Shor’s Algorithms

Giovanni Galatro

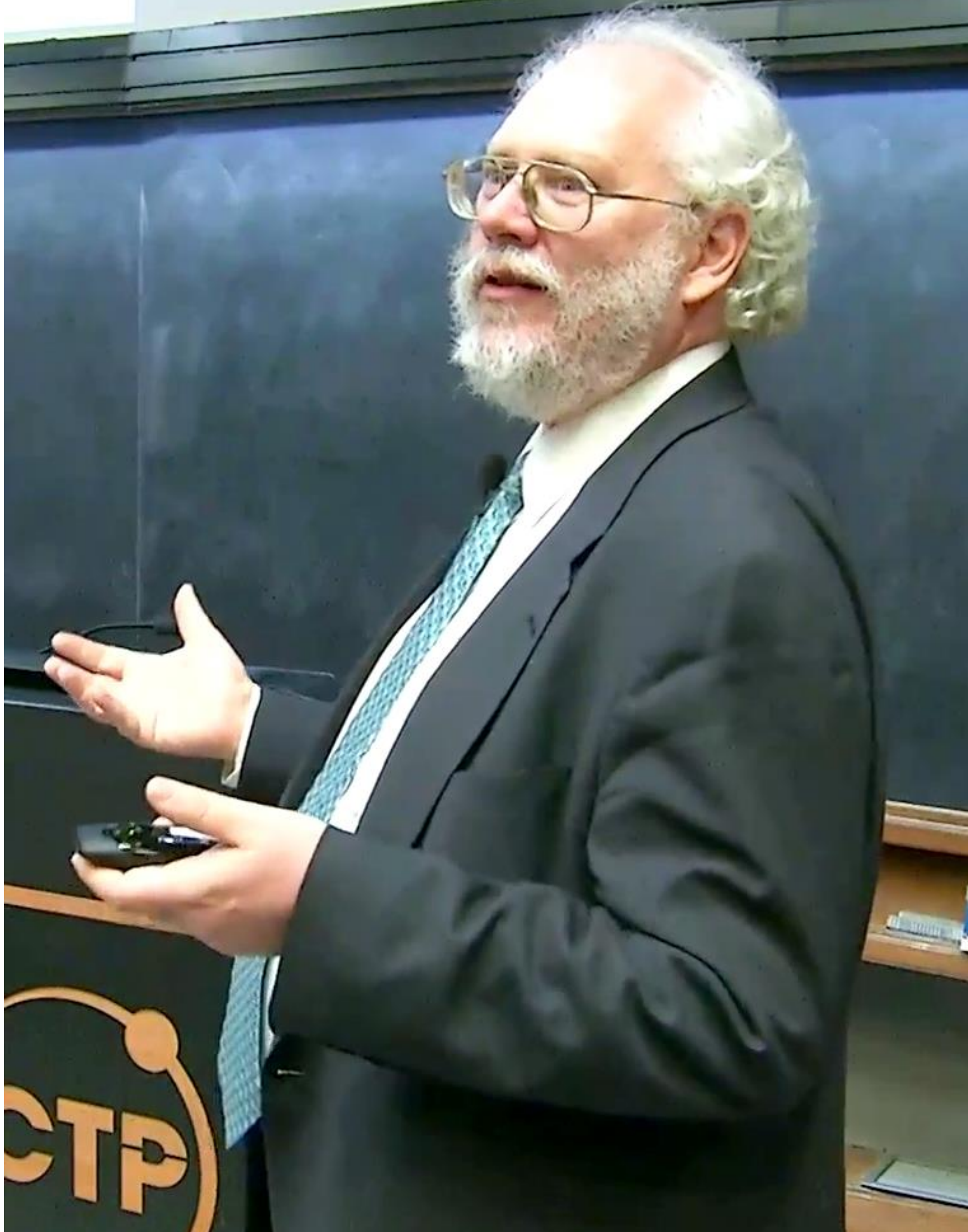
Shor’s Algorithm

Applications

- Factorization of large integers
- Cryptanalysis (breaking RSA encryption)

Generalization

- Discrete logarithm problem
- Period-finding problem



Reminder

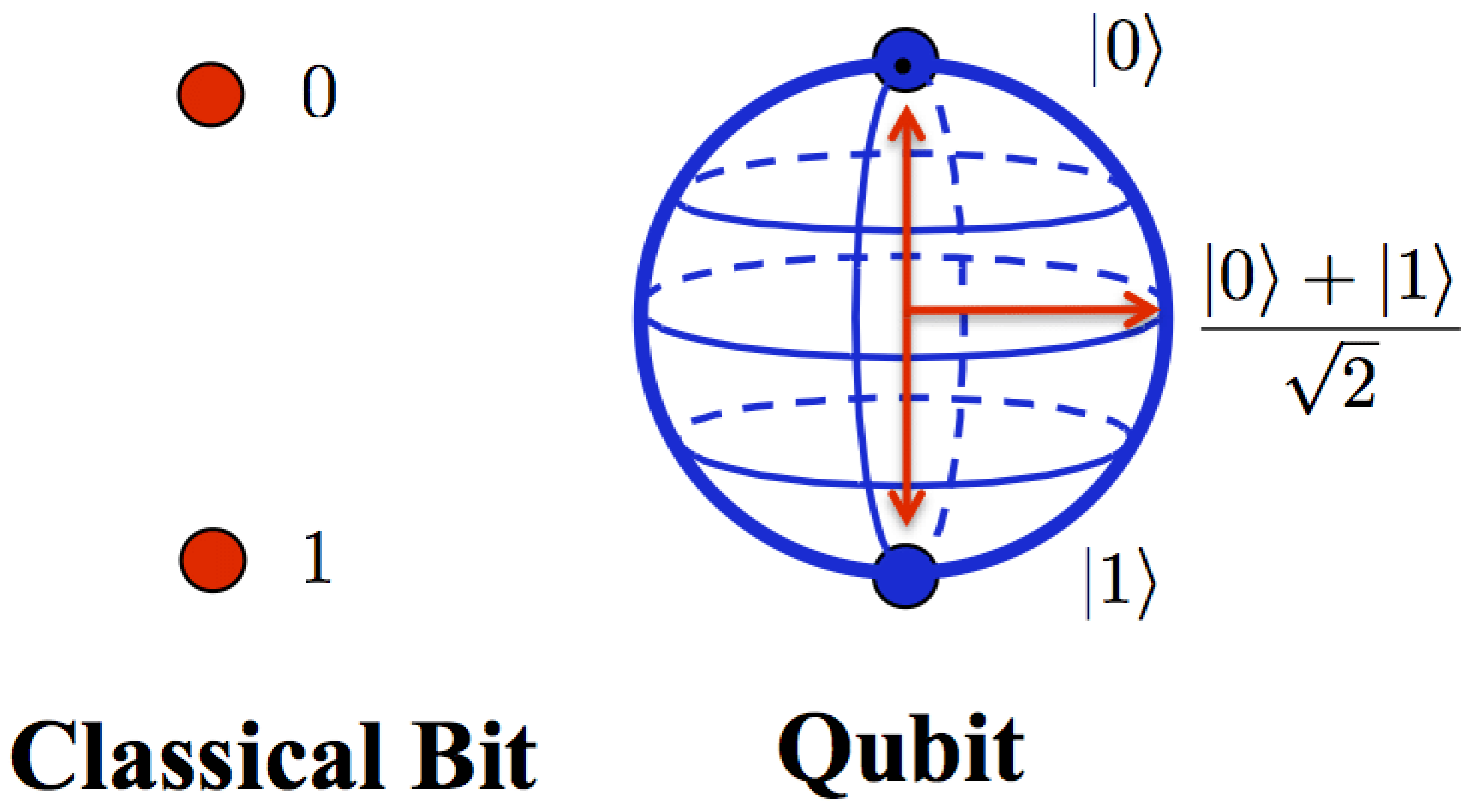
Classical bits have two states (0 or 1), but qubits can exist in a superposition of both simultaneously:

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle$$

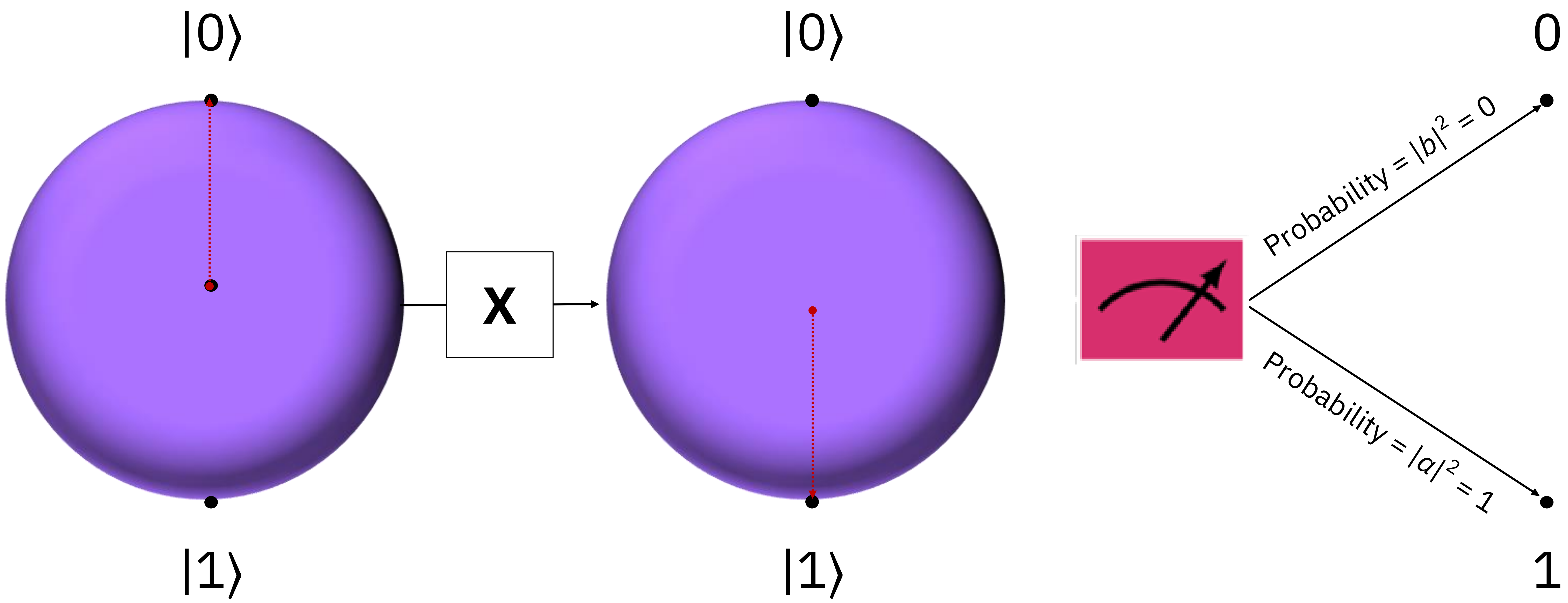
Quantum gates manipulate qubit states.

Es. Hadamard Transform:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$



Bits and qubits: the effect of the X gate on $|0\rangle$



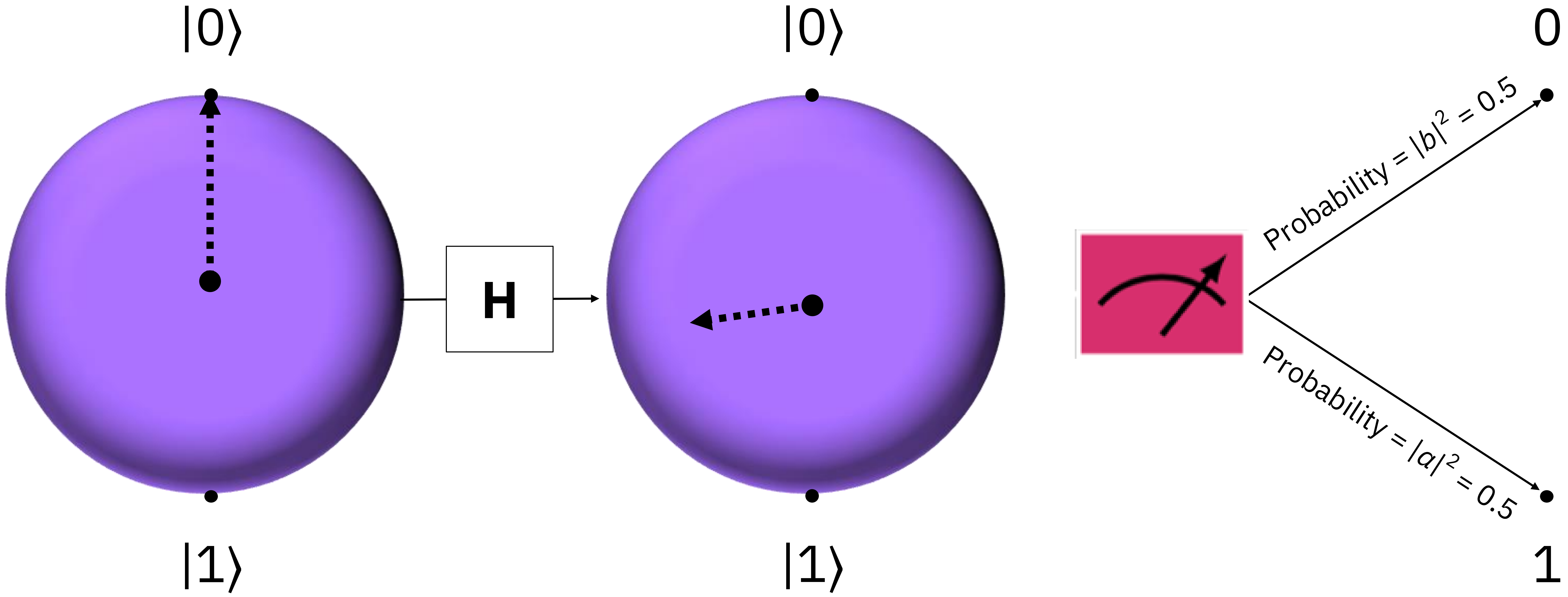
The **X** gate reverses $|0\rangle$ and $|1\rangle$:

$$a\,|0\rangle + b\,|1\rangle \;\mapsto\; b\,|0\rangle + a\,|1\rangle$$

$a = 1$ and $b = 0$, so $|0\rangle$ is mapped to $|1\rangle$.

When measured, the result is **1** with 100% probability.

Bits and qubits: the effect of the H gate on $|0\rangle$



The **H** gate maps $|0\rangle$ via

$|0\rangle \mapsto (1/\sqrt{2})\,|0\rangle + (1/\sqrt{2})\,|1\rangle = a\,|0\rangle + b\,|1\rangle$

Since $a = b = 1/\sqrt{2}$, $|a|^2 = |b|^2 = ½$.

When measured, the probability of getting **0** or **1** is the same, 0.5.
Quantum randomness!

Qiskit is the most preferred and performant quantum SDK

74%

Quantum computing programmers prefer the Qiskit SDK

(2024 Unitary Foundation Open Source Software Survey)

7400+

Dependent projects

(Next nearest: PennyLane, 1000+)

83x

Mean transpilation time improvement

(Qiskit 2.2.0 vs TKET 2.6.0)

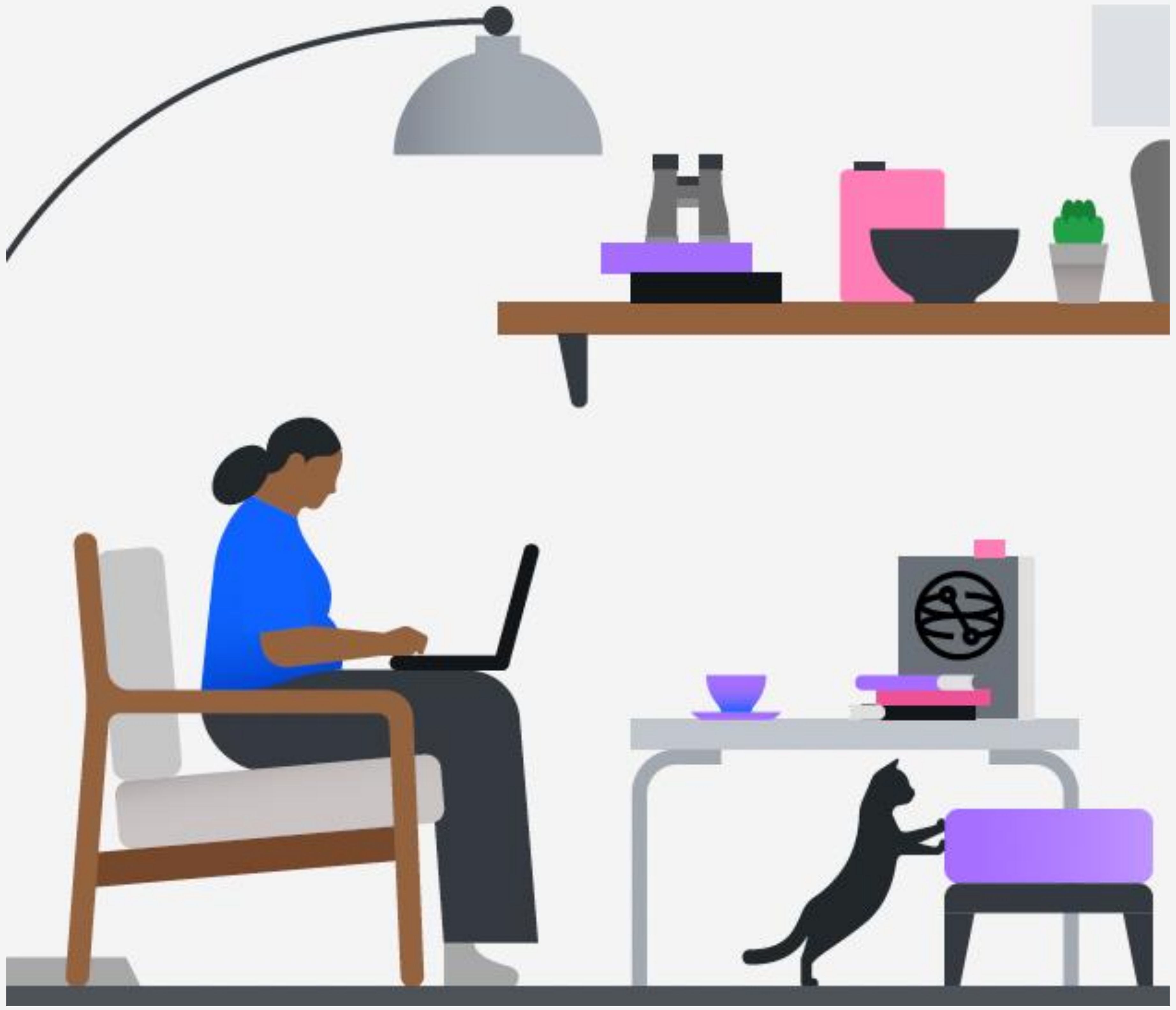
29%

Fewer 2Q gates

(Qiskit 2.2.0 vs TKET 2.6.0)

Results reported from Benchpress tests of ~1000 circuits (https://github.com/Qiskit/benchpress) and published in *Nature Computational Science* (2025): https://www.nature.com/articles/s43588-025-00792-y. Timing and quality measured using only completed tests. Dependencies from Github insights. Current as of 10/14/25.

IBM Certified Quantum Computation using Qiskit v2.X Developer



1370 Developers with Qiskit certification

71 Countries with Qiskit-certified developers

The new and updated Qiskit certification is available on IBM Training now!

Validate your Qiskit 2.X skills, with topics such as performing quantum operations, creating quantum circuits, as well as running quantum circuits.

A free study guide and sample test are available through IBM Training.

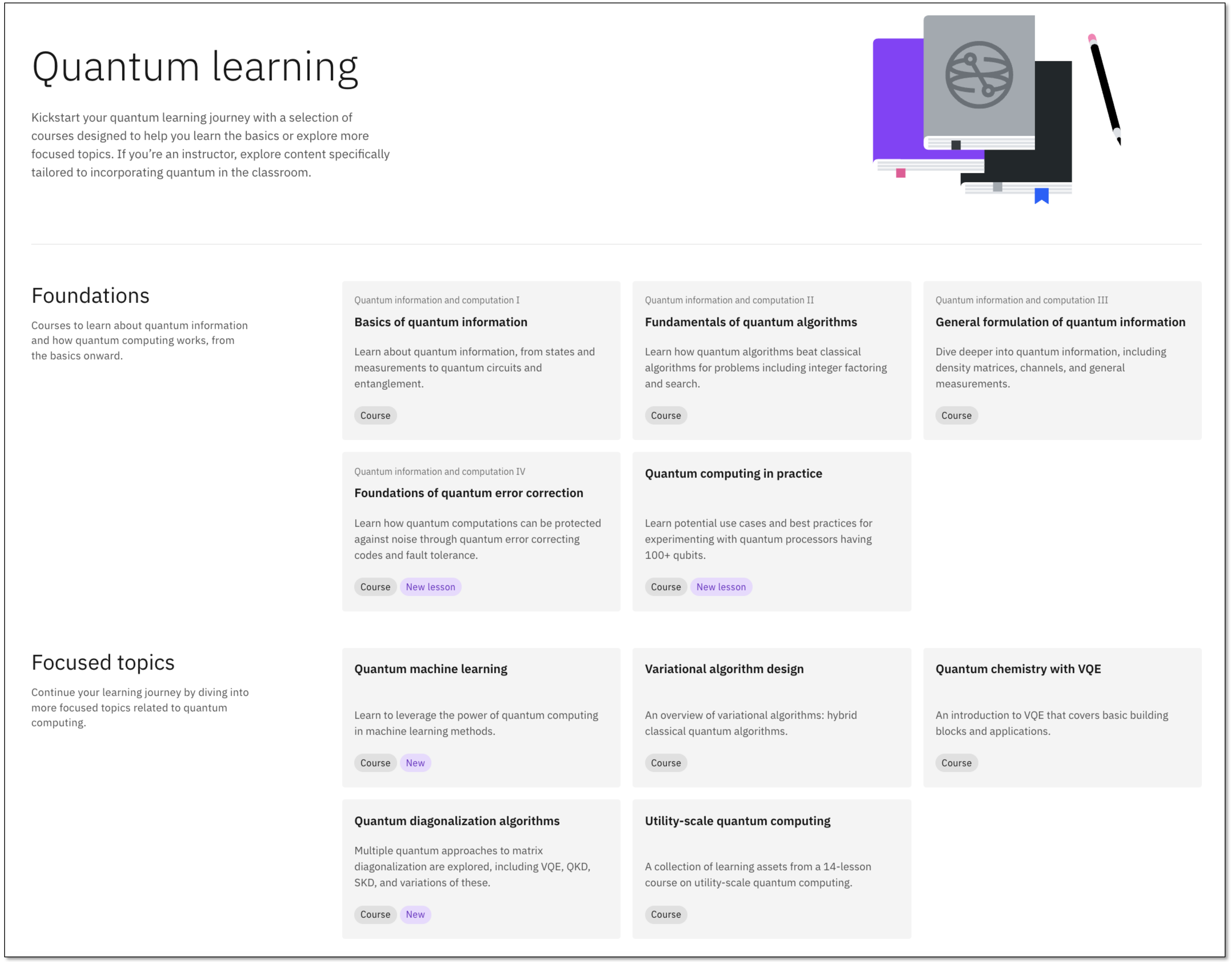
www.ibm.biz/qiskit-cert

IBM Quantum Learning

Learn the basics of quantum computing and how to solve real-world problems with IBM Quantum services and systems

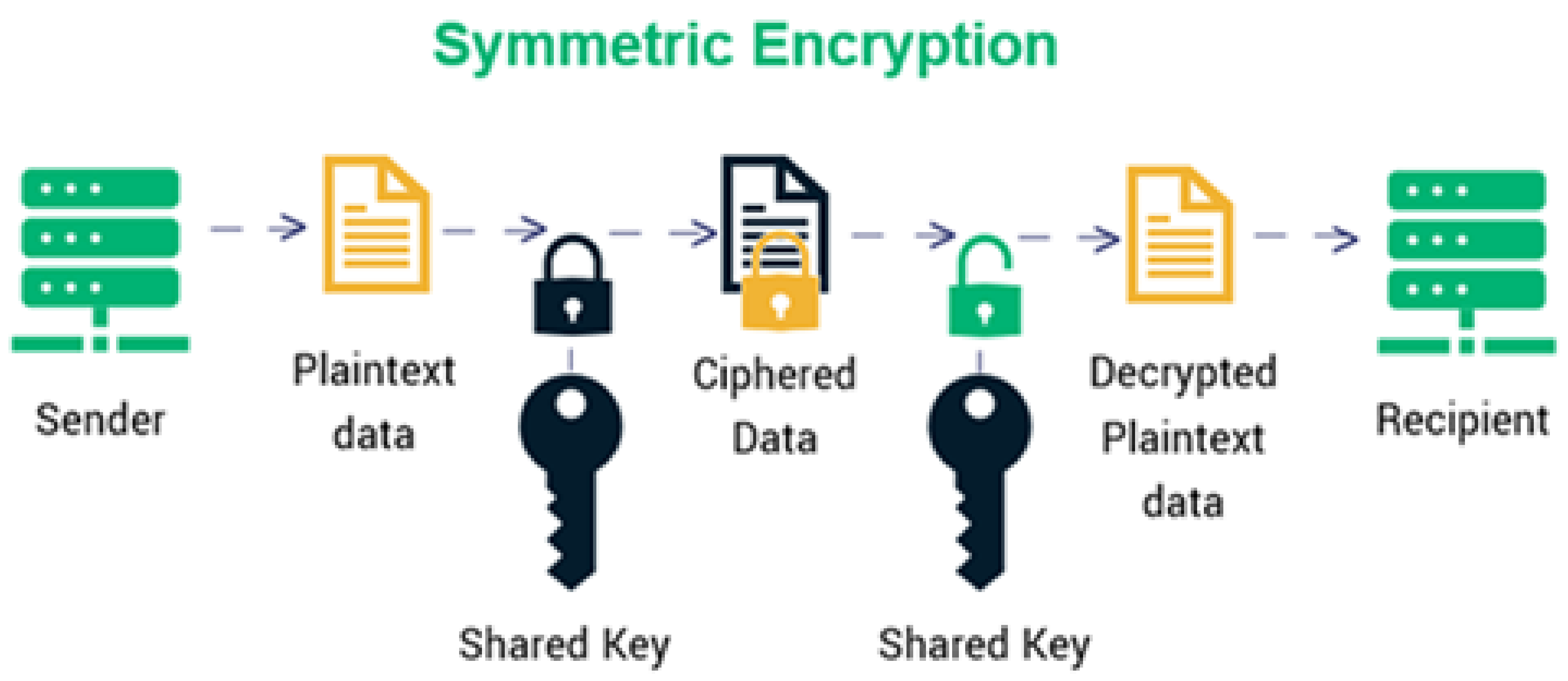
Courses, tutorials, and educational resources by leading quantum experts.

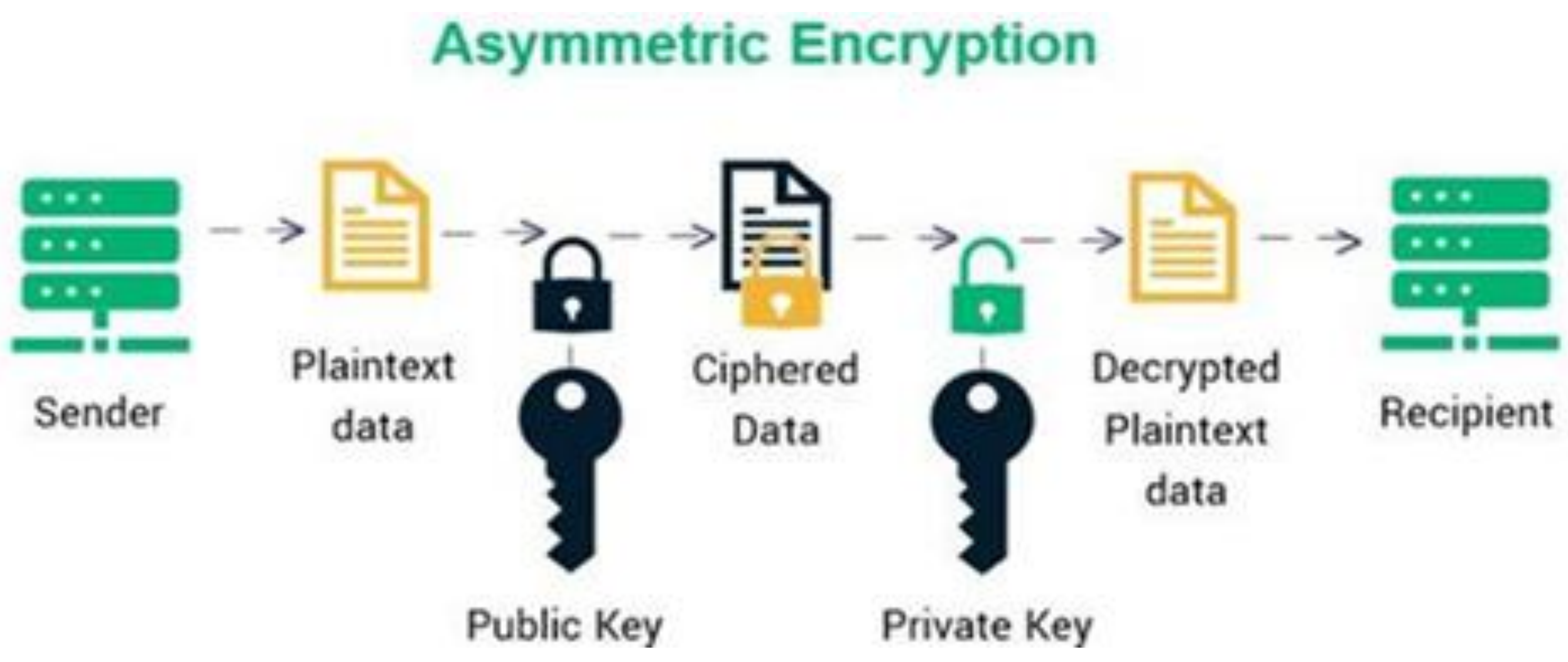
Qiskit Quantum Seminar



Shor's Algorithm

The fundamental crypto primitives





- Use **the same (secret) Shared key** for encryption and decryption.
- Simple and efficient
- Problem: the shared key need to be .... Shared
- Most common: Advanced Encryption Standard (AES), ChaCha20, (3DES, DES (deprecated))

- Uses two keys — a **public key** for encrypting data and a **private key** for decrypting it.
- Computationally more expensive
- Most common: Rivest-Shamir-Adleman (RSA), Elliptic Curve Cryptography (ECC), Digital Signature Algorithm (DSA)

Image source :https://sectigostore.com/blog/public-key-vs-private-key-how-do-they-work

How can we say they are safe?

- Both kinds of primitives are using varying degrees of mathematical structure. Breaking a primitive means solve an "hard" mathematical problem.
- Hardness assumptions: the hypothesis that a particular computational problem cannot be solved efficiently (in polynomial time).

Today's **asymmetric** encryption relies mostly on two mathematical problems with the hardness assumption:

Factoring

Given a number N, find a pair of prime numbers (p, q), $p < q$, $p \approx q$, whose product is equal to $N = p \cdot q$

Discrete Logarithms (DLOG)

Given a finite cyclic group G, a generator $\mathrm{g} \in G$ and an element $h \in G$ find the integer x where $g^x = h$

What changes in last 40 years?

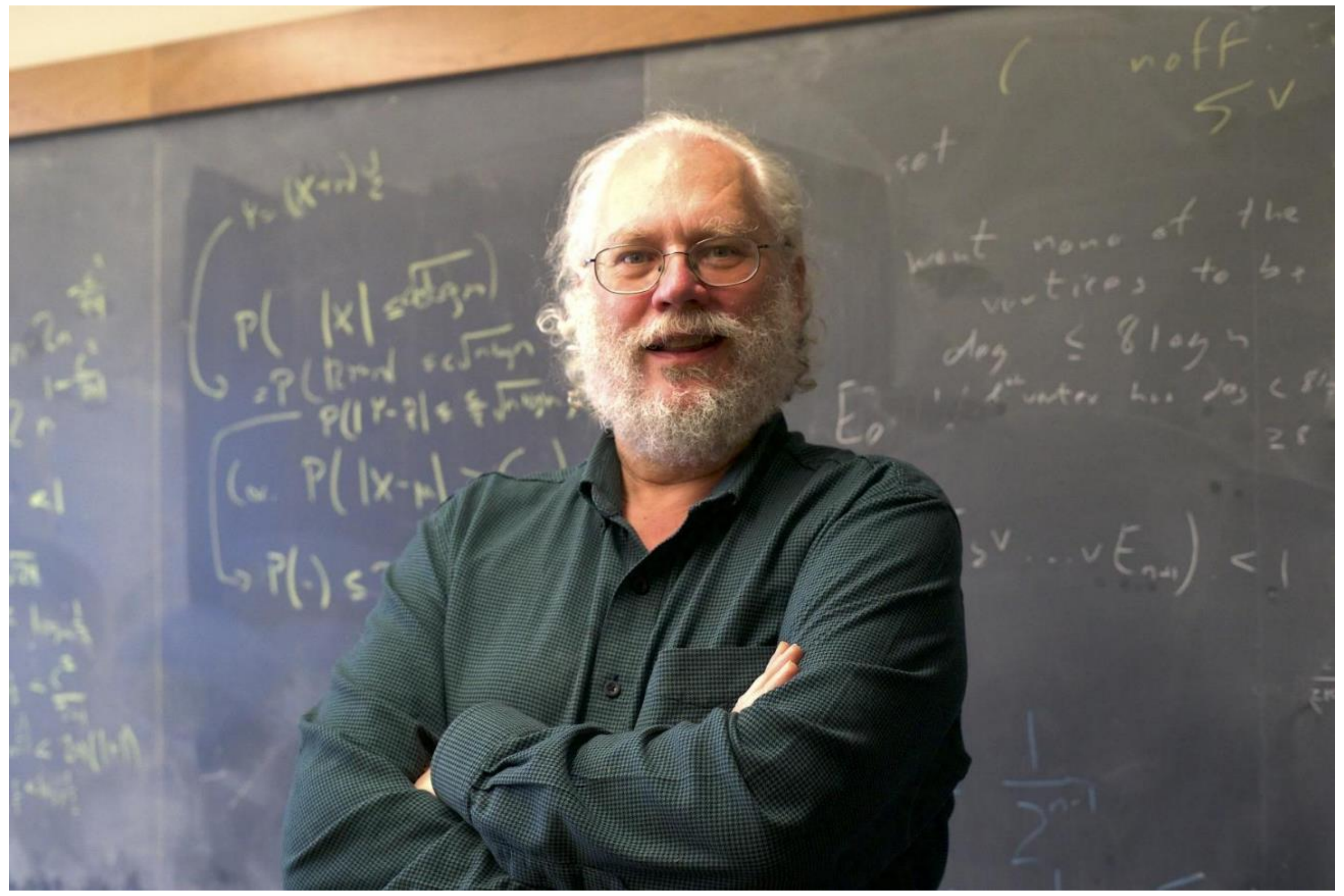
Shor's Algorithm

- General number field sieve (GNFS) is the most efficient classic algorithm for factoring integers larger than $\mathbf{10^{100}}$. Complexity:

$$\exp\left(\left((64/9)^{1/3}+o(1)\right)(\log n)^{1/3}(\log\log n)^{2/3}\right)$$

- Shor algorithm have a complexity of:

$$O\left((\log N)^2(\log\log N)(\log\log\log N)\right)$$

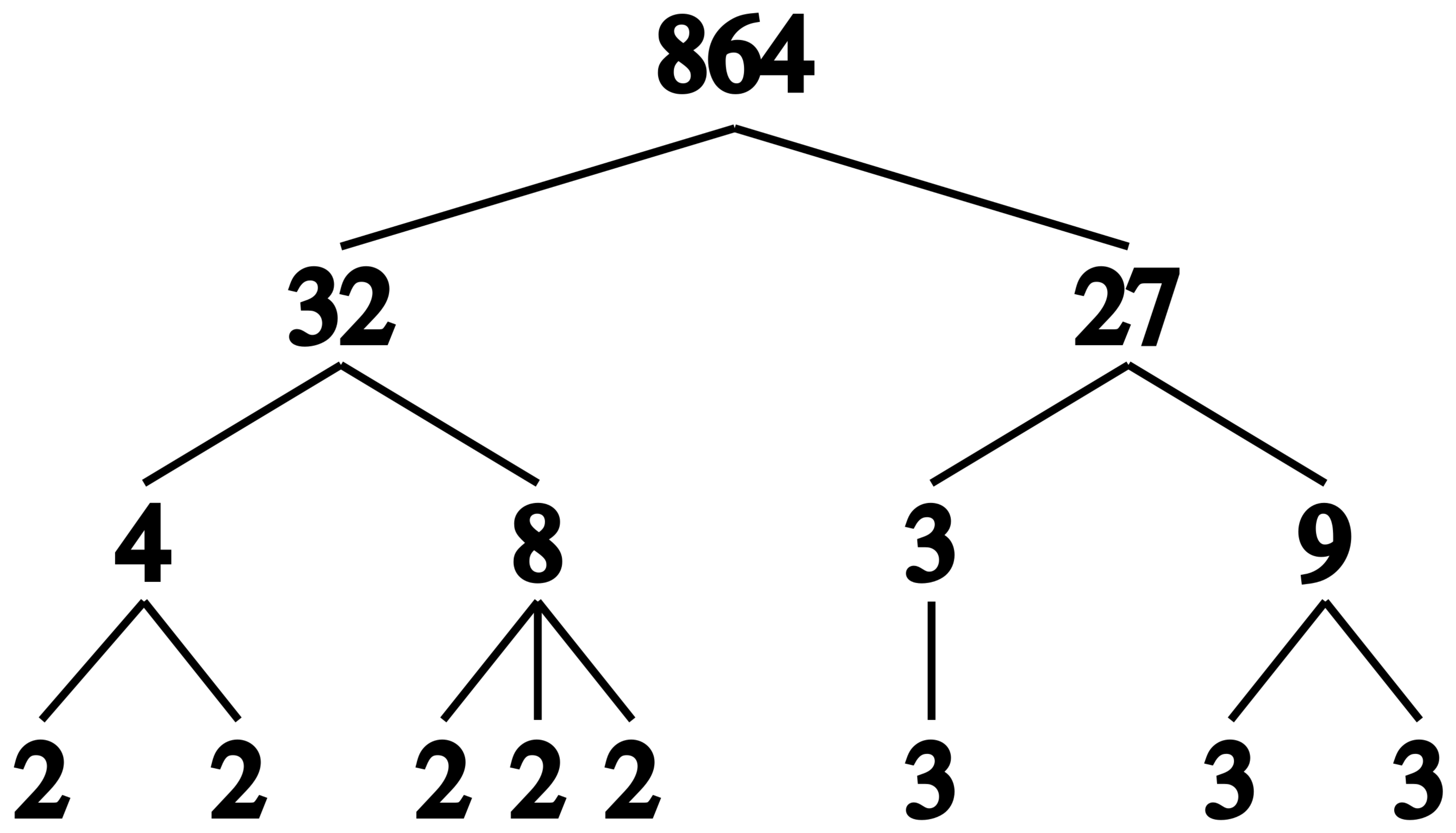


- We moved from subexponential in log N (hard) to poly-logarithmic (easy)
- Shor proposed multiple similar algorithms for solving the factoring_problem, the discret proposed e logarithm problem, and the period-finding problem
- In just one hit we lost two families of hardness assumptions widely used for PKC

Factorization problem

Fundamental theorem of arithmetic:

every integer greater than 1 can be represented uniquely as a product of prime numbers



How can we do it?

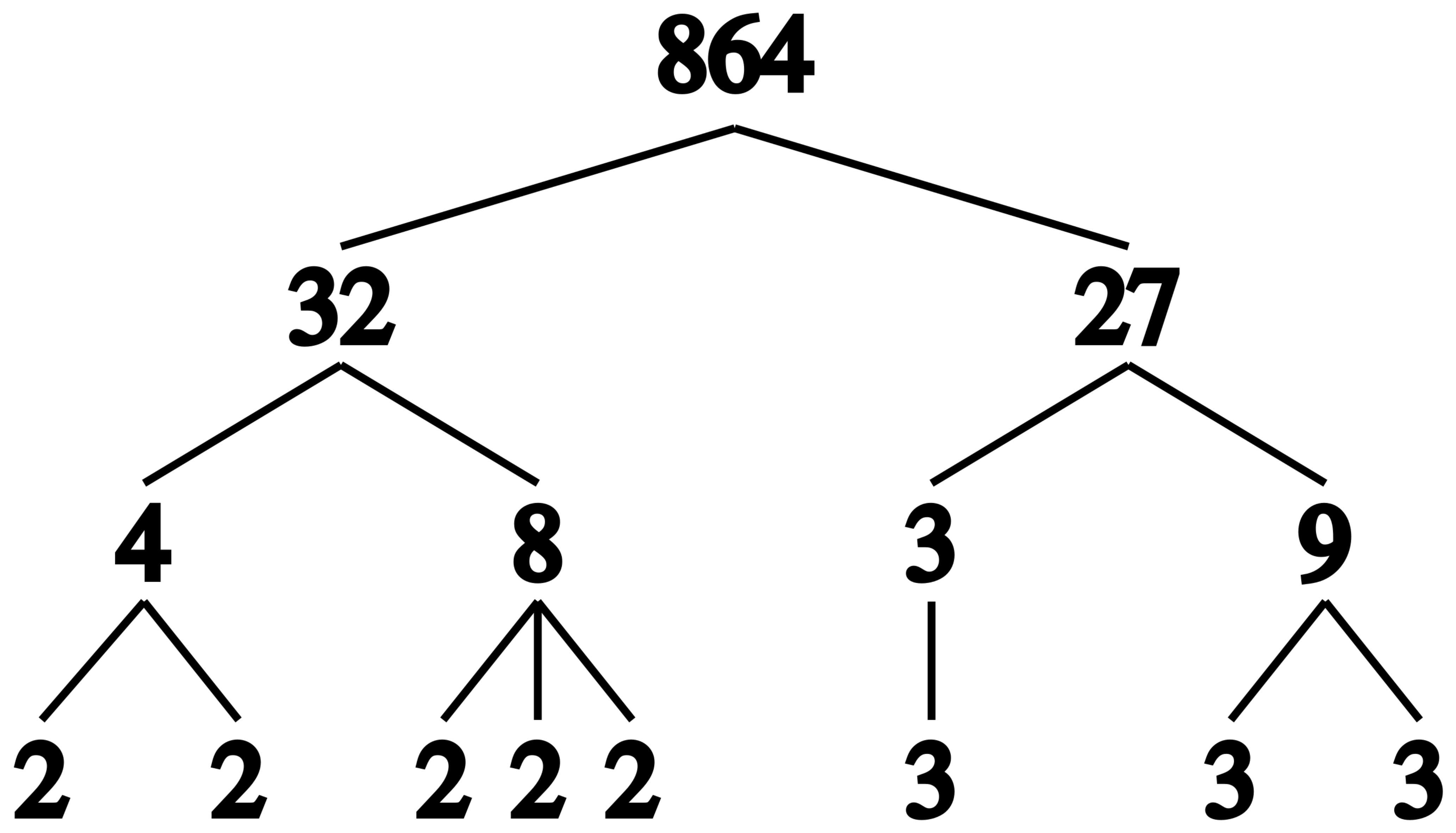
Base approach:

whether the number is divisible by the prime numbers, 2, 3, 5 and so on up to the square root of the number.

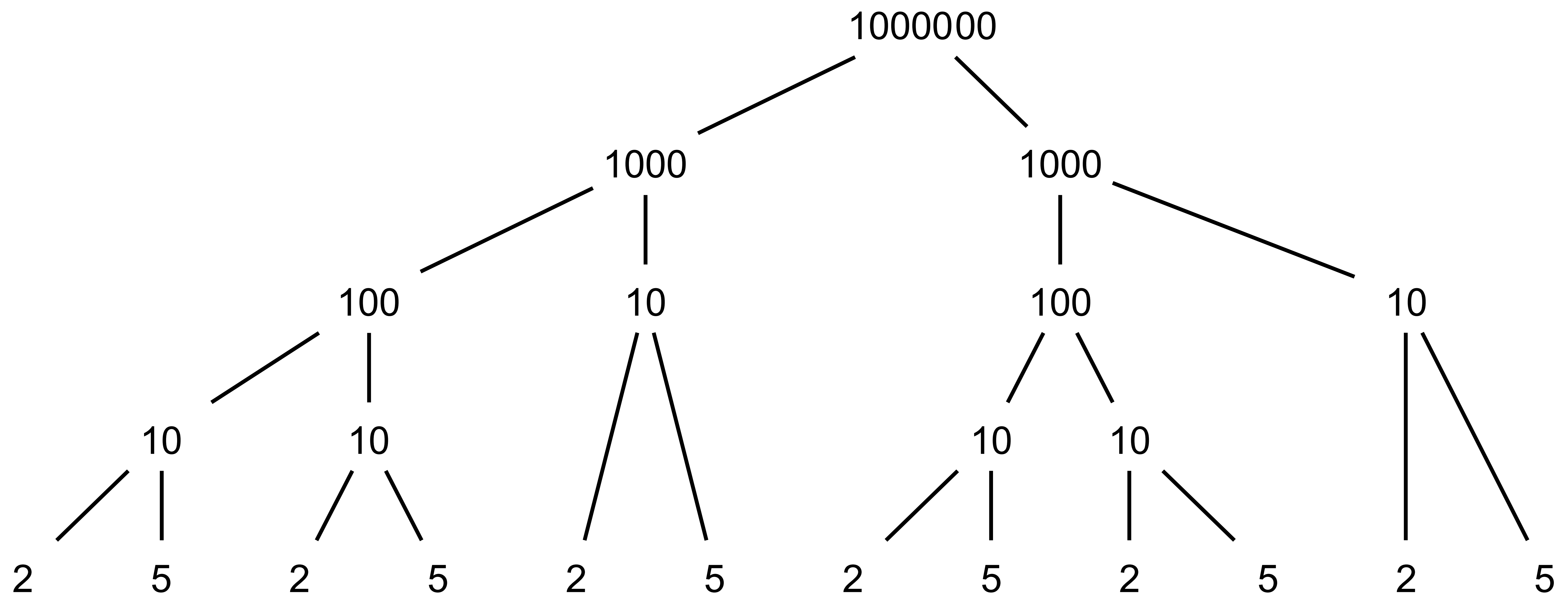
Best approach:

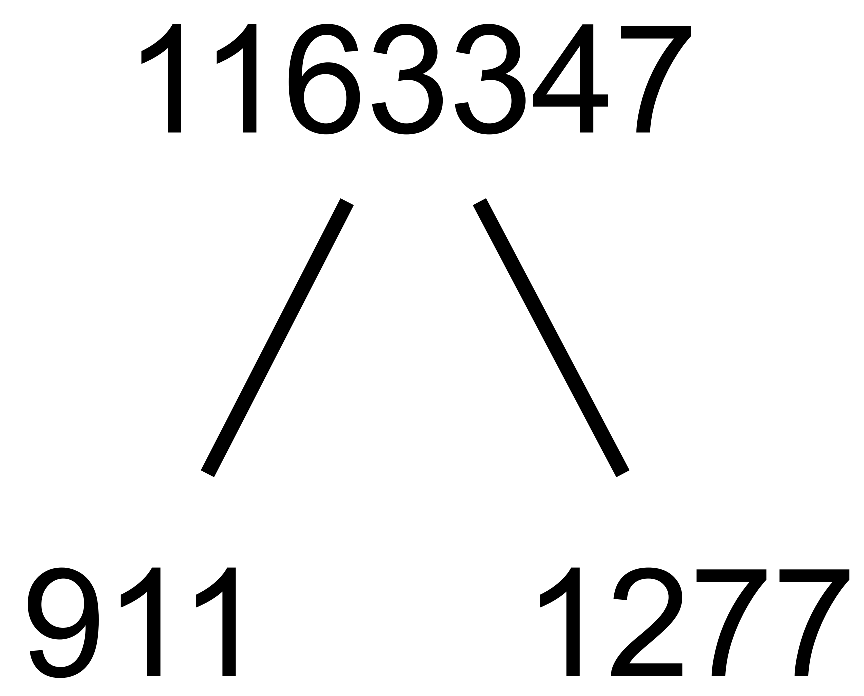
The General number field sieve (1993), for numbers > 10^{100}

$$\exp\left(\left(\left(\frac{8}{3}\right)^{\frac{2}{3}} + o(1)\right)(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}\right).$$



Different difficulties





Factorization in practice

Factoring a number N coincides with choosing a random integer m that is coprime (i.e. has no divisor in common) with N and finding the multiplicative order P , i.e. finding P such that:

$$m^P \equiv 1(modN)$$

where the symbol ≡ stands for congruence in modulus.

Thus, ∃k ∈ N such that

$$m^P - 1 = kN.$$

Shor's solution

Principle of Operation

Shor's algorithm is a masterpiece.
It works by reducing the factorization problem to a period-finding problem. Here are the key steps:

- Transforms the factorization problem into a period-finding problem
- Uses the Quantum Fourier Transform to determine this period
- Utilizes the discovered period to compute the factors of the original number

The true magic lies in the **Quantum Fourier Transform**, which can be executed exponentially faster on a quantum computer compared to a classical one.

Implementation Challenges

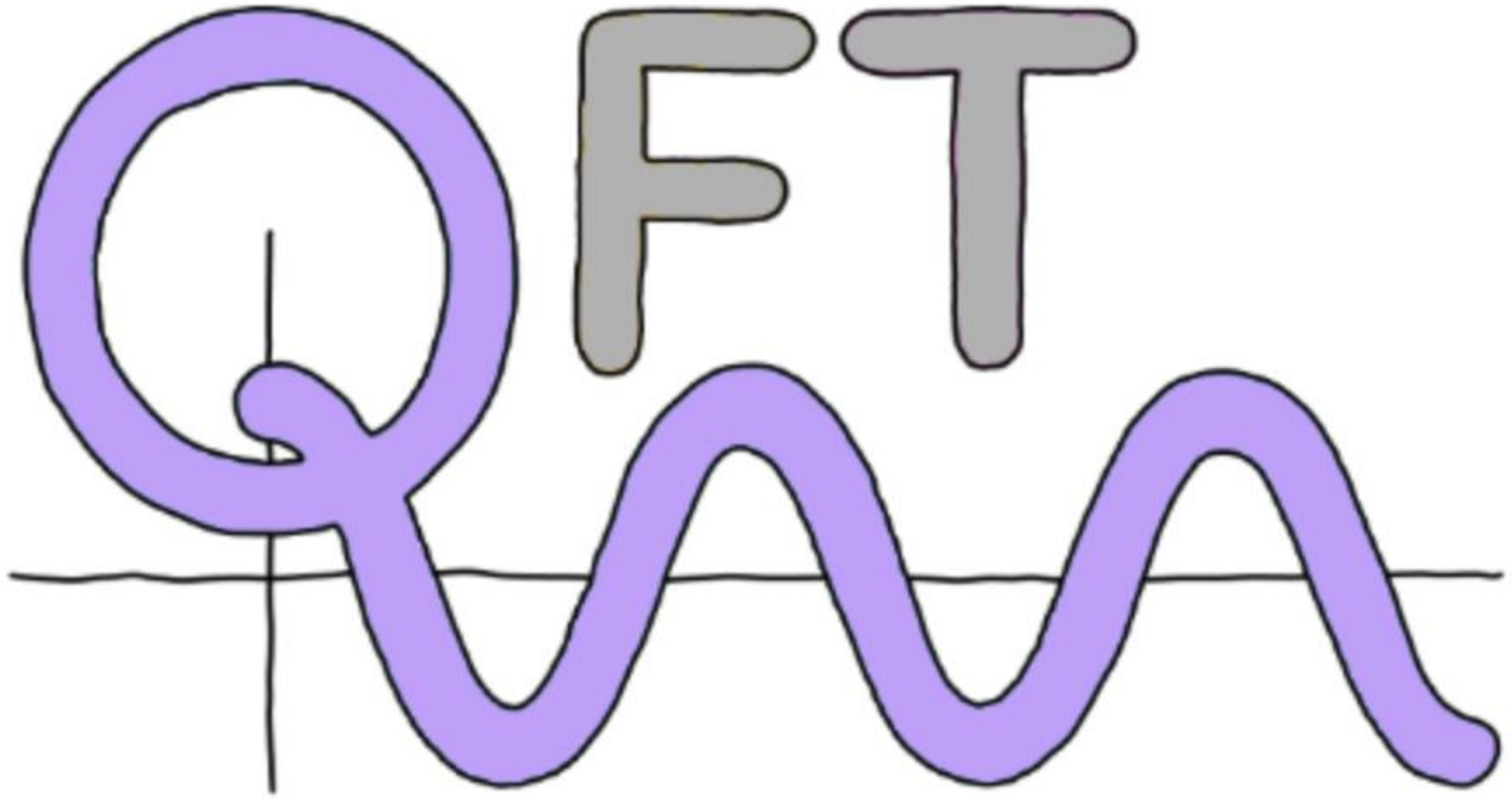
Despite its theoretical elegance, the practical implementation of Shor's algorithm presents significant challenges:

- Requires a large number of stable qubits
- Is highly sensitive to errors and decoherence
- Needs high-precision quantum gates

Quantum Fourier transform

The Quantum Fourier Transform (QFT) is a quantum analog of the discrete Fourier transform — the main tool of digital signal processing — which is used to analyze periodic functions by mapping between time and frequency representations.

The Discrete Fourier Transform (DFT) is a transform that converts a finite collection of N equispaced points of a function into a collection of coefficients of a linear combination of complex sinusoids, ordered with increasing frequency.



Quantum Fourier transform

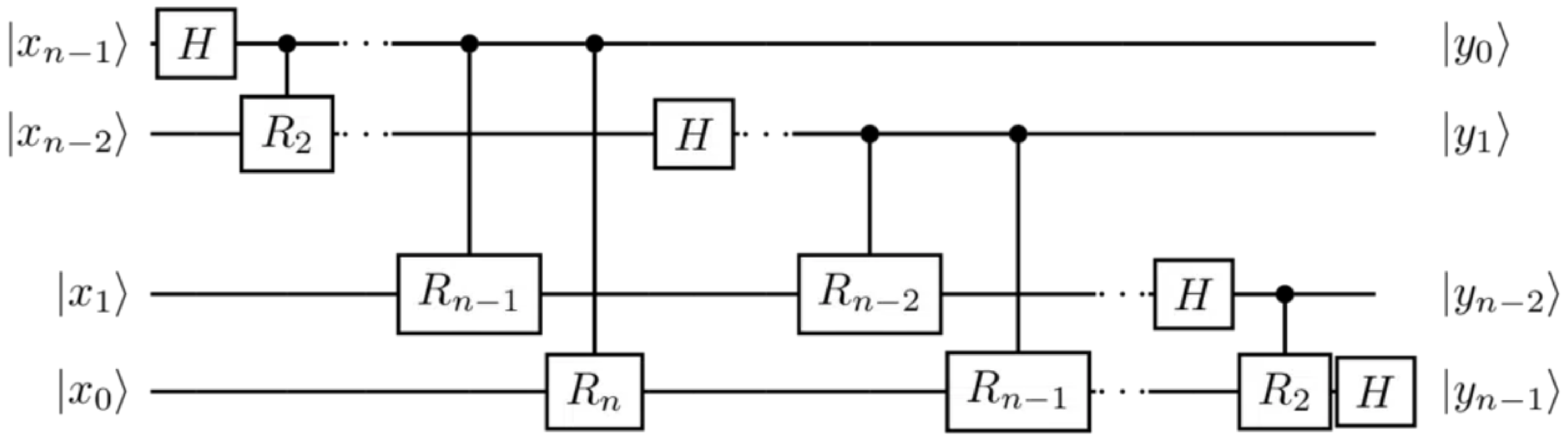
The Quantum Fourier Transform of the n-qubit |x⟩ is defined as

$$QFT(|x\rangle) = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi xy}{2^n}} |y\rangle.$$

The definition is then extended by linearity to all n-qubits.
Considering $w_l := e^{\frac{2\pi i}{2^l}}$,

$$e^{\frac{2\pi ixy}{2^n}} = w_n^{xy} = w_1^{xy_{n-1}} w_2^{xy_{n-2}} \cdots w_n^{xy_0}.$$

$$\begin{aligned} QFT(|x\rangle) &= \frac{1}{\sqrt{2^n}} \sum_{y_0,\ldots y_{n-1}\in\{0,1\}} w_1^{xy_{n-1}} \cdots w_n^{xy_0} |y_{n-1} \ldots y_0\rangle \\ &= \frac{|0\rangle + w_1^x|1\rangle}{\sqrt{2}} \frac{|0\rangle + w_2^x|1\rangle}{\sqrt{2}} \cdots \frac{|0\rangle + w_n^x|1\rangle}{\sqrt{2}} \end{aligned}$$



Quantum Phase Estimation

Quantum Phase Estimation (QPE) is a fundamental quantum algorithm used to determine with high precision the phase φ associated with an eigenvalue of a unitary operator U. This algorithm is crucial in various quantum algorithms, including Shor's algorithm for factorization and quantum simulation.

Given a unitary operator U and an eigenstate |u⟩ with the corresponding eigenvalue

$$e^{2\pi i\theta}$$

the goal is to estimate the phase φ, which is a fractional number between 0 and 1.

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

The QPE algorithm allows us to obtain an estimate of φ with increasing precision as the number of qubits used increases.

Shor's Algorithm

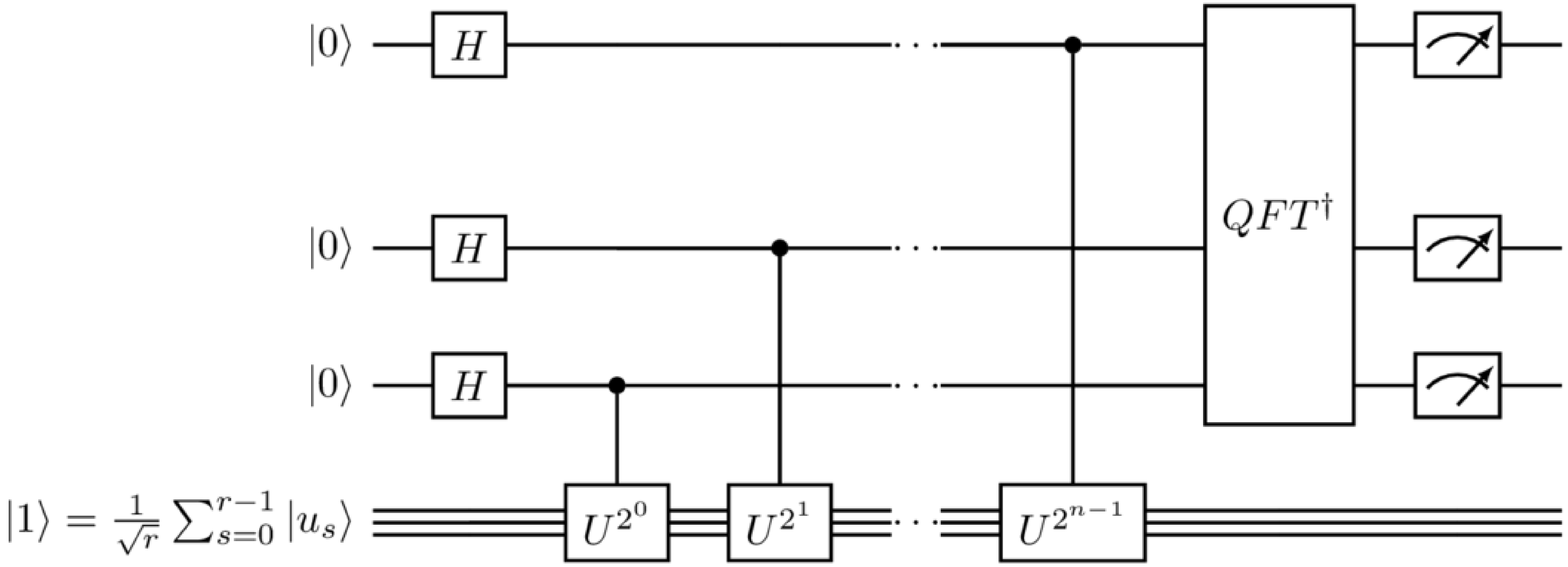
Let N be a positive integer and b a coprime integer with N, the objective of Shor's quantum algorithm is to find the period of the function

$$f : x \mapsto b^x \mod N$$

in a polynomial number of log N steps. We use a unitary operator

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

Thus, if one proceeds to the QPE of U using |1⟩ which is a superposition of eigenstates. If you do not have a solution, you only get an estimate of the desired value and proceed with continuous fractions to find it.



Shor's Algorithm in practice

N=15
b=2

The sequence $2^x mod\ 15$ produce 2, 4, 8, 1, 2, 4, 8, 1 ... a function with period r=4.

$$2^{\frac{4}{2}} - 1 = 3$$

$$2^{\frac{4}{2}}+1 = 5$$

3 and 5 are the factors of 15

Recap: From Hardness to Vulnerability

Classical World (today)

Security relies on hard mathematical problems

- Factoring large integers (RSA)
- Discrete logarithm (Diffie–Hellman, ECC)

Best classical algorithms: subexponential in logN

Practically secure — impossible to break in reasonable time

Quantum World (tomorrow)

Shor's algorithm solves factoring and discrete log in polynomial time

Grover's algorithm weakens symmetric crypto (square-root speed-up)

Public-key cryptography as we know it becomes obsolete

Implications

Transition to Post-Quantum Cryptography (PQC)

- Based on lattice, hash-based, and code-based assumptions
- NIST standardization in progress

"Harvest now, decrypt later" risk → migration must begin today

Shor’s Algorithm in practice