Zero Knowledge Proofs for Various Primality Tests

Dr.Pratik Soni, Raghava, Mahanthi

Contents

1 Introduction

This document explores various primality testing methods in the context of Zero Knowledge Proofs (ZKPs). Each method is described in detail, including its algorithm, pseudocode, advantages, disadvantages, and its behavior for large numbers. The aim is to understand how these tests can be used for cryptographic applications and their suitability for ZKP systems.

2 Fermat Primality Test

Algorithm Explanation

The Fermat Primality Test is a simple and efficient probabilistic method for checking if a number n is likely to be prime. It is based on **Fermat's Little Theorem**, which states:

$$\text{If } n \text{ is prime and } a \text{ is an integer such that } 1 \leq a < n, \text{ then:}$$
$$a^{n-1} \equiv 1 \pmod{n}.$$

The algorithm randomly selects integers a (called bases) within the range $[1, n-1]$ and checks whether the above condition holds. If it fails for any base a, n is composite. If it holds for several bases, n is declared as *probably prime*.

Pseudocode

Fermat Primality Test

```
1.  If $n \leq 1$, return "Composite"
2.  For $i = 1$ to $k$ do:
     a.  Randomly choose an integer $a$ such that $1 \leq a < n$
     b.  Compute $x = a^{n-1} \bmod n$
     c.  If $x \neq 1$, return "Composite"
3.  If all tests passed, return "Probably Prime"
```

Advantages

- **Efficiency**:
  - Computationally efficient with complexity $O(k \cdot \log(n)^2)$, where k is the number of iterations and $\log(n)$ is the cost of modular exponentiation.
  - Suitable for small to medium-sized numbers.
- **Simplicity**: Easy to implement with minimal additional mathematical knowledge.
- **Scalability**: Works well with modern hardware for moderately large numbers.

Disadvantages

- **Pseudoprimes**: Some composite numbers (called *Fermat pseudoprimes*) pass the test for specific bases a.
- **Base Dependence**: The result depends on the choice of base a; poor selection of bases can lead to incorrect results.
- **No Certificate of Primality**: The test does not provide a deterministic proof or certificate of primality.

Advantages Over Trial Division

The Fermat test outperforms trial division for numbers larger than 10^6. While trial division requires $O(\sqrt{n})$ checks for divisibility, Fermat operates with $O(k \cdot \log(n)^2)$ complexity, making it faster for large inputs.

Behavior for Large Numbers

- Modular exponentiation ensures efficient computation for large numbers.
- The probability of declaring a composite number as "Probably Prime" decreases exponentially with the number of iterations k.
- For k iterations, the error rate is at most $1/2^k$, assuming bases are chosen uniformly at random.

Expected Error

- **Error Rate**: At most $1/2^k$, where k is the number of iterations.
- **Worst-Case Error**: For Carmichael numbers (a special class of pseudoprimes), all bases a with $\gcd(a, n) = 1$ incorrectly pass the test.

Example

Test $n = 561$ with bases $a = 2$ and $a = 3$.

- Compute $2^{560} \mod 561$:
$$2^{560} \equiv 1 \pmod{561}$$
- Compute $3^{560} \mod 561$:
$$3^{560} \equiv 1 \pmod{561}$$

Despite 561 being composite (it is the smallest Carmichael number), it passes the Fermat test for bases $a = 2$ and $a = 3$, showcasing the limitation of the test.

Conclusion

The Fermat Primality Test is a fast and simple probabilistic test for identifying prime numbers. However, its susceptibility to pseudoprimes makes it less reliable for applications requiring absolute certainty, such as cryptography.

3 Lehmann Primality Test

Algorithm Explanation

The Lehmann Primality Test builds upon Fermat's Little Theorem but includes an additional condition to address some of the shortcomings of the Fermat test, such as its susceptibility to pseudoprimes.

For a number n to be prime, and a chosen base a, the following must hold:

1. $a^{n-1} \equiv 1 \pmod{n}$ 2. $a^{(n-1)/2} \equiv \pm 1 \pmod{n}$

If $a^{(n-1)/2} \not\equiv \pm 1 \pmod{n}$, then n is composite. By testing with multiple random bases, the probability of error can be reduced significantly.

Pseudocode

Lehmann Primality Test

```
1.  If $n \leq 1$, return "Composite"
2.  For $i = 1$ to $k$ do:
     a.  Randomly choose an integer $a$ such that $1 \leq a < n$
     b.  Compute $x = a^{(n-1)/2} \mod n$
     c.  If $x \neq 1$ and $x \neq n-1$, return "Composite"
3.  If all tests passed, return "Probably Prime"
```

Advantages

- **Increased Accuracy**: By adding the second condition $a^{(n-1)/2} \equiv \pm 1 \pmod{n}$, the Lehmann test catches more composites compared to the Fermat test.
- **Efficiency**: Still retains a time complexity of $O(k \cdot \log(n)^2)$, where k is the number of iterations.
- **Simplicity**: Easy to implement and provides more reliable results than Fermat.

Disadvantages

- **Probabilistic Nature**: Like Fermat, the Lehmann test is probabilistic and cannot guarantee primality.
- **Dependency on Randomness**: Its accuracy depends on the quality of the random number generator for selecting bases.
- **No Certificate of Primality**: The Lehmann test does not produce a verifiable certificate of primality.

Advantages Over Fermat Test

- By checking $a^{(n-1)/2} \equiv \pm 1 \pmod{n}$, the Lehmann test detects pseudoprimes that would otherwise pass the Fermat test.
- Significantly reduces the error rate compared to Fermat.

Behavior for Large Numbers

- Handles large numbers efficiently using modular exponentiation for calculations.
- For k iterations, the probability of a composite number passing as ”Probably Prime” is at most $1/2^k$, assuming uniformly random bases.

Expected Error

- **Error Rate**: At most $1/2^k$, where k is the number of iterations.
- **Worst-Case Error**: The Lehmann test is still susceptible to Carmichael numbers, but the second condition reduces the chances of error compared to Fermat.

Example

Test $n = 341$ with bases $a = 2$ and $a = 3$.

- Compute $2^{(341-1)/2} \mod 341$:
$$2^{170} \equiv 1 \pmod{341}$$
- Compute $3^{(341-1)/2} \mod 341$:
$$3^{170} \equiv 340 \pmod{341} \quad \text{(which is } n-1\text{)}$$

Since both 2^{170} and 3^{170} satisfy the Lehmann conditions, $n = 341$ is declared ”Probably Prime,” even though n is composite (it is a pseudoprime).

Conclusion

The Lehmann Primality Test improves upon Fermat by adding an additional condition to reduce the likelihood of errors. While still probabilistic, it is more reliable than Fermat and is well-suited for applications requiring quick primality checks for moderately large numbers.

4 Rabin-Miller Primality Test

Algorithm Explanation

The Rabin-Miller Primality Test is a probabilistic algorithm widely used to test primality. It improves on the Lehmann test by using the concept of nontrivial square roots of 1 modulo n. The test is based on the following properties:

1. Decompose $n-1$ as $2^s \cdot d$, where d is odd. 2. A number n is composite if for a randomly chosen base a: - $a^d \not\equiv 1 \pmod{n}$, and - $a^{2^r \cdot d} \not\equiv -1 \pmod{n}$ for all r in $[0, s-1]$.

If n passes the above conditions for several random bases, it is declared as *probably prime.*

Pseudocode

Rabin-Miller Primality Test

```
        a.  Randomly choose an integer $a$ such that $1 \leq a < n$
        b.  Compute $x = a^d \mod n$
        c.  If $x = 1$ or $x = n-1$, continue to the next iteration
        d.  Repeat $s-1$ times:
           i.  Compute $x = x^2 \mod n$
           ii.  If $x = n-1$, break
        e.  If $x \neq n-1$, return "Composite"
    4.  If all tests passed, return "Probably Prime"
```

Advantages

- **High Accuracy**: The test detects most composites that would pass Fermat or Lehmann tests.
- **Efficiency**: Modular exponentiation ensures the test is computationally efficient, with complexity $O(k \cdot \log(n)^2)$.
- **Widely Used**: A practical and reliable algorithm for many cryptographic applications.

Disadvantages

- **Probabilistic Nature**: Like Fermat and Lehmann, Rabin-Miller cannot provide a deterministic proof of primality unless all possible bases are tested.
- **Computational Cost**: Slightly more computationally intensive than Fermat or Lehmann due to the additional checks.
- **Carmichael Numbers**: It can still fail for carefully constructed pseudoprimes, though the likelihood is extremely low.

Advantages Over Lehmann Test

- Incorporates multiple rounds of squaring, reducing the chance of error compared to the Lehmann test.
- Detects nontrivial square roots of 1 modulo n, which many pseudoprimes would otherwise pass in Lehmann.

Behavior for Large Numbers

- Efficiently handles large numbers using modular exponentiation for both the initial test and subsequent rounds.
- For k iterations, the probability of error decreases exponentially, making the test suitable for cryptographic applications.

Expected Error

- **Error Rate**: At most $1/4^k$, where k is the number of iterations.
- **Worst-Case Error**: Although extremely unlikely, certain pseudoprimes may pass as "Probably Prime."

Example

Test $n = 561$ with $n - 1 = 560 = 2^4 \cdot 35$ and base $a = 2$.

- Compute $x = 2^{35} \mod 561$:
$$x \equiv 263 \pmod{561}$$
- Square x four times (since $s = 4$):
$$x^2 \mod 561 = 166, \quad x^4 \mod 561 = 67, \quad x^8 \mod 561 = 1$$
- Since $x^{2^r \cdot d} \not\equiv -1 \pmod{561}$ for any r, n is composite.

Conclusion

The Rabin-Miller Primality Test is a powerful probabilistic method for primality testing. Its ability to detect nontrivial square roots of 1 modulo n makes it far more reliable than Fermat and Lehmann tests. While it remains probabilistic, its high accuracy and efficiency make it the preferred choice for practical primality testing, especially in cryptographic applications.

5 Pratt Primality Certificate

Algorithm Explanation

The Pratt Primality Certificate is a deterministic algorithm that provides a verifiable certificate of primality. It is based on Lucas's theorem, which states:

A number n is prime if there exists a base a such that:

1. $a^{n-1} \equiv 1 \pmod{n}$,
2. $a^{(n-1)/p} \not\equiv 1 \pmod{n}$ for all prime divisors p of $n-1$.

The Pratt certificate works recursively: to prove that n is prime, we must prove that all prime factors of $n-1$ are also prime. This recursion ends at small primes that are trivially verifiable.

Pseudocode

Pratt Primality Certificate

```
1.  If $n \leq 1$, return "Not Prime"
2.  Compute $n-1$ and find its prime factorization:  $n-1 = \prod p_i^{e_i}$
3.  Choose a base $a$ such that $1 < a < n$ and compute:
     a.  $a^{n-1} \bmod n$
     b.  $a^{(n-1)/p_i} \bmod n$ for all prime factors $p_i$
4.  If $a^{n-1} \not\equiv 1 \pmod{n}$ or any $a^{(n-1)/p_i} \equiv 1 \pmod{n}$, return "Composite"
5.  Recursively verify the primality of each $p_i$
6.  Return the certificate:  $(n, a, \{(p_i, e_i)\}, \{p_i$'s certificates$\})$
```

Advantages

- **Deterministic Proof**: Provides a verifiable certificate of primality.
- **Recursive Verification**: Ensures complete rigor by verifying all factors of $n-1$.
- **Polynomial-Time Verification**: Once generated, the certificate can be verified in $O(\log(n)^2)$.

Disadvantages

- **Computationally Intensive**: Factorizing $n-1$ is expensive, especially for large n.
- **Recursive Complexity**: The recursion grows with the number of prime factors in $n-1$, increasing the computation time.

Advantages Over Rabin-Miller Test

- Unlike Rabin-Miller, Pratt certificates are *deterministic* and provide an absolute proof of primality.
- Certificates can be stored and reused, making verification efficient and independent of the original computation.

Behavior for Large Numbers

- The method remains deterministic but becomes computationally expensive for large primes due to the factorization of $n-1$.
- Large numbers with smooth $n-1$ (i.e., small prime factors) are easier to handle.
- For cryptographic applications, this method is often impractical due to its recursive nature.

Expected Error

- **Error Rate**: None; Pratt provides a deterministic proof.
- **Worst-Case Complexity**: Depends on the size and factorization of $n-1$.

Example

Let $n = 23$. Then $n-1 = 22 = 2 \times 11$.

1. Choose $a = 2$. Compute:
$$2^{22} \mod 23 \equiv 1 \pmod{23}$$
and
$$2^{22/2} = 2^{11} \mod 23 \not\equiv 1 \pmod{23}$$
$$2^{22/11} = 2^{2} \mod 23 \not\equiv 1 \pmod{23}.$$
2. Verify $p_1 = 2$ and $p_2 = 11$. Both are prime.
3. Certificate for $n = 23$:
$$(23, 2, \{(2,1), (11,1)\}, \{\text{Certificate for } 2, \text{Certificate for } 11\}).$$

Conclusion

The Pratt Primality Certificate is a deterministic method that rigorously proves the primality of a number. While computationally expensive, its recursive structure provides verifiable proof, making it a useful theoretical tool for understanding primality in cryptography and mathematics.

6 Elliptic Curve Primality Proving (ECPP)

Algorithm Explanation

The Elliptic Curve Primality Proving (ECPP) algorithm is a deterministic method for proving the primality of a number n. It is based on the properties of elliptic curves over finite fields. The core idea is to construct an elliptic curve E and a point P on it, such that the order of P satisfies certain divisibility properties, proving the primality of n.

Key steps: 1. Choose an elliptic curve $E : y^2 = x^3 + ax + b \pmod{n}$. 2. Select a point $P = (x, y)$ on the curve. 3. Compute the order of P, denoted m, using $mP \equiv 0 \pmod{n}$. 4. Verify that the order $m = n - 1$ and satisfies the necessary divisibility conditions.

Pseudocode

Elliptic Curve Primality Proving (ECPP)

```
1.  If $n \leq 1$, return "Not Prime"
2.  Choose an elliptic curve $E : y^2 = x^3 + ax + b \pmod{n}$.
3.  Select a point $P = (x, y)$ on $E$.
4.  Compute the order $m$ of $P$:
      a.  Use scalar multiplication to compute $mP \bmod n$.
5.  Verify that $m = n - 1$:
      a.  Ensure $mP \equiv 0 \pmod{n}$.
6.  Check divisibility conditions:
      a.  $m = n - 1$ is divisible by small primes $p_i$.
7.  Recursively verify the primality of $p_i$.
8.  Return the certificate: $(n, E, P, m,$ certificates for $p_i)$.
```

Advantages

- **Deterministic Proof**: Provides rigorous proof of primality.
- **Polynomial-Time Verification**: The certificate can be verified in polynomial time.
- **Efficient for Large Primes**: ECPP is highly optimized for proving the primality of very large numbers.

Disadvantages

- **Complexity in Setup**: Requires careful selection of elliptic curves and points.
- **Computational Intensity**: The algorithm is computationally expensive due to the order computation and recursive primality checks.
- **Dependence on Elliptic Curve Theory**: Requires advanced mathematical knowledge for implementation.

Advantages Over Pratt and Rabin-Miller Tests

- Unlike Pratt, ECPP does not rely on the smoothness of $n - 1$ and is more practical for general numbers.
- ECPP provides a deterministic proof, unlike Rabin-Miller, which is probabilistic.
- For very large primes, ECPP is faster and more scalable than Pratt certificates.

Behavior for Large Numbers

- The algorithm efficiently handles primes with hundreds of digits.
- Modular arithmetic and elliptic curve operations are optimized for large numbers.
- Recursive primality checks ensure the robustness of the proof.

Expected Error

- **Error Rate**: None; ECPP provides deterministic proofs.
- **Worst-Case Complexity**: Depends on the selection of elliptic curves and the recursive depth.

Example

Let $n = 11$. To prove 11 is prime using ECPP:

1. Choose the elliptic curve $E : y^2 = x^3 + x + 1 \pmod{11}$.
2. Select a point $P = (2, 7)$ on E.
3. Compute the order m of P:
$$m = 10 \quad (\text{since } 10P \equiv 0 \pmod{11}).$$
4. Verify that $m = n - 1$ and m satisfies divisibility conditions.
5. Recursive verification:
   - Verify p_i (prime factors of m).
   - Return the certificate:
$$(11, E, P, 10, \text{certificates for } p_i).$$

Conclusion

ECPP is a state-of-the-art primality proving algorithm that combines the power of elliptic curves and modular arithmetic. Its scalability and rigorous proof make it ideal for cryptographic applications requiring absolute certainty of primality.

7 Circom Utility Functions and Optimizations

To implement various primality tests in Circom, we relied on essential utility functions and employed specific optimizations to overcome limitations and challenges. This section elaborates on these components and their significance in the circuits.

Utility Functions

Num2Bits

Purpose: Converts an integer into its binary representation. This function was critical for:

- Efficiently performing modular exponentiation using repeated squaring.
- Ensuring that the input values conform to binary constraints, reducing errors in circuit behavior.

Role in Primality Tests:

- In modular exponentiation, the binary representation of the exponent enabled efficient computation.
- Played a central role in breaking down large numbers into manageable binary operations for verification purposes.

ModularExponentiation

Purpose: Computes modular exponentiation $\text{base}^{\text{exp}} \mod \text{mod}$. This is the core arithmetic operation used in all primality tests, including Fermat, Lehmann, Pratt, and ECPP.

Advantages:

- Provides an efficient way to handle large inputs, essential for primality testing of large numbers.
- Reduces computational overhead by using the repeated squaring method, which has a time complexity of $O(\log(\text{exp}))$.

Tricks and Optimizations in Pratt and ECPP

Static Sizes

Circom does not support dynamic array sizes, so we hardcoded sizes for arrays representing factors, primes, and elliptic curve parameters.

Reason for Static Sizes:

- To simplify implementation and testing on constrained systems.
- Ensure compatibility with Circom's strict constraints on signal declarations.

Implications:

- In Pratt, the number of factors per prime (`maxFactors`) was limited to 5.
- In ECPP, the number of primes (`maxPrimes`) and factors per prime were fixed to 5.

Challenges:

- Larger inputs require increasing the static sizes, which significantly increases the number of constraints and computational complexity.
- Testing larger certificates is challenging on limited hardware due to the exponential growth of constraint size.

Extra Input Columns

In circuits like Pratt and ECPP, additional input columns were introduced to simplify checks and reduce redundancy.

Added Inputs in Pratt:

- **Factors**: Precomputed prime factors of $n-1$.
- **Exponents**: Corresponding exponents $\frac{n-1}{\text{factor}_i}$ for modular checks.

Added Inputs in ECPP:

- Elliptic curve parameters a and b.
- Coordinates (x, y) of a point on the curve.
- Order m of the point, precomputed for efficiency.

Benefits:

- Simplified verification by offloading complex computations to preprocessing.
- Reduced the overall number of constraints in the circuit by leveraging precomputed values.

Non-Zero Constraints

Dynamic conditions, such as if factor $\neq 0$, are not directly supported in Circom. To handle this, additional signals were introduced to enforce binary constraints.

Why This Was Necessary:

- To ensure that unused slots in arrays (e.g., unused factors) do not affect the computation.
- To meet Circom's requirement of explicitly defining all signal constraints.

Testing and Practical Considerations

Testing on Smaller Inputs

Due to system constraints, all circuits were tested on small primes and elliptic curve parameters. For example:

- In Pratt, $n - 1$ was factored into at most 5 primes.
- In ECPP, elliptic curve parameters were selected for small primes (e.g., $p = 23$, $p = 29$).

Limitations:

- Larger primes require higher values for `maxFactors` and `maxPrimes`, which increases computational cost.
- Proof generation time grows significantly with larger certificates due to the increased number of constraints.

Proof Generation Complexity

- Increasing `maxFactors` and `maxPrimes` results in quadratic growth in constraint size.
- For ECPP, elliptic curve validation adds additional modular operations, further increasing computational cost.

Key Learnings and Future Improvements

- Explore dynamic sizing techniques in Circom to handle varying numbers of factors and primes.
- Optimize modular arithmetic and elliptic curve operations for larger inputs.
- Parallelize circuits to leverage multicore hardware for constraint computation.

Conclusion

The Circom implementations for Pratt and ECPP are functional for small inputs but require significant optimization for large-scale applications. These circuits provide a foundation for further research and development in cryptographic primality testing. Future contributions are encouraged to improve efficiency, scalability, and compatibility for real-world use cases.