ISSN NO: 2770-4491
Date of Publication:24-03-2023

A method for improving Milner-Rabin algorithm to reduce the number of false witnesses

Oydin Ahmedova Pulatovna
Research department of information security and cryptology
State Unitary Enterprise Unicon.uz
Tashkent, Uzbekistan
o.ahmedova@unicon.uz
Ulugbek Mardiyev Rasulovich
Cryptology department
TUIT named after Muhammad al-Khwarizmi
Tashkent, Uzbekistan
twofine@mail.ru

Abstract— nowadays, along with the increase in the volume of data, ensuring its protection is one of the important issues. The best way to ensure data confidentiality is the cryptographic method. One of the main parameters of public-key cryptoalgorithms is a prime number. This paper presents an analysis of probabilistic algorithms to check the primality number and improves the Rabin Milner algorithm to reduce the number of false witnesses.

Keywords — Miller-Rabin test, prime number, recursive function, deterministic algorithms, probabilistic algorithms

Introduction

There are various ways or algorithms to check if a number is a prime number. Some algorithms are specific to numbers with certain properties or structures, while others can be applied to any number. The algorithms that work for any number are highly useful both in theory and in practice. Generally, all primality-testing algorithms can be grouped into three major categories [1]:

- One-way error probabilistic algorithms;
- probabilistic runtime algorithms;
- deterministic algorithms.

Although exact tests return, a definite answer about whether a number is prime or not, these types of tests are rarely used in practice because they are slow because of the complexity of the algorithm.

Probabilistic tests - the result of this test is true with a fairly high probability. Repeating them multiple times with different parameters for the same number makes the probability of error quite small.

The methods used to determine whether a number is prime or not can be categorized into different classes based on their execution complexity:

- An algorithm is called continuous if its complexity value does not depend on the size of the initial value, i.e., O(1);
- An algorithm is called linear if its order of complexity is $O(n)$;
- Exponential rank algorithms - an estimate of the complexity level $O(c\hat{}(log\ n))$ for some constant $c > 1$;
- Subexponential level algorithms - complexity level estimate $O(c^{(\log n)^{\gamma}(\log\log n)^{1-\gamma}})$ for any constant $c > 1$ and $0 < \gamma < 1$;
- Polynomial algorithms - complexity estimate $O(log^{c}n)$ for some constantc≥1.

Probabilistic algorithms with one-sided error **-**Historically, the first one-way error probability algorithms were algorithms with polynomial execution complexity.

Algorithms whose execution time is probabilistic - The next important step in the development of primality testing is related to the emergence of algorithms whose execution time is a polynomial time probability. Algorithms belonging to this class are elliptic curve tests (ECC). Algorithms of this type have very high polynomial complexity, due to which they are not used in practice.

Deterministic algorithms - Deterministic algorithms for checking integers for prime numbers have been around for more than two thousand years. One of the earliest algorithms we know is Eratosthenes' algorithm, which determines a prime number by dividing it by the prime numbers preceding it. In general, it is enough to check that $P_i \leq \lfloor\sqrt{n}\rfloor$ is divisible by all prime numbers. This algorithm is also called the trial division algorithm.

Algorithms of this type make an explicit decision as to whether the incoming value is prime or complex. However, such algorithms are impractical because they require large computational registers for very large numbers.

In practical applications, tests that have a higher degree of polynomial complexity are employed.

Main Part

A. *Fermat primality test*

The tiny Fermat theorem-based algorithm is the first of these class of algorithms. This test procedure is based on the theorem proposed by renowned French mathematician Pierre Fermat, known as the "little ferme theorem" in the 17^{th} century [2].

If n is a prime, then, according to Fermat's small theorem, the equation $a^{n-1} \equiv 1\ (mod\ n)$ holds, where a is arbitrary and n is not divisible by a. The fulfilment of the equation $a^{n-1} \equiv 1\ (mod\ n)$ is a necessary and sufficient condition for determining the primality of a given number n. That is, if $a^{n-1} \neq 1\ (mod\ n)$ for any a, then n is a complex number, otherwise, it is difficult to say anything definite, but the probability of the number increases. If the comparison $a^{n-1} \equiv 1\ (mod\ n)$ is performed for a complex number n, then the number n is called pseudo-prime based on a.

However, when the number n is complex, a is found such that the comparison $a^{n-1} \equiv 1\ (mod\ n)$ is not performed, such a number a is called evidence of the complexity of the number n, and the previous number a, which made the comparison, is called a "false witness" of primality n.

Consequently, when testing a number for prime according to the Ferm theorem, the number a is chosen. $a^{n-1} \equiv 1\ (mod\ n)$ the larger the number a that satisfies the condition, the more likely that the number n is prime. But there are complex numbers n such that for $a^{n-1} \equiv 1\ (mod\ n)$ the comparison holds in an arbitrary number a, which is prime with n. Such numbers are called Carmichael numbers. The set of Carmichael numbers is an infinite set, the smallest of which is $n = 561 = 3 \cdot 11 \cdot 17$. In spite of this, the Ferma test is effective in determining composite numbers [3]. The Time complexity of this test is $O(k\ logn)^3$.

```
Input: An odd integer number $n \geq 3$ and hidden parameter $t \geq l$;
Output: "is $n$ prime?" the answer shoul be "prime" or "complex".
  1. for $i = 1$ to $t$ do the following:
    1.1. randomly selected the number $a$, satisfying condition $2 \leq a \leq n - 2$;
    1.2. Calculated $r = a^{n-1}$;
    1.3. If $(r = 1)$, then returns a "complex number";
Returns "prime number" [4].
```

Fermat's theorem suggests that, when checking for primality, several numbers of a should be selected. The more numbers of a that satisfy the condition $a^{n-1} \equiv 1\ (mod\ n)$, the higher the likelihood that n is a prime number. However, there are n complex numbers for which the comparison $a^{n-1} \equiv 1\ (mod\ n)$ is performed for any prime number a that is coprime with n. These numbers are known as Carmichael numbers. The Carmichael number set is an infinite set, and the smallest known Carmichael number is $n = 561 = 3 \cdot 11 \cdot 17$. Nevertheless, the Fermat test remains an effective way of determining primes.

ISSN NO: 2770-4491
Date of Publication:24-03-2023

B. *Solovay–Strassen primality test*

Another primality-testing algorithm in this category is the Solovay-Strassen test, which always correctly detects prime numbers but may give a wrong answer with a certain probability for composite numbers. The primary advantage of this test is that it can identify Carmichael numbers as composite, which the Fermat test cannot do.

The essence of the test is to test not every number in the entire sequence, but a random set of each random number for k times. This algorithm based on Fermat's little theorem.

If p–simple number, and $a - a_n$ integer that is mutually prime with n, so: $a^{n-1} \equiv 1(mod\ n)$.

In this case, the Jacobi symbol is used to define the Carmichael numbers.

$$\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}}(mod\ n)$$

where $(\frac{a}{n})$–Jacobi symbol, called witness of primality n [4].

This test uses the Euler criterion. It is known that according to the Euler criterion, if $a^{(n-1)/2} = (\frac{a}{n})\ mod\ n$ condition is satisfied for all witnesses of prime numbers a that do not have the greatest common divisor with n, then n is an odd number of prime numbers.

That is, this algorithm focuses on the elements 1 and -1, which are formed in the column corresponding to the "prime witnesses" a, which are mutually simple with n in the $(n-1)/2$ row of the table of levels of prime numbers [5].

If at the end of the test witnesses, the prime number n has been discovered as much as iterations k, the number n is probably simple, with a probability $1 - 2^{-k}$. Odd n satisfying the test condition and not being prime is called pseudo-simple Euler numbers at the base a [6c]. The complexity value of this test is $O(log^3\text{p})$.

The Fermat and Solovay-Strassen tests rely on translating a congruence modulus of primes, or Fermat's little theorem, or Euler's congruence into a set of composite numbers and hoping it will fail there.

C. *Miller-Rabin primality test*

The Rabin-Miller test is one of the probabilistic primality tests based on the strong concept of pseudoprimality.

The test algorithm developed by Michael Rabin, based in part on the ideas of Jerry Miller, is now widely used in the design of public key cryptosystems. This algorithm is recognized as a powerful algorithm for testing pseudo prime numbers. Miller-Rabin is a polynomial-time algorithm with a time complexity of $O(k\ logn)^3$.

As in the Fermat and Solovay–Strassen tests, we are using the term “witness” to mean a number that proves n is composite. An odd prime has no Miller–Rabin witnesses, so when n has a Miller–Rabin witness it must be composite [7].

It is based on the representation of $p - 1$in the representation $2^s * r$. Where s is the number of divisions of $p - 1$ by two, r is an odd number. The Rabin-Miller algorithm is based on the following definition [8].

Definition. Suppose p is an odd composite integer, and $p - 1 = 2^s * r$, where r is an odd number. If $a^r \neq 1\ (mod\ p)$ and all $j, 0 \leq j \leq s - 1$, $a^{2^j r} \neq -1$, then for p, a is called a "strong witness" (composite). Otherwise, if $a^r = 1\ (mod\ p)$ or for all $0 \leq j \leq s - 1$, $a^{2^j r} \neq -1$, then p, a is called the number of the "strong pseudo-prime" according to the basis a. An integer a is called a "strong liarwitness" number for p.

The probability of a test error does not exceed 2^{-2} for a single value of x, and if the test is repeated k times for different values of x, the probability of error decreases to 2^{-2k}. Numbers that satisfy this condition but are not prime are called x-based strong pseudoprime numbers. The complexity of this test is $O(log^3 p)$. In the figure below, the prime number generation time represents the dependence of the number of bits.

D. *Lucas primality test*

The Lucas test was developed on the Lucas side in 1891, an algorithm for determining the numbers (not just the Mersen and Farm numbers) for the primality, based on the probability used to determine the primality.

According to the algorithm, an input value n is called prime if for any prime q, which is a divisor of $n-1$, there exists a such that $a^{n-1} \equiv 1\ (mod\ n)$ and $a^{(n-1)/q} \neq 1\ (mod\ n)$, if the conditions are satisfied. This algorithm requires that the prime divisors of n-1 are known. At present, there is no known complex number that does not pass a certain number of Rabin-Miller and Lucas tests [9].

E. *Proth primality test*

Proth theorem is a probabilistic algorithm that is used to test numbers for primality of a certain kind. This test usually tests numbers of the form $k * 2^n + 1$, where k is an odd integer such that $k < 2^n$. A number p is called a prime number if the condition $a^{(p-1)/2} \equiv -1\ (mod\ p)$ holds for such an integer a. Prime numbers of this type are called Prot prime numbers. Prot's theorem quickly determines whether a number is prime or not. However, it is very slow in determining whether a given number is composite(it is necessary to check every number from 2 to n). This algorithm is recommended for finding prime numbers within a certain range [10]. The time complexity of Proth test is $O((k\ logk\ +\ logn)\ logn)$.

F. *Pocklington primality test*

The Pocklington test, developed by Pocklington and Lammer, determines whether an incoming prime number can be identified. An input number N is prime if, for any prime number q, which is a divisor of number N-1, there exists an integer a such that $a^{N-1} \equiv 1\ (mod\ N)$ and $\gcd\left(a^{(N-1)/q} - 1, N\right) = 1$. This algorithm requires that the prime divisors of N-1 are known [11].

Table 1. Below is an analysis of probabilistic algorithms for checking numbers for primality.

Name of test	Advantage	Disadvantage	Complexity
Fermat	- Very simple to implement. - Base for many tests.	-Failure probability may reach 1. -Pseudoprime can pass the test.	$O\ (k \log n)^3$
Solovay Strassen	Pseudoprimes are successfully announced as composites.	- an Euler pseudoprime can pass the test. - Computation of Jacobi symbol adds more computation overhead.	$O\ (k \log n)^3$
Miller-Rabin	- Fast & efficient. - Euler Pseudoprimes are successfully announced as composites.	Strong pseudoprimes can pass the test.	$O\ (k \log n)^3$
Pocklington	Very efficient if there is a factor $q > \sqrt{n-1}$	Prime factors of $n-1$are required to be already known.	$O\ (lnln\ n)$
Lucas	Valid for any generic or special form numbers.	- Prime factors of $n-1$are required to be already known. - Worst case scenario may take long time. (if n is composite, this test may not terminate).	$O\ (p^2\ \log_p n)$
Proth	Very fast and reliable test to decide about proth number.	Working well only with proth numbers.	$O\ ((k \log k + \log n) \log n$

G. *Releated Work*

The authors of [12] conducted an analysis of the Miller-Rabin and Solovay-Strassen tests, which are based on probabilistic testing. They concluded that the Miller-Rabin test is highly effective. This analysis was carried out using a mathematical model.

In reference [13], the authors conducted an analysis of the Miller-Rabin probabilistic test, the deterministic AKS test, and an elliptic curve test. Based on their analysis, they concluded that the Miller-Rabin test is the most effective, while the latter two tests are typically used in practice. They also provided mathematical evidence to demonstrate the superiority of the Miller-Rabin test over the Solovay-Strassen test.

The authors of [14] conducted a study on the correlation between the length of test numbers and the number of Miller-Rabin rounds required to obtain an accurate result. They also provided suggestions for

ISSN NO: 2770-4491
Date of Publication:24-03-2023

selecting a suitable set of bases that can improve the efficiency of Miller-Rabin. The paper ends with a discussion on several theoretical issues that can enhance the implementation of Miller-Rabin.

In [15], the author discusses the Solovay-Strassen, Miller, and AKS primality tests and presents results from implementing these methods using Maple. The study aimed to determine the number of steps required for numbers of varying sizes (ranging from 4 to 12 digits) and to assess the results obtained.

The article [16] provides C++ implementations of several randomized and deterministic primality tests, including Miller-Rabin, Fermat, Solovay-Strassen, and AKS. The author proves several theorems to help understand these algorithms and provides explanations of the necessary concepts from number theory. While the author provides a brief overview of primality tests, they focus on the AKS test and compare its effectiveness to Fermat's test.

The paper published in [17] describes the implementation of the Lucas probabilistic primality test and focuses on developing a hardware architecture that is suitable for this test. The effectiveness of this algorithm was evaluated for numbers of different sizes.

In [18], the authors conducted a comprehensive study of 14 primality algorithms, including both deterministic and probabilistic tests. They found that deterministic tests were very slow, so probabilistic algorithms were more suitable for real-world applications. However, there is a chance of failure for probabilistic algorithms in certain situations. The authors concluded that the LLR method is the most effective deterministic primality test, while the Miller-Rabin algorithm is the most effective probabilistic primality test.

The authors of [19] provide theoretical and practical justifications for why the Miller-Rabin primality test requires improvements. They present alternative, more effective approaches for testing primality using Miller-Rabin probability error reduction estimations.

In [20], a comprehensive examination of various primality testing methods is presented, along with details on their characteristics, capabilities, limitations, and time complexities. The tests are divided into four subcategories: deterministic, heuristic, monte-carlo randomized, and las vegas randomized. Additionally, eleven of the algorithms are implemented in both Java and Python to assess their effectiveness. The findings reveal that no single primality test is appropriate for all situations and number formats. Thus, it is necessary to choose the appropriate algorithm from among these methods for each instance.

The analysis shows that the Rabin-Milner algorithm is the best probability-based algorithm for checking the numbers for primality. But there will be false witnesses in this method, too.

II. An improved Rabin-Milner algorithm

As mentioned above, the Rabin-Milner algorithm treats some complex numbers as prime numbers, which affects the reliability of cryptographic algorithms. Below is an improved Rabin-Milner algorithm that uses a recursive function to solve this problem.

A recursive function (Latin recursio - return) is a numeric function of a numeric argument, which is used in the form of the function itself. That is, $f(n-1), f(n-2),\ldots$ are used to calculate $f(n)$. To complete the calculation for an arbitrary n, it is required that the function value for some n be determined without recursion (for example, for $n = 0, 1$).

An example of a recursive function is the n-counter of the Fibonacci number:

$$F = \begin{cases} F(0) = 1; \\ F(1) = 1; \\ F(n) = F(n-1) + F(n-2), \;\; n > 1. \end{cases}$$

Using this formula, you can find the value of $F(n)$ in a finite step for any natural number n. In this case, to find the desired value, it is necessary to calculate the values $F(n-1), F(n-2), \ldots, F(2)$.

For any number n to be prime, the following factorial-recursive function can be expressed for any number a taken in the interval from one to $n-1$:

$$D_k = D_k + (k + a) * (1 + D_k * b) mod\; n,$$

here:

$D_{k=a} = a;$

k is recursion step, from $k = a$ to n;
$a = \{x: x = 1,\overline{n-1}\}$, an element of a finite set;
b is the computing base, an integer in the interval $[1, n-1]$;
D_k is the value of the recursive function of the factor in k-steps.
The algorithm of a factorial recursive function is as follows:

```
**Input:** $a$ - element and $n$ - module values, $k$ - number of steps.
**Output:** the value of the recursive function of the $D_k$-factor in $k$ steps.
1. $j \leftarrow 1,\ D_1 \leftarrow a$.
2. When the condition ($D_k \neq n-1 \ || \ j \neq k$) is satisfied, do the following:
    2.1. $D1_j \leftarrow D_j + 1, a1 \leftarrow j + a, D2_j \leftarrow D1_j * a1, D_{j+1} \leftarrow D_j + D2_j$;
    2.2. $D_{j+1} \leftarrow D_{j+1} mod\ n$
    2.3. $j++$
3. Return $D_j$.
```

In the factorial-recursive function table, since the cells related to factors n occupy a relatively large area when n is a complex number, using such a function in advanced algorithms allows you to speed up the determination of the complexity of the number being checked. Also, when the number being checked is a prime number, it greatly increases the number of witnesses to its prime number, resulting in a lower probability of committing an error.

The Rabin-Miller algorithm can be improved based on this algorithm for computing the recursive function [21].

```
**Input**: Choose $n \geq 3$ odd integers, $a$-random number satisfying the condition $2 \leq a \leq n-2$, and recursion step $k$.
**Чиқиш**: Is $n$ prime? the answer to the question is "prime" or "complex"..
1. It is written in the form $n - 1 = 2^s * r$, where $r$- is odd number.
2. $y = a^r mod\ n$ is calculated..
3. Go to next iteration if $y = 1$ or $y = n - 1$.
4. $s - 1$ times $y = y^2\ mod\ n$ is calculated.
    4.1. If $y = 1$, then the answer should be “complex”.
    4.2. If $y = n - 1$ then:
        4.2.1. Do the following until ($D_j \neq n - 1 \ \| \ j \neq k$) is satisfied:
        4.2.2. $D_1 = a; j = 1$;
        4.2.3. $D1_{\text{j}} \leftarrow D_{\text{j}} + 1,\ \ a1 \leftarrow j + a,\ \ D2_{\text{j}} \leftarrow D1_{\text{j}} * a1$,
        4.2.4. $D_{\text{j+1}} \leftarrow D_{\text{j}} + D2_{\text{j}} mod\ n.$
        4.2.5. $j++$.
```

ISSN NO: 2770-4491
Date of Publication:24-03-2023

```
        4.2.6. $a = D_j$
5. Return $D_j$.
```

The improved algorithm, compared to the existing algorithm, notices earlier when n is a complex number, which significantly reduces the number of "false witnesses".

Conclusion

Probabilistic algorithms are more practical for most applications since deterministic tests are very slow. Nonetheless, these algorithms may not always provide accurate results, so it is important to strike a balance between the efficiency of the algorithm and the correctness of the generated results. In fact, the Miller-Rabin algorithm has been identified as the most efficient probabilistic primality test according to reference [6].

When generating prime numbers using algorithms, there is always a probability involved in determining whether a number is prime or not. To determine this, test algorithms are used, which rely on the number of "prime witnesses" that they find. In other words, the more "prime witnesses" a test algorithm finds, the higher the probability that the number being tested (n) is actually prime.

References

1. Горбенко И., Вервейко В. Тестирование чисел на простоту: теория и практика. – 2003.
2. Nagell, T. (2021). Introduction to number theory (Vol. 163). American Mathematical Soc.
3. Adleman L. M., Huang M. D. Algorithmic Number Theory First International Symposium, ANTS-I Ithaca, NY, USA, May 6–9, 1994 Proceedings //Conference proceedings ANTS. – 1994. – С. 292.
4. N. Y. Myzdrikov et al., “Modification and optimization of solovey-strassen’s fast exponentiation probablistic test binary algorithm,” in 2019 IEEE East-West Design and Test Symposium, EWDTS 2019, 2019, doi: 10.1109/EWDTS.2019.8884469.
5. Robert, Solovay., Volker, Strassen. "A Fast Monte-Carlo Test for Primality." SIAM Journal on Computing, 6 (1977).:84-85. doi: 10.1137/0206006.
6. C.-L. Duta, L. Gheorghe, and N. Tapus, “Duta, C.-L., Gheorghe, L., & Tapus, N. (2015). Framework for evaluation and comparison of primality testing algorithms. Proceedings - 2015 20th International Conference on Control Systems and Computer Science, CSCS 2015, 483–490. https://doi.org/10.1109/C,” in Proceedings - 2015 20th International Conference on Control Systems and Computer Science, CSCS 2015, 2015, pp. 483–490, doi: 10.1109/CSCS.2015.153.
7. K. Conrad, “The Miller–Rabin Test,” Univ. Connect., pp. 1–17, 2016, [Online]. Available: http://www.math.uconn.edu/~kconrad/blurbs/ugradnumthy/millerrabin.pdf.
8. M. O. Rabin, “Probabilistic Algorithms”, in Academic Press, Algorithms and Complexity, New Directions and Recent Results, 1976, pp. 21-24.
9. R. Baillie, S.W. Jr. Wagstaff, “Lucas Pseudoprimes”, in Math. Comput. vol 35, 1980, pp. 1391-1417.
10. Grau J., Oller-Marcén A., Sadornil D. A primality test for $Kp^{n}+ 1$ numbers //Mathematics of Computation. – 2015. – Т. 84. – №. 291. – С. 505-512.
11. Buchmann, Johannes, and Volker Müller. "Primality testing." (1992).
12. Louis Monier. Evaluation and comparison of two efficient probabilistic primality-testing algorithms. Theoretical Computer Science, 12(1):97–108, 1980.
13. R. Schoof, “Four primality testing algorithms”, in Algoritmic Number Theory, vol. 44, 2008, pp. 101-126.
14. Ishmukhametov, S., and B. Mubarakov. "On practical aspects of the Miller-Rabin primality test." Lobachevskii Journal of Mathematics 34.4 (2013): 304-312.
15. R. M. Canfield, “Three Primality Tests and Maple Implementation”, available at https://getd.libs.uga.edu/pdfs/canfield_renee_m_200805_ma.pdf (Last Accessed: February 2015).

ISSN NO: 2770-4491
Date of Publication:24-03-2023

16. M. Perrenoud, “Randomized and Deterministic Primality Testing”, available at http://algo.epfl.ch/_media/en/projects/bachelor_semester/ randomized_and_deterministic_primality_testing.pdf (Last Accessed: February 2015).
17. Masle, W. Luk, C. A. Moritz, “Parametrized Hardware Arhitectures for the Lucas Primality test”, in Proceedings of International Conference on Embedded Computer Systems (SAMOS), 2011, pp. 124-131.
18. L. Duta, L. Gheorghe and N. Tapus, "Framework for Evaluation and Comparison of Primality Testing Algorithms", Proc. 20 th Int. Conf. on Control Systems and Computer Science Bucharest , pp. 483-490, 2015.
19. Ishmukhametov, S. T., Rubtsova, R., & Savelyev, N. (2018). The Error Probability of the Miller–Rabin Primality Test. Lobachevskii Journal of Mathematics, 39(7), 1010–1015. doi:10.1134/s1995080218070132
20. AbuDaqa A., Abu-Hassan A., Imam M. Taxonomy and Practical Evaluation of Primality Testing Algorithms //arXiv preprint arXiv:2006.08444. – 2020.
21. Ahmedova O.P., Mardiyev U.R., Karimov A.A., Tursunov O.O. Advanced Probabilistic Primality Test Using by Recursive Function. International Journal of Advanced Science and Technology Vol. 29, No.4, (2020), pp.8839 –8849