Exact quantum factoring over a fixed finite gate set

OpenAI

September 25, 2026

Abstract

We give a polynomial-time uniform quantum circuit family that outputs the complete prime factorization of every integer $N \geq 2$ with probability one. A fixed finite set of bounded-arity gates suffices, and both the gate count and the number of qubits have polynomial worst-case bounds in the input length. The construction first gives an order trial with a prescribed dyadic success probability. A factoring run then generates verifiable auxiliary factorizations that determine its transition probabilities; filters evaluated from the completed data give the full preparation a known exact success probability. A final dyadic correction makes this probability $1/4$, so one amplitude-amplification step using sign reflections yields certainty.

2020 Mathematics Subject Classification. Primary 81P68; Secondary 11Y05, 68Q12.
Keywords. Quantum factoring, exact quantum computation, finite gate sets, amplitude amplification, dyadic probabilities.

1 Introduction

The integer factorization problem asks, on input a binary integer $N \geq 2$, for the nondecreasing list of primes whose product is N, including multiplicities. We consider quantum circuits on qubits with classical basis inputs. A circuit family is polynomial-time uniform if a deterministic classical algorithm generates the circuit for inputs of each bit length in time polynomial in that length.

Theorem 1.1. *There is a polynomial-time uniform family of polynomial-size quantum circuits that outputs the complete prime factorization of every input integer $N \geq 2$ with probability one. The number of gates and qubits is bounded by a fixed polynomial in the bit length of N. One fixed finite gate set suffices: NOT, CNOT, Toffoli, Hadamard,* $\mathrm{diag}(1, i)$*, their inverses, and their singly controlled versions.*

Shor's quantum algorithms established polynomial-time factorization and discrete logarithms with bounded error [17]. Their order-based factoring reduction builds on Miller's work [10, 17]: suitable multiplicative orders produce nontrivial square roots of unity and hence gcd splits. The splitting argument used here is proved directly in Section 4. Verifying a proposed factorization makes repetition reliable, but a procedure that retries until success need not have a polynomial worst-case running time. Approximating an arbitrary-angle rotation to increasingly high precision likewise does not by itself yield probability-one correctness over a fixed finite gate set.

1.1 How the proof reaches certainty

A preparation with success probability exactly 1/4 can be amplified to certainty by one pair of sign reflections. The difficulty is to obtain that exact probability using the finite gates while the factorization is still unknown. The construction makes a successful factoring run produce the auxiliary information needed to calculate its own transition probabilities. It then tests the recorded transitions using those verified data. A proof that these deferred tests mark exactly the same runs as tests using the true data gives a known success probability before the algorithm runs.

The argument proceeds through four constructions.

1. *An order trial with a prescribed probability (Section 3).* For a unit a modulo m, its order is the least positive integer r with $a^r \equiv 1 \pmod{m}$. Proposition 3.1 constructs a trial that returns no candidate or a positive multiple of r, and gives the true order a specified dyadic probability. Here a dyadic rational has a power-of-two denominator. A four-phase auxiliary state implements a piecewise-linear approximation to Fourier phases; two small sampling corrections make its true-order probability exact.

2. *A factoring process that produces its own auxiliary data (Section 4).* Besides factoring the input, the process recursively factors $p - 1$ for each odd prime factor p that it encounters. This occurrence tree has polynomial size. Proposition 4.1 shows that the completed tree determines exact orders and transition probabilities, is efficiently verifiable, and is generated whenever all transition outputs are good. Prime labels are checked by the deterministic test proved in Appendix A.

3. *A known success probability from deferred tests (Section 5).* Ordinary transitions are mixed with uniform guesses of their outputs. Lemma 5.1 gives filters with the same retained mass at each transition. The algorithm computes these filters after verifying the completed tree. Proposition 5.2 identifies its acceptance event, on each recorded run, with the event that tests every transition against the true data. Passing all those tests itself forces the tree to be generated, which justifies the probability calculation.

4. *A quarter-success preparation and exact amplification (Section 5.3).* Lemma 5.3 makes a second dyadic correction, now to the whole preparation, so that its success probability is exactly 1/4. The closing proof of Theorem 1.1 applies the two sign reflections. Retained classical records and reversible control from Section 2 give the required unitary preparation and its inverse.

The exact probability calculation uses the auxiliary data only after the run has generated them. Its analysis compares predicates on the same recorded runs; it neither supplies advice to the circuit nor conditions on eventual completion. This separates the arithmetic problem of making useful events have computable exact masses from the verification problem of recovering the information needed to evaluate those masses.

1.2 Relation to earlier exact algorithms

Exact transforms and algorithms have been developed in several quantum models. Mosca and Zalka [11] construct exact Fourier transforms using computed rotations and obtain exact discrete-logarithm algorithms for known group orders. Their discussion also reports Mosca's exact factoring algorithm in a generalized model permitting gate parameters obtained during the computation [11]; see also [8]. Imran's exact order-finding algorithm takes a supplied multiple of the unknown

order [8]. A suitable multiple of polynomial bit length is additional information, which the factoring problem does not provide initially. These are material model and input distinctions. Nishimura and Ozawa [12] study exact computation by finitely generated uniform circuit families; their results also explain why approximate universality cannot replace an explicit gate analysis in an exact algorithm. Theorem 1.1 uses the single finite gate set stated there and receives only the integer to be factored.[1]

The controlled modular shifts in the order trial are related to the phase-kickback construction of Cleve, Ekert, Macchiavello, and Mosca [5]. Our auxiliary register is a four-phase step state rather than a Fourier eigenstate; testing its return yields the interpolated matrix with its actual, unnormalized probability. Lev's four-phase approximation to the order-sampling transform [9] provides a related approach to avoiding fine phase rotations.[2] The interpolation gives a different matrix on the auxiliary-register return event, and the corrections prescribe the trial's true-order success law. Its residue-class probabilities have dyadic majorants whose sums can be evaluated through a bounded number of polynomial pieces and exact power sums. Mosca and Zalka use power sums to approximate a transform's success probability to arbitrary precision, then use a rotation with a computed angle to obtain an exact transform [11]. Here the finite polynomial pieces give exact dyadic sums. The trial requires neither the true order nor its factorization.

The auxiliary $p-1$ tree is related to Pratt's succinct primality certificates [15]. Its role here is to determine orders and transition probabilities; the separate primality test in Appendix A adapts the polynomial-congruence method of Agrawal, Kayal, and Saxena [1], with its constants and auxiliary-prime choice established directly. Quantum rejection sampling [13] is a related use of selective retention and amplification, with different state-conversion inputs and parameterized rotations. The one-transition completion argument in Lemma 5.1 also requires, for its efficient application here, verification of the auxiliary data and the all-good generation property in Proposition 4.1.

The final reflection step belongs to amplitude amplification, originating in quantum search [6, 4]. Exact probability engineering already appears in Brassard and Høyer's exact algorithm for Simon's problem [3], and Mosca and Zalka explicitly use success probability $1/4$ followed by one ordinary amplification step [11]. General exact amplification may employ phases depending on the success probability [7, 4]. The dyadic construction here reaches that probability with the fixed gates, so the established pair of sign reflections suffices.

Recent quantum factoring algorithms also address resource costs. Regev's multidimensional construction [16] initially used a number-theoretic hypothesis; Pilatte [14] proves unconditional correctness for modified algorithms. Their success guarantees retain a nonzero failure probability. The present construction addresses certainty and worst-case polynomial bounds in a specified exact gate model. The resource bounds count quantum gates and qubits. Its large polynomial parameters are not intended as practical resource improvements.

2 Exact circuits and classical records

Exact dyadic coins and retained classical records will let us implement the probability calculations coherently over the finite gate set. All logarithms in bit bounds are to base two. For an input integer $N \geq 2$, put

$$n = \max\{128, \lfloor \log_2 N \rfloor + 1\}, \qquad B = 2^n. \tag{1}$$

[1] The model and input distinctions are discussed in [11, Section 5.2] and [8, Sections 1 and 3].

[2] The four-phase transform is described in [9, Sections 3–4].

Thus $N < B$. The cutoff at 128 is a constant and has no effect on polynomial-time uniformity. Unless indicated otherwise, a polynomial bound is in n.

Our fixed gate set contains NOT, CNOT, Toffoli, the Hadamard gate H, and the phase gate S, where

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

We also allow their inverses and singly controlled versions. This is still a fixed finite set of bounded-arity gates. Gates with more controls will be implemented by computing the conjunction of the controls into a scratch bit, applying a singly controlled gate, and uncomputing the conjunction. In particular, signs conditioned on a polynomial-size Boolean test are exact: compute the test, apply S^2 to its result, and uncompute.

Recall that a *dyadic rational* has the form $A/2^T$ with $A \in \mathbb{Z}$ and an integer $T \geq 0$. Its representation has polynomial bit length if T and the bit length of $|A|$ are polynomially bounded. Integers, rational numbers, and Gaussian rationals are always represented exactly. Euclidean division, gcd, and modular powering with polynomial-length binary exponents take polynomial bit time. We use the reversible simulation and cleanup method of Bennett [2], compiled into the Boolean gates of Toffoli [18, Section 5]. A deterministic computation with a polynomial clock bound can be unrolled into a uniformly generated Boolean circuit. Each intermediate AND is computed into fresh scratch with a Toffoli gate; NOT and CNOT implement complements, exclusive-or operations, and copies. Retaining the intermediate values, copying the answer, and reversing the gates clears scratch with polynomial time and space overhead. Exact arithmetic networks over this basis are also developed by Vedral, Barenco, and Ekert [19].

Lemma 2.1 (Exact dyadic sampling)**.** *Let a classical basis label specify a dyadic $q \in [0, 1]$ by a polynomial-time computation with polynomially bounded bit lengths. A circuit over the gates of Theorem 1.1 can append a flag that equals* 1 *with probability exactly q, in polynomial time, without altering the label.*

Proof. Choose a common polynomial bound T on the denominator exponent, and compute A such that $q = A/2^T$. Applying T Hadamards to fresh zero bits gives a uniform label $u \in \{0, \ldots, 2^T - 1\}$. The reversible comparison $[u < A]$ sets the flag on exactly A labels. The circuit retains u and all required work bits. Padding to the common T handles a denominator depending on the input label. □

When a procedure is said to *retain* an event with probability q, it appends such a flag and requires that flag for its eventual success. It does not discard and renormalize the other branches. An ancillary projection likewise denotes a tested event whose actual squared norm contributes to the success probability.

Lemma 2.2 (Retained classical records)**.** *Consider a procedure with polynomially bounded time and workspace that uses the gates of Theorem 1.1, computational-basis measurements, and polynomial-time classical control. Suppose subsequent quantum operations are controlled by the measurement outcomes as classical data. It has a polynomial-size uniform unitary implementation with the same distribution on its classical outputs and success flag, while retaining polynomially many history and work qubits. The inverse implementation uses the same fixed gate set.*

Proof. In place of a basis measurement, copy its outcome by CNOT gates into fresh history bits. Retain this copy and thereafter use it only as a classical control. The branches indexed by different recorded histories remain orthogonal. Induction over the procedure shows that their squared

norms and subsequent outcome distributions equal those in the measured procedure. Quantum work not measured in the original procedure remains quantum work; in particular, no label is copied before an intended interference computation.

Run the reversible classical controller for a fixed polynomial number of ticks, keeping its history and using a halted flag after termination. The circuit implements the controller directly, with its history retained. For a bounded-arity instruction that selects registers by addresses, it enumerates all tuples of addresses in the polynomial workspace. For each tuple, compute whether the controller selects it, apply the corresponding singly controlled primitive, and uncompute this test. The number of tuples is polynomial because the arity is fixed. A polynomial bound on ticks, workspace, and stored records therefore gives a polynomial circuit whose description can be generated in polynomial time. Reverse the gate list and invert each primitive to obtain the inverse. □

We will also use the following consequence when analyzing an adaptive procedure. Fix a retained classical history h. Its current classical inputs are fixed basis labels, while old quantum work may be in an arbitrary state ρ_h. If the next trial uses fresh zero registers and leaves that old work alone, it acts as $I_{\mathrm{old}} \otimes U_h$. Its outcome probabilities are those of U_h alone, independently of ρ_h. This observation permits conditional probability calculations along recorded histories even though the full implementation is unitary.

Finally, exact deterministic primality is needed to verify prime-factor lists. Lemma A.1 in Appendix A gives an elementary polynomial-time test for all $2 \le m < B$. For the main construction, its only required interface is an exact yes-or-no decision with a polynomial bit-time bound.

3 An order trial with prescribed success probability

For a unit a modulo m, write $\mathrm{ord}_m(a)$ for its multiplicative order. We construct a trial whose probability of returning the true order is an explicit dyadic function of that order. This additional control of the probability will allow us to make the final factoring algorithm exact. As usual, let

$$\varphi(d) = \big|\{0 \le j < d : \gcd(j, d) = 1\}\big|, \qquad \varphi(1) = 1.$$

Proposition 3.1. *Let n, B be as in (1). Let $2 \le m < B$ and let a be a unit modulo m, of order r. There is a uniform quantum trial of worst-case cost polynomial in n that returns either no candidate or an integer d satisfying $1 \le d < B$ and $r \mid d$. Moreover,*

$$\Pr(d = r) = \varphi(r)\lambda(r), \qquad \lambda(d) = 2^{-\lceil \log_2 d \rceil - 10}.$$

The trial uses only the gate set of Theorem 1.1 and does not require r or any factorization as input.

The construction has two parts. First we replace the Fourier phases in order sampling by a piecewise linear phase function. The resulting probabilities admit exact dyadic arithmetic. We then add two small classical sampling branches to make the mass at each reduced fraction of denominator r exactly $\lambda(r)$. The proof of Proposition 3.1 is completed at the end of this section.

3.1 A phase function implemented by a circuit

Set

$$Q = B^{16} = 2^b, \qquad b = 16n. \tag{2}$$

This power of two makes the later bin-rounding errors and correction probabilities small, while the register length remains linear in n. Let $g : \mathbb{R} \to \mathbb{C}$ be the continuous, 1-periodic function obtained by linear interpolation of $\exp(-2\pi \mathrm{i} z)$ at the quarter-integers. Explicitly, for $q \in \mathbb{Z}$ and $0 \le t \le 1$,

$$g((q+t)/4) = (1-t)(-\mathrm{i})^q + t(-\mathrm{i})^{q+1}. \tag{3}$$

In particular, $g(z + 1/2) = -g(z)$.

The controlled-shift mechanism is a form of phase kickback [5, Section 7]. The auxiliary state below is not a shift eigenstate: its overlap with a shifted copy will give g. Testing its return to the initial state realizes the matrix in the next lemma. The matrix describes a tested branch whose squared norm contributes to the trial's success probability.

Lemma 3.2. *Let Q, b be as in (2), and let g be the periodic phase function defined by (3). There is a circuit of size polynomial in b on a b-qubit data register and ancillary registers initially in the all-zero state, with a designated ancillary success outcome, such that its unnormalized matrix on that outcome is*

$$A_{k,x} = Q^{-1/2} g(kx/Q), \qquad 0 \le x, k < Q.$$

The circuit uses only the gate set of Theorem 1.1.

Proof. We first realize g as a shift overlap, then combine the shifts with Hadamard matrix elements to obtain the required data-register matrix.

Shift overlap. Prepare a b-qubit phase register in the state

$$|\chi\rangle = Q^{-1/2} \sum_{y=0}^{Q-1} \mathrm{i}^{\lfloor 4y/Q \rfloor} |y\rangle.$$

This requires Hadamards on its bits, followed by a $Z = S^2$ phase on the most significant bit and an S phase on the next bit. For $D \in \mathbb{Z}$, let T_D denote the forward shift $|y\rangle \mapsto |y + D \bmod Q\rangle$ on this register. We first verify that

$$\langle \chi | T_D | \chi \rangle = g(D/Q). \tag{4}$$

Write $D = qQ/4 + s$ modulo Q, where $0 \le q < 4$ and $0 \le s < Q/4$. Among the Q positions, a fraction $1 - 4s/Q$ crosses exactly q quarter-bins, and the remaining fraction $4s/Q$ crosses $q + 1$. The corresponding products of the initial phase with the conjugate final phase are i^{-q} and i^{-q-1}, respectively. Their average is precisely (3) at D/Q. The sign is fixed by the use of a forward shift: a shift by $Q/4$ has overlap $-\mathrm{i}$.

Controlled shifts. Number input and output bits from the units bit: $x = \sum_{i=0}^{b-1} 2^i x_i$ and $k = \sum_{j=0}^{b-1} 2^j k_j$. For $j = 0, \ldots, b-1$, perform the following operations in order: apply a Hadamard to the physical bit that initially carried x_{b-1-j}, regard its new value as k_j, and, for every $i < b-1-j$, apply the shift

$$T_{2^{i+j} x_i k_j}$$

to the phase register. Both controls are available: k_j is the bit just produced, whereas x_i has not yet been overwritten. The logical output bits are thus in the reverse physical order from the input bits; a wire reversal, or ordinary SWAP gates, puts them in the conventional order.

Matrix entries and success test. For fixed input x and output k, the single Hadamard on each data wire fixes its input and output values. Thus every shift control has its specified basis value

along the contributing path. The product of the Hadamard matrix elements is $Q^{-1/2}(-1)^h$, and the total phase-register shift is D, where

$$h = \sum_{i+j=b-1} x_i k_j, \qquad D = \sum_{i+j<b-1} 2^{i+j} x_i k_j.$$

All omitted terms in the product xk are accounted for by

$$xk \equiv D + (Q/2)h \pmod{Q}.$$

Using (4) and $g(z+1/2) = -g(z)$ gives the claimed matrix entry after projecting the phase register back onto $|\chi\rangle$:

$$Q^{-1/2}(-1)^h \langle\chi|T_D|\chi\rangle = Q^{-1/2} g(xk/Q).$$

To perform that test, undo the preparation of $|\chi\rangle$ and designate the all-zero phase-register outcome as success. Its actual probability is retained as part of the trial's success probability.

There are $O(b^2)$ controlled shifts. Each is a reversible modular binary addition on b bits, with its scratch bits uncomputed afterwards, and therefore has polynomial cost in the stated finite gate set. All other operations also have polynomial cost. □

3.2 Recovering fraction labels and their probabilities

For every reduced fraction j/d with $0 \le j < d < B$, define

$$k(d,j) = \left\lfloor \frac{jQ}{d} + \frac{1}{2} \right\rfloor, \qquad \eta = k(d,j) - \frac{jQ}{d}. \tag{5}$$

Thus $|\eta| \le 1/2$, and $0 \le k(d,j) < Q$. We call (d,j) the label of this bin.

Lemma 3.3. *Let Q be as in (2), and let $k(d,j)$ be the bin function in (5). Given $0 \le k < Q$, one can determine in polynomial bit time whether $k = k(d,j)$ for a reduced fraction $0 \le j < d < B$, and recover (d,j) when it exists. Such a label is unique.*

Proof. If k has a valid label, separation of reduced fractions will force a sufficiently close continued-fraction convergent to be the corresponding reduced fraction. Compute the finite simple continued fraction of k/Q, and let p/v be its last convergent with denominator less than B. If all convergents have denominator less than B, use the terminal one. We have

$$\left| \frac{k}{Q} - \frac{p}{v} \right| \le \frac{1}{vB}. \tag{6}$$

Indeed, suppose p/v is not terminal. Let (p', v') be the preceding numerator–denominator pair, using the formal pair $(1,0)$ when p/v is the first convergent. The remaining continued-fraction tail θ gives

$$\frac{k}{Q} = \frac{p\theta + p'}{v\theta + v'}, \qquad |pv' - p'v| = 1.$$

The next denominator is $vc + v' \ge B$, where c is the integer part of θ. Consequently the error is $1/[v(v\theta + v')] \le 1/[v(vc + v')] \le 1/(vB)$. The determinant identity follows inductively from the convergent recurrence, which replaces each numerator–denominator pair by c times the latest pair plus the preceding pair. If the convergent is terminal, its error is zero.

Suppose k has a label (d, j). If $j/d \neq p/v$, integrality of the cross-product gives

$$\left|\frac{j}{d} - \frac{p}{v}\right| \geq \frac{1}{dv} \geq \frac{1}{(B-1)v} > \frac{1}{vB} + \frac{1}{2Q}.$$

For the strict inequality, note that $2Q > vB(B-1)$ because $v < B$ and $Q = B^{16}$. This contradicts (6) and $|k/Q - j/d| \leq 1/(2Q)$. Thus $j/d = p/v$. It remains only to check $0 \leq p < v < B$ and the exact rule (5); these checks accept precisely the valid labels. They also prove uniqueness. When $k = 0$, the terminal convergent is $0/1$, so the case $d = 1$ is included. The Euclidean algorithm and all these checks use integers with $O(n)$ bits and take polynomial bit time. □

We now apply Lemma 3.2 to the state

$$Q^{-1/2} \sum_{x=0}^{Q-1} |x\rangle |a^x \bmod m\rangle.$$

Reversible modular powering prepares this state in polynomial time; its temporary work is uncomputed, leaving no additional information about x. This is the order-sampling setup of the quantum factoring algorithm [17], with the transform of Lemma 3.2 in place of the Fourier transform.

For any reduced label (d, j) and $0 \leq t < d$, define the arithmetic quantity

$$P(d, j, t) = \left| \frac{1}{Q} \sum_{\substack{0 \leq x < Q \\ x \equiv t \pmod d}} g\big(k(d, j)x/Q\big) \right|^2. \tag{7}$$

When $d = r = \mathrm{ord}_m(a)$, it has the following direct probabilistic meaning:

$$\Pr\big(\text{phase-register success and } k = k(r, j)\big) = \sum_{t=0}^{r-1} P(r, j, t). \tag{8}$$

To see this, the unnormalized second-register vector for a fixed output k is

$$\frac{1}{Q} \sum_{x=0}^{Q-1} g(kx/Q) |a^x \bmod m\rangle.$$

The labels $a^x \bmod m$ coincide exactly on residue classes modulo r; the distinct classes give orthogonal basis states. Taking its squared norm proves (8), including the actual probability of the phase-register test.

The quantity $P(d, j, t)$ is defined arithmetically for every candidate label. Its physical-probability interpretation in (8) is restricted to $d = r$. The correction below evaluates this arithmetic at candidate labels; its probability calculation will use the identity at the true order.

3.3 A dyadic majorant with an exact sum

We will supplement the quantum experiment with classical random choices so that, at the true order $d = r$, every reduced numerator contributes exactly $\lambda(d)$. The correction needs a dyadic majorant U for each progression quantity P in (7), together with an efficiently computable total S. Computing each value separately is insufficient because there may be exponentially many residues.

The next lemma obtains the majorant by approximating these values with a bounded collection of rational polynomial pieces and rounding their coefficients upward. Mosca and Zalka use power sums to approximate a success probability to arbitrary precision [11, Sections 4.1–4.2]. Here the polynomial pieces have bounded degree, so their dyadic envelope can be summed exactly.

Lemma 3.4. *Let n, B, Q be as in (1) and (2), and let P be defined by (7). For every $1 \le d < B$ and $0 \le j < d$ with $\gcd(j, d) = 1$, there are nonnegative dyadic numbers $U(d, j, u)$, $0 \le u < d$, such that*

$$0 \le U(d, j, jt \bmod d) - P(d, j, t) \le \frac{128}{Q} \qquad (0 \le t < d), \tag{9}$$

$$\frac{1}{128d} \le S(d, j) := \frac{1}{2} \sum_{u=0}^{d-1} U(d, j, u) \le 2. \tag{10}$$

Given their integer arguments, $P(d, j, t)$, $U(d, j, u)$, and $S(d, j)$ are exactly computable in time polynomial in n, and are dyadic numbers of polynomial bit length. In particular, computing $S(d, j)$ does not require enumerating its d summands.

Proof. Exact progression sums. Fix d, j, and recall the bin parameters $k = k(d, j)$ and $\eta = k - jQ/d$. For $x \equiv t \pmod d$, put $u = jt \bmod d$, represented in $\{0, \dots, d-1\}$. The fractional part of jx/d is then fixed at u/d, so periodicity gives the exact identity

$$g(kx/Q) = g(u/d + \eta x/Q).$$

As x traverses the progression, this last argument moves a distance at most $1/2$. Thus it meets only a bounded number of the quarter-integer breakpoints of g. Split the progression at those breakpoints by rational comparisons and integer division. On each resulting part, the real and imaginary coordinates are affine in the progression index, so their sums are obtained from the number of terms and the sum of consecutive integers. This computes $P(d, j, t)$ with a bounded number of arithmetic operations on $O(n)$-bit integers and rationals. Its dyadic nature is also immediate from the original sum: each $g(kx/Q)$ has real and imaginary denominators dividing Q, so the squared modulus after division by Q has denominator dividing Q^4.

Integral estimates. Although each $P(d, j, t)$ is efficiently computable, direct summation over t requires d evaluations, which need not be polynomial in n. To construct a majorant with an efficiently computable total, first define, for real $0 \le u < d$,

$$I(u) = \frac{1}{d} \int_0^1 g(u/d + \eta v)\, dv, \qquad J(u) = |I(u)|^2. \tag{11}$$

We first record two uniform estimates:

$$|P(d, j, t) - J(jt \bmod d)| \le \frac{32}{Q}, \qquad \frac{1}{8d} \le |I(u)| \le \frac{1}{d}. \tag{12}$$

For the first estimate, let $h = d/Q < 1$ and $f(v) = g(u/d + \eta v)$. The function g has magnitude at most one and Lipschitz constant $4\sqrt{2}$, so f has Lipschitz constant less than four. The progression points x/Q are the left endpoints of intervals of length h. Together these intervals start within h of zero and end between 1 and $1 + h$. Their total length is at most $1 + h$. The error of the left rectangle rule is at most $2h(1 + h)$, and the two endpoint discrepancies contribute at most

$2h$. Dividing these rectangle estimates by d matches the weight $1/Q$ in the progression sum. Consequently, writing

$$A_t = \frac{1}{Q} \sum_{\substack{0 \le x < Q \\ x \equiv t \pmod{d}}} g(kx/Q),$$

we have $|A_t - I(u)| \le 6/Q \le 8/Q$. Since $|A_t| \le 1/d + 1/Q$ and $|I(u)| \le 1/d$, this implies the first inequality in (12).

The lower bound will keep the majorant's total away from zero. Replace g in (11) by $e^{-2\pi i z}$. The resulting integral has magnitude

$$\frac{1}{d}\left|\frac{\sin(\pi\eta)}{\pi\eta}\right| \ge \frac{2}{\pi d},$$

where the quotient is interpreted as one at $\eta = 0$. The inequality follows from concavity of sine on $[0, \pi/2]$. Linear interpolation on intervals of length $1/4$ changes the exponential by at most

$$\frac{1}{8}\left(\frac{1}{4}\right)^2 \sup_z \left|\frac{d^2}{dz^2} e^{-2\pi i z}\right| = \frac{\pi^2}{32}.$$

This interpolation bound follows by Taylor expansion at the interpolated point and linear averaging of the two endpoint remainders; it applies equally to complex-valued functions. Therefore

$$|I(u)| \ge \frac{1}{d}\left(\frac{2}{\pi} - \frac{\pi^2}{32}\right) > \frac{1}{8d}.$$

For completeness, $\pi < 2\sqrt{3}$, obtained from $\tan(\pi/6) > \pi/6$, bounds the parenthesis below by $1/\sqrt{3} - 3/8 > 1/8$. The upper bound follows from $|g| \le 1$.

Polynomial pieces and dyadic rounding. The integral approximation has the additional advantage that its squared magnitude is piecewise polynomial. Let G be the continuous primitive of g satisfying $G(0) = 0$. Its real and imaginary parts are quadratic polynomials with rational coefficients on each quarter interval. When $\eta \ne 0$,

$$I(u) = \frac{G(u/d + \eta) - G(u/d)}{d\eta}. \tag{13}$$

The two arguments range in $[-1/2, 3/2]$, so only a bounded number of pieces occur. Their boundaries in u arise when either argument is a quarter-integer. On each piece, I is a complex polynomial in u of degree at most two; hence J is a rational polynomial of degree at most four. If $\eta = 0$, the identity $I(u) = g(u/d)/d$ gives the same conclusion. All boundaries and coefficients have $O(n)$ bits and can be computed with a bounded number of rational operations. In particular, $\eta = (kd - jQ)/d$, so every nonzero η has $|\eta| \ge 1/d$; the division in (13) has no hidden precision cost.

The polynomial J provides the simple pieces to sum, and upward coefficient rounding will make their values at integer residues dyadic. On each piece write $J(u) = \sum_{q=0}^{4} a_q u^q$, allowing zero coefficients, and set $D = QB^8$. Replace a_q by $\widetilde{a}_q = \lceil Da_q \rceil / D$. Denote the resulting piecewise polynomial by $\widetilde{J}$, assigning boundary points consistently to one adjacent piece, and define

$$U(d, j, u) = \widetilde{J}(u) + \frac{64}{Q} \qquad (u = 0, \ldots, d-1). \tag{14}$$

Because $u \geq 0$, rounding each coefficient upward increases the value even when that coefficient is negative. Moreover,

$$0 \leq \widetilde{J}(u) - J(u) \leq \frac{1 + u + u^2 + u^3 + u^4}{QB^8} \leq \frac{1}{Q}.$$

Together with (12), this yields $32/Q \leq U(d, j, jt \bmod d) - P(d, j, t) \leq 97/Q$, proving (9). It also gives

$$\frac{1}{128d} \leq \frac{1}{2} \sum_{u=0}^{d-1} U(d, j, u) \leq \frac{1}{2d} + \frac{65d}{2Q} \leq 2,$$

where the last inequality uses $d < B$ and $Q = B^{16}$.

Exact summation. Floors and ceilings give the integer points of each rational piece interval in $0, \ldots, d-1$, with each boundary integer assigned only once. The total of each polynomial is therefore a linear combination of integer power sums through degree four. These sums are computed, for example, recursively from

$$M^{q+1} = \sum_{\ell=0}^{q} \binom{q+1}{\ell} \sum_{u=0}^{M-1} u^{\ell} \qquad (0 \leq q \leq 4).$$

Here the zeroth power sum is the count M, with $u^0 = 1$ also at zero. These sums have $O(n)$ bits for $M \leq B$, and there are only a bounded number of pieces. This computes $S(d, j)$ in polynomial time without enumerating the residues. Since D is a power of two, both U and S are dyadic with $O(n)$-bit representations. All operations used to compute them are rational arithmetic, integer division, and rounding. □

3.4 Three modes with a prescribed mass

The first mode uses the quantum experiment, and the second adds the discrepancy between P and U by uniform random choices. At the true order, their combined mass is a dyadic multiple of $S(d, j)$. The third mode supplies the remainder needed to reach $\lambda(d)$ exactly.

Completion of the proof of Proposition 3.1. Scaling and remainder. For each reduced label, the natural scaling coefficient $\lambda(d)/S(d, j)$ need not be dyadic. We round it downward and reserve the remaining mass for the third mode. Define

$$c(d, j) = 2^{-10n} \left\lfloor 2^{10n} \frac{\lambda(d)}{S(d, j)} \right\rfloor, \qquad \rho(d, j) = \lambda(d) - c(d, j) S(d, j). \tag{15}$$

By (10) and $\lambda(d) \leq 1/(1024d)$,

$$0 \leq c(d, j) \leq \frac{1}{8}, \qquad 0 \leq \rho(d, j) \leq 2 \cdot 2^{-10n}. \tag{16}$$

Both numbers are exactly computable dyadics of polynomial bit length.

The three modes. Choose one of the following three modes with probabilities $1/2, 1/4, 1/4$, respectively. In every mode, reject unless

$$1 \leq d < B, \qquad 0 \leq j < d, \qquad \gcd(j, d) = 1, \qquad a^d \equiv 1 \pmod{m}.$$

The label conditions are checked before evaluating any formula involving the label. An invalid label or a failed retention test produces no candidate.

1. Run the quantum experiment. Require its phase-register success, recover the reduced label (d, j) from the observed k, and retain the candidate d with probability $c(d, j)$.

2. Choose independent uniform n-bit labels d, j, t. Require also $t < d$, and retain d with probability
$$2B^3\big(U(d, j, jt \bmod d) - P(d, j, t)\big)c(d, j). \tag{17}$$

3. Choose independent uniform n-bit labels d, j, and retain d with probability
$$4B^2\rho(d, j). \tag{18}$$

Validity of the retentions. All retention probabilities are dyadic and belong to $[0, 1]$. Nonnegativity follows from (9) and (16); the last two probabilities are at most
$$2B^3\frac{128}{Q} \le 256B^{-13} \le 1, \qquad 4B^2 \cdot 2 \cdot 2^{-10n} = 8B^{-8} \le 1.$$
Each retention test can therefore be implemented exactly by comparing fresh fair bits with an integer numerator. The number of bits and all arithmetic and quantum costs are polynomial in n.

True-order mass. To compute the mass of the correct order, fix $d = r$ and a reduced numerator j. By (8), Mode 1 contributes
$$\frac{c(d, j)}{2}\sum_{t=0}^{d-1} P(d, j, t).$$
Each triple in Mode 2 has probability B^{-3} conditional on choosing that mode. Thus Mode 2 contributes
$$\frac{c(d, j)}{2}\sum_{t=0}^{d-1}\big(U(d, j, jt \bmod d) - P(d, j, t)\big).$$
Multiplication by j permutes the residues modulo d, so these two contributions sum to $c(d, j)S(d, j)$. Mode 3 contributes $\rho(d, j)$, making the total exactly $\lambda(d)$. The bin labels are distinct by Lemma 3.3, and there are $\varphi(r)$ reduced numerators. Hence
$$\Pr(\text{the returned candidate is } r) = \varphi(r)\lambda(r).$$
This includes $r = 1$, whose sole reduced label is $j = 0$.

Every returned candidate satisfies $a^d \equiv 1 \pmod m$, so it is a multiple of the true order r. The quantum mass calculation used (8) only at $d = r$; at other labels, $P(d, j, t)$ remains an arithmetic quantity. □

4 Factoring with verifiable auxiliary data

We next organize the order trial into a factoring process. Its successful output contains enough auxiliary information to compute, retrospectively, the exact probabilities of all its useful transitions. The process generates this information together with the factorization. Throughout this section, n and $B = 2^n$ retain their values for the original input N, even when we work on a smaller integer. We use K repetitions within each randomized transition, at most L transition slots, and W bits per transition output, where
$$K = n^5, \qquad L = n^{10}, \qquad W = n(K + 1). \tag{19}$$

For $M \geq 2$, define $\mathcal{D}(M)$ to be the following rooted occurrence tree. The root records M and its distinct prime factors in increasing order, together with their positive exponents. For each distinct odd prime p in that factorization, it has one child containing $\mathcal{D}(p-1)$; the children are ordered by p. Equal labels under different parents are separate occurrences. There is no child for $p = 2$. Every descent decreases the label, so this defines unique finite data. For example, the root of $\mathcal{D}(21)$ has children $\mathcal{D}(2)$ and $\mathcal{D}(6)$, associated with its prime factors 3 and 7. The second child has its own $\mathcal{D}(2)$ child, associated with the prime factor 3 of 6; the two occurrences of $\mathcal{D}(2)$ remain separate.

The recursive $p-1$ structure recalls Pratt's succinct primality certificates [15]. Here the recorded data consist of the factorization tree, with prime labels checked by Lemma A.1. The auxiliary factorizations determine exact orders and transition probabilities retrospectively. The order-to-gcd splitting mechanism follows the Miller–Shor reduction [10, 17]; the proof below includes its unconditional counting argument.

When the controller works with a current factor m, it uses the following encodings. A list output consists of K fields of $b_m = \lceil \log_2 m \rceil$ bits, followed by zero padding to width W; an order output consists of one n-bit integer followed by zero padding to width W. A dummy output is the all-zero W-bit string. The controller rejects malformed padding, and every good-output predicate below requires a valid encoding.

Proposition 4.1 (A process with verifiable transition probabilities)**.** *Let n, B be as in (1), and let K, L, W be as in (19). There is a uniformly polynomial-time process, using the order trial of Proposition 3.1 and the transition-output encodings, with the following properties.*

1. *It either aborts or generates $\mathcal{D}(N)$. A deterministic polynomial-time verifier accepts exactly these complete data. The process's classical controller makes at most L transitions, and completed runs are padded to exactly L by constant-output dummy transitions. Each transition output has a fixed W-bit encoding.*

2. *Given $\mathcal{D}(N)$ and the recorded classical input of a transition, one can compute a good-output predicate, one canonical good encoding, and the exact probability s that an ordinary execution of that transition has a good output. The probability s is dyadic, has polynomial bit length, and obeys*
$$s \geq 1 - 2^{-2n}.$$
For a dummy transition, its constant output is good and canonical, and $s = 1$.

3. *Every evolution whose transition outputs are all good completes the tasks and generates $\mathcal{D}(N)$ without aborting or reaching a resource cap. This assertion also holds when good outputs are supplied directly. The work is polynomially bounded on arbitrary output sequences as well, with malformed outputs causing an abort.*

Remark. The W-bit encoding contains only the classical output consumed by the controller. The completed tree supplies arithmetic information for later tests. Internal order-trial registers, computational history, and other quantum work form separate workspace; they may be retained for the inverse circuit. Thus W does not bound the whole workspace.

Proof. We first recover transition probabilities from the completed tree, then construct the controller and prove that good outputs complete its tasks. Size bounds and verification finish the argument.

Recorded transitions. Each transition uses fresh work registers and fresh randomness. Its classical output is copied to a history register before the controller uses it; earlier quantum work is retained as inactive workspace. Consequently, conditional on any recorded classical prefix, the next transition has exactly the distribution specified by its recorded classical inputs, even if earlier work registers remain entangled.

Arithmetic from the completed tree. Suppose that a subfactor m is being split while factoring a node M. From $\mathcal{D}(N)$ we can factor m by division from the prime list for M. Moreover, $\mathcal{D}(N)$ supplies the factorization of $p-1$ for every odd prime $p \mid m$. Thus it supplies the factorization of

$$\varphi(m) = \prod_{p^e \| m} (p-1)p^{e-1}. \tag{20}$$

Here the formula follows by counting units modulo a prime power and using the Chinese remainder bijection; the latter follows from Bézout's identity. The contribution from $p = 2$ needs no child.

Given this factorization and a unit a modulo m, its order is computable exactly. Start with exponent $v = \varphi(m)$. For each prime q in its known factorization, repeatedly replace v by v/q while $q \mid v$ and $a^{v/q} \equiv 1 \pmod{m}$. The order divides $\varphi(m)$ by the coset proof of Lagrange's theorem. Throughout the procedure it divides v, and at termination no excess prime factor can remain. Hence the final v is the order, and the remaining prime exponents give its factorization. There are polynomially many modular powerings. The same procedure applies to each prime-power component of m. These are retrospective computations: they are used to evaluate recorded transitions once the data have been generated.

We shall use the elementary bound

$$\frac{\varphi(d)}{d} \geq \frac{1}{n+1} \qquad (1 \leq d < B). \tag{21}$$

Indeed, if $p_1 < \cdots < p_k$ are the distinct prime divisors of d, then $k \leq n$ and $p_i \geq i+1$, so the product formula gives $\varphi(d)/d \geq \prod_{i=1}^{k} i/(i+1) = 1/(k+1)$. The empty product covers $d = 1$.

The controller. To factor a node, apply the deterministic primality test in Lemma A.1 to each current factor. A prime is recorded; an even composite can be split by 2; and a composite perfect power can be split by finding an integral root. Root searches for exponents at most n take polynomial time. Consequently the remaining case is an odd m having at least two distinct prime divisors. We use the following two kinds of transition for this case.

List transitions. A list transition generates K independent uniform b_m-bit integers. Entries outside $[0, m)$ and nonunits are ignored. A list is good if some usable entry a has unequal values of

$$\nu_2\big(\mathrm{ord}_{p^e}(a)\big), \qquad p^e \parallel m,$$

where ν_2 denotes the exponent of 2 in a positive integer. To compute this probability, we count the order levels in each prime-power component, combine the counts by the Chinese remainder bijection, and then transfer the favorable-residue count to the uniform bit labels.

The nonzero elements of a prime field form a cyclic group. For completeness, let T be the least common multiple of their orders. Lagrange's theorem gives $T \mid p-1$, while the root bound for $X^T - 1$ gives $p - 1 \leq T$. The exponent of a finite abelian group is attained: for each prime power dividing that exponent, take a suitable power of an element whose order contains that prime power, then multiply these elements of pairwise coprime orders. Thus there is an element of order $T = p - 1$.

Write $p - 1 = h2^\ell$, with h odd. In this cyclic group the counts at levels $0, 1, \ldots, \ell$ are $h, h, 2h, \ldots, 2^{\ell-1}h$. Reduction of units modulo p^e to units modulo p has an odd kernel of size p^{e-1}.

The order of an element and the order of its reduction therefore have the same 2-adic level, and each residue has p^{e-1} lifts. The corresponding counts modulo p^e are

$$p^{e-1}h, \quad p^{e-1}h, \quad 2p^{e-1}h, \quad \ldots, \quad 2^{\ell-1}p^{e-1}h. \tag{22}$$

No level has more than half the units.

Let $c_i(t)$ be these counts for the prime-power components of m, extended by zero outside their ranges. For a uniform unit modulo m, the Chinese remainder bijection makes the prime-power components uniform and independent, and hence also their levels. Since there are at least two components, the probability that all levels agree is at most $1/2$: for the first two level laws q_1, q_2, $\sum_t q_1(t)q_2(t) \le \max_t q_1(t) \le 1/2$. The number of favorable residues is therefore exactly

$$F = \varphi(m) - \sum_t \prod_i c_i(t) \ge \frac{\varphi(m)}{2}.$$

There are at most n components and $n+1$ levels, so this formula is polynomial-time computable from $\mathcal{D}(N)$. A uniform b_m-bit label is favorable with probability $F/2^{b_m}$. Out-of-range labels and nonunits contribute zero to this probability. Independence of the K labels gives the list's exact good probability below; its lower bound uses (21) and $2^{b_m} \le 2m$:

$$s = 1 - \left(1 - \frac{F}{2^{b_m}}\right)^K, \qquad \frac{F}{2^{b_m}} \ge \frac{1}{4(n+1)}. \tag{23}$$

A canonical good list has first entry equal to the Chinese remainder residue that is 1 on the component corresponding to the smallest prime and -1 on all others, followed by zeros. Its first entry has levels 0 and 1.

Order transitions. For every usable entry a, an order transition runs K independent order trials. It outputs the smallest returned candidate, or 0 if there was none. If $r = \mathrm{ord}_m(a)$, goodness means that this output is r, which is also the canonical good output. By Proposition 3.1, every nonempty candidate is a positive multiple of r, and the probability of the candidate r in one trial is $\varphi(r)\lambda(r)$. Recall $\lambda(r) = 2^{-\lceil \log_2 r \rceil - 10}$ from Proposition 3.1. The minimum is r precisely when at least one trial returns r. Independence therefore gives

$$s = 1 - \big(1 - \varphi(r)\lambda(r)\big)^K, \qquad \varphi(r)\lambda(r) \ge \frac{1}{2048(n+1)}. \tag{24}$$

The last bound follows from (21) and $2^{\lceil \log_2 r \rceil} \le 2r$, also when $r = 1$.

Both (23) and (24) are exactly computable dyadics. Their denominator exponents are at most nK and $(n+10)K$, respectively. For either transition, the lower bound on the one-trial success parameter x gives

$$Kx \ge \frac{n^5}{2048(n+1)} \ge \frac{n^4}{4096} \ge 2n \qquad (n \ge 128).$$

Thus $(1-x)^K \le e^{-2n} < 2^{-2n}$, proving the required uniform bound on s. The good predicates, probabilities, and canonical outputs are all computable from $\mathcal{D}(N)$ and the classical inputs m or (m, a).

Splitting from good outputs. For every positive even order output r', try $\gcd(a^{r'/2} - 1, m)$ and accept it only if it is a proper nontrivial divisor. If no entry produces a split, abort. If the list and its order transitions are good, choose an entry with unequal local levels and let r be its true order. By the Chinese remainder bijection,

$$r = \operatorname*{lcm}_{p^e \| m} \mathrm{ord}_{p^e}(a).$$

The largest local 2-adic level is positive. The local orders at smaller levels divide $r/2$, whereas at a largest-level component $a^{r/2}$ has order two. The only square roots of 1 modulo an odd prime power are 1 and -1: in $(x-1)(x+1) \equiv 0 \pmod{p^e}$ the entire odd prime power divides one factor. Hence $a^{r/2}$ equals 1 on the smaller-level components and -1 on the largest-level components. The gcd is a proper nontrivial divisor. This proves that all-good outputs prevent an abort at every split, regardless of how those outputs were obtained.

After completely factoring a node, the controller creates its specified children and factors them in a fixed order. Once all these tasks have finished, it fills unused transition slots with dummy transitions.

Size bounds. We bound the logarithmic label mass at each depth and the number of depths. Together they bound both the auxiliary tree and its split tasks. The binary splitting tree for one node M has at most $\log_2 M$ leaves: all leaf labels are at least two and their product is M. A partial run has at most $\log_2 M$ split attempts as well, including a possible final failed attempt. At each auxiliary-tree descent,

$$\sum_{\substack{p \mid M \\ p>2}} \log_2(p-1) \le \sum_{p \mid M} \log_2 p \le \log_2 M.$$

Thus the total logarithmic size at any depth is at most $\log_2 N < n$. In two successive descents $M \to p-1 \to q-1$, the intermediate label is even and $q > 2$ divides it, so $q-1 < (p-1)/2 < M/2$. There are therefore fewer than $2n+2$ levels. Since every node label is at least two, each contributes at least one to the logarithmic size; the tree has at most $(2n+2)n$ node occurrences. Summing the split-attempt bound over the same levels gives at most $(2n+2)n$ split tasks. Each uses at most $K+1$ transitions, whence

$$\#\{\text{list and order transitions}\} \le (2n+2)n(K+1) \le 6n^7 < n^{10} = L.$$

All deterministic work and all K trials within an order transition also have polynomial bounds.

These bounds hold even for bad output sequences. Only genuine gcd splits are accepted, prime leaves are verified, and children are created only from the resulting actual prime factors. An abort only shortens the work. Fixed polynomial caps on work and storage can therefore be imposed without excluding any all-good evolution. Each node's prime list has at most n entries of polynomial bit length, so the occurrence bound also bounds the storage for the complete data. A list has at most nK unpadded bits, and an order value, including zero, has n bits. Thus the specified encodings fit within W bits and give each canonical output a unique string. The controller depends only on these classical outputs, and internal quantum work can be retained without being treated as part of an output.

Verification. The complete data are polynomially verifiable. The verifier first requires the root label to be N. Before arithmetic, it rejects node labels outside $[2, B)$, proposed prime labels outside $[2, M]$ at a node M, exponents outside $[1, n]$, and lists or trees exceeding the preceding polynomial size caps. At a node M, require strictly increasing prime labels, check primality using Lemma A.1, and check the factorization's product. Then require exactly the ordered children $p-1$ for the distinct odd primes in that list and apply the same checks recursively. Reject missing or extra children and malformed or oversized records; a partial product may be rejected as soon as it exceeds M. These guards make the verification polynomial even on arbitrary invalid records, and no true data violate them. Unique prime factorization implies that an accepted tree is exactly $\mathcal{D}(N)$; only the operational split history may vary. The all-good splitting argument ensures that every all-good evolution supplies this complete tree. This proves the proposition. □

5 From verified runs to certain output

The process in Proposition 4.1 has two useful properties: every transition has a high probability of a good output, and a completed run supplies the data needed to calculate that probability exactly. We now use these data only after they have been generated. The aim is a preparation whose success probability is known before it runs. There are two normalization stages. First, every transition will have the same retained mass z, so L slots have known success probability z^L. A second correction, applied to the complete preparation, will make that probability exactly $1/4$.

5.1 Equalizing one transition

We first equalize a transition's passing probability by rounding its retention downward and filling the remaining mass with a rare uniform guess. The distinguished output used to retain a guess need not be known when the guess is made.

Lemma 5.1 (Dyadic completion)**.** *Let W be a nonnegative integer. Consider an ordinary transition with output in $\{0,1\}^W$, a designated set of good outputs of dyadic probability $0 < s \le 1$, and a specified good output y_0. Let δ, z be dyadic rationals with $0 < \delta < 1$ and $0 < z \le A := (1-\delta)s$, and suppose $\delta = 2^{-t}$ for a positive integer t.*

With probability $1-\delta$ run the ordinary transition, and otherwise guess a uniform W-bit output. For $E = W + t + 2$, define

$$C = 2^{-E}\left\lfloor 2^E \frac{z}{A} \right\rfloor, \qquad R = z - CA. \tag{25}$$

In the ordinary branch require a good output and retain with probability C. In the guessing branch require the particular output y_0 and retain with probability $R2^W/\delta$.

These are valid dyadic retention probabilities, and the total probability of passing is exactly z. The arithmetic computations have polynomial bit complexity whenever the input dyadics have polynomial-size representations and the values of the integers W, t are polynomially bounded.

Proof. Since $z/A \le 1$, we have $0 \le C \le 1$. Downward rounding gives

$$0 \le R < A2^{-E} \le 2^{-E}, \qquad 0 \le R2^W/\delta \le \tfrac{1}{4}.$$

The two passing masses are AC and

$$\delta\, 2^{-W}\,(R2^W/\delta) = R,$$

whose sum is z. Although z/A need not be dyadic, computing its scaled floor is an exact integer division. Every probability that is actually sampled is dyadic. The products, subtraction, and denominator exponents have polynomial size. □

5.2 Filters computed after completion

Use the process of Proposition 4.1. Recall its repetition count, slot bound, and output width from (19):

$$K = n^5, \qquad L = n^{10}, \qquad W = n(K+1).$$

Every completed execution has exactly L slots. A run that finishes after t real transitions continues through $L - t$ constant-output dummy slots, each with good probability 1 and one fixed canonical

string. For example, $N = 2$ can have only dummy slots. A prime root greater than 2 first generates its required $p - 1$ descendants. Set

$$\delta = 2^{-2n}, \qquad z = 1 - 2^{-n}, \qquad E = W + 2n + 2. \tag{26}$$

At each slot run the ordinary transition with probability $1 - \delta$, and otherwise guess its W-bit output uniformly. Record the branch and the output. Subsequent deterministic computation uses that output in the usual way. Malformed encodings may abort, and a proposed divisor is always checked before a split is made. In an ordinary order transition the internal trial registers are retained separately; the guessed string encodes only the transition's classical output.

The algorithm first attempts to complete all the factorizations defining $\mathcal{D}(N)$, recording the outputs for testing after completion. If it aborts or its generated data fail the complete verification in Proposition 4.1, it sets its success flag to zero. Otherwise, for every recorded slot it reconstructs the transition inputs and computes, from the verified data, the good-output predicate, its probability s, and the canonical output y_0. It then applies the two filters of Lemma 5.1, using fresh independent dyadic coins. Success requires every filter to pass.

The filters are well-defined: writing $\varepsilon = 2^{-n}$, the transition bound gives

$$(1 - \delta)s \geq (1 - \varepsilon^2)^2 \geq 1 - \varepsilon = z. \tag{27}$$

Indeed the difference on the right is $\varepsilon - 2\varepsilon^2 + \varepsilon^4 > 0$ for the present range. The same bound holds in dummy slots. Exact dyadic sampling is provided by Lemma 2.1.

Proposition 5.2 (Known probability from deferred filters)**.** *For the process of Proposition 4.1, the mixed transitions with parameters* (26) *and the deferred filters of Lemma 5.1, implemented in Section 5.2, give a preparation with success probability exactly*

$$p_* = (1 - 2^{-n})^{n^{10}}. \tag{28}$$

Every successful branch outputs the correct prime factorization of N. The preparation has polynomial worst-case time and workspace, and $p_* > 1/2$ is an initially known dyadic with polynomial bit length.

Proof. An event defined from the true data. Fix the true mathematical data $\mathcal{D}(N)$ for the purpose of analysis only. Allocate fresh filter coins in advance, with a common polynomial bound on their lengths. Consider the event obtained by testing each slot immediately against these true data, using the same coins that the actual algorithm will use after completion. Declare this analytical event false on a run that aborts before all L slots are recorded. These hypothetical tests only mark an event on the same recorded runs; their outcomes do not alter the controller's evolution.

Probability along passing prefixes. Condition on a recorded controller prefix of fewer than L slots together with the coins used by its previous filters, all of which have passed. Later filter coins remain unexposed. Every output in this prefix is good. Continuing with canonical good outputs gives an all-good evolution, so Proposition 4.1 ensures that the controller reaches the next real or dummy slot. This extension argument applies to every passing prefix, before considering any later filter.

Its transition inputs are fixed. Its new ordinary transition uses fresh registers, so its good-output probability is s, even if unobserved old quantum work is entangled; this is the consequence of Lemma 2.2 explained in Section 2. The branch choice, uniform guess, and next filter coins are also fresh. Lemma 5.1 therefore gives conditional passing probability exactly z for this prefix. This

statement holds separately for every passing prefix and for every dummy slot. Induction over the L slots gives probability z^L for the analytical event.

Equality of the two events. Figure 1 summarizes the two implications. Every test uses the same recorded outputs and coins in both descriptions. Every analytical passing transition has a good output. Proposition 4.1 then guarantees completion of the full true data, without abort or truncation by a cap. Such a run passes the actual data verification, and its deferred thresholds and canonical outputs agree with those in the analytical event. Conversely, if an implemented run passes, its verified complete prime factorizations are precisely the true data, by unique factorization. Every deferred test is consequently the corresponding true-data test on the same output and the same coins. The two events coincide on each recorded run. In particular, the conditional calculation above never conditions on eventual completion, and the implementation never uses data it has not generated.

Coherent implementation. The retained records also justify the probability calculation in a unitary implementation. Every result used for later classical control has an immutable orthogonal history copy. Distinct histories cannot interfere during the preparation; internal interference within a quantum trial is preserved because its pre-interference labels are not copied. All filtering computations are classical controls and fresh-coin comparisons. Thus Lemma 2.2 preserves the event probability just established.

Resource bounds and the size of p_*. The underlying process has polynomial caps on every branch. The s values have $O(nK)$ denominator bits, and the rounded filters have polynomial bit length. There are L of them, so evaluating all deferred tests and retaining their histories still has polynomial cost. The dyadic p_* has at most nL denominator bits and can be computed exactly by integer powering. Finally, Bernoulli's inequality yields

$$p_* \geq 1 - n^{10}2^{-n} > \tfrac{1}{2}.$$

At $n = 128$ the subtracted term is 2^{-58}, and $x^{10}2^{-x}$ decreases for real $x \geq 128$, since its logarithmic derivative is $10/x - \log_e 2 < 0$. □

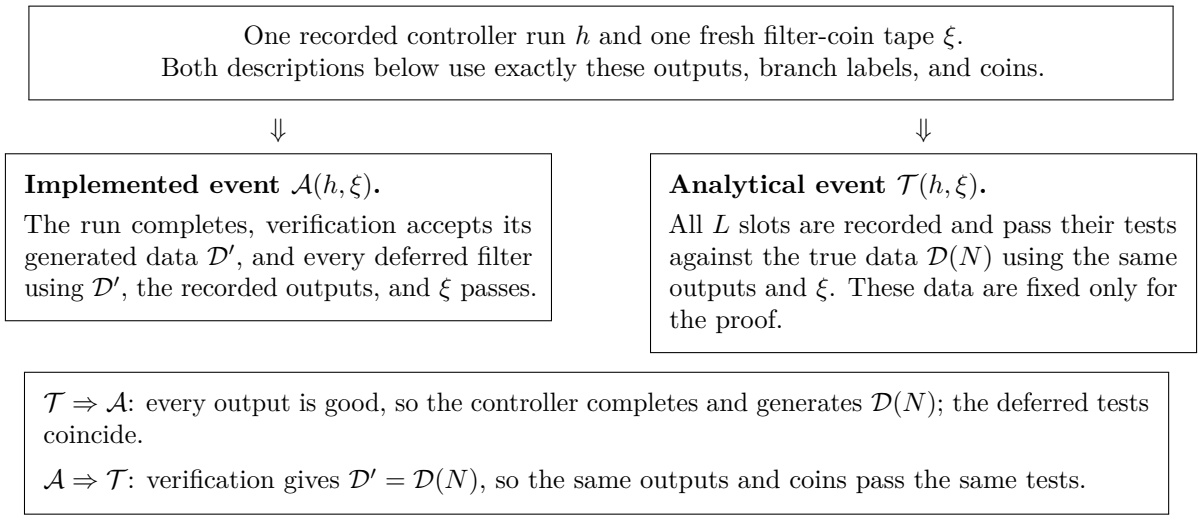


Figure 1: Two descriptions of one acceptance event. The probability calculation conditions on passing prefixes. The implementation evaluates its filters after generating and verifying the required data; the analytical tests leave the controller evolution unchanged.

5.3 A known quarter of success and one amplification step

The target probability $1/4$ is the one-step exact-amplification case of the standard reflection formula [4, Section 2], used explicitly for exact transforms by Mosca and Zalka [11, Section 2.1]. The remaining task is to reach that probability with the fixed gates. As in Lemma 5.1, downward rounding leaves a dyadic remainder that a rare valid guess can supply. This correction is independent of the factoring process.

Encode the root factorization by n entries of n bits each: positive primes in nondecreasing order with multiplicity, followed by zeros. Put $J = n^2$. Because $N < 2^n$, there are fewer than n prime factors with multiplicity, and each prime is below 2^n. Exactly one J-bit string is valid. Lemma A.1, the ordering check, and the product check recognize it in polynomial time. The product of an arbitrary guessed list has at most n^2 bits, so this verification is also polynomial on invalid guesses.

Lemma 5.3 (Quarter-success preparation)**.** *For $N \geq 2$ and n as in (1), there is a polynomial-size uniform unitary preparation V_N whose flag is 1 with probability exactly $1/4$ and whose flagged output is always the n^2-bit prime-list encoding of the prime factorization of N.*

Proof. With probability $1/2$, run Proposition 5.2 and retain its success with the further probability

$$c_* = 2^{-(J+2)} \left\lfloor \frac{2^{J+2}}{2p_*} \right\rfloor .$$

With probability $1/2$, guess a uniform J-bit string, require it to be the valid root factor list, and retain with probability

$$2^{J+1}e, \qquad e = \frac{1}{4} - \frac{p_* c_*}{2}.$$

Since $1/2 < p_* \leq 1$, we have $0 \leq c_* \leq 1$. Rounding gives

$$0 \leq e < p_* 2^{-(J+3)}, \qquad 0 \leq 2^{J+1}e \leq \frac{1}{4}.$$

Thus both retentions are valid polynomial-bit dyadics. The first branch contributes $p_* c_*/2$, and the unique valid guess contributes

$$\frac{1}{2}\, 2^{-J}\, 2^{J+1} e = e.$$

Their sum is exactly $1/4$. Place the same fixed-encoding root factor list in the output register on either successful branch. All work and flags are retained when applying Lemma 2.2, which gives the unitary V_N. □

Proof of Theorem 1.1. Let $\psi = V_N|0\rangle$, let Π project onto flag 1, and put $g_* = \Pi\psi$. By Lemma 5.3, $\|g_*\|^2 = 1/4$. First apply the sign flip $I - 2\Pi$, and then the reflection $2|\psi\rangle\langle\psi| - I$. Since

$$\langle \psi, \psi - 2g_* \rangle = 1 - 2\|g_*\|^2 = \frac{1}{2},$$

the resulting state is

$$(2|\psi\rangle\langle\psi| - I)(\psi - 2g_*) = 2g_*. \tag{29}$$

It has norm one and lies entirely in the successful subspace. Every basis outcome in that subspace has the correct factor list, so measuring the output gives the factorization with certainty.

Both reflections are exact over our finite gates. The success sign is a bit test. For fixed classical input N, the other reflection is

$$V_N(2|0\rangle\langle 0| - I)V_N^{-1},$$

where the middle test concerns all initially zero workspace, and the read-only input is held fixed. Its sign convention may differ by an irrelevant global minus sign from a sign flip on zero. Multi-control signs use the reversible gates and S^2 described in Section 2. A circuit for inputs of a fixed bit length performs these same operations controlled by the unchanged input bits, so its description remains polynomial-time uniform. Inverting V_N includes all of its work and history registers; none needs to be erased separately.

Each use of V_N or its inverse has polynomial size and workspace. The single amplification step (29), at the exactly known probability $1/4$, is the corresponding special case of amplitude amplification [4, Section 2]. It uses the exact sign reflections above and has a fixed number of calls to the preparation and its inverse. This proves the asserted worst-case bound and exact output. □

A An elementary polynomial-time primality test

The factorization procedure needs an exact test for its prime leaves and for the prime factors in its auxiliary data. We give the required test here, including its correctness proof. It is a convenient variant of the polynomial-congruence method of Agrawal, Kayal, and Saxena [1, Section 4]; the auxiliary prime and constants used below are proved directly.

Lemma A.1. *There is a uniform deterministic algorithm which, given integers $n \geq 128$ and $2 \leq m < 2^n$, decides whether m is prime using a number of bit operations polynomial in n.*

Proof. The test checks polynomial congruences after excluding perfect powers and small divisors. If a composite passed, those congruences would imply incompatible lower and upper counts of polynomial values. The auxiliary prime in the algorithm makes that comparison possible.

1. Reject m if it is a perfect power b^e with integers $b, e \geq 2$.

2. Find a prime $s \le n^8$ such that

$$\gcd(s, m) = 1, \qquad \operatorname{ord}_s(m) > n^2. \tag{30}$$

3. If $m \le 8s$, decide primality by trial division and stop. Otherwise reject if any integer from 2 through $8s$ divides m.

4. Accept if and only if all the congruences

$$(X + a)^m \equiv X^m + a \pmod{m,\, X^s - 1}, \qquad 1 \le a \le 8s, \tag{31}$$

hold.

We prove that the search in the second step succeeds, that the algorithm is correct, and that its running time is polynomial.

Existence of s. A prime dividing neither m nor any of $m^j - 1$ for $1 \le j \le n^2$ satisfies (30). We show that the primes at most n^8 cannot all divide this product. Let $R = 2\lfloor n^8/2 \rfloor$ and let P_R be the product of the primes at most R. The central binomial coefficient $C = \binom{R}{R/2}$ satisfies $C \ge 2^R/(R+1)$. For every prime ℓ, the factorial valuation formula gives

$$v_\ell(C) = \sum_{j \ge 1} \left(\left\lfloor \frac{R}{\ell^j} \right\rfloor - 2 \left\lfloor \frac{R/2}{\ell^j} \right\rfloor \right) \le \lfloor \log_\ell R \rfloor.$$

Each summand is zero or one, and it vanishes when $\ell^j > R$. Consequently, if $\pi(R)$ denotes the number of primes at most R, then

$$\log_2 C \le \pi(R) \log_2 R \le (\log_2 P_R) \log_2 R.$$

Thus

$$\log_2 P_R \ge \frac{R - \log_2(R+1)}{\log_2 R} > n^6. \tag{32}$$

For the last inequality, $8 \log_2 n + 1 < n$ for $n \ge 128$; the difference is positive at 128 and increasing thereafter. Since $R \ge n^8 - 1$ and $R + 1 \le 2n^8$, the displayed quotient is larger than $(n^8 - 1 - n)/n$, which is larger than n^6. On the other hand,

$$\log_2 \left(m \prod_{j=1}^{n^2} (m^j - 1) \right) < n \left(1 + \frac{n^2(n^2+1)}{2} \right) < n^6.$$

It follows from (32) that some prime $s \le R$ divides neither m nor any $m^j - 1$ with $1 \le j \le n^2$. This is precisely (30).

Prime inputs. Primes pass the perfect-power and trial-division checks. If m is prime, the binomial theorem in characteristic m gives $(X + a)^m = X^m + a^m = X^m + a$, so they also pass (31).

Congruences for an accepted composite. Suppose, conversely, that a composite m is accepted. It is not a perfect power, and every prime p dividing it satisfies $p > 8s$. Choose one such p. In an algebraic extension of $\mathbb{F}_p$, choose a root w of $1 + X + \cdots + X^{s-1}$. Its value at 1 is $s \ne 0$ in $\mathbb{F}_p$, so $w \ne 1$. Since s is prime, w has multiplicative order exactly s.

For any product f of the tested linear factors $X + a$, reduction of (31) modulo p gives $f(X)^m \equiv f(X^m) \pmod{X^s - 1}$. Frobenius gives the same identity with exponent p. These

identities compose: substitution $X \mapsto X^b$ preserves the ideal $(X^s - 1)$, so identities for exponents b, c imply the identity for bc. Hence, for all nonnegative integers u, v,

$$f(w)^k = f(w^k), \qquad k = m^u p^v. \tag{33}$$

Let H be the subgroup of $(\mathbb{Z}/s\mathbb{Z})^\times$ generated by the residues of m and p, and put $h = |H|$. By (30),

$$n^2 < h \le s - 1.$$

The nonnegative powers $m^u p^v$ already range over H, because inverses in a finite group are nonnegative powers. We now obtain incompatible lower and upper bounds on the number of values $f(w)$. These bounds adapt the root-counting arguments in [1, Lemmas 4.7–4.8 and their footnotes], which credit the lower-bound argument to Lenstra and independently Macaj, and the upper-bound formulation to Kalai, Sahai, and Sudan.

A lower bound from distinct polynomial values. Take all products of $h - 1$ distinct factors from $X + 1, \ldots, X + 8s$. The factors are pairwise distinct over $\mathbb{F}_p$ because $p > 8s$, so different subsets give different monic polynomials. If two such polynomials f_1, f_2 had the same value at w, then (33) would give $f_1(w^k) = f_2(w^k)$ for every $k \bmod s \in H$. This would give h distinct roots of the nonzero polynomial $f_1 - f_2$, whose degree is at most $h - 1$. Thus all these values are distinct, and their number is at least

$$\binom{8s}{h-1} \ge \left(\frac{8s}{h-1}\right)^{h-1} \ge 8^{h-1}. \tag{34}$$

Here $\binom{M}{t} \ge (M/t)^t$ follows by comparing each factor $(M - i)/(t - i)$ with M/t. These values are also nonzero: if $f(w) = 0$, (33) would give h roots of the nonzero polynomial f, of degree $h - 1$.

An upper bound from an exponent collision. Put $q = \lfloor \sqrt{h} \rfloor$. The $(q+1)^2 > h$ pairs $0 \le u, v \le q$ give two distinct pairs for which the corresponding integers $k_1 = m^u p^v$ and $k_2 = m^{u'} p^{v'}$ agree modulo s. They are different as integers: since m is composite and not a perfect power, it has a prime divisor $p' \ne p$. Equality of k_1 and k_2 would force $u = u'$ by taking p'-adic valuations, and then $v = v'$. Now $w^{k_1} = w^{k_2}$, so (33) makes every value counted in (34) a root of the nonzero polynomial $T^{k_1} - T^{k_2}$. Its degree is at most

$$\max(k_1, k_2) \le m^{2q} < 2^{2nq} \le 2^{2n\sqrt{h}} < 2^{2h}.$$

But (34) is at least $8^{h-1} = 2^{3h-3} > 2^{2h}$, since $h > n^2 \ge 128^2$. This contradiction proves correctness.

Running time. Every exponent in a representation $m = b^e$ with $b \ge 2$ satisfies $e < n$, so binary search for integer roots with exponents $2, \ldots, n$ implements the first step in polynomial time. The search for s uses trial division to recognize primes up to n^8, followed by gcd and successive modular powers to test (30). Trial division, gcd, and modular powers give a polynomial bound for this search independently of the primality test being proved. The trial divisions involving m are also bounded by a polynomial in n. Finally, elements of $(\mathbb{Z}/m\mathbb{Z})[X]/(X^s - 1)$ are represented by s coefficients of at most n bits. Schoolbook cyclic multiplication and repeated squaring test each of the $8s$ congruences using polynomially many bit operations, since $s \le n^8$ and the exponent m has at most n bits. The prime factor p, the extension field, and the exponents in the root count serve only as witnesses in the correctness proof. All loops have uniform polynomial bounds, completing the proof. □

References

[1] M. Agrawal, N. Kayal, and N. Saxena, *PRIMES is in P*, Ann. of Math. (2) **160** (2004), no. 2, 781–793. doi:10.4007/annals.2004.160.781.

[2] C. H. Bennett, *Logical reversibility of computation*, IBM J. Res. Develop. **17** (1973), no. 6, 525–532. doi:10.1147/rd.176.0525.

[3] G. Brassard and P. Høyer, *An exact quantum polynomial-time algorithm for Simon's problem*, Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems, IEEE Computer Society, 1997, 12–23. arXiv:quant-ph/9704027v1.

[4] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, *Quantum amplitude amplification and estimation*, Contemp. Math. **305** (2002), 53–74. doi:10.1090/conm/305/05215. Preprint: arXiv:quant-ph/0005055v1.

[5] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, *Quantum algorithms revisited*, Proc. R. Soc. Lond. A **454** (1998), no. 1969, 339–354. doi:10.1098/rspa.1998.0164. Preprint: arXiv:quant-ph/9708016v1.

[6] L. K. Grover, *A fast quantum mechanical algorithm for database search*, Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC 1996), ACM, 1996, 212–219. arXiv:quant-ph/9605043v3.

[7] P. Høyer, *Arbitrary phases in quantum amplitude amplification*, Phys. Rev. A **62** (2000), 052304. doi:10.1103/PhysRevA.62.052304. Preprint: arXiv:quant-ph/0006031v1.

[8] M. Imran, *An exact quantum order finding algorithm and its applications*, arXiv:2205.04240v4, 2022.

[9] F. M. Lev, *An integral version of Shor's factoring algorithm*, 2001. arXiv:quant-ph/0109103v1.

[10] G. L. Miller, *Riemann's hypothesis and tests for primality*, J. Comput. System Sci. **13** (1976), no. 3, 300–317. doi:10.1016/S0022-0000(76)80043-8.

[11] M. Mosca and C. Zalka, *Exact quantum Fourier transforms and discrete logarithm algorithms*, Int. J. Quantum Inf. **2** (2004), no. 1, 91–100. Preprint: arXiv:quant-ph/0301093v1.

[12] H. Nishimura and M. Ozawa, *Perfect computational equivalence between quantum Turing machines and finitely generated uniform quantum circuit families*, Quantum Inf. Process. **8** (2009), no. 1, 13–24. doi:10.1007/s11128-008-0091-8. Preprint: arXiv:quant-ph/0511117v3.

[13] M. Ozols, M. Roetteler, and J. Roland, *Quantum rejection sampling*, Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS 2012), ACM, 2012, 290–308. doi:10.1145/2090236.2090261. Preprint: arXiv:1103.2774v4.

[14] C. Pilatte, *Unconditional correctness of recent quantum algorithms for factoring and computing discrete logarithms*, Forum Math. Pi **14** (2026), e5. doi:10.1017/fmp.2025.10023. Preprint: arXiv:2404.16450v3.

[15] V. R. Pratt, *Every prime has a succinct certificate*, SIAM J. Comput. **4** (1975), no. 3, 214–220. doi:10.1137/0204018.

[16] O. Regev, *An efficient quantum factoring algorithm*, 2023, revised 2024. arXiv:2308.06572v3.

[17] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput. **26** (1997), no. 5, 1484–1509. doi:10.1137/S0097539795293172. Preprint: arXiv:quant-ph/9508027v2.

[18] T. Toffoli, *Reversible computing*, in *Automata, Languages and Programming* (J. de Bakker and J. van Leeuwen, eds.), Lecture Notes in Computer Science, vol. 85, Springer, Berlin–Heidelberg, 1980, pp. 632–644. doi:10.1007/3-540-10003-2_104.

[19] V. Vedral, A. Barenco, and A. Ekert, *Quantum networks for elementary arithmetic operations*, Phys. Rev. A **54** (1996), no. 1, 147–153. doi:10.1103/PhysRevA.54.147. Preprint: arXiv:quant-ph/9511018v1.