Primality Testing Assignment Report

Aniket Basak (CrS2401)
Cryptographic and Security Implementation

August 2025

1. Miller–Rabin Primality Test

In the Miller–Rabin primality test, an integer parameter k is used to denote the number of independent rounds (with different random bases) of the test.

(a) Role of Parameter k:

Explain the role of k in reducing the error probability of the algorithm. Why is increasing k important for practical applications such as cryptography?

Answer: The Miller–Rabin test is a **probabilistic primality test**. For a given odd integer $n > 3$, we pick a random base a with $2 \leq a \leq n - 2$.

- The test checks whether a is a *witness* to the compositeness of n.
- If it finds a witness, n is declared **composite**.
- If not, n is declared **probably prime**.

However, because the test is probabilistic, it can make an error: a composite number may sometimes "pass" the test for some choices of a. Such numbers are called **strong pseudoprimes** to base a.

Role of k (Number of Rounds)

The parameter k specifies how many times we repeat the test with independent random bases $a_1, a_2, \ldots, a_k$. Each round independently reduces the chance that a composite number is incorrectly identified as prime.

Error Probability

For any odd composite n:

$$\Pr[\text{composite } n \text{ passes a single test}] \leq \frac{1}{4}.$$

Thus, the probability of error after k independent rounds is: $P_{\text{error}} \leq \left(\frac{1}{4}\right)^k$. Some concrete values are:

$$\begin{aligned} k = 1 \quad &\Rightarrow \quad P_{\text{error}} \leq \tfrac{1}{4} = 25\%, \\ k = 5 \quad &\Rightarrow \quad P_{\text{error}} \leq \tfrac{1}{1024} \approx 0.1\%, \\ k = 20 \quad &\Rightarrow \quad P_{\text{error}} \leq (1/4)^{20} \approx 10^{-12}. \end{aligned}$$

Why Increasing k is Important (Especially in Cryptography)

1. **Security requirement:** In cryptographic schemes such as RSA or Diffie–Hellman, primes are fundamental. If a composite is mistakenly accepted, the security of the entire system collapses, since factorization becomes trivial.

2. **Negligible error probability:** By choosing a sufficiently large k, we ensure that $P_{\text{error}} \ll 2^{-\lambda}$ where λ is the security parameter. For example, with $k = 40$: $P_{\text{error}} \leq 2^{-80}$, which is negligible compared to practical attack probabilities.

3. **Trade-off:** Increasing k increases computation linearly. But primality testing is much cheaper than cryptanalytic attacks, so a reasonably large k is easily affordable.

The parameter k controls the confidence level of the Miller–Rabin test. Each additional round reduces the probability that a composite number is mistakenly declared prime by a factor of 4. In cryptography, increasing k is critical to ensure the error probability is negligible, making it virtually impossible for weak (composite) numbers to be used as "primes" in secure key generation.

The parameter k denotes the number of independent rounds of the Miller–Rabin primality test. Each round chooses a random base a and checks whether n passes the strong probable prime test. If n is composite, then in at least 3/4 of bases, a is a *witness* to its compositeness. Thus, the probability that n passes one round is at most 1/4. Therefore, after k independent rounds, the error probability is:

$$\Pr[\text{composite passes all } k \text{ rounds}] \leq \left(\tfrac{1}{4}\right)^k$$

In cryptographic applications (RSA, Diffie–Hellman, ECC), primes must be certain. Even a small probability of error is unacceptable, so we increase k until the error probability becomes negligible (e.g., $< 2^{-80}$).

(b) Proof of Error Bound:

Show that if n is a composite number, the probability that it passes all k rounds of the Miller–Rabin test is at most $(\frac{1}{4})^k$.

Answer Let us repeat the test independently k times using k randomly chosen bases $a_1, a_2, \ldots, a_k$. The events are independent because each base is chosen randomly and independently. Therefore, the probability that n passes **all** k **rounds** is:

$$\Pr[\text{n passes all } k \text{ rounds}] = \Pr[\text{n passes round 1} \cap \text{round 2} \cap \cdots \cap \text{round k}].$$

By independence:

$$\Pr[\text{n passes all } k \text{ rounds}] = \prod_{i=1}^{k} \Pr[\text{n passes round i}] \leq \prod_{i=1}^{k} \frac{1}{4} = \left(\frac{1}{4}\right)^k.$$

Thus, if n is composite, the probability that it passes all k rounds of the Miller–Rabin test satisfies:

$$\boxed{\Pr[\text{composite } n \text{ passes all } k \text{ rounds}] \leq \left(\frac{1}{4}\right)^k}.$$

This shows that increasing k exponentially decreases the error probability.

(c) Value of k for 512-bit Candidates

For a **512-bit candidate prime**, calculate the minimum value of k required so that the probability of a composite number being falsely declared prime is below 2^{-80}.

Answer: We want the error probability below 2^{-80}: in other words $\left(\frac{1}{4}\right)^k < 2^{-80}$.
Taking logarithms on both sides yields: $-2k < -80 \quad \Longrightarrow \quad k > 40$.
Thus, the minimum value is $\mathbf{k = 41}$.

2. Experimental Verification

(a) Prime Generation

Two random 256-bit primes p and q were generated using a prime generation routine. Their product $n = pq$ is therefore a composite number of about 512 bits.
I implemented a C program using the GMP library to generate two 256-bit primes p and q using the Miller-Rabin test with $k = 41$ rounds, ensuring an error probability below 2^{-80}. Random candidates were generated, set to 256 bits, made odd, and tested for primality. Their product $n = pq$ is a 512-bit composite.

```
Generated 256-bit prime p:
102155177732347944725661660259905027558427513783027716268096742834085549523369

Generated 256-bit prime q:
74232884290951714480438305031688596732237723925958222061363346481617914560343

Composite n = p * q (≈512 bits):
75832734883269921327389820515696040639676291389882612333154972999605046097082232979017870082771583567097715352696031422874365096814906615426711088391555567
```

Figure 1: Snippets of generated primes p and q using Miller-Rabin and their product

(b) Implementation and Experiment

A Miller–Rabin test (single-round) was implemented. It was executed 20,000 times on 512 bit n, each time with a random base. The number of times n was incorrectly declared 'probably prime' or 'false positive' was recorded.
The code of core Miller-Rabin function is attached in the submitted zip file. In my case:

- Number of false positive: 0 out of 20,000.
- Experimental Error rate: 0.0000

```
Bit-length of n: 512 bits

Out of 20000 single-round tests on n: falsely declared 'probably prime' = 0 times
Experimental probability ≈ 0.000000

Prime generation time: 0.04 s  |  Testing time: 23.49 s  |  Total: 23.53 s
```

Figure 2: False Positive and runtime

(c) Comparison with Theory

The theoretical bound states that a composite passes a Miller-Rabin round with probability at most 0.25. For the 512-bit n, I observed 0 false positives in 20,000 trials (error rate 0.000000), which is well below 0.25, supporting the bound. For large semiprimes, the actual rate is typically much lower, often $< 10^{-6}$, due to the algebraic structure of $p - 1$ and $q - 1$. With 20,000 trials, the expected number of false positives is < 1, so a result of 0 is consistent.

3. Resources

To complete this assignment, I used the following resources:

- Sample Miller–Rabin code and explanations from xAI, which helped me understand the algorithm's structure and implementation in C using the GMP library.
- GMP library documentation for handling large integers and random number generation.
- Lecture notes on primality testing and the Miller–Rabin algorithm for theoretical insights.