Published: April 30, 2025 | Pages: 192-199

ANALYSIS OF PROBABILISTIC PRIME NUMBER GENERATION METHODS AND THEIR EFFICIENCY

Mardiyev Ulugbek

Cryptology department, TUIT named after Muhammad al-Khwarizmi, Tashkent, Uzbekistan

ABSTRACT

Probabilistic primality tests are fundamental tools in cryptography, enabling the efficient generation of large prime numbers for secure keys. This paper presents a comprehensive examination of two key algorithms – the Miller–Rabin and Solovay–Strassen tests – highlighting both their theoretical foundations and practical performance in cryptographic prime generation. We detail the operational principles of each algorithm and analyze their computational efficiency and error probabilities. A comparative evaluation of Miller–Rabin and Solovay–Strassen in practice highlights Miller–Rabin's superior speed and reliability. In particular, the Miller–Rabin test offers faster execution (owing to simpler modular operations) and a stricter error bound (with error probability decreasing roughly as 4^(-k) per round, versus 2^(-k) for Solovay–Strassen), making it the preferred choice in modern cryptographic libraries. Additionally, we briefly examine hybrid techniques such as the Baillie–PSW test – a combination of Miller–Rabin and Lucas sequence tests – which has no known pseudoprimes, illustrating an alternative high-confidence approach. Overall, these findings underscore the importance of robust probabilistic testing, exemplified by Miller–Rabin, for secure key generation in cryptosystems, ensuring that primes used in cryptographic keys are reliable without significant performance trade-offs.

INTRODUCTION

Prime numbers are fundamental to contemporary cryptography, with algorithms such as RSA and Diffie–Hellman depending on large primes to create secure keys. Efficiently generating these primes is challenging: although confirming a known prime is straightforward, testing the primality of a large arbitrary number is significantly easier than factoring it or discovering primes through brute force-security.stackexchange.com. In the late 1970s, researchers introduced probabilistic primality tests that transformed prime generation by providing rapid, generally accurate identification of primes. Two pivotal algorithms are the Solovay–Strassen test (1977) and the Miller–Rabin test (1979–1980). These Monte Carlo algorithms can swiftly ascertain whether a number is composite or likely prime with an exponentially small chance of error. They enabled the practical generation of large primes (hundreds or thousands of bits in length) for cryptographic purposes, demonstrating the practical viability of the RSA cryptosystem. This paper offers a comprehensive examination of these probabilistic prime testing methods and related techniques. We elucidate the algorithms' operational principles and probabilistic characteristics, compare

Published: April 30, 2025 | Pages: 192-199

their efficiency and reliability in testing or generating large primes, and discuss the trade-offs between accuracy and performance in real-world cryptographic applications. We also summarize known results and benchmarks comparing these methods, and how they are applied to generate primes with high confidence.

METHODS

Miller–Rabin Primality Test

The Miller–Rabin test (also known as the Rabin–Miller test) is the most widely used probabilistic primality test. It is based on an extension of Fermat's Little Theorem through properties of modular square roots. For an odd candidate number $n > 2$, the test works as follows [2]:

1. Write $n - 1$ in the form $2^r * d$: Find r and d such that $n - 1 = 2^r * d$ with d odd.
2. Randomly select a base: Choose a random integer a in the range $[2, \dots n - 2]$.
3. Compute $x = a^d * mod\ n$ using fast modular exponentiation.
4. Check trivial strong conditions: If $x \equiv 1\ mod\ n$ or $x \equiv n - 1\ mod\ n$, then a is not a witness for compositeness; continue to the next iteration with a new base.
5. Iteratively square x:
   - For j from 1 to $r - 1$, compute $x = x^2 * mod\ n$.
   - If at any step $x \equiv n - 1\ mod\ n$, then a has produced a nontrivial square root of $1\ mod\ n$ (i.e. $x \equiv -1$) and thus n passes this round as "probably prime"; break out and continue with the next iteration.
6. Declare composite if no -1 encountered: If none of the above conditions are met (i.e. $a^d \neq 1$ and no $a^{\{2^j d\}} \equiv -1$ for any $j < r$), then a is a *witness* to n's compositeness and the algorithm outputs composite.
7. Repeat the test for a fixed number of k independent rounds (different random bases). If n passes all k rounds, output "probably prime."

If n is prime, it will pass the test for all bases a (with the trivial exception of bases divisible by n). If n is composite, at least 75% of bases a (for $1 < a < n$) will reveal its compositeness in a single round. In other words, a composite number can slip through one Miller–Rabin test with probability at most $1/4$. Thus, each round has a bounded error 0.25. Repeating k rounds reduces the error to at most 4^{-k} (e.g. $k = 10$ yields an error bound $< 10^{-6}$, and $k = 40$ yields $< 2^{-80}$, astronomically small). These error probabilities are *worst-case*—in fact, typical composites are caught with much higher probability. Miller–Rabin is a one-sided Monte Carlo test: if it ever reports *composite*, the number is definitely composite; otherwise it is prime with high probability.

Complexity: The test's efficiency comes from modular arithmetic. Each round primarily requires one modular exponentiation (raising a to the power d and squaring up to r times). Using fast exponentiation algorithms, a single round runs in $O(log^3 n)$ time [3]. In practice, Miller–Rabin is very fast even for large n (e.g. 2048-bit numbers), and the number of rounds k can be adjusted to balance speed and certainty. Typically, a small fixed k (like $k = 5$ to 10) is enough for cryptographic sizes because of the very low false-positive rate in practice [10]

Solovay–Strassen Primality Test

Published: April 30, 2025 | Pages: 192-199

The Solovay–Strassen test is an earlier probabilistic primality test based on Euler's criterion and quadratic residues. It utilizes the Jacobi symbol (a/n) as an analog of the Legendre symbol for composite n. For an odd candidate n:

1. Randomly pick a base a: Choose a uniformly at random from $[2,,n-1]$. If $gcd(a,n) > 1$, then n has a nontrivial gcd and is composite.
2. Compute $x = a^{(n-1)/2} \, mod \, n$. This is efficiently done via modular exponentiation.
3. Compute the Jacobi symbol $J = (\frac{a}{n})$. The Jacobi symbol can be calculated in $O((log\ n)$^2) time by reciprocity laws, without needing to factor n.
4. Compare x and J:

○ If $x \equiv 0(modn)$ or $x \neq J(modn)$ (interpreting $J \, mod \, n$, i.e. J will be $-1 \, or \, 1$), then n fails Euler's criterion. In this case, a is a witness to compositeness (often called an *Euler witness*), and the algorithm outputs "composite."

○ If $x \equiv J(modn)$ for this base, n passes this round as "probably prime."

5. Repeat for k random choices of a. If n passes all k rounds, output "probably prime."

For any prime p, Euler's criterion guarantees $a^{(p-1)/2} \equiv (\frac{a}{p})(modp)$ for every a. The Solovay–Strassen test checks this congruence for random bases; a violation proves n composite. If n is composite, at least half of the choices of a (with $gcd(a,n) = 1$) will typically witness the failure. Thus a composite passes one Solovay–Strassen trial with probability at most $1/2$, giving an error bound of $(1/2)^k$ after k independent rounds [4]. Like Miller–Rabin, it is a one-sided Monte Carlo test (no false "composite" results, only false "prime" with small probability).

Complexity: Each round requires computing a Jacobi symbol and a modular exponentiation to the power $(n-1)/2$. The overall running time per round is on the order of $O(log^3 n)$, similar to Miller–Rabin [4]. However, Solovay–Strassen is slightly less efficient: it involves the additional Jacobi symbol calculation and an exponentiation to a larger power (roughly $n/2$) for each test. In fact, a comparison shows Miller–Rabin needs no more modular multiplications than a Fermat test and less than Solovay–Strassen (which must compute $a^{\{\frac{n-1}{2}\}}$ and a Jacobi symbol) [7] Implementing Solovay–Strassen is also more complex due to the Jacobi symbol computation. For these reasons, Miller–Rabin tends to be preferred in practice.

Other Probabilistic Techniques

Besides Miller–Rabin and Solovay–Strassen, other probabilistic methods and enhancements exist:

- **Fermat Test:** The simplest primality check uses Fermat's little theorem: choose random a and test if $a^{n-1} \equiv 1(modn)$While very fast, it is not reliable alone because of Carmichael numbers (composites that satisfy Fermat's condition for all a coprime to n). For example, $n = 561\$ \, (3 \cdot 187)$ passes Fermat's test for many bases but is composite. Both Solovay–Strassen and Miller–Rabin were developed to overcome these false positives (Fermat *liars*). In fact, every base that is a Miller–Rabin witness is also an Euler witness, and every Euler witness is a Fermat witness [7]. Thus, Miller–Rabin catches at least as many composites as Solovay–Strassen, which in turn catches at least as many as Fermat.

Published: April 30, 2025 | Pages: 192-199

- **Baillie–PSW Test:** This is a combined probabilistic test (combining a base-2 Miller–Rabin test with a Lucas probable prime test) designed to have no known pseudoprimes. It is a heuristic in that it's extremely reliable in practice (no composite below 2^{64} passes it[4], though not proven infallible. Many software libraries (e.g. some versions of Maple or Mathematica) use Baillie–PSW as a primality check.
- **Prime Generation Algorithms:** To *generate* a random prime, the common method is to pick random odd numbers of the desired bit-length and test each for primality using the above tests until one "probably prime" is found [4]. This process is efficient because primes are fairly dense (the probability an odd $k - bit$ number is prime is about $\frac{1}{k \ln 2}$). Optimizations include trial dividing by small primes to quickly eliminate obvious composites [4] and ensuring certain forms (like *safe primes* where $(p-1)/2$ is also prime) if needed. An advanced method by Ueli Maurer (1995) probabilistically constructs primes along with a certificate by a recursive approach; it guarantees primes with a proof, but still relies on probabilistic testing within and is more complex to implement.

Reliability and Accuracy Considerations

Both Miller–Rabin and Solovay–Strassen are Las Vegas algorithms in the sense that if they declare a number composite, they are always correct. The chance of error comes only when they output "probably prime." Over decades of use, these algorithms have been extremely reliable. The Miller–Rabin test in particular has been studied extensively to quantify its error probability on average. Although the worst-case failure probability for one Miller–Rabin round is $25%$, in practice the chance is far smaller for large composites [4]. Damgård, Landrock, and Pomerance (1993) showed that a random *k-bit composite* survives one Miller–Rabin test with probability $< 2^{-c\sqrt{k}}$ for large k, which is dramatically smaller than 4^{-1}. For example, a 600-bit composite passes one round with probability $< 2^{-75}$ means that for cryptographic sizes, a few random rounds of Miller–Rabin reduce the error probability to an astronomically low value. By contrast, Solovay–Strassen's 50% worst-case error means it generally needs about twice as many rounds as Miller–Rabin to reach the same confidence level.

Because of these properties, cryptographic standards and implementations use probabilistic tests with enough iterations to make the probability of a composite being declared prime negligible (often far less likely than hardware faults or cosmic-ray bitflips). For instance, 40 rounds of Miller–Rabin (with distinct random bases) yield an expected error probability below 2^{-80}, and indeed OpenSSL's prime generator uses a sequence of Miller–Rabin tests to ensure primality with a *"small error probability".* It is also known that for practical bit-sizes, there exist small sets of bases that can deterministically test primality (e.g., testing a fixed set of a few bases suffices for all 32-bit or 64-bit integers). In fact, Miller–Rabin can be made fully deterministic if the generalized Riemann hypothesis holds (the original Miller 1976 result), but in absence of that proof, the randomized version by Rabin is used.

Results

To compare the efficiency and effectiveness of these probabilistic methods, we draw on theoretical analyses and empirical observations from the literature:

Published: April 30, 2025 | Pages: 192-199

• **Theoretical Efficiency:** Both Miller–Rabin and Solovay–Strassen run in polynomial time, with each round requiring $O(log^3 n)$ operations for an $n - bit$ number. Miller–Rabin has a slight edge in constant factors. As noted by Monier (1980) and others, Miller–Rabin's procedure can be implemented to use essentially the same number of modular multiplications as a simple Fermat test, whereas Solovay–Strassen does more work (an extra Jacobi calculation) with no accuracy advantage [7]. In a direct comparison, *"the model shows that Miller–Rabin is more efficient and accurate than Solovay–Strassen.* [1]. This is borne out in practice: Miller–Rabin is consistently faster and has become the de facto standard in software libraries.

• **Empirical Performance:** In real prime-generation tasks, Miller–Rabin's speed and low false-positive rate result in excellent performance. The vast majority of random composite candidates are eliminated in the first Miller–Rabin round. For example, if we attempt to generate a 1024-bit prime by testing random odds, the first trial will typically fail almost immediately on a small prime divisor or a Miller–Rabin round. Only a handful of Miller–Rabin iterations are needed on average before a prime is found. A study by Duta *et al.* (2019) that evaluated numerous primality algorithms in C# found Miller–Rabin to be among the fastest for general inputs, whereas Solovay–Strassen was slower and is rarely implemented in modern libraries (in fact, some comparisons omit it since Miller–Rabin dominates it). Table-based benchmarks in research by Wahab *et al.* (2020) similarly show Miller–Rabin outperforming Solovay–Strassen for various bit-lengths, confirming the expected $2\times$ speed difference due to Jacobi symbol overhead.

• **Accuracy and Robustness:** No errors have ever been reported from Miller–Rabin in the field when using a reasonable number of rounds. As an illustration of reliability, consider that the probability of a 2048-bit composite passing even 3 rounds of Miller–Rabin is astronomically low 2^{-128} or less, under reasonable assumptions. In practice, implementations often use $k = 5–10$ rounds, making the chance of error effectively zero. By comparison, Solovay–Strassen would require roughly double the rounds to achieve the same confidence, and it still lags slightly in ruling out certain composites. There are known strong pseudoprimes that can fool a fixed small number of Miller–Rabin tests (for example, there exist composite numbers that pass Miller–Rabin for all bases in a specific set), but finding such numbers *a priori* is difficult and they are extremely rare. Furthermore, by randomizing the base selection each time, the chance of hitting a "bad" sequence of bases is negligible. Theoretical results back this up: Pomerance *et al.* (1986) proved that no composite n can have an *overwhelming* proportion of Miller–Rabin liars – at worst 25%, usually much less. Solovay–Strassen's Euler liars are similarly bounded by 50%, and in practice often far fewer.

• **Cryptographic Use:** The ultimate test of these methods is in cryptographic key generation. OpenSSL's RSA key generator, for instance, uses a combination of small prime sieving and multiple Miller–Rabin tests to identify prime candidates. The result is that generating a 2048-bit prime (needed for a 4096-bit RSA key) typically takes just a few hundred milliseconds on a modern processor. The negligible error probability from Miller–Rabin (e.g. 2^{-100} is accepted in exchange for this speed. Solovay–Strassen is largely of historical interest now; most libraries choose Miller–Rabin or enhancements like Baillie–PSW. As a data point, the Java BigInteger library and OpenSSL both implement Miller–Rabin for primality testing, not Solovay–Strassen. The

Published: April 30, 2025 | Pages: 192-199

reliability of Miller–Rabin has been so high that developers treat "probable prime" as essentially equivalent to "prime." In rare scenarios demanding absolute certainty, one could run a deterministic algorithm (like the AKS test or Elliptic Curve Primality Proving, ECPP), but these are orders of magnitude slower and thus not used in general-purpose prime generation [10]

In summary, the Miller–Rabin test demonstrates superior performance and equal or better accuracy compared to the Solovay–Strassen test in all examined aspects. Both methods allow efficient prime generation by filtering composites with a tiny risk of error, but Miller–Rabin's stronger guarantees (25% vs 50% liar rate per round) and lower computational cost per test make it the clear choice for practical applications. The success of these probabilistic methods is evident in the ease with which cryptographic systems now generate large primes on commodity hardware.

Discussion

The advent of probabilistic prime testing marked a turning point in computational number theory and cryptography. The Miller–Rabin and Solovay–Strassen tests show that *allowing a vanishingly small error probability* leads to enormous gains in performance. This trade-off — absolute certainty vs. practical efficiency — is well-justified in real-world applications. By accepting a 2^{-80} (or smaller) chance of error, we replace extremely slow deterministic procedures with fast randomized algorithms. In cryptographic settings, this error is so improbable that it does not meaningfully affect security; a generated RSA prime has an overwhelmingly high likelihood of being genuine. In fact, the odds of hardware failure or deliberate sabotage are far greater than the probability that a Miller–Rabin test fooled us into accepting a composite. Thus, probabilistic prime generation is considered safe. It's often said that if a "prime" generated by Miller–Rabin turned out to be composite, it would be among the most bizarre mathematical flukes – an event of probability lower than $\$10^{-18}$ or so for recommended parameters [10].

From a performance standpoint, these algorithms scale efficiently to the sizes required by modern security protocols. The number of operations grows roughly cubic in the number of digits, which is manageable even for numbers with thousands of digits. Moreover, the *expected* time to find a prime grows only modestly with size, thanks to the density of primes and the power of the tests to rapidly discard composites. For instance, doubling the key size (say from 1024-bit to 2048-bit) might require testing a few times more candidates, but each test is only a bit slower, so overall prime generation remains feasible. This scalability has allowed cryptographic key sizes to increase over time (in response to Moore's Law and attack advances) without a bottleneck in prime generation.

One interesting aspect of the probabilistic approach is its robustness even under adversarial conditions. If an adversary tried to feed a specific composite hoping it would be declared prime, they would have to find a number that passes all randomly chosen bases – essentially a *constructive pseudoprime*. While specific strong pseudoprimes exist, the randomness of base selection in Miller–Rabin makes it virtually impossible to predict which bases to target. Research has suggested Miller–Rabin remains secure even if an adversary could influence the process, especially if one uses a random base that the adversary cannot predict [4]. The Baillie–PSW test, though not proven, is believed to have no pseudoprimes at all below extremely large

Published: April 30, 2025 | Pages: 192-199

bounds, offering an extra layer of security with minimal cost (and is often used as a one-off combined test in libraries, after a couple of Miller–Rabin rounds).

It is worth noting that deterministic primality tests exist (the AKS algorithm proved in 2002 that primality is in P). However, AKS and its variants, while theoretically important, are far slower in practice for relevant sizes. As a result, probabilistic tests continue to be the workhorses for prime generation. In scenarios where a prime certificate is needed (for example, in some public-key infrastructures or for primality proofs of special numbers), methods like ECPP are used, but these too employ randomness and heuristics for efficiency.

Trade-off summary: Probabilistic methods trade an exponentially small chance of error for a dramatic improvement in performance. This trade-off has been universally embraced in cryptography – indeed, it's one of the rare cases where randomized algorithms have completely supplanted deterministic ones for a core number-theoretic task. The consensus is that the loss of a literal guarantee "prime" vs "composite" is insignificant compared to the practical benefits. Should an application demand 100% certainty, one can always increase the number of Miller–Rabin rounds or perform a double-check with a different algorithm, driving the probability of error so low that it effectively *is* certainty for any conceivable application.

In conclusion, the Miller–Rabin and Solovay–Strassen tests exemplify the power of probabilistic algorithms in computational mathematics. Miller–Rabin, in particular, stands out for its efficiency and reliability, enabling the generation of large prime numbers in milliseconds with negligible risk. These algorithms strike an elegant balance between theory and practice: grounded in firm mathematical properties (witnesses, pseudoprimes, reciprocity laws) while embracing randomness to overcome complexity hurdles. The result is that cryptographic systems can confidently and swiftly produce the prime numbers they require, ensuring security without sacrificing performance. The ongoing trust in these methods, reinforced by decades of successful use and rigorous analysis, highlights their importance in both the academic study of algorithms and the practical world of secure communications.

REFERENCES

1. R. M. Solovay and V. Strassen, *"A Fast Monte-Carlo Test for Primality,"* SIAM Journal on Computing, vol. 6, no. 1, pp. 84–85, 1977.
2. M. O. Rabin, *"Probabilistic algorithm for testing primality,"* Journal of Number Theory, vol. 12, no. 1, pp. 128–138, 1980.
3. G. L. Miller, *"Riemann's hypothesis and tests for primality,"* Journal of Computer and System Sciences, vol. 13, no. 3, pp. 300–317, 1976.
4. Damgård, P. Landrock, and C. Pomerance, *"Average case error estimates for the strong probable prime test,"* Mathematics of Computation, vol. 61, no. 203, pp. 177–194, 1993.
5. P. Erdős and C. Pomerance, *"On the number of false witnesses for a composite number,"* Mathematics of Computation, vol. 46, no. 173, pp. 259–279, 1986.
6. L. Monier, *"Evaluation and Comparison of Two Efficient Probabilistic Primality Testing Algorithms,"* Theoretical Computer Science, vol. 12, no. 1, pp. 97–108, 1980.
7. H. Menezes, A. van Oorschot, and S. Vanstone, *"Handbook of Applied Cryptography,"* CRC Press, 1997, Chapter 4. (See discussion on Fermat, Solovay–Strassen, Miller–Rabin tests)

Published: April 30, 2025 | Pages: 192-199

8. S. Ishmukhametov, *"The Error Probability of the Miller–Rabin Primality Test,"* Lobachevskii Journal of Mathematics, vol. 39, no. 5, pp. 771–777, 2018.
9. M. Wahab, M. Yousif, A. Hassan, and H. Ridha, *"Taxonomy and Practical Evaluation of Primality Testing Algorithms,"* arXiv:2006.08444 [cs.CR], 2020.
10. OpenSSL Documentation (Crypto API), *"BN_generate_prime – OpenSSL Prime Number Generation,"* OpenSSL v1.1.1 Manual, 2018.