Miller–Rabin Primality Testing: Analysis and Benchmarks

Course Project

December 1, 2025

Abstract

We present a rigorous analysis of the Miller–Rabin (MR) primality test, its error bounds, and time complexity, alongside benchmarks comparing trial division, Fermat, randomized MR with k bases, and deterministic MR for 64-bit integers. Experiments span multiple input distributions (random odd, Carmichael, small-factor composites, primes), and we include a toy RSA demo built on the same MR implementation.

1 Problem Setting and Model

We seek to decide primality of an integer $n \geq 2$. The analysis is stated in the uniform RAM model with unit-cost operations on $\Theta(\log n)$-bit words. No implementation details appear in this section; they are deferred to later benchmarking notes.

2 Miller–Rabin Algorithm and Proofs

2.1 Preprocessing for odd inputs

For every odd $n > 2$, factor

$$n - 1 = 2^s \cdot d, \qquad d \text{ odd}, \ s \geq 1.$$

The pair (s, d) is computed once and reused across all bases.

2.2 One Miller–Rabin round

Let $n > 2$ be odd and a satisfy $2 \leq a \leq n - 2$ and $\gcd(a, n) = 1$.

1. Compute $x = a^d \bmod n$ by repeated squaring.
2. If $x \in \{1, n-1\}$, accept the round.
3. Otherwise, for $r = 1, \ldots, s-1$:
   (a) Set $x \leftarrow x^2 \bmod n$.
   (b) If $x = n - 1$, accept; if $x = 1$, reject (non-trivial square root of 1).
4. If no acceptance occurs, reject.

2.3 Correctness on prime inputs

Assume n is prime. The multiplicative group $G = (\mathbb{Z}/n\mathbb{Z})^\times$ has order $|G| = n - 1 = 2^s d$.

Lemma 1 (Square roots of unity)**.** *If* $x^2 \equiv 1 \pmod{n}$*, then* $x \equiv \pm 1 \pmod{n}$*.*

Proof. We have $n \mid (x-1)(x+1)$. Since n is prime, one factor vanishes modulo n. $\square$

Lemma 2 (Allowed trajectory)**.** *Let $x = a^d \in G$. Then $x^{2^s} = a^{n-1} \equiv 1$. In the sequence $x, x^2, x^4, \ldots, x^{2^s}$, any transition from a value other than -1 to 1 would create a non-trivial square root of 1, contradicting Lemma 1. Therefore the sequence must either start at 1 or hit -1 exactly once before reaching 1.*

Proof. The equality $x^{2^s} = 1$ follows from Lagrange's theorem. Suppose some iterate $x^{2^j} = 1$ with $0 < j < s$ and $x^{2^{j-1}} \not\equiv -1$. Then $x^{2^{j-1}}$ is a non-trivial square root of 1, impossible by Lemma 1. Hence acceptance in a prime modulus always occurs along the Miller–Rabin path. $\square$

Theorem 1 (No false negatives)**.** *For prime n and any admissible base a, one Miller–Rabin round accepts.*

Proof. Lemma 2 shows the round's control flow accepts exactly in the situations mandated by the group structure; no rejection is possible. $\square$

2.4 Error bound on composite inputs

Let n be odd and composite; write $n - 1 = 2^s d$ with d odd. Define a base a coprime to n as a *strong liar* if the round accepts and as a *strong witness* otherwise. We prove that at most a quarter of all admissible bases are liars, yielding the classical $(1/4)^k$ error bound for k independent rounds.

CRT decomposition. Factor $n = \prod_{i=1}^{r} p_i^{e_i}$. By the Chinese Remainder Theorem,

$$(\mathbb{Z}/n\mathbb{Z})^\times \cong \prod_{i=1}^{r} G_i, \qquad G_i = \left(\mathbb{Z}/p_i^{e_i}\mathbb{Z}\right)^\times.$$

Each G_i is cyclic of order $p_i^{e_i-1}(p_i - 1) = 2^{t_i} m_i$ with m_i odd and $t_i \geq 1$. Choose generators g_i so every a corresponds to exponents u_i with $0 \leq u_i < 2^{t_i} m_i$.

Component-wise acceptance. Write $u_i = 2^{\lambda_i} v_i$ with v_i odd and $0 \leq \lambda_i \leq t_i$. The MR acceptance condition in component i is:

$$\text{either } g_i^{u_i d} \equiv 1 \pmod{p_i^{e_i}} \quad \text{or} \quad g_i^{u_i 2^j d} \equiv -1 \text{ for some } j < t_i.$$

Because d is odd, $g_i^{u_i d} \equiv 1$ iff $m_i \mid u_i$. Moreover, $g_i^{u_i 2^j d} \equiv -1$ requires $m_i \mid u_i$ and $u_i/m_i \equiv 2^{t_i - 1} \pmod{2^{t_i}}$, the unique exponent producing order 2 after the squaring ladder.

Counting liars per component. Acceptable u_i are exactly those with $m_i \mid u_i$ and $u_i \bmod 2^{t_i} \in \{0, 2^{t_i-1}\}$. The count is

$$|L_i| = 2 \cdot \frac{2^{t_i} m_i}{2^{t_i}} = \frac{|G_i|}{2^{t_i - 1}},$$

so the liar fraction in component i is $\rho_i = 2^{-(t_i-1)} \leq \frac{1}{2}$, with $\rho_i \leq \frac{1}{4}$ whenever $t_i \geq 3$.

Global product bound. Independence of the CRT components gives $|L| = \prod_i |L_i|$ and

$$\frac{|L|}{|G|} = \prod_{i=1}^{r} \rho_i = 2^{-\sum_i (t_i - 1)}.$$

For any odd composite n, $\sum_i (t_i - 1) \geq 2$:

- With at least two distinct prime factors ($r \geq 2$), two components each contribute a factor at most $\frac{1}{2}$, yielding $|L|/|G| \leq \frac{1}{4}$.
- For a prime power p^e with $e \geq 2$, the 2-adic valuation of $|G_1|$ satisfies $t_1 \geq 2$, so $\rho_1 \leq \frac{1}{4}$.

Thus $|L| \leq \frac{1}{4}|G|$. Random bases are chosen uniformly from admissible residues, so one round accepts a composite with probability at most $\frac{1}{4}$, and k independent rounds accept with probability at most $(1/4)^k$.

2.5 Deterministic base set for 64-bit inputs

The fixed bases $\{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37\}$ suffice for all odd $n < 2^{64}$: passing MR for all of them implies primality. Therefore `is_prime_mr_det64` is exact on 64-bit integers.

2.6 Time Complexity

Let $M(n)$ denote the cost of one modular multiplication on $\Theta(\log n)$-bit words; in the RAM model $M(n) = O(1)$.

- Fast exponentiation of $a^d \bmod n$ uses $\lfloor \log_2 d \rfloor + \text{popcount}(d)$ multiplications, i.e., $O(\log n)$.
- One MR round consists of that exponentiation and up to $s - 1 \leq \log_2(n-1)$ additional squarings, so $O(\log n)$.
- k rounds take $O(k \log n)$ time and $O(1)$ space.
- Fermat-k matches these time bounds but lacks the 4^{-k} error decay; trial division is $O(\sqrt{n})$.

2.7 Summary of Bounds

Algorithm	Time	Error	Space
Trial Division	$O(\sqrt{n})$	0	$O(1)$
Fermat-k	$O(k \log n)$	Can be 1 on Carmichael	$O(1)$
MR-k	$O(k \log n)$	$\leq (1/4)^k$	$O(1)$
MR-det-64	$O(\log n)$	0 (for $n < 2^{64}$)	$O(1)$

Table 1: Asymptotic behavior of implemented algorithms.

3 Algorithms Implemented (C++)

- **Trial Division (TD)** (`is_prime_td`): odd divisors up to $\lfloor \sqrt{n} \rfloor$; disabled for bits > 48.
- **Fermat-k** (`is_probable_prime_fermat`): k random bases; rejects on any violation of $a^{n-1} \equiv 1$.
- **MR-k** (`is_probable_prime_mr`): k random bases with the standard strong probable prime check; per-round error $\leq 1/4$, total $\leq 4^{-k}$.
- **MR-det-64** (`is_prime_mr_det64`): fixed 12 bases; exact for $n < 2^{64}$.

4 Benchmark Workflow

All benchmarks live in `src/main.cpp`; `scripts/run_benchmarks.py` drives the binary `mr_bench`.

- **CLI**: `./mr_bench <algo_id> <dist_id> <bits> <sample_count> <rounds> <seed_base> <reps>`.
- **Distributions** (`dist_id`): random odd, Carmichael, composite with small factor, primes (generated via MR-det-64).
- **Metrics**: each repetition outputs `<time_ns_total> <error_count>` across the sample of size S, with truth labels provided by MR-det-64.
- **Rounds**: Fermat/MR use k supplied on the CLI (e.g., $k = 10$); TD ignores it.
- **Python aggregation**: `results/raw_results.csv` records all rows; plots are generated by `plot_results.py`.

5 Results

5.1 Plots

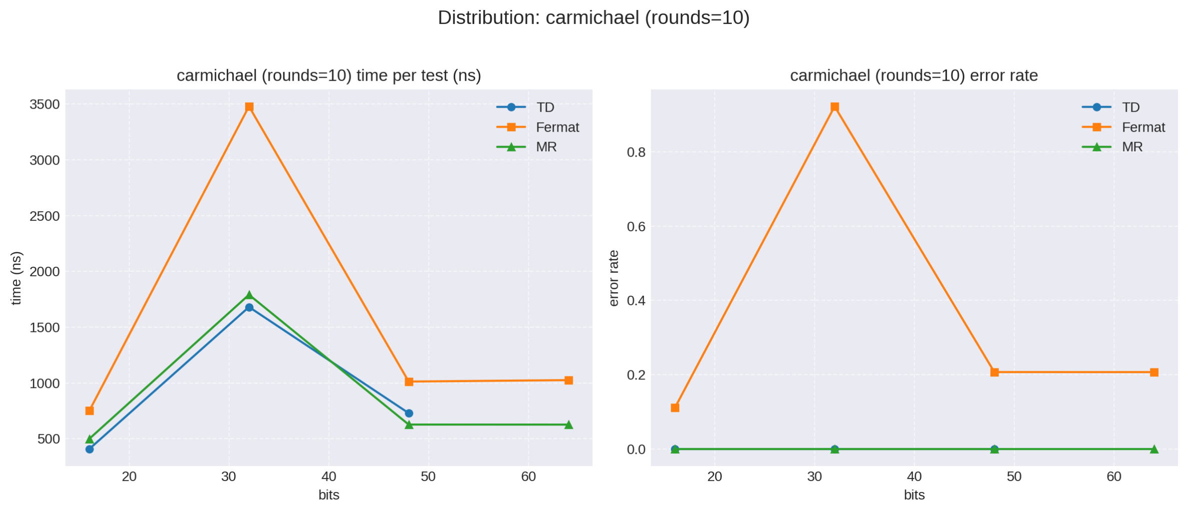

Figure 1: Carmichael distribution, $k = 10$: time per test (ns) and error rate.

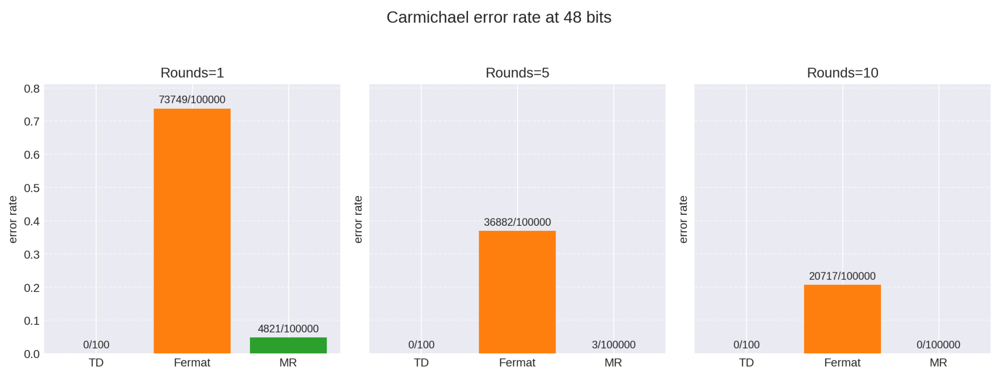

Figure 2: Carmichael, 48 bits: error rate vs rounds for Fermat and MR.

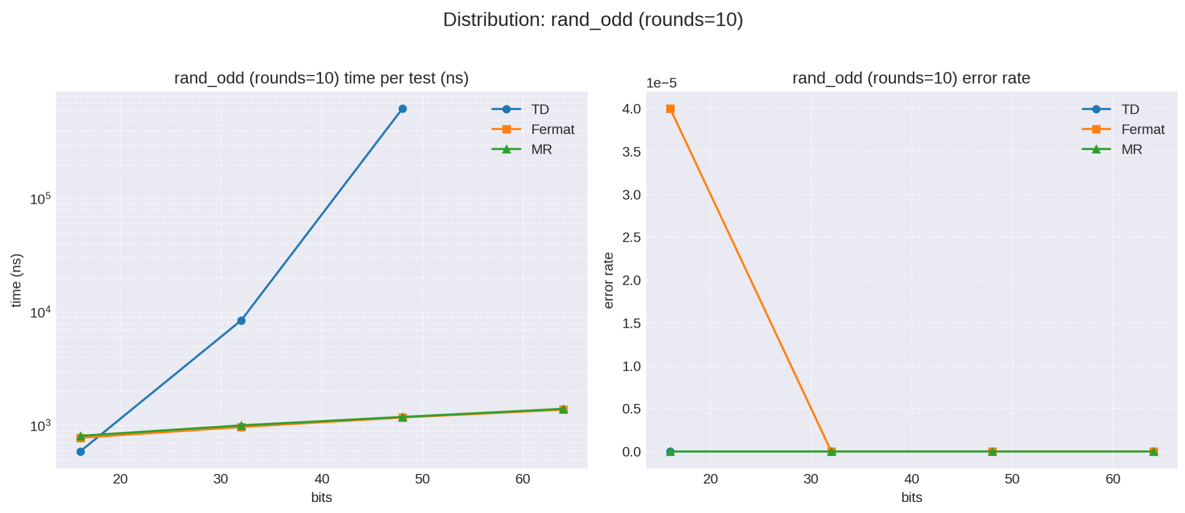

Figure 3: Random odd inputs, $k = 10$: time per test (ns) and error rate.

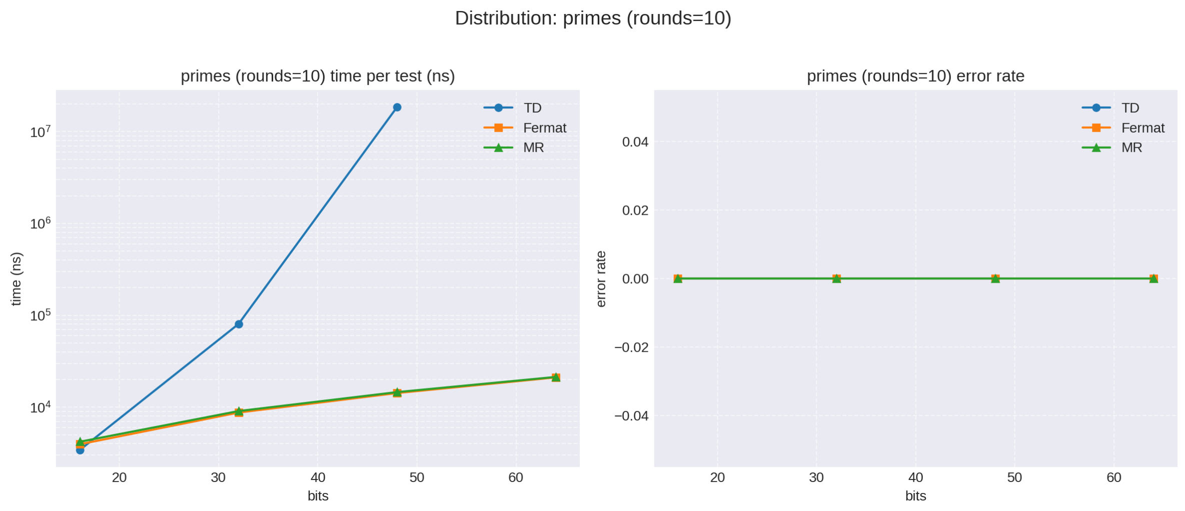

Figure 4: Prime inputs (ground truth), $k = 10$: time per test (ns) and false-negative rate.

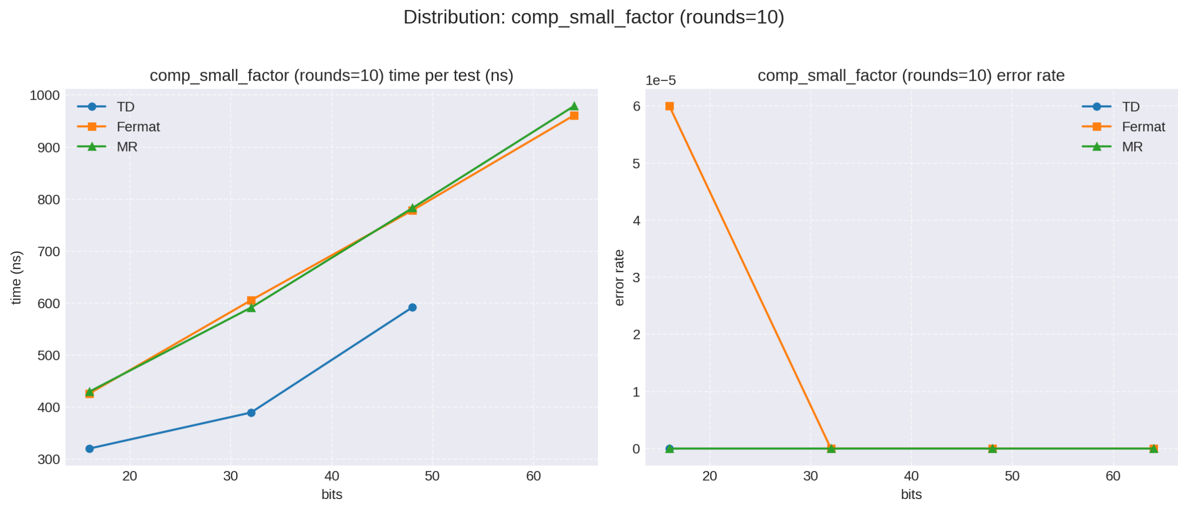

Figure 5: Composites with small factor, $k = 10$: time per test (ns) and error rate.

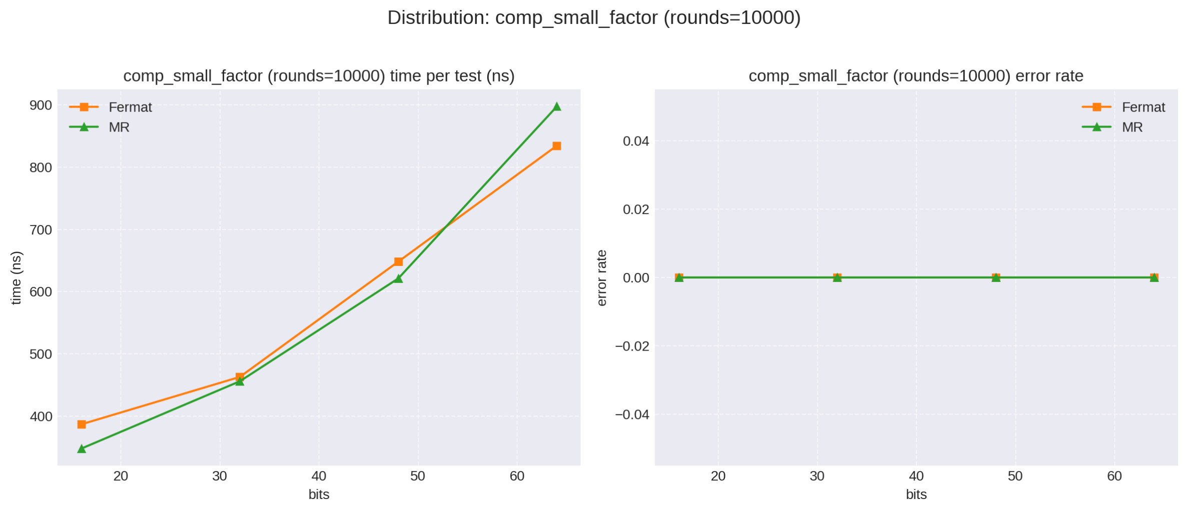

Figure 6: Composites with small factor, $k = 10000$: time per test (ns) and error rate.

5.2 Observations

- **Carmichael sensitivity**: Figure 1 shows MR attaining near-zero error on Carmichael inputs at $k = 10$; Figure 2 highlights the exponential decay, with error essentially gone by $k = 5$.
- **Random odd mix**: In Figure 3, Fermat and MR have similar runtimes; MR keeps error negligible, and Fermat's empirical error is extremely rare (about 4×10^{-5}).
- **Prime inputs**: Figure 4 shows no false negatives for MR (as guaranteed); Fermat also very rarely rejects primes.
- **Small-factor composites**: Figures 5 and 6 demonstrate that MR's error vanishes quickly with k; time scales linearly with k, matching the $O(k \log n)$ or $O(k \log^2 n)$ bounds depending on the cost model.

6 Real-world Usage

- **RSA key generation**: OpenSSL applies multiple MR rounds (and sieving) to candidate primes; documented error probabilities are bounded by powers of two. The toy RSA demo in this repository reuses the same MR code for small key generation.
- **Big-integer libraries**: GMP, Java `BigInteger.isProbablePrime`, and similar libraries rely on MR (often combined with Lucas/Baillie-PSW) to deliver negligible error on large inputs.
- **Computer algebra systems**: CAS tools (Maple, Mathematica, PARI/GP, SageMath) embed MR-style strong probable prime tests as core building blocks.
- **Toy RSA implementation**: A pedagogical RSA demo (local CLI and simple client/server) is included; it generates small primes using the MR routines documented here and walks through textbook RSA key generation, encryption, and decryption for illustration.

7 Conclusions

Miller–Rabin delivers logarithmic-time probabilistic primality testing with exponentially small error in the number of rounds. Deterministic base sets make it exact for 64-bit integers, and randomized MR dominates Fermat, especially on adversarial inputs like Carmichael numbers. Benchmarks confirm the theoretical bounds: time grows linearly with $k \log n$, and empirical error aligns with the $(1/4)^k$ guarantee.