Probabilistic primality test: Miller-Rabin algorithm

Håkon Nymo Matland
CommTech student at Norwegian University of Science and Technology

April 2013

1 Introduction

Prime numbers are of great importance in number theory, and when you consider that cryptography is all about number theory, the importance of knowing if a integer is a prime or a composite should be clear. For clearance a composite number is a number that is not prime.

Many cryptographic algorithms require several large random prime numbers. The common RSA public-key cryptosystem is a good example of a system in need of prime numbers. A private key is generated consisting of two large primes and a public key consisting of the product. Being able to find large prime numbers are of great importance.

A primality test is an algorithm used to determine whether an input number is prime or not. The most basic primality tests are very easy to implement, but incredible slow for large numbers. In fact, there is no simple and efficient mean to accomplish the task with certainty. There are several different algorithms to test if an integer is prime or not.

In 1980 Michael Rabin discovered a randomized polynomial-time algorithm to test whether a number is prime. Gary Miller was given partial credit due to Rabin's research being closely related to a deterministic algorithm Miller studied in 1976. The Miller-Rabin algorithm is widely used in software, like OpenSSL, that rely on RSA encryption.

2 Finding composites and possible primes

To understand the algorithm you will first have to remember Fermat's little theorem: If p is prime and a is a positive integer not divisible by p, then $a^{p-1} \equiv 1 \, mod \, p$
The theorem is very important in this context because it can be used to show that a number is composite. Proof can be found at [1].
The Fermat primality test is quite easy:
If you want to check if integer p is prime, randomly pick an integer a from $1 < a < p$ and see if the equality holds.
If the equality does not hold for the value of a, p is compisite. If the equality holds for a, we have to make sure that the integer is not a Fermat liar. A Fermat liar is when a composite passes the test. Because of this uncertainty, you can only find probably primes with this method. By testing with several different values as a, the probability that the probable

prime is indeed a composite will decrease.

The Fermat primality test has a large flaw, and can't be used in a real robust, secure system. The problem with the test is values knows as Carmichael numbers. Carmichael numbers are pseudoprime to any base a with $gcd(a, p) = 1$ [4]. The Fermat primality test will see these numbers as probably primes, and decrease the oppertunity to gain a really high probability on the value being prime.
Altough Carmichael numbers occur fairly seldom, it is advisable to rather use an algorithm able to distinguish Carmichael numbers from prime numbers.
In addition to find a factorization, or a Fermat witness there is a third way to prove that a number is composite.

The Miller-Rabin algorithm uses other properties of prime numbers to find probable primes and composites than the Fermat primality test. This test also relies on equalties that hold true for prime numbers. One of the following conditions must be true when p is a prime.

- $a^q \equiv 1\,(mod\,p)$, $1 < a < n-1$
- $a^{2^{j-1}q} \equiv -1\,(mod\,p)$ for any number j in the range $1 \leq j \leq k$, $1 < a < n-1$

Proof of the conditions can be found at [3, p. 26].
The value k and q is found by finding the correct values so that $p - 1 = 2^k q$ where $k > 0$ and integer, and q is odd. The equation holds because $p - 1$ is even, and if we factorize out the larger power of 2 from n-1, the remainder q will be odd.

This can be used to check if a number is composite. If there exists a integer a making both conditions unsatisfied, the number p is composite. If one of the conditions hold for the integer a, then the number p must be either composite or a prime.

However, knowing that a given number is either composite or prime is not really helpful in a primality test. This is were the use of probability is helpful. The probability that an integer is composite while holding true to one of the conditions with a given value a, i.e., finding a strong liar to the number is at most 0.25 [5, p. 276]. Actually, for large primes it will usually be far less. Checking for several different values a in the range $1 < a < n-1$ will make the probability of a number being a false positive lower for each value a used.

This is perhaps better understood with an example: Want to determine if $n = 91$ is a prime.
$n - 1 = 90 = 2^1 * 45$
That makes k = 1, and q = 45. Randomly select a value a withing 1 and n, lets say 9.
$a^{2^0 q}\,mod\,n = 9^{45}\,mod\,91 = 1$
The number 91 passes the first condition with a = 9, and we do not yet know if the number 91 is prime or composite. Running the same process with another value a will either increase the probability that 91 is prime, or it will be proven a composite.
Trying with $a = 18$
$a^{2^0 q}\,mod\,n = 18^{45}\,mod\,91 = 57 \neq 1, n-1$
Both conditions have failed, due to k limiting the value of j to 0. We now know 91 is composite.

3 Is a probable prime enough?

The Miller-Rabin algorithm will give you probable primes with a very high probability if you use enough iterations. For most practical applications this is enough. In cryptography, the test is run with enough iterations, the probability of finding a false positive will be negligible to other factors such as hardware computational faults etc. If we decide to use 50 iterations, the probability that a found probably prime is indeed prime will be at least $1 - 0.25^{50} = 0.99999999999999999999999999999921$. Note that this probability is worst case. The probability of this happening is far less then winning the EuroMillions lottery.

What if you against all odds use a composite in for example a RSA key? First of all a bad key would probably end up in a bad signature, and you would create a new set of keys. If by chance the key including a composite ends up in a good signature, the attacker would have no idea this is the case. The probability of guessing a normal key would likely be much higher then finding a key including a composite.

References

[1] Chris K. Caldwell. Proof of Fermat's Little Theorem. http://primes.utm.edu/notes/proofs/FermatsLittleTheorem.html. [Online; accessed 20-April-2013].

[2] Joe Hurd. Verication of the Miller–Rabin probabilistic primality test. http://www.gilith.com/research/papers/miller.pdf. [Online; accessed 21-April-2013].

[3] Avinash Kak. Miller-Rabin Algorithm for Primality Testing. https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture11.pdf. [Online; accessed 21-April-2013].

[4] Ben Lynn. Carmichael Numbers. http://crypto.stanford.edu/pbc/notes/numbertheory/carmichael.xhtml. [Online; accessed 19-April-2013].

[5] William Stallings. *Cryptography and Network Security, Principles and practice.* Pearson Education, Limited, 2010.