain reduced form a/b of s/r.
4. Check whether $x^b = 1 \bmod N$. If so, then $b = r$. Otherwise, go back to Step 1.

Each loop succeeds with probability at least $\frac{1}{\log\log N}$, so we only have to repeat a few times.

Summary of Factoring via Quantum Computers

1. If we can factor, we can break RSA.

Summary of Factoring via Quantum Computers

1. If we can factor, we can break RSA.
2. If we can solve Order Finding, we can factor integers.

Summary of Factoring via Quantum Computers

1. If we can factor, we can break RSA.
2. If we can solve Order Finding, we can factor integers.
3. By running Phase Estimation with Modular Multiplication unitary a few times, we can get noisy estimates of s/r.

Summary of Factoring via Quantum Computers

1. If we can factor, we can break RSA.
2. If we can solve Order Finding, we can factor integers.
3. By running Phase Estimation with Modular Multiplication unitary a few times, we can get noisy estimates of s/r.
4. We can "decode" these estimates by using Continued Fractions algorithm.

How far away is Shor's factoring algorithm?

Gidney 2025: given current methods for error correction, we would need

1. 1 million noisy qubits
2. 1 week

to factor a 2048-bit RSA key.

How far away is Shor's factoring algorithm?

Gidney 2025: given current methods for error correction, we would need

1. 1 million noisy qubits
2. 1 week

to factor a 2048-bit RSA key.

IBM Roadmap: 1 million qubits by 2030.

What will replace RSA?

US National Institute for Standards and Technology (NIST) just concluded a multiyear competition to find **postquantum** cryptosystems to replace RSA. The winners are...

What will replace RSA?

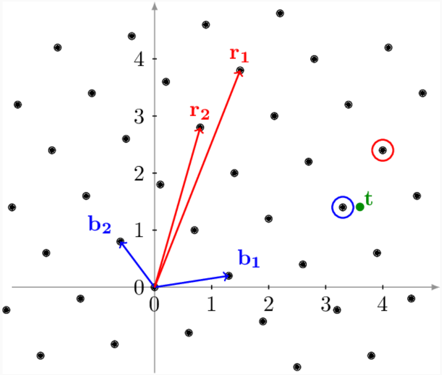
US National Institute for Standards and Technology (NIST) just concluded a multiyear competition to find **postquantum** cryptosystems to replace RSA. The winners are...

1. CRYSTALS-Kyber
2. CRYSTALS-Dilithium
3. FALCON
4. SPHINCS+

Kyber is for encryption, and the last three are for digital signatures.

Post-quantum cryptography from lattices

Kyber, Dilithium, and Falcon are all based on **lattice problems**, where the goal is to find short vectors in a high-dimensional lattice.



It is believed that these problems cannot be quickly solved by quantum computers.

Post-quantum cryptography from lattices

It is an important research problem to find better evidence that lattice problems are hard for quantum computers.

But there's always the possibility that someone can find a fast quantum algorithm for them...

It will take time to build confidence in these new cryptosystems.