Primality Testing Assignment: Miller-Rabin Test Analysis

CRS2404

August 25, 2025

1 Question 1: Theoretical Analysis

1.1 (a) Role of k in Error Reduction

The Miller-Rabin test determines if an odd integer $n > 2$ is prime by running k rounds, each using a random base a where $1 < a < n$ and $\gcd(a, n) = 1$. It expresses $n - 1 = 2^s \cdot d$ (with d odd) and checks if $a^d \equiv 1 \pmod{n}$ or $a^{2^r d} \equiv -1 \pmod{n}$ for some $0 \leq r < s$. If any round fails, n is composite; if all pass, it's declared "probably prime."

The parameter k represents the number of rounds. For a composite n, the probability of passing one round (a false positive) is at most $1/4$. Since each round uses a different random base, they are independent, so the probability of passing all k rounds is at most $(1/4)^k$. Increasing k reduces this error exponentially, which is crucial for cryptographic applications like RSA, where a composite mistaken for a prime could lead to a factorable key, breaking security. For 512-bit primes, we need an error below 2^{-80}, achievable with $k \approx 40$ or higher, balancing accuracy and computational efficiency (each round is $O(\log^3 n)$).

1.2 (b) Error Probability Bound

To prove that a composite n passes all k rounds with probability at most $(1/4)^k$:

For a single round:

- Write $n - 1 = 2^s \cdot d$, where d is odd.
- Choose a random a with $1 < a < n$, $\gcd(a, n) = 1$.
- Compute $x = a^d \mod n$.
- If $x \equiv 1 \pmod{n}$ or $x \equiv -1 \pmod{n}$, the round passes.
- For $r = 0$ to $s - 1$, compute $x = x^2 \mod n$; if $x \equiv -1 \pmod{n}$, the round passes.
- Otherwise, a is a strong witness, proving n is composite.

The Monier-Rabin theorem states that for an odd composite $n > 9$ (not a prime power), at least $3/4$ of valid bases a are strong witnesses, so the proportion of strong liars (bases that cause a pass) is at most $1/4$. Thus, the probability of passing one round is $\leq 1/4$. With k independent rounds, the probability of passing all is $\leq (1/4)^k$.

1.3 (c) Minimum k for 512-Bit Candidate

For a 512-bit number $n \approx 2^{512}$, we need the error probability $(1/4)^k < 2^{-80}$. Since $1/4 = 2^{-2}$, we have $(1/4)^k = (2^{-2})^k = 2^{-2k}$. Solving $2^{-2k} < 2^{-80}$: $-2k < -80 \implies k > 40$. The smallest integer k is 41, as $2^{-82} \approx 2.22 \times 10^{-25} < 2^{-80} \approx 8.27 \times 10^{-25}$, while $k = 40$ gives 2^{-80}, which is not strictly less.

2 Question 2: Implementation and Results

2.1 (a) Generating 256-Bit Primes and Computing $n = pq$

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 $n = p \cdot q$ is a 512-bit composite. All results were saved to `resultsmiller.txt`. A snippet from the file shows:

```
Generating two 256-bit primes...
p: 68379896888971817223699176392041404787140365021717644680019912736661821788869
q: 62459192574650402322084353628893594209572118695062216220475135974718426342427
n: 427095314802302867255423283347651672622005251197500620957784527782659777276 63.....
```

The full numbers are available in resultsmiller.txt, but they are shortened here.

2.2 (b) Miller-Rabin Implementation and Testing

The program implements the Miller-Rabin test and runs it 1,000,000 times on the 512-bit n, counting false positives (when n is declared 'probably prime'). To verify the implementation, I tested $n = 221 = 13 \cdot 17$, a semiprime, over 100 trials.

Results from `resultsmiller.txt` (after correction):

- For $n = 221$: 4 false positives out of 100 trials (error rate 0.04).
- For the 512-bit n:

```
Number of false positives: 0 out of 1000000
Experimental error rate: 0.000000
```

The core Miller-Rabin function is shown below:

Listing 1: Core Miller-Rabin Function

```c
int miller_rabin_single(const mpz_t n, const mpz_t a, int debug, FILE *fp) {
    mpz_t nm1, d, x;
    mpz_init(nm1); mpz_init(d); mpz_init(x);
    mpz_sub_ui(nm1, n, 1); mpz_set(d, nm1);
    unsigned long s = 0;
    while (mpz_even_p(d)) { mpz_divexact_ui(d, d, 2); s++; }
    mpz_powm(x, a, d, n);
    if (debug) { /* Print a, s, d, x */ }
    if (mpz_cmp_ui(x, 1) == 0 || mpz_cmp(x, nm1) == 0) { /* Handle pass */ }
    for (unsigned long r = 0; r < s; r++) {
        if (r > 0 || s == 1) { mpz_powm_ui(x, x, 2, n); }
        if (debug) { /* Print r, x */ }
        if (mpz_cmp(x, nm1) == 0) { /* Handle pass */ }
    }
    if (debug) { /* Print fail */ }
    mpz_clear(nm1); mpz_clear(d); mpz_clear(x);
    return 0;
}
```

2.3 (c) Comparison with Theoretical Bound

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 1,000,000 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 1,000,000 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.