An Efficient Deterministic Primality Test

Joseph M. Shunia

December 2023

Abstract

A deterministic primality test with a polynomial time complexity of $\tilde{O}(\log^3(n))$ is presented. The test posits that an integer n satisfying the conditions of the main theorem is prime. Combining elements of number theory and combinatorics, the proof operates on the basis of simultaneous modular congruences relating to binomial transforms of powers of two.

1 Introduction

Primality testing has seen remarkable advancements over the past few decades. A significant breakthrough in this field was the AKS primality test, introduced by Agrawal, Kayal, and Saxena (2002) [1]. The AKS test was the first to offer determinism and polynomial-time complexity, a monumental achievement that resolved a longstanding open question in computational number theory [2]. However, despite its theoretical importance, the AKS test has practical limitations due to its relatively high polynomial time complexity, rendering it inefficient for most applications. Agrawal, Kayal, and Saxena gave a time complexity of $\tilde{O}(\log^{12}(n))$ for the AKS test [1]. This bound was lowered significantly by Lenstra and Pomerance (2011) to $\tilde{O}(\log^6(n))$ [3]. Despite this reduction, AKS remains impractical and is mostly unused.

In the field of cryptography, the unique properties of prime numbers are widely exploited to create cryptographic primitives. It is often the case that many large primes must be generated in rapid succession [4]. To make these cryptographic operations practical, fast probabilistic primality tests such as the Baille-PSW primality test (BPSW) [5] or Miller-Rabin (MR) [6] [7] are used instead of AKS when searching for large primes. Probabilistic primality tests are by definition non-deterministic and may erroneously report a composite integer as being prime. Composite integers which pass a probabilistic primality test are relatively rare and are known as pseudoprimes (PSPs) for the respective test [8]. When generating primes for cryptographic purposes, probabilistic primality tests are often combined or repeated with different parameters in order to achieve an acceptable error-bound that makes it almost certain that no composite integer will pass. However, reducing the error-bound requires additional compute and increases running-time, creating a trade-off.

We present a new deterministic primality test that operates in polynomial time with a time complexity of $\tilde{O}(\log^3(n))$. This efficiency gain opens new avenues for practical applications, particularly in cryptography, where fast and reliable primality testing is desirable [9].

Our test is based on a famous conjecture issued by Manindra Argawal while he was an undergraduate student [1]. Let n and d be two coprime positive integers. If the following polynomial congruence holds, then n is prime or $n^2 = 1 \pmod{d}$:

$$(x-1)^n \equiv x^n - 1 \pmod{n, x^d - 1} \tag{1}$$

We alter the conditions slightly to use roots of 2 instead of roots of unity. Through the proof of our main theorem, we demonstrate that our modified test is equivalent to checking the polynomial congruence $(1+x)^n \equiv 1 + x^n \pmod{n}$, a congruence which is known to hold for only prime integers n [10].

1.1 Structure of the Paper

This paper is structured as follows: We begin by presenting the main theorem which defines our primality test. We follow up with supporting theorems, identities, and lemmas. Then, we present the proof of our main theorem. The proof of our main theorem demonstrates the test's validity for odd prime numbers and its failure for odd composite numbers. Through this, we establish the deterministic nature of our test. We then describe the algorithm used to compute our test and analyze its computational complexity. We conclude with pseudocode for our test, and a link to an open source implementation, to demonstrate how our test can be implemented.

2 Statement of Main Theorem

Theorem 1 (Main theorem)**.** Let n be an odd integer > 3 such that $2^{n-1} \equiv 1 \pmod{n}$. Denote d as the least integer greater than 2 such that $n \not\equiv 1 \pmod{d}$. If the following polynomial congruence holds, then either n is prime or n has a prime divisor $p \leq d$:

$$(1+x)^n \equiv 1 + x^n \pmod{n, x^d - 2} \tag{2}$$

3 Supporting Identities

Identity 1 (Powers of two)**.** Let $n, d \in \mathbb{Z}^+$, with $1 \leq d < n$:

$$2^n = \sum_{j=0}^{d-1} \sum_{k=0}^{n} \binom{n}{kd+j} \tag{3}$$

Proof. By the binomial theorem, $2^n = \sum_{k=0}^{n} \binom{n}{k}$. We can split this sum up into d groups of at most $\lfloor \frac{n}{d} \rfloor$ terms, like so:

$$2^n = \sum_{k=0}^{n} \binom{n}{k} = \sum_{j=0}^{d-1} \sum_{k=0}^{\lfloor \frac{n}{d} \rfloor} \binom{n}{kd+j} \tag{4}$$

For $k > \lfloor \frac{n}{d} \rfloor$, we have $dk + j > n$. In such case, $\binom{n}{dk+j} = 0$. Hence, we can say:

$$\sum_{j=0}^{d-1} \sum_{k=0}^{\lfloor \frac{n}{d} \rfloor} \binom{n}{kd+j} = \sum_{j=0}^{d-1} \sum_{k=0}^{n} \binom{n}{kd+j} \tag{5}$$

□

Identity 2 (Binomial transform of floored powers of two)**.** Let $n, d \in \mathbb{Z}^+$, with $1 \leq d < n$:

$$\sum_{k=0}^{n} \binom{n}{k} 2^{\lfloor \frac{k}{d} \rfloor} = \sum_{j=0}^{d-1} \sum_{k=0}^{n} \binom{n}{kd+j} 2^k \tag{6}$$

Proof. By Identity 1, we have:

$$\sum_{k=0}^{n} \binom{n}{k} = \sum_{j=0}^{d-1} \sum_{k=0}^{\lfloor \frac{n}{d} \rfloor} \binom{n}{kd+j} \tag{7}$$

Applying this to $\sum_{k=0}^{n}\binom{n}{k}2^{\left\lfloor\frac{k}{d}\right\rfloor}$, since the exponent in $2^{\left\lfloor\frac{k}{d}\right\rfloor}$ is partitioned by d, it splits over the groups. We may rewrite our equation as:

$$\sum_{k=0}^{n}\binom{n}{k}2^{\left\lfloor\frac{k}{d}\right\rfloor}=\sum_{j=0}^{d-1}\sum_{k=0}^{\left\lfloor\frac{n}{d}\right\rfloor}\binom{n}{kd+j}2^{k} \tag{8}$$

As in Identity 1, for $k>\left\lfloor\frac{n}{d}\right\rfloor$, we have $dk+j>n$. In such case, $\binom{n}{kd+j}=0$. Hence, we can say:

$$\sum_{j=0}^{d-1}\sum_{k=0}^{\left\lfloor\frac{n}{d}\right\rfloor}\binom{n}{kd+j}2^{k}=\sum_{j=0}^{d-1}\sum_{k=0}^{n}\binom{n}{kd+j}2^{k} \tag{9}$$

□

4 Polynomial Ring Theorems

Theorem 2. Let $n,d\in\mathbb{Z}^{+}$ with $d>2$. Let x^{k} be a polynomial in $(\mathbb{Z}/n)[x]/(x^{d}-2)$. Taking the sum of the coefficients modulo n in the polynomial expansion of x^{k}, gives $2^{\left\lfloor\frac{k}{d}\right\rfloor} \pmod{n}$.

Proof. Examining $d=2$, we can see:

$$\begin{aligned}
x^{0}&=1=2^{\lfloor 0/2\rfloor}\\
x^{1}&=1=2^{\lfloor 1/2\rfloor}\\
x^{2}&=2=2^{\lfloor 2/2\rfloor}\\
x^{3}&=2x=2=2^{\lfloor 3/2\rfloor}\\
x^{4}&=2x^{2}=4=2^{\lfloor 4/2\rfloor}\\
x^{5}&=4x=4=2^{\lfloor 5/2\rfloor}\\
x^{6}&=4x^{2}=8=2^{\lfloor 6/2\rfloor}\\
&\vdots
\end{aligned}$$

We proceed by induction. For $k=0$, we have $x^{0}=1$, and the sum of coefficients is $2^{\left\lfloor\frac{0-1}{d}\right\rfloor}=2^{0}=1$. Assume the theorem holds for some $k\geq 0$, that is, the sum of coefficients modulo n in x^{k} is $2^{\left\lfloor\frac{k}{d}\right\rfloor} \pmod{n}$. We need to show it also holds for $k+1$. Consider $x^{k+1}=x^{k}\cdot x$. By the definition of the ideal $I=x^{d}-2$, the degree of x^{k+1} gets reduced every time it reaches d. The number of times this reduction happens is $\left\lfloor\frac{k}{d}\right\rfloor$. Therefore, the sum of coefficients in x^{k+1} should be $2^{\left\lfloor\frac{k+1}{d}\right\rfloor}$, as each reduction doubles the sum of coefficients. Hence, the theorem holds for $k+1$. By the principle of mathematical induction, the theorem is proven. □

Theorem 3. Let n an integer >0. Let d be an integer >2. Let $(1+x)^{n}$ be a polynomial in $(\mathbb{Z}/n)[x]/(x^{d}-2)$. Taking the sum of the coefficients modulo n in the polynomial expansion of $(1+x)^{n}$ is equivalent to $\sum_{k=0}^{n}\binom{n}{k}2^{\left\lfloor\frac{k}{d}\right\rfloor} \pmod{n}$.

Proof. By the binomial theorem, we have:

$$(1+x)^{n}=\sum_{k=0}^{n}\binom{n}{k}x^{k} \tag{10}$$

Hence, when we evaluate this polynomial at $x = 1$, we sum the coefficients of the polynomial to get:

$$\sum_{k=0}^{n} \binom{n}{k} 2^{\left\lfloor \frac{k}{d} \right\rfloor} \quad \text{(by Theorem 2)} \tag{11}$$

□

5 Supporting Lemmas

Lemma 1. Given $a, b \in \mathbb{Z}^+$ with $b \nmid a$ and $1 < b < \left\lfloor \frac{a}{b} \right\rfloor$, then $\left\lfloor \frac{a}{b} \right\rfloor$ cannot divide a.

Proof. Let $q = \left\lfloor \frac{a}{b} \right\rfloor$. By definition, q is the greatest integer that is less than $\frac{a}{b}$. Thus, $q \cdot b < a < b \cdot (q + 1)$.

Suppose, for contradiction, that q divides a. Then there exists an integer k such that $a = k \cdot q$. Substituting $a = k \cdot q$ into the inequality $q \cdot b < a < b \cdot (q + 1)$, we get $q \cdot b < k \cdot q < b \cdot (q + 1)$. Dividing this inequality by q, we obtain $b < k < b + \frac{b}{q}$.

Since k is an integer, and $b \nmid a$ implies $k \neq b$, the next possible integer value for k is $b + 1$. Therefore, $k = b + 1$, which gives $a = k \cdot q = q \cdot (b + 1)$. However, this leads to a contradiction: $a = q \cdot (b + 1)$ implies $a \geq b \cdot (q + 1)$, contradicting the established fact that $a < b \cdot (q + 1)$. Hence, our assumption that q divides a is false. Therefore, $\left\lfloor \frac{a}{b} \right\rfloor \nmid a$. □

Lemma 2 (Upper bound on floor function and indivisibility)**.** Given $a, b \in \mathbb{Z}^+$ with $1 < b < \left\lfloor \frac{a}{b} \right\rfloor$, then $b \leq \lfloor \sqrt{a} \rfloor$.

Proof. Assume $a, b \in \mathbb{Z}^+$ and $1 < b < \left\lfloor \frac{a}{b} \right\rfloor$. By definition, $\left\lfloor \frac{a}{b} \right\rfloor$ is the greatest integer less than or equal to $\frac{a}{b}$. Hence, $\left\lfloor \frac{a}{b} \right\rfloor \leq \frac{a}{b}$. Since $b < \left\lfloor \frac{a}{b} \right\rfloor$, we have $b^2 < b \cdot \left\lfloor \frac{a}{b} \right\rfloor \leq a$. Taking square roots on both sides of the inequality $b^2 < a$ and considering that b and $\sqrt{a}$ are both positive, we get $b < \sqrt{a}$. Since b and $\sqrt{a}$ are positive integers, and $b < \sqrt{a}$, it follows that $b \leq \lfloor \sqrt{a} \rfloor$. □

Lemma 3 (Bounds on least non-divisor)**.** Let n be an integer such that $n > 3$, then there exists an integer $1 < D \leq \lfloor \log_2(n - 1) \rfloor + 2$ which does not divide $n - 1$.

Proof. If n is a composite integer > 3, clearly $n - 1$ can have at most $\lfloor \log_2(n - 1) \rfloor$ prime factors (when $n - 1$ is a power of 2). Since 2 is the least prime that may divide $n - 1$, there must exist a $D \leq \lfloor \log_2(n - 1) \rfloor + 2$ which does not divide $n - 1$. □

Lemma 4 (Multiplicative order inequality)**.** Let n be an odd composite integer greater than 3 such that $2^{n-1} \equiv 1 \pmod{n}$. Denote by D the least integer $2 < D < n$ which does not divide $n - 1$. Then, $\left\lfloor \frac{n-1}{D} \right\rfloor \neq \operatorname{ord}_n(2)$.

Proof. Consider an odd composite integer $n > 3$ for which $2^{n-1} \equiv 1 \pmod{n}$. According to the properties of the multiplicative order modulo n, the smallest positive integer k such that $2^k \equiv 1 \pmod{n}$ defines $\operatorname{ord}_n(2)$, that is, $k = \operatorname{ord}_n(2)$. Since n is composite and $2^{n-1} \equiv 1 \pmod{n}$, this order, $\operatorname{ord}_n(2)$, must divide $n - 1$.

Given D is the least integer greater than 2 and less than n that does not divide $n - 1$, and $\operatorname{ord}_n(2)$ divides $n - 1$, it follows that D cannot equal $\operatorname{ord}_n(2)$. Furthermore, by Lemma 1, we know that if an integer b does not divide an integer a, and $1 < b < \left\lfloor \frac{a}{b} \right\rfloor$, then $\left\lfloor \frac{a}{b} \right\rfloor$ does not divide a. Applying this to our current context with $a = n - 1$ and $b = D$: The least odd composite integer such that $2^{n-1} \equiv 1 \pmod{n}$ is 341 [11]. It follows from Lemma 3 that D must be less than or equal to $\lfloor \log_2(n - 1) \rfloor + 2$. For $n \geq 341$, clearly $D < \lfloor \sqrt{n} \rfloor$ which satisfies Lemma 2. Thus we have $1 < D < \left\lfloor \frac{n-1}{D} \right\rfloor$ and hence, $\left\lfloor \frac{n-1}{D} \right\rfloor$ cannot divide $n - 1$.

Since $\operatorname{ord}_n(2)$ is a divisor of $n - 1$ and $\left\lfloor \frac{n-1}{D} \right\rfloor$ is not, it must be that $\left\lfloor \frac{n-1}{D} \right\rfloor$ is not equal to $\operatorname{ord}_n(2)$, as this would imply a contradiction with the nature of $\operatorname{ord}_n(2)$ as a divisor of $n-1$. Therefore, $\left\lfloor \frac{n-1}{D} \right\rfloor \neq \operatorname{ord}_n(2)$. □

Lemma 5. Let $n, d \in \mathbb{Z}^+$, with $1 \le d < n$. For $(1+x)^n \in (\mathbb{Z}/n)[x]/(x^d - 2)$, we have:

$$[x^j](1+x)^n \equiv \sum_{k=0}^{n} \binom{n}{dk+j} 2^k \pmod{n} \tag{12}$$

Proof. By Identity 2, we have $\sum_{k=0}^{n} \binom{n}{k} 2^{\lfloor \frac{k}{d} \rfloor} = \sum_{j=0}^{d-1} \sum_{k=0}^{n} \binom{n}{dk+j} 2^k$. In Theorem 3, we showed that evaluating $(1+x)^n \in M$ at $x = 1$ yields $\sum_{k=0}^{n} \binom{n}{k} 2^{\lfloor \frac{k}{d} \rfloor}$. Upon expansion $(1+x)^n$ will contain at most d terms. By the nature of exponentiation within our ring, these groups are spaced uniformly over the binomial coefficients. That is, the coefficient of the term x^j in the expanded polynomial corresponds to the sum $\sum_{k=0}^{n} \binom{n}{dk+j} 2^k \pmod{n}$. $\square$

Lemma 6 (Upper bound on d)**.** Let n be an integer such that $n > 3$, then there exists an integer d such that $1 < d \le \lfloor \log_2(n-1) \rfloor + 2$ and $n \not\equiv 1 \pmod{d}$.

Proof. In order to have $n \equiv 1 \pmod{d}$ for some integer d, there must exist an integer k such that $d \cdot k + 1 \equiv 0 \pmod{n}$. This implies that any such d will divide $n - 1$. Given n is a composite integer > 3, clearly $n - 1$ can have at most $\lfloor \log_2(n-1) \rfloor$ prime factors (if $n - 1$ is a power of 2). Since 2 is the least prime that may divide $n - 1$, there must exist a $d \le \lfloor \log_2(n-1) \rfloor + 2$ which does not divide $n - 1$. This implies $n - 1 \not\equiv 0 \pmod{d}$ and hence it follows, $n \not\equiv 1 \pmod{d}$. $\square$

5.1 Primes Case

Lemma 7 (Primes pass)**.** Let n be an odd prime integer such that $n > 3$. Denote d as the least integer greater than 2 such that $n \not\equiv 1 \pmod{d}$. Then, the conditions of Theorem 1 hold unconditionally and n will pass the test.

Proof. For this proof, we will examine all of the cases individually and show that they are satisfied for prime n.

Case 1: $j = 0$

If $j = 0$, then all $\binom{n}{dk} \equiv 0 \pmod{n}$ except for $k = 0$. Hence, we have:

$$\sum_{k=0}^{\lfloor \frac{n}{d} \rfloor} \binom{n}{dk} 2^k \equiv \binom{n}{0} 2^0 \equiv 1 \cdot 1 \equiv 1 \pmod{n} \tag{13}$$

Case 2: $0 < j < d$ and $j \ne n \bmod d$

If $0 < j < d$ and $j \ne n \bmod d$, all $\binom{n}{dk+j} \equiv 0 \pmod{n}$. Hence, the entire sum $\sum_{k=0}^{\lfloor \frac{n}{d} \rfloor} \binom{n}{dk+j} 2^k$ must be equivalent to $0 \pmod{n}$.

Case 3: $j = n \bmod d$

If $j = n \bmod d$, then all $\binom{n}{dk} \equiv 0 \pmod{n}$ except for $k = \lfloor \frac{n}{d} \rfloor$. Hence, we have:

$$\sum_{k=0}^{\lfloor \frac{n}{d} \rfloor} \binom{n}{dk + (n \bmod d)} 2^k \equiv \binom{n}{n} 2^{\lfloor \frac{n}{d} \rfloor} \equiv 1 \cdot 2^{\lfloor \frac{n}{d} \rfloor} \equiv 2^{\lfloor \frac{n}{d} \rfloor} \pmod{n} \tag{14}$$

Conclusion: All of the individual conditions are satisfied when $n > 3$ is an odd prime. Hence, n will pass the test unconditionally. $\square$

5.2 Composites Case

A fundamental theorem in polynomial ring theory states that an integer n is prime if and only if $(1+x)^n \equiv 1+x^n \pmod{n} \in \mathbb{Z}[x]$ [10]. This congruence was used as the basis for the AKS test [1]. A short proof of the theorem, given by Granville (2004) [10], is that since $(x+1)^n - (x^n+1) = \sum_{k=1}^{n-1}\binom{n}{k}x^k$, we may have $(1+x)^n \equiv 1+x^n \pmod{n}$ if and only if n divides $\binom{n}{k}$ for all k in the range $1 \leq k \leq n-1$. It is important to note that for the polynomial congruence to be valid, x^n and $(1+x)^n$ must be irreducible in $(\mathbb{Z}/n)[x]$.

Kopparty and Wang (2014) [12] proved several theorems related to limits on the counts of consecutive zero coefficients in polynomials over finite fields. We take a similar approach to show that composite integers will always fail our test.

Lemma 8 (Composites fail)**.** Let $n = pq$ be an odd composite integer > 3 with p a prime divisor. Let d be the least positive integer > 2 such that $n \not\equiv 1 \pmod{d}$. Suppose $2^{\lfloor \frac{n}{d} \rfloor} \not\equiv 1 \pmod{n}$, and consider the polynomial $f(x) = (1+x)^n - (1+x^n) \in \mathbb{Z}_p[x]$. If n does not have a prime divisor $\leq d$, then $f(x)$ is nonzero when reduced modulo $x^d - 2$.

Proof. Let p be a prime divisor of n. Consider the polynomial ring $\mathbb{Z}_p[x]$. We examine the reduction of $f(x)$ modulo x^d-2, which gives us a polynomial $f(x) \pmod{x^d-2} \in \mathbb{Z}_p[x]$. After reduction modulo x^d-2, the polynomial $f(x)$ has $\deg(f(x)) = d-1$, and can be written as $f(x) = \sum_{i=0}^{d-1} c_i x^i$.

The condition $2^{\lfloor \frac{n}{d} \rfloor} = x^n \not\equiv 1 \pmod{n}$ implies that $2^{\lfloor \frac{n}{d} \rfloor} = x^n \not\equiv 1 \pmod{p}$ for at least 1 prime divisor p of n. Recall also that we are given d which does not divide n, and hence $p \neq d$. Together, these imply that the powers x^k in $f(x)$ do not behave in a cyclical manner when reduced modulo p, and hence, the polynomial $f(x)$ cannot simplify to the zero polynomial due to any cyclical patterns in the exponents. Furthermore, the fact that $x^d \equiv 2$ in our quotient ring, and not 1, ensures that x is not a dth root of unity in $\mathbb{Z}_p$ for any prime p dividing n. Hence, $f(x)$ does not exhibit any cyclical reduction that would occur if x were a root of unity.

Assume for contradiction that $f(x)$ is the zero polynomial in $\mathbb{Z}_p[x]/(x^d-2)$. This would imply that all coefficients c_i are zero in $\mathbb{Z}_p$.

Since p is prime, $\mathbb{Z}_p[x]$ is a finite field. Further, since $2^{\lfloor \frac{n}{d} \rfloor} \not\equiv 1 \pmod{n}$, there must exist at least 1 prime divisor p of n such that $2^{\lfloor \frac{n}{d} \rfloor} \not\equiv 1 \pmod{p}$. This implies that 2 is not a dth power residue modulo p. That is, $a^d \not\equiv 2 \pmod{p}$ for all integers a. Hence, it follows that $x^d - 2$ is irreducible over $\mathbb{Z}_p[x]$ and therefore, $\mathbb{Z}_p[x]/(x^d-2)$ forms a field.

By the Fundamental Theorem of Algebra for finite fields, if $\mathbb{Z}_p[x]/(x^d-2)$ is a field and $\deg(f(x)) = d-1$, then $f(x)$ can have at most $d-1$ roots in $\mathbb{Z}_p[x]/(x^d-2)$. The assumption that $f(x)$ is zero would imply it has p roots, which is a contradiction unless $p \leq d-1$. However, this is clearly false, since we are given n which does not have a prime divisor $\leq d$.

Therefore, $f(x)$ must be nonzero in $\mathbb{Z}_p[x]/(x^d-2)$ for at least one prime p that divides n. □

6 Proof of the Main Theorem

Proof of Theorem 1. Let n be an odd integer > 3. If n is prime, then by Lemma 7, the congruences for our test hold and n will pass the test. Conversely, if n is composite, then by Lemma 8 the congruences for our test cannot hold and n will fail the test. Hence, the theorem is proven. □

7 Algorithm

INPUT: An integer $n > 1$.

1. If $n \equiv 0 \mod 2$:
   (a) If n equals 2, output PRIME.
   (b) Otherwise, output COMPOSITE.
2. If n equals 3, output PRIME.
3. Find the least integer d that is greater than 2 such that $n \not\equiv 1 \pmod{d}$.
4. Compute the polynomial expansion of $x^n \bmod n$ in the ring $(\mathbb{Z}/n)[x]/(x^d - 2)$ with degree d, and store the result.
5. Compute the polynomial expansion of $(1+x)^n \bmod n$ in the ring $(\mathbb{Z}/n)[x]/(x^d - 2)$ with degree d, and store the result.
6. If $(1+x)^n \not\equiv 1 + x^n$, output COMPOSITE.
7. Output PRIME;

7.1 Time Complexity Analysis

7.1.1 Algorithm Overview

The given algorithm is a primality test that involves several computational steps, including modular arithmetic and polynomial exponentiation in the ring $(\mathbb{Z}/n)[x]/(x^d - 2)$.

7.1.2 Analysis of Individual Operations

1. **Check for Even n:**

   This step involves calculating $n \bmod 2$ and has a time complexity of $O(1)$.

2. **Finding d:**

   Finding the least integer $d > 2$ such that $n \not\equiv 1 \pmod{d}$ takes at most $O(\log(n))$ steps, with each step requiring $O(1)$ time for the mod operation. Hence, the overall complexity is $O(\log(n))$.

3. **Computing** $x^n \bmod n \in (\mathbb{Z}/n)[x]/(x^d - 2)$:

   Computing the polynomial expansion of $x^n \pmod{n}$ in the ring $(\mathbb{Z}/n)[x]/(x^d - 2)$ with degree d is equivalent to calculating $2^{\lfloor \frac{n}{d} \rfloor} \pmod{n}$. This step requires modular exponentiation with a $\log(n)$-digit base and a $\log(n)$-digit exponent. The time complexity of modular exponentiation is $O(\log(n)M(n))$.

4. **Computing** $(1+x)^n \bmod n \in (\mathbb{Z}/n)[x]/(x^d - 2)$:

   Computing the polynomial expansion of $(1+x)^n \pmod{n}$ in the ring $(\mathbb{Z}/n)[x]/(x^d - 2)$ with degree d involves exponentiating a polynomial in $(\mathbb{Z}/n)[x]/(x^d - 2)$ with $D = O(\log(n))$ terms. Exponentiation using repeated squaring takes $O(\log(n))$ steps, and each step requires $O(M(n)\log(n))$ time due to the multiplication of polynomials of size $O(\log(n))$. Therefore, the overall complexity is $O(\log^2(n)m(n))$.

5. **Checking the equality** $(1+x)^n = 1 + x^n \pmod{n} \in (\mathbb{Z}/n)[x]/(x^d - 2)$: The final steps involve comparing the equality of coefficients in the polynomials $(1+x)^n$ and $1 + x^n$. This requires $O(\log(n))$ comparisons, which are themselves $O(1)$ operations.

7.1.3 Overall Time Complexity

The dominant time complexity in the algorithm comes from computing $(1+x)^n \pmod{n} \in (\mathbb{Z}/n)[x]/(x^d-2)$. Therefore, the overall time complexity of the algorithm is $T(n) = O(\log^2(n)M(n))$.

Harvey and van Der Heoven (2021) have given an algorithm for integer multiplication which has a time complexity $M(n) = O(\log(n)\log\log(n))$ [13]. This would give our algorithm an overall time complexity of:

$$T(n) = O(\log^2(n)M(n)) = O(\log^2(n)\log(n)\log\log(n)) = O(\log^3(n)\log\log(n)) = \tilde{O}(\log^3(n)) \tag{15}$$

7.1.4 Conclusion

The overall complexity is polynomial in the size of n when expressed in terms of bit operations, making the algorithm efficient for large values of n.

8 Implementation Details

8.1 Reference Implementation

Sample open source .NET and Python implementations, along with test data, are available on the author's Github page [14].

8.2 Pseudocode Implementation

To demonstrate how our test may be implemented, we offer a pseudocode implementation of the key functions involved.

8.2.1 IsPrime Function

```
Require: An integer $n > 1$
  function IsPrime($n$)
    if $n \bmod 2 = 0$ then
      if $n = 2$ then
        return true                                   ▷ $n$ is prime
      else
        return false                                  ▷ $n$ is composite
      end if
    end if
    if $n = 3$ then
      return true                                     ▷ $n$ is prime
    end if
    $d \leftarrow 2$
    $log \leftarrow$ Log2($n$)
    for $i \leftarrow 3$ to Max($log + 2, 3$) do      ▷ Find $d$, the least non-divisor of $n - 1$
      $d \leftarrow i$
      $m \leftarrow (n-1) \bmod d$
      if $m \neq 0$ then
        break
      end if
    end for
    $v_x \leftarrow$ POW$(2, \lfloor n/d \rfloor, n)$
    $polyDegree \leftarrow d - 1$                    ▷ Subtract 1 to account for zero indexing in polynomials
    $poly \leftarrow$ POLYPOW$([1,1], n, n, polyDegree, [2])$
    if $poly[0] \neq 1$ then
        return false                                                          ▷ $n$ is composite
    end if
    $x_{\text{index}} \leftarrow n \bmod d$
    for $i \leftarrow 1$ to $d$ do                                      ▷ Check polynomial equality
        if $i = x_{\text{index}}$ then
            if $poly[i] \neq v_x$ then
                return false                                                  ▷ $n$ is composite
            end if
        else
            if $poly[i] \neq 0$ then
                return false                                                  ▷ $n$ is composite
            end if
        end if
    end for
    return true                                                                   ▷ $n$ is prime
end function
```

8.2.2 PolyPow Function

```
function POLYPOW($poly_A$, $k$, $n$, $d$, $polyMapping$)
    $polyB \leftarrow [1]$                        ▷ Initialize polyB as a polynomial with constant term 1
    while $k > 0$ do
        if $k \bmod 2 = 1$ then
            $polyB \leftarrow$ POLYMUL$(polyA, polyB, n)$            ▷ Multiply polynomials modulo $n$
            $polyB \leftarrow$ POLYREDUCE$(polyB, d, polyMapping, n)$       ▷ Reduce degree of polyB
            if $k = 1$ then
                break
            end if
        end if
        $polyA \leftarrow$ POLYMUL$(polyA, polyA, n)$                      ▷ Square polyA modulo $n$
        $polyA \leftarrow$ POLYREDUCE$(polyA, d, polyMapping, n)$          ▷ Reduce degree of polyA
        $k \leftarrow k/2$
    end while
    return $polyB$
end function
```

8.2.3 PolyReduce Function

```
function POLYREDUCE($polyA$, $d$, $mappingPoly$, $n$)
    if LENGTH$(polyA) \leq d$ then
        return $polyA$
    end if
    $polyB \leftarrow$ CLONE$(polyA)$                          ▷ Copy the polynomial to a new array
    $polyDegree \leftarrow$ DEGREE$(mappingPoly)$           ▷ Find degree of the mapping polynomial
    $degreeDelta \leftarrow d - polyDegree$
    for $i \leftarrow$ LENGTH$(polyB) - 1$ downto $d + 1$ do
        if $polyB[i] = 0$ then
```

```
            continue
        end if
        for $j \leftarrow i - 1 - degreeDelta$, $k \leftarrow polyDegree$ downto 0 do
            $polyB[j] \leftarrow polyB[j] + polyB[i] \times mappingPoly[k]$        ▷ Degree reduction step
        end for
        $polyB[i] \leftarrow 0$
    end for
    $polyC \leftarrow$ new array of size $d + 1$
    for $i \leftarrow 0$ to $\min(\text{LENGTH}(polyC), \text{LENGTH}(polyB)) - 1$ do        ▷ Take coefficients modulo $n$
        $polyC[i] \leftarrow polyB[i]$
        if $n \neq 0$ then
            $polyC[i] \leftarrow polyC[i] \bmod n$
        end if
    end for
    return $polyC$
end function
```

9 Acknowledgements

The author would like to thank the kind users at `mersenneforum.org` for their helpful feedback.

References

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of Mathematics*, pages 781–793, 2002.

[2] Oded Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, 2008.

[3] Hendrik W Lenstra and Carl Pomerance. Primality testing with gaussian periods. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1951): 3376–3390, 2011.

[4] Hendrik W Lenstra. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987.

[5] Robert Baillie and Samuel S Jr Wagstaff. Lucas pseudoprimes. *Mathematics of Computation*, 35(152): 1391–1417, 1980.

[6] Michael O Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1): 128–138, 1980.

[7] Gary L Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300–317, 1976.

[8] Samuel S Jr Wagstaff. Pseudoprimes and a generalization of artin's conjecture. *Acta Arithmetica*, 41 (2):141–150, 1983.

[9] Carl Pomerance. The use of elliptic curves in cryptography. *Advances in Cryptology*, pages 203–208, 1984.

[10] Andrew Granville. It is easy to determine whether a given integer is prime. *Bulletin of the American Mathematical Society*, 42:3–38, 2004. URL `https://www.ams.org/journals/bull/2005-42-01/S0273-0979-04-01037-7`.

[11] N. J. A. Sloane. Entry a001567 in the on-line encyclopedia of integer sequences. `https://oeis.org/A001567`, 2023. Fermat pseudoprimes to base 2.

[12] Swastik Kopparty and Qiang Wang. Roots and coefficients of polynomials over finite fields. *Finite Fields and Their Applications*, 29:198–201, 2014. ISSN 1071-5797. doi: https://doi.org/10.1016/j.ffa.2014.04.002. URL `https://www.sciencedirect.com/science/article/pii/S1071579714000574`.

[13] Joris van Der Hoeven David Harvey. Integer multiplication in time o(n log n). *Annals of Mathematics*, 2021. doi: 10.4007/annals.2021.193.2.4.hal-02070778v2.

[14] Joseph M. Shunia. A sample .net implementation of the primality test. `https://github.com/jshunia/Shunia.Primes`, 2023. Accessed: December 2023.