Algorithms to Compute a Generator of the Group \((\mathbb{Z}_p^*, \times_p)\) and Safe Primes

Pinaki Mitra, M Durgaprasada Rao and M Kranthi Kumar

\(^a\)Department of Computer Science and Engineering, Indian Institute of Technology, Guwahati - 781039, India.

In this paper, we consider the problem of computing a generator of the group \((\mathbb{Z}_p^*, \times_p)\) for a given prime number \(p\). In [4], Motwani and Raghavan present a randomized polynomial-time Las Vegas algorithm to compute a generator of the group \((\mathbb{Z}_p^*, \times_p)\) where the prime factorization of \(p - 1\) is provided. In this paper we improve the time complexity analysis of this Las Vegas algorithm for generator computation. The expected running-time complexity is improved from \(O((\log \log p)^2)\) to \(O((\log^3 p)/(\log \log p)^2)\). Our approach is based on providing stronger bounds on both the number of distinct prime factors and on the Euler’s phi function of an arbitrary composite integer. We also present a deterministic polynomial-time algorithm to compute a generator in the case of safe prime numbers and an efficient randomized algorithm to generate safe prime numbers.

1. INTRODUCTION

The set \(\mathbb{Z}_n^*\) is defined as the set of all natural numbers not exceeding \(n\) that are relatively prime to \(n\). Its cardinality is denoted by Euler’s phi function \(\phi(n)\). \(\mathbb{Z}_n^*\) forms a group under \(\times_n\) where \(\times_n\) denotes the multiplication modulo \(n\) operation. An element \(a \in \mathbb{Z}_n^*\) is said to be a generator or primitive root of the group \((\mathbb{Z}_n^*, \times_n)\) if \(\text{ord}(a) = \phi(n)\), where \(\text{ord}(a)\) denotes the smallest positive integer \(k\) such that \(a^k \equiv 1 \text{ mod } n\). In other words the cyclic group generated by \(a\), i.e., \(\{a^0, a^1, a^2, \ldots\}\) has cardinality \(\phi(n)\). \(\mathbb{Z}_n^*\) is said to be cyclic if a generator exists within it.

Now the question arises as to whether \(\mathbb{Z}_n^*\) is cyclic for every integer \(n\) or not. The answer is no, for some composite integers. The values of \(n > 1\) for which \(\mathbb{Z}_n^*\) is cyclic are 2, 4, \(p^n, 2p^n\) for all odd primes \(p\) and \(\alpha \geq 1\) [1]. So \((\mathbb{Z}_p^*, \times_p)\) is cyclic for every prime \(p\), i.e., a generator always exists within it.

The problem of computing a generator or a primitive root of a prime number \(p\) is an extremely important one in computational number theory and cryptography. In Diffie and Hellman key exchange [2], used in public key cryptography, both parties must agree on a shared key to be used for the subsequent encryption of messages. A generator or primitive root of a prime number is required for this algorithm. In Section 3 we describe a randomized \(O(\log^3 p)\) time algorithm for computing the generator. In Sections 4 and 5 we reduce the time complexity by a factor of \((\log \log p)^2\).

A safe prime is a prime number of the form \(2q + 1\), where \(q\) is also a prime. In Section 6 we present a deterministic polynomial-time algorithm to compute a generator in \(\mathbb{Z}_p^*\) when \(p\) is a safe prime. In Section 7 we provide an efficient algorithm to generate safe prime numbers.

2. MATHEMATICAL PRELIMINARIES

**Theorem 2.1** Every integer \(n > 1\) can be factored in a unique way as \(n = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k}\) where \(p_1 > p_2 > \ldots > p_k\) are prime numbers and each \(\alpha_i > 0\) [3].

The above theorem is known as the fundamental theorem of arithmetic, also known as the unique factorization theorem.

**Theorem 2.2** If \(p\) is a prime then \(a^{p-1} \equiv 1 \text{ mod } p\) \(\forall a \in \mathbb{Z}_p^*\) [3].

This theorem is known as Fermat’s little theorem. From the above theorem it may happen that \(\text{ord}(a) < p - 1\). So in the computing of generators we have to test for \(\text{ord}(a) = p - 1\).
Let us consider the problem of the generation of safe primes. Suppose \( q \) is a prime such that \( p = 2q + 1 \) is also a prime. As we know, \( p \) will be a safe prime and \( q \) is called a Sophie Germain prime. A widely believed conjecture about the density of Sophie Germain primes is as follows:

**Sophie Germain Prime Density Conjecture:** The number of primes \( q \leq m \) such that \( 2q + 1 \) is also a prime, is asymptotically \( \frac{2C_2m}{\ln^2 m} \) where \( C_2 \) is the twin prime constant (estimated to be approximately 0.66).

A straight-forward method of finding safe primes is to randomly pick an odd number \( q \) and check if both \( q \) and \( 2q + 1 \) are prime with a standard primality testing method such as the Miller-Rabin randomized primality test. The speed of this method can be improved by a factor of 2 by checking both \( 2q + 1 \) and \((q - 1)/2\) for primality. But even then, this procedure takes a lot of time because we need to do two primality tests for each prime \( q \), and at least one even if \( q \) is not prime. If we believe the above conjecture, we have to check \( \frac{ln^2 m}{ln^2 2} \) \( q \)'s on an average. We must therefore do at least \( \frac{ln^2 m}{ln^2 2} \) primality tests, on an average, in this method. We will try to improve upon the above naïve algorithm using the following result about safe primes:

**Theorem 7.3** If \( q \) is a prime, and \( 2^{2q} \mod (2q + 1) = 1 \), then \( (2q + 1) \) is a prime.

**Proof:** Let \( p = 2q + 1 \). Consider the group \((Z^*_p, \times_p)\). Since \( 2q + 1 \) is an odd number, 2 is relatively prime to \( 2q + 1 \), and 2 is hence a member of the set \( Z^*_p \). Let the order of 2 in the above group be \( h \). Therefore, \( 2^h \mod (2q + 1) = 1 \). Since \( 2^{2q} \mod (2q + 1) = 1 \), it follows that \( h \mid 2q \). Now, since \( q \) is a prime, \( 2q \) has only four factors: 1, 2, \( q \) and \( 2q \). Hence \( h \) should be one of these four numbers. The only number with order 1 in the group is 1. Suppose \( h = 2 \). From *Theorem 2.6* only one element can have order 2. The only number with order 2 is the number \((p - 1)\), and hence 2 must be \((p - 1)\). But this is not possible, since \( p \geq 5 \). Therefore, \( h \) must either be \( q \) or \( 2q \). In either case, \( q \mid h \).

Let the order of the group \((Z^*_p, \times_p)\) be \( n \). Since \( h \) is the order of 2 in this group, we have \( h \mid n \). Now, we also know that \( q \mid h \), so that \( q \mid n \). Also, we know that \((p - 1)\) has an order of 2 in this group. We know therefore that \( 2 \mid n \). If the prime \( q > 2 \), then \( gcd(2, q) = 1 \). Hence, \( 2q \mid n \). But, \( n \leq (p - 1) \), i.e., \( n \leq 2q \). Thus \( n = 2q = p - 1 \). Consider a number \( p \) with \( \phi(p) = p - 1 \implies Z^*_p = p - 1 \). Since \( gcd(a, p) = 1, \forall a \in Z^*_p \), no number less than \( p \) is a factor of \( p \), and hence \( p \) is a prime [7].

We can use above theorem to condense two Miller-Rabin primality tests to one base-2 Fermat’s test, and if it succeeds, one Miller-Rabin primality test. An outline of the algorithm to find safe primes of \( m + 1 \) bits is provided below.

**Algorithm 2** Function FindSafePrime(m)

1: Pick an odd random number \( q \) of \( m \) bits.
2: if \( 2^{2q} \mod (2q + 1)! = 1 \) then
3: goto Step 1
4: end if
5: if \( q \) is not a prime then
6: goto Step 1
7: else
8: return \( 2q + 1 \) as a safe prime
9: end if

This algorithm can be further adapted to use the above idea of checking both \( 2q + 1 \) and \((q - 1)/2\), thus accelerating it by a factor of 2. In this case, most numbers will fail at Step 2. Those numbers that pass this test will then be tested for primality and, if this succeeds, the process is complete. Therefore, this is a much faster way to produce safe primes.

**8. CONCLUSIONS**

In this paper we have improved the expected running-time complexity analysis of the polynomial-time Las Vegas algorithm for the
computation of a generator of the group \((Z_p^*, \times_p)\) for a given prime number \(p\) that was described in Motwani and Raghavan [4]. We have presented a deterministic polynomial-time algorithm to compute a generator in the case of safe prime numbers. But it is still an open problem to design a polynomial-time randomized algorithm (Las Vegas or Monte Carlo) to compute a generator of the group \((Z_p^*, \times_p)\) for any prime number \(p\) without the knowledge of the prime factorization of \(p - 1\). The expected time complexity of the randomized algorithm can be further improved if we can give a stronger lower bound on the product of the first \(k\) prime numbers. The lower bound \(k!\) used in Theorem 4.2 is indeed a weak lower bound for this purpose. This will also give us an estimate of the number of distinct prime factors of an arbitrary integer \(n\). We have also presented a faster algorithm for generating safe prime numbers by condensing two Miller-Rabin primality tests to one base-2 Fermat’s test and one Miller-Rabin primality test. This algorithm works faster when implemented because most of the numbers fail at the base-2 Fermat’s test.

**REFERENCES**