Modifying Agrawal's Conjecture for Efficient Primality Testing

Joseph M. Shunia

December 2023
Revised: May 2024 (Version 3)

Abstract

We propose a new approach to primality testing based on a modified version of Agrawal's conjecture, a significant yet unresolved conjecture in the field. By slightly altering the conditions of Agrawal's conjecture and leveraging specific number-theoretic properties, we arrive at a deterministic primality test that achieves improved efficiency compared to existing methods.

1 Introduction

Primality testing has witnessed remarkable advancements over the past several decades [1, 2, 3, 4, 5]. A significant breakthrough in this field was the AKS primality test, introduced by Agrawal, Kayal, and Saxena (2002) [5]. 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 [6]. 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 [5]. This bound was lowered significantly by Lenstra and Pomerance (2019) to $\tilde{O}(\log^{6} n)$ [7]. 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 [8]. To make these cryptographic operations practical, fast probabilistic primality tests such as the Baille-PSW primality test (BPSW) [2] or Miller-Rabin (MR) [1, 3] 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 [4]. 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 propose a deterministic primality test with polynomial time complexity $\tilde{O}(\log^{3} n)$. This efficiency gain opens new avenues for practical applications, particularly in cryptography, where fast and reliable primality testing is desirable [8]. Our test is based on a famous conjecture issued by Manindra Agrawal at the conclusion of the PRIMES is in P paper [5].

Conjecture (Agrawal)**.** *Let n and r be two coprime positive integers such that $n^2 \not\equiv 1 \pmod{r}$. Let $(x^r - 1, n)$ is the ideal generated by $x^r - 1$ and n in the polynomial ring $\mathbb{Z}[x]$. Suppose the following polynomial congruence holds:*

$$(x-1)^n \equiv x^n - 1 \pmod{(x^r - 1, n)}.$$

Then, n is prime.

We alter the conditions of Agrawal's conjecture to use the generator $x^r - 2$ instead of $x^r - 1$, and update the restriction on r so that r is the least odd prime such that $r \nmid n(n-1)$. These changes allow us to leverage specific number theoretic properties to establish determinism, which were previously unavailable.

Through the proof of our main theorem (§ 3) and its supporting lemmas (§ 2), we demonstrate that our modified test is equivalent to checking the polynomial congruence $(x+1)^n \equiv x^n + 1 \pmod{n} \in \mathbb{Z}[x]$, which is known to hold for only prime integers n [9].

2 Supporting Lemmas

We begin by presenting the supporting lemmas for our main theorem (§ 3).

Lemma 1. *Let $n \in \mathbb{Z}_{>1}$ be a Carmichael number. Hence, $n = p_1 p_2 \cdots p_m$ is odd, composite, and squarefree, where the p_i are distinct odd prime factors. Furthermore, $(p_i - 1) \mid (n-1)$ for all $p_i \mid n$. Let $r \in \mathbb{P}_{\geq 3}$ be the least odd prime such that $r \nmid n(n-1)$. Let $(x^r - 2, n)$ be the ideal generated by $x^r - 2$ and n in the polynomial ring $\mathbb{Z}[x]$. Consider the polynomial $f(x) := (x+1)^n - x^n - 1 \in \mathbb{Z}[x]$.*

Suppose $x^n \not\equiv x \pmod{(x^r - 2, n)}$. Then, $f(x) \not\equiv 0 \pmod{(x^r - 2, n)}$.

Proof. The assumption $r \nmid n(n-1)$ implies that $r \nmid n$ and $r \nmid (n-1)$. We will begin by briefly show why this is necessary. Suppose $r \mid n$, therefore $r = p$ where $p \mid n$. Then

$$x^n \equiv 2 \pmod{(x^r - 2, p)} \implies (x+1)^n \equiv 3 \pmod{(x^r - 2, p)}.$$

Hence, we have trivially

$$f(x) \equiv (x+1)^n - x^n - 1 \equiv 0 \pmod{(x^r - 2, p)}.$$

Next, suppose $r \mid (n-1)$. Then $r \mid (p-1)$ for some $p \mid n$. Since p is prime, $(p-1) = \phi(p)$. Leading to

$$x^n \equiv x \pmod{(x^r - 2, p)} \implies (x+1)^n \equiv x + 1 \pmod{(x^r - 2, p)}.$$

Again, it is easy to see that $f(x) \equiv 0 \pmod{(x^r - 2, p)}$ in such case.

We will now show that $f(x) \equiv 0 \pmod{(x^r - 2, n)}$ leads to a contradiction under the given conditions. Assume, for the sake of contradiction, that

$$f(x) \equiv (x+1)^n - x^n - 1 \equiv 0 \pmod{(x^r - 2, n)}.$$

Since the congruence holds mod $(x^r - 2, n)$, it must also hold mod $(x^r - 2, p)$ for each prime factor p of n. Otherwise, n could not divide $f(x)$. Thus, for all primes $p \mid n$, we have

$$\begin{aligned} f(x) \equiv (x+1)^n - x^n - 1 \equiv (x+1)^p - x^p - 1 \equiv 0 \pmod{(x^r - 2, p)} \\ \iff (x+1)^n - x^n \equiv (x+1)^p - x^p \equiv 1 \pmod{(x^r - 2, p)}. \end{aligned}$$

From this, we deduce

$$\left((x+1)^{n/p} - x^{n/p}\right)^p \equiv 1 \pmod{(x^r - 2, p)}.$$

Leading to

$$\left((x+1)^{n/p} - x^{n/p}\right)^p \equiv (x+1)^n - x^n \equiv (x+1)^p - x^p \equiv 1 \pmod{(x^r - 2, p)}.$$

This also implies

$$\zeta_p \equiv (x+1)^{n/p} - x^{n/p} \pmod{(x^r-2,p)},$$

where ζ_p is a p-th root of unity modulo (x^r-2,p). By the Chinese Remainder Theorem (CRT), since the congruences hold mod (x^r-2,p) for each prime factor p of n, they also hold mod (x^r-2,n). Thus, we have

$$\begin{aligned}\zeta_n &\equiv (x+1)^{n/n} - x^{n/n} \pmod{(x^r-2,n)}\\ &\equiv (x+1)^1 - x^1 \pmod{(x^r-2,n)}\\ &\equiv (x+1) - x \pmod{(x^r-2,n)}\\ &\equiv 1 \pmod{(x^r-2,n)}.\end{aligned}$$

This is consistent with ζ_p being a trivial p-th root of unity modulo (x^r-2,n). That is

$$\zeta_p \equiv 1 \pmod{(x^r-2,p)}.$$

Hence, we have

$$\Big((x+1)^{n/p} - x^{n/p}\Big)^p \equiv (x+1)^{n/p} - x^{n/p} \equiv (x+1)^n - x^n \equiv (x+1)^p - x^p \equiv 1 \pmod{(x^r-2,p)}.$$

Then, for each p, we must consider the following mutually exclusive cases:

(i) $x^p \equiv x^{n/p} \pmod{(x^r-2,p)} \iff (x+1)^p \equiv (x+1)^{n/p} \pmod{(x^r-2,p)}$,

(ii) $x^n \equiv x^p \pmod{(x^r-2,p)} \iff (x+1)^n \equiv (x+1)^p \pmod{(x^r-2,p)}$.

Each case, taken individually, allows for $f(x) \equiv 0 \pmod{(x^r-2,p)}$. These cases are mutually exclusive, since satisfying both (i) and (ii) leads to

$$x^{n/p} \equiv x^p \equiv x^n \pmod{(x^r-2,p)},$$

implying that $p = r$ and $r \mid n$, contradicting the theorem.

Now, suppose cases (i) and (ii) are both false. If $x^n \equiv \zeta_p x \pmod{(x^r-2,p)}$, where ζ_p is a non-trivial p-th root of unity modulo (x^r-2,p), then $(x+1)^n \equiv \zeta_p(x+1) \pmod{(x^r-2,p)}$. This is possible because the polynomial ring $\mathbb{Z}[x]/(x^r-2,p)$ is isomorphic to the direct product of fields $\mathbb{F}_p[x]/(x-\alpha_1)\times\cdots\times\mathbb{F}_p[x]/(x-\alpha_r)$, where the α_i are the roots of x^r-2 in an algebraic closure of $\mathbb{F}_p$. In some of these fields, there may exist non-trivial p-th roots of unity, allowing for this. However, we showed above that $\zeta_n \equiv 1 \pmod{(x^r-2,n)}$, so this would imply $x^n \equiv \zeta_n x \equiv x \pmod{(x^r-2,n)}$, contradicting the assumption in the theorem that $x^n \not\equiv x \pmod{(x^r-2,n)}$.

Finally, suppose either case (i) or (ii) is true for all primes $p \mid n$. For n, the two cases (i) and (ii) collapse to a single case, since p is replaced by n in the exponents when lifting via the CRT:

$$x^n \equiv x \pmod{(x^r-2,n)} \iff (x+1)^n \equiv x+1 \pmod{(x^r-2,n)}.$$

However, this is a contradiction, since again, $x^n \not\equiv x \pmod{(x^r-2,n)}$ by assumption in the theorem. Therefore $f(x) \not\equiv 0 \pmod{(x^r-2,n)}$. This completes the proof. □

Lemma 2. *Let $n, r \in \mathbb{Z}_{>1}$ such that n is odd, $r \geq 3$, and $r \nmid n$. Consider the polynomial*

$$f(x) := (x+1)^n - x^n - 1 \in \mathbb{Z}[x].$$

Let (x^r-2,n) be the ideal generated by x^r-2 and n in the polynomial ring $\mathbb{Z}[x]$. Suppose

$$f(x) \equiv 0 \pmod{(x^r-2,n)}.$$

Then it is necessary, but not sufficient, that n is a Carmichael number.

Proof. Let $p \mid n$ be prime. Consider

$$f(x) \equiv (x+1)^n - x^n - 1 \equiv 0 \pmod{(x^r - 2, p)}.$$

Expanding the term $(x+1)^n$ via the Binomial Theorem and simplifying, we see

$$f(x) = \sum_{k=1}^{n-1} \binom{n}{k} x^k.$$

From Lucas Theorem, since $p \neq n$, we know that p cannot divide all $\binom{n}{k}$. Furthermore, since $r \geq 3$, the reduction of $f(x)$ mod $(x^r - 2)$ will leave a polynomial remainder of degree $d \geq 1$. In other words, $f(x)$ is not a constant modulo $(x^r - 2)$. Then, for $f(x)$ to be zero, it must vanish for all $a \in \mathbb{Z}$ when evaluated modulo p. That is,

$$\forall a \in \mathbb{Z}, \quad f(a) \equiv (a+1)^n - a^n - 1 \equiv 0 \pmod{p}.$$

Re-arranging this, we have

$$\forall a \in \mathbb{Z}, \quad f(a) \equiv (a+1)^n \equiv a^n + 1 \pmod{p}.$$

Thus,

$$\forall a \in \mathbb{Z}, \quad a^n \equiv a \pmod{p}.$$

By Fermat's Little Theorem (FLT), this implies $(p-1) \mid (n-1)$. Since this must be true for all primes $p \mid n$, we conclude that n must be a Carmichael number. However, this condition is not sufficient, as we did not prove the conditions under which $f(x) \equiv 0 \pmod{(x^r - 2, n)}$ holds for Carmichael numbers. □

Lemma 3. *Let $n \in \mathbb{Z}_{>1}$ be a Carmichael number. Hence, n is odd, composite, and squarefree. Let $r \in \mathbb{P}_{\geq 3}$ be the least odd prime such that $r \nmid n(n-1)$. Let $(x^r - 2, n)$ be the ideal generated by $x^r - 2$ and n in the polynomial ring $\mathbb{Z}[x]$. Then, $x^n \not\equiv x \pmod{(x^r - 2, n)}$.*

Proof. The reduction of $x^n \pmod{(x^r - 2)}$ leaves a polynomial remainder of degree $d = (n \bmod r)$. Formally, we have

$$x^n \equiv 2^{\lfloor n/r \rfloor} x^d \equiv 2^{\lfloor n/r \rfloor} x^{n \bmod r} \pmod{(x^r - 2)}$$

Since $n = p_1 p_2 \cdots p_m$ is a Carmichael number, we know that $(p_i - 1) \mid (n-1)$ for all prime factors $p_i \mid n$. In such case

$$r \nmid (n-1) \quad \Longrightarrow \quad n \not\equiv 1 \pmod{r}.$$

Thus, $d \neq 1$. Leading to

$$x^n \not\equiv x \pmod{(x^r - 2, n)}.$$

□

3 Main Theorem

Theorem 4. *Let $n \in \mathbb{Z}_{>1}$ such that n is odd. Let $r \in \mathbb{P}_{\geq 3}$ be the least odd prime such that $r \nmid n(n-1)$. Let $(x^r - 2, n)$ be the ideal generated by $x^r - 2$ and n in the polynomial ring $\mathbb{Z}[x]$. Consider the polynomial $f(x) := (x+1)^n - x^n - 1 \in \mathbb{Z}[x]$. Suppose $f(x) \equiv 0 \pmod{(x^r - 2, n)}$. Then, n is prime.*

Proof. Expanding $f(x)$ by the Binomial Theorem and simplifying, we see

$$f(x) = (x+1)^n - x^n - 1 = \sum_{k=1}^{n-1} \binom{n}{k} x^k.$$

If n is prime, then $\binom{n}{k} \equiv 0 \pmod{n}$ for all k in the sum. Clearly then, $f(x) \equiv 0 \pmod{n}$ when n is prime.

On the other hand, suppose n is composite. In this case, Lucas Theorem tells us that n cannot possibly divide all $\binom{n}{k}$ in the sum. Applying Lemma 2, we see that n must be a Carmichael number for $f(x)$ to be zero. Furthermore, by Lemma 3, since $r \geq 3$ and $r \nmid (n-1)$, we have $x^n \not\equiv x \pmod{(x^r - 2, n)}$. Under these constraints, we have

$$f(x) \equiv (x+1)^n - x^n - 1 \not\equiv 0 \pmod{(x^r - 2, n)} \quad \text{(By Lemma 1)}.$$

Therefore, under the given conditions, if $f(x) \equiv 0 \pmod{(x^r - 2, n)}$, then n is prime. □

4 Time Complexity

4.1 Proofs

Lemma 5. *Let $n \in \mathbb{Z}_{>6}$. Then, there exists a prime p coprime to n, such that*

$$p < \log n \left(\log\log n + \log\log\log n\right)$$

Proof. From the Prime Number Theorem, n must have fewer than $\log n$ distinct prime factors. Therefore, we need to establish an upper bound on the $\log n$-th prime number. Let p_n denote the n-th prime number. Then, for $n \geq 6$, we have the inequality [10]:

$$\frac{p_n}{n} < \log n + \log\log n.$$

Re-arranging terms, we see

$$p_n < n\log n + n\log\log n.$$

Next, we want to establish an upper bound for $p_{\log n}$, the $\log n$-th prime number. Substituting $\log n$ for n in the inequality, we get

$$p_{\log n} < \log n \left(\log\log(n) + \log\log\log n\right).$$

This expression gives an upper bound for the smallest prime p that is coprime to n. □

Theorem 6. *Let $n \in \mathbb{Z}_{>6}$. Then, there exists a prime $r = \tilde{O}(\log n)$ such that $r \nmid (n-1)$.*

Proof. By Lemma 5, there exists a prime integer r coprime to $n-1$, satisfying

$$r < \log n \left(\log\log n + \log\log\log n\right).$$

Considering the soft-O notation, which simplifies the time complexity by ignoring sublogarithmic factors, we have

$$\tilde{O}(\log n \left(\log\log n + \log\log\log n\right)) = \tilde{O}(\log n).$$

It follows that this bound can be expressed as $\tilde{O}(\log n)$, indicating the existence of an algorithm capable of finding r within this time complexity. □

5 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 prime integer $r \geq 3$ such that $n \not\equiv 1 \pmod{r}$.
4. If n has a prime divisor less than r, output COMPOSITE.
5. Compute the polynomial remainder of x^n in the ring $(\mathbb{Z}/n\mathbb{Z})[x]/(x^r - 2)$ and store the result as f. The intermediate terms and polynomial remainder will be of degree at most $r - 1$.
6. Compute the polynomial expansion of $(x + 1)^n$ in the ring $(\mathbb{Z}/n\mathbb{Z})[x]/(x^r - 2)$ and store the result as g. The intermediate terms and polynomial remainder will be of degree at most $r - 1$.
7. If $g - 1 \neq f$, output COMPOSITE.
8. Output PRIME;

5.1 Time Complexity Analysis

The given algorithm is a primality test that involves several computational steps, including modular arithmetic and polynomial exponentiation in the ring $(\mathbb{Z}/n\mathbb{Z})[x]/(x^r - 2)$. In this subsection, we use $M(n)$ to denote the worst-case time complexity of integer multiplication in terms of n.

5.1.1 Analysis of Individual Operations

1. **Check for Even** n:
   This step involves calculating n mod 2 and has a time complexity of $T_1(n) = O(1)$.
2. **Finding** r:
   Finding the least $r \in \mathbb{P}_{\geq 3}$ such that $n \not\equiv 1 \pmod{r}$ takes at most $O(\log n(\log\log n + \log\log\log n))$ steps (Theorem 6), with each step requiring $O(1)$ time for the mod operation. Hence, the overall complexity of this step is $T_2(n) = O(\log n(\log\log n + \log\log\log n))$.
3. **Computing** $x^n \pmod{(x^r - 2, n)}$:
   Computing the polynomial expansion of x^n in the ring $(\mathbb{Z}/n\mathbb{Z})[x]/(x^r - 2)$ results in intermediate terms and a polynomial remainder of degree at most $r - 1$. This step requires modular exponentiation with a $\log n$-digit base and a $\log n$-digit exponent. The time complexity of modular exponentiation is $T_3(n) = O(\log n \cdot M(n))$.
4. **Computing** $(x + 1)^n \pmod{(x^r - 2, n)}$:
   Computing the polynomial expansion of $(x + 1)^n \pmod{n}$ in the ring $(\mathbb{Z}/n\mathbb{Z})[x]/(x^r - 2)$ involves performing modular exponentiation of a polynomial in $(\mathbb{Z}/n\mathbb{Z})[x]/(x^r - 2)$ with $r = T_2(n)$ terms. Exponentiation using repeated squaring takes $O(\log n)$ steps, and each step requires $O(M(n) \cdot T_2(n))$ time due to the multiplication of polynomials of size $T_2(n)$. Therefore, the overall complexity of this step is $T_4(n) = O(\log n \cdot M(n) \cdot T_2(n))$.

5. **Checking the congruence** $(x+1)^n \equiv x^n + 1 \pmod{(x^r - 2, n)}$:

   The final steps involve comparing the equality of coefficients in the polynomials $(x+1)^n$ and $x^n + 1$. This requires $O(\log n)$ comparisons, which are themselves $O(1)$ operations. The overall time complexity of this step is $T_5(n) = O(\log n)$.

5.1.2 Overall Time Complexity

The dominant time complexity in the algorithm comes from computing $(x+1)^n \pmod{(x^r - 2, n)}$, which is $T_4(n)$. Therefore, the overall time complexity of the algorithm is $T(n) = O(\log n \cdot M(n) \cdot T_2(n))$.

Harvey and van Der Heoven (2021) [11] have given an algorithm for integer multiplication which has a time complexity $M(n) = O(\log n \log\log n)$. This would give our algorithm an overall time complexity of

$$\begin{aligned} T(n) &= O(\log n \cdot M(n) \cdot T_2(n)) \\ &= O(\log n \log n \log\log n \cdot T_2(n)) \\ &= O(\log^2 n \log\log n \cdot T_2(n)). \end{aligned}$$

In soft-O notation [12], typically denoted as $\tilde{O}$, simplicity is achieved by omitting slower-growing logarithmic and lower-order factors that do not significantly contribute to the overall growth rate of the function.

In the context of our overall time complexity $T(n) = O(\log^2 n \log\log n \cdot T_2(n))$, where the dominant term is $O(\log^2 n)$, the linear factors $\log\log n$ are omitted, and the time complexity simplifies to

$$\tilde{T}(n) = \tilde{O}(\log^2 n \cdot \tilde{T}_2(n)).$$

We have $T_2(n) = O(\log n(\log\log n + \log\log\log n))$, which in soft-O simplifies to $\tilde{T}_2(n) = \tilde{O}(\log n)$. By substitution, we have our final time complexity

$$\tilde{T}(n) = \tilde{O}(\log^2 n \log n) = \tilde{O}(\log^3 n).$$

5.1.3 Conclusion

The overall complexity $\tilde{O}(\log^3 n)$ is polynomial in the size of n when expressed in terms of bit operations, making the algorithm efficient for large values of n.

6 Implementation Details

Sample open source .NET and Python implementations, along with test data, are available on the author's Github page [13].

References

[1] G. L. Miller. Riemann's Hypothesis and Tests for Primality. *Journal of Computer and System Sciences*, 13(3):300–317, 1976. ISSN 0022-0000. URL https://sciencedirect.com/science/article/pii/S0022000076800438.

[2] R. Baillie and S. Wagstaff. Lucas Pseudoprimes. *Mathematics of Computation*, 35(152):1391–1417, 1980. ISSN 00255718, 10886842. URL http://jstor.org/stable/2006406.

[3] M. O. Rabin. Probabilistic Algorithm for Testing Primality. *Journal of Number Theory*, 12(1): 128–138, 1980. ISSN 0022-314X. URL https://sciencedirect.com/science/article/pii/0022314X80900840.

[4] S. Wagstaff. Pseudoprimes and a Generalization of Artin's Conjecture. *Acta Arithmetica*, 41(2): 141–150, 1982. URL https://eudml.org/doc/205837.

[5] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics*, pages 781–793, 2002. URL https://jstor.org/stable/3597229.

[6] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. ISBN 978-0-521-88473-0.

[7] H. W. Lenstra and C. Pomerance. Primality Testing with Gaussian Periods. *Journal of the European Mathematical Society: JEMS*, 21(4):1229–1269, 2019. ISSN 1435-9855.

[8] H. W. Lenstra. Factoring Integers with Elliptic Curves. *Annals of Mathematics*, 126(3):649–673, 1987. ISSN 0003486X. URL https://jstor.org/stable/1971363.

[9] A. Granville. It is Easy to Determine Whether a Given Integer is Prime. *Bulletin of the American Mathematical Society*, 42:3–38, 2004. URL https://ams.org/journals/bull/2005-42-01/S0273-0979-04-01037-7.

[10] B. Rosser. Explicit Bounds for Some Functions of Prime Numbers. *American Journal of Mathematics*, 63(1):211–232, 1941. ISSN 00029327, 10806377. URL https://jstor.org/stable/2371291.

[11] D. Harvey and J. van Der Hoeven. Integer Multiplication in Time O(n log n). *Annals of Mathematics*, 2021. URL https://annals.math.princeton.edu/2021/193-2/p04.

[12] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3rd edition, 2013. ISBN 978-1107039032.

[13] Joseph M. Shunia. A Sample .NET Implementation of the Primality Test, 2023. URL https://github.com/jshunia/Shunia.Primes. GitHub repository.