Primality testing

Mathematical Cryptography,
Lectures 19 - 20

Contents

Primality testing

In the previous chapter, we used the *IsPrime*(n) algorithm for primality testing as a "black box". In this chapter, we will introduce some primality tests, especially the Miller-Rabin test.

In the second part, we calculate the time complexity of generating random primes if primality is tested by the Miller-Rabin test, possibly improved by dividing with small primes up to a certain bound.

Deterministic primality testing

Trial division

Claim: A natural number $n > 1$ is prime if and only if it is not divisible by any prime $p \leq \sqrt{n}$.

The brute force test of primality: Divide n by all (prime) numbers up to $\sqrt{n}$.

Time complexity: exponential, $O(2^{\frac{1}{2}\operatorname{len}(n)}\operatorname{len}(n)^2)$
Advantage: for n composite we will find its divisors

Deterministic primality testing

Deterministic polynomial primality testing

There exists a deterministic algorithm for primality testing operating in polynomial time, which uses properties of polynomials over a ring $\mathbb{Z}_n$, or polynomials over a field in case n is prime. The algorithm was investigated by Agrawal, Kayal and Saxena and published in 2004.

The algorithm operates in time $O(\mathrm{len}(n)^{16.5})$. If using faster algorithms for integer and polynomial arithmetic, it works in time $O(\mathrm{len}(n)^{10.5+o(1)})$.

Note

This algorithm is an important theoretical result, but it is meaningless in practice - the polynomial power is too high.

Deterministic primality testing

Note

If a computer performes a billion ($= 10^9$) divisions per second, then primality testing of a number $n = 10^{100} \doteq 2^{330}$ would take

- 10^{33} years by brute fource,
- approximately 10^7 years by the deterministic AKS algorithm,
- only one second by the probabilistic Miller-Rabin algorithm, with a probability of error less than 2^{-100}, which is almost zero.

Probabilistic primality testing

Probabilistic primality testing with one-sided error

Probabilistic primality tests use some property that holds for all $a \in \mathbb{Z}_n^*$ in case n is prime (all elements in $\mathbb{Z}_p^*$ truthfully testify that p is prime), while for n composite, the property holds only for some $a \in \mathbb{Z}_n^*$ (these elements are false witnesses to primality of the composite number n).

In the test, we $k-$times randomly choose some $a \in \mathbb{Z}_n^*$ and verify our property. If all chosen $a \in \mathbb{Z}_n^*$ have the property, we declare n to be prime. The probability that we can be wrong and declare the composite number n to be prime depends on the number of false witnesses. The test is therefore burdened by one-sided error.

Probabilistic primality testing

We introduce the Fermat test and the Miller-Rabin test for primality.

First, we always look at the property which they use, and we estimate number of false witnesses to primality for a composite number n in each test.

We also show that the properties may not characterize the prime numbers, or there exist pseudo-prime numbers.

The Fermat test

Fermat's little theorem

Let p be a prime. For every $a \in \mathbb{Z}_p^*$, $a^{p-1} = 1$ in $\mathbb{Z}_p$.

Witnesses to primality for the Fermat test

Let $n > 1$. Let's denote $K_n = \{a \in \mathbb{Z}_n^*,\ a^{n-1} = 1\}$.

We could define $K_n = \{a \in \mathbb{Z}_n,\ a^{n-1} = 1\}$, yet $K_n \subseteq \mathbb{Z}_n^*$.
Indeed, if $a^{n-1} = 1$, then a has an inverse $a^{-1} = a^{n-2}$, so $a \in \mathbb{Z}_n^*$.

It doesn't matter whether we choose a random element from $\mathbb{Z}_n$ or from $\mathbb{Z}_n^*$, the number of witnesses to primality is the same.

The Fermat test

Theorem

If n is prime, then $K_n = \mathbb{Z}_n^* = \mathbb{Z}_n^+$.
If n is composite and $K_n \neq \mathbb{Z}_n^*$, then $|K_n| \leq \frac{1}{2}|\mathbb{Z}_n^*| < \frac{1}{2}|\mathbb{Z}_n^+|$.

The proof relies on the fact that K_n is a subgroup of $\mathbb{Z}_n^*$.
The situation that $K_n = \mathbb{Z}_n^*$ can occur for the so-called Carmichael numbers.

The Fermat test

Testing if $a \in K_n$ (a boolean procedure)

Input: $n > 1$, $a \in \mathbb{Z}_n^*$ (or $a \in \mathbb{Z}_n^+$)
Output: *True* or *false*

- $b \leftarrow a^{n-1}$ in $\mathbb{Z}_n$
- if $b = 1$ then return *true*
  else return *false*

Time complexity: $O(\mathrm{len}(n)^3)$ (repeated squaring algorithm)

The Fermat test

The Fermat primality test - algorithm $\mathbf{F}(\cdot, \mathbf{k})$

Input: $n > 1$; (we test whether n is prime)
parameter $k \geq 1$ (number of random witnesses)
Output: *True* or *false*

- repeat k times
  - $a \overset{\not{c}}{\leftarrow} \mathbb{Z}_n^+$ (or $a \overset{\not{c}}{\leftarrow} \mathbb{Z}_n^*$)
  - if $a \notin K_n$ then return *false* endif enddo
- return *true*

Time complexity in the worst case: $O(k \operatorname{len}(n)^3)$
The expected time for n composite (but not Carmichael) is $O(2 \operatorname{len}(n)^3)$.

The Fermat test

Probability of error

If n is prime, then the Fermat test always answers *true* correctly.

If n is composite, but not Carmichael, then the probability of error (the test answers *true*) is at most $\epsilon = \frac{1}{2^k}$, where k is the number of independently randomly chosen witnesses $a \in \mathbb{Z}_n^+$.

For Carmichael numbers, the probability of error is much greater (especially when choosing $a \overset{\not{c}}{\leftarrow} \mathbb{Z}_n^*$, Carmichael numbers are indistinguishable from primes by the Fermat test).

Remark

The random choice $a \in \mathbb{Z}_n^+ \setminus \mathbb{Z}_n^*$ allows to find a factor of n, which is $d = \gcd(a, n) > 1$.

Carmichael numbers

Definition

A *Carmichael number* is a composite number n such that $a^{n-1} = 1$ holds for every $a \in \mathbb{Z}_n^*$.

Carmichael numbers are rare, yet they are infinitely many.

The only Carmichael number up to 1000 is $561 = 3 \cdot 11 \cdot 17$, the nexts are $1105 = 5 \cdot 13 \cdot 17$, $1729 = 7 \cdot 13 \cdot 19$.

Within 10^{16} there are roughly $2.7 \cdot 10^{14}$ prime numbers and only $2.4 \cdot 10^5$ Carmichael numbers.

Carmichael numbers

Proposition

A composite number n is Carmichael if and only if $\lambda(n) \mid n-1$, where $\lambda(n) = \exp(\mathbb{Z}_n^*)$ is a Carmichael function.

Proposition

Each Carmichael number n is of the form $n = p_1 \cdot \dots \cdot p_r$, where

- p_i are different odd primes (or n is odd and square free),
- $r \geq 3$,
- $p_i - 1 \mid n - 1$ for every $1 \leq i \leq r$.

The Miller-Rabin test

Proposition

Let $p > 2$ be a prime.
The equation $x^2 = 1$ has exactly two solutions in the group $\mathbb{Z}_p^*$, namely $x = \pm 1$, so there are only trivial square roots of 1 in $\mathbb{Z}_p^*$.

Witnesses to primality for the Miller-Rabin test

Let $n > 1$ be an odd number, $n - 1 = t\,2^h$ for t odd.

$$L_n = \{a \in \mathbb{Z}_n^*,\ a^{n-1} = 1 \text{ and when } a^{t\,2^j} = 1, \text{then } a^{t\,2^{j-1}} = \pm 1 \\ \text{for all } 1 \leq j \leq h\}$$

We could define L_n as a subset of Z_n and define the same set L_n. Obviously, $L_n \subseteq K_n$.

The Miller-Rabin test

Note

The property that the equation $x^2 = 1$ has exactly two solutions $x = \pm 1$ in $\mathbb{Z}_n^*$ does not characterize prime numbers. This property holds in every cyclic group $\mathbb{Z}_n^*$, so it also holds for $n = p^e$, where $p > 2$ is a prime, $e \geq 1$. (And also for $n = 2$, $n = 4$, $n = 2p^e$, where $p > 2$ is prime, but we are not interested in even n now.)

For such n is $L_n = K_n$ (the Miller-Rabin test has as many false witnesses to primality as the Fermat test).

The Miller-Rabin test

Theorem

Let n be an odd number. If n is prime, then $L_n = \mathbb{Z}_n^* = \mathbb{Z}_n^+$.
If $n > 9$ is composite, then $|L_n| \leq \frac{1}{4}|\mathbb{Z}_n^*| < \frac{1}{4}|\mathbb{Z}_n^+|$.

Proof notes:

- For $n = p^e$, $e \geq 2$, where p is an odd prime, is $L_n = K_n$ and since $\mathbb{Z}_n^*$ is cyclic, we can compute $|K_n| = p - 1 = \frac{1}{p^{e-1}}|\mathbb{Z}_n^*|$.
- For $n = \prod_{i=1}^r p_i^{e_i}$, $r \geq 2$, p_i odd prime numbers, we can show (it takes some work) that $|L_n| \leq \frac{2}{2^r}|Ker\,\rho_{t2^g}| \leq \frac{1}{2^{r-1}}|K_n|$, where $\rho_{t2^g} : x \mapsto x^{t2^g}$ and $g = \min\{h, h_1, \ldots, h_r\}$, where $n - 1 = t\,2^h$, $\varphi(p_i^{e_i}) = t_i\,2^{h_i}$ and t, t_i are odd.

  If n is not a Carmichael number, then $|L_n| \leq \frac{1}{2}|K_n| \leq \frac{1}{4}|\mathbb{Z}_n^*|$.
  If n is Carmichael, than $r \geq 3$, so $|L_n| \leq \frac{1}{4}|K_n| = \frac{1}{4}|\mathbb{Z}_n^*|$.

The Miller-Rabin test

Testing if $a \in L_n$ (a boolean procedure)

Input: $n > 1$ odd, where $n - 1 = t\,2^h$ for t odd;
$a \in \mathbb{Z}_n^*$ (or $a \in \mathbb{Z}_n^+$)

Output: *True* or *false*

- $b \leftarrow a^t$ in $\mathbb{Z}_n$
- if $b = 1$ then return *true* endif
- for $j \leftarrow 0$ to $h - 1$ do
  - if $b = -1$ then return *true* endif
  - it $b = 1$ then return *false* endif
  - $b \leftarrow b^2$ in $\mathbb{Z}_n$ enddo
- return *false*

The time complexity is $O(\mathrm{len}(n)^3)$. The algorithm sequentially calculates a^{n-1} v $\mathbb{Z}_n$ using the repeated squaring algorithm.

The Miller-Rabin test

The Miller-Rabin primality test - algorithm MR$(\cdot, \mathbf{k})$

Input: $n > 1$ (we test whether n is prime),
parameter $k \geq 1$ (number of random witnesses)

Output: *True* or *false*

- if $n = 2$ then return *true* endif
- if n is even then return *false* endif
- repeat k times (*n* is odd for now)
  - $a \stackrel{\not{c}}{\leftarrow} \mathbb{Z}_n^+$ (or $a \stackrel{\not{c}}{\leftarrow} \mathbb{Z}_n^*$)
  - if $a \notin L_n$ then return *false* endif enddo
- return *true*

The time complexity is at worst $O(k \operatorname{len}(n)^3)$.
The expected time for n composite is $O(\frac{4}{3} \operatorname{len}(n)^3)$.

The Miller-Rabin test

Probability of Error

If n is prime, then the Miller-Rabin test always answers *true*.

If n is composite, then the probability of error (that $MR(\cdot, k)$ still answers *true*) is at most $\epsilon = \frac{1}{4^k}$.

Note

A random choice of $a \in K_n \setminus L_n$ allows us to factorize n into two factors. The element a generates in its powers a non-trivial square root of 1 ($c \neq \pm 1$, but $c^2 = 1$ in $\mathbb{Z}_n$), thus $d = \gcd(c \pm 1, n) > 1$ are factors of n.

Generating random primes

RP algorithm (=Random Prime)

Input: a natural number $m \geq 2$, (let's denote $l = \mathrm{len}(m)$),
Output: a random prime number between 2 and m

- repeat $n \stackrel{\not c}{\leftarrow} \{2, \ldots m\}$
- until *IsPrime*(n)
- output n

IsPrime$(\cdot)$ will be implemented as the Miller-Rabin test $MR(\cdot, k)$ with parameter k for now.

Generating random primes

Analysis of RP algorithm using MR$(\cdot, \mathbf{k})$ - OUTPUT

$MR(\cdot, k)$ is a probabilistic test with one-sided error, for n composite, the probability of error is at most $\epsilon = \frac{1}{4^k}$.
We know from the previous chapter:

- Every prime up to $m \doteq 2^l$ can be found with equal probability.
- The probability that the output is a composite number up to $m \doteq 2^l$ is $O(\epsilon l) = O(\frac{1}{4^k} l)$.

If we want to find a random 1024−bit prime with error probability at most $\frac{1}{2^{100}}$, we should choose $k = 55$ witnesses.

Generating random primes

Remark

In reality, the probability of error is even smaller, especially when generating large primes. We estimated the number of false witnesses to primality in the Miller-Rabin test by

$$|L_n| \leq \frac{2}{2^r}|K_n| \leq \frac{1}{2^r}|\mathbb{Z}_n^*| \text{ (for } n \text{ not Carmichael),}$$

where r is the number of primes in the factorization of n.
So most of composite numbers has very few false witnesses to primality.

Let's denote by $\gamma(m,k)$ the probability that the output of the algorithm $RP(m)$ using $MR(\cdot,k)$ is a composite number.
For large m, $\gamma(m,1)$ (one witness) is already very small.

$$\gamma(2^{200},1) \leq \frac{1}{8},\ \gamma(2^{300},1) \leq \frac{1}{2^{19}},\ \gamma(2^{500},1) \leq \frac{1}{2^{55}}$$

To generate a 512−bits prime with error probability less than $\frac{1}{2^{100}}$, it is sufficient to choose $k=2$ witnesses.

Generating random primes

Time analysis of RP algorithm using MR$(\cdot, k)$

Since last time, we know:

- The expected number of loops is $O(l)$, where $l = \mathrm{len}(m)$, since *LOOPS* has a geometric distribution with parameter $p > \frac{\pi(m)}{m-1} \in O(\frac{1}{l})$ (Chebyshev's theorem).
  So we will have to test on average l numbers up to $m = 2^l$ before we find one prime number.
- The algorithm $MR(\cdot, k)$ works in time $O(kl^3)$ at worst.
- The expected time is $E(TIME) \in O(kl^4)$.

Generating random primes

Time analysis of RP algorithm using MR$(\cdot, \mathbf{k})$

However, this estimate of time is very pessimistic because if n is composite the probability of finding a witness to compositeness is at least $\frac{3}{4}$. The random variable $LOOPS$ in the Miller-Rabin test has an "almost geometric" distribution, so we can expect $E(LOOPS) = \frac{4}{3}$ choises.

A composite n is usually recognized by one or two Miller-Rabin witnesses, only for a prime n we check k witnesses in order to be more sure. Hence:

- The expected time is $E(TIME) \in O(l^4 + kl^3)$.
  We test roughly l composite numbers each in time $O(l^3)$ before finding one prime number to test in time $O(kl^3)$.

The Miller-Rabin test - improvement

The Miller-Rabin test with division by small primes - MRS$(\cdot, \mathbf{k})$

Most composite numbers are divisible by small primes (every second by two, every third by three, etc.).
To test the divisibility by small primes can take $O(\mathrm{len}(n))$ time, while the Miller-Rabin test takes $O(\mathrm{len}(n)^3)$ time.

We can speed up the primality testing if we first check divisibility by any prime up to a certain bound s.

The Miller-Rabin test - improvements

The Miller-Rabin test with division by small primes - algorithm MRS$(\cdot, k)$

Input: $n > 1$ (we test if n is prime),
parameter $k \geq 1$ (number of witnesses to primality)
parameter $s > 1$ (we divide by primes up to the bound s)
Output: *True* or *false*

- for each prime $p \leq s$ do
  - if $p \mid n$ then if $p = n$ then return *true*
    else return *false* endif enddo
- repeat k times
  - $a \overset{\not c}{\leftarrow} \mathbb{Z}_n^+$ (or $a \overset{\not c}{\leftarrow} \mathbb{Z}_n^*$)
  - if $a \notin L_n$ then return *false* endif enddo
- return *true*

Generating random primes

Time analysis of RP algorithm using MRS$(\cdot, \mathbf{k})$

We estimate how many numbers go into the Miller-Rabin test.
We know that every $p-$th number is divisible by the prime p.
The probability that a random number n is not divisible by p is then $(1 - \frac{1}{p})$. We will assume that beeing not divisible by different primes are independent events (heuristic argument).
Let $\tilde{p}$ denote the probability that random n is not divisible by any prime $p \leq s$, then:

$$\tilde{p} = \prod_{p \leq s} (1 - \frac{1}{p}) \in O(\frac{1}{\mathrm{len}(s)})$$

Merton's theorem

$\prod_{p \leq s} (1 - \frac{1}{p}) \in \Theta(\frac{1}{\ln(s)})$, where the product is over all primes up to s.

Generating random primes

Time analysis of RP algorithm using MRS$(\cdot, \mathbf{k})$

Thus, we can expect that before we find the prime $\leq m$, we will test roughly $l = (m)$ numbers, of which

- $\frac{1}{\mathrm{len}(s)}\, l$ numbers will go into the Miller-Rabin test and one or two witnesses will prove their compositeness (in time $O(l^3)$);
- the other composite numbers (there are at most l) will be divisible by some prime up to s, which will be uncovered in time $O(\pi(s)\, l) = O(\frac{s}{\mathrm{len}(s)}\, l)$ for each;
- one prime will be tested by all k witnesses in the Miller-Rabin test in time $O(kl^3)$;
- The expected time is $E(\mathit{TIME}) \in O(\frac{1}{\mathrm{len}(s)} l^4 + \frac{s}{\mathrm{len}(s)} l^2 + kl^3)$.

Generating random primes

Time analysis of RP algorithm using MRS$(\cdot, \mathbf{k})$

- We choose the bound s such that $l \leq s \leq l^2$, or $s \doteq l$, then the expected time to find a random prime within 2^l using the $MRS(\cdot, k)$ algorithm is

$$E(TIME) \in O(\frac{1}{\mathrm{len}(l)} l^4 + kl^3)$$

For example, to find a random 1024−bit prime, we will divide by primes up to the bound $s = 1024$. For $k = 55 < 2^6$ we expect time $c(\frac{1}{10}2^{40} + k2^{30}) \doteq c2^{37}$ for a small constant $c \doteq 1$.
Supercomputers operating at a rate of 1000 billions $(= 10^{12})$ operations per second will find a 1024−bit prime in one second with nearly a zero probability of error. Computers with a speed of one billion operations per second would do it in 15 minutes.

Eratosthenes sieve

Let's give an algorithm how to find all primes up to the bound s.

Eratosthenes sieve algorithm

Input: $s > 1$

Output: an array $A[2, \ldots, s]$,
where $A[i] = 1$ only if i is a prime

- for $i \leftarrow 2$ to s do $A[i] \leftarrow 1$ enddo
- for $i \leftarrow 2$ to $\lfloor \sqrt{s} \rfloor$ do
  - if $A[i] = 1$ then
    - $j \leftarrow i + i$
    - while $j \leq s$ do $A[j] \leftarrow 0$, $j \leftarrow j + i$ enddo
    - endif
  - enddo

Eratosthenes sieve

Analysis of the Eratosthenes sieve algorithm

The space complexity is exponential $O(s) = O(2^{\{len(s)})$!

We estimate the time complexity:
For each prime $p \leq \sqrt{s}$ we perform $\frac{s}{p}$ simple operations.

$$TIME = \sum_{p \leq \sqrt{s}} \frac{s}{p} < s \int_1^{\sqrt{s}} \frac{1}{y}\, dy = \frac{1}{2} s \ln(s) \in O(s \operatorname{len}(s))$$

A more accurate estimate: $TIME \in O(s \operatorname{len}(\operatorname{len}(s))$, thanks to the following theorem.

Theorem

Sum over all primes $\sum_{p \leq \sqrt{s}} \frac{1}{p} = \ln(\ln(s)) + O(1)$.

The Miller-Rabin test

Note

If the generalized Riemann hypothesis holds, then for every composite number n there is a witness to compositeness, $a \in \mathbb{Z}_n \setminus L_n$, of size $a \leq 2\,\mathrm{len}(n)^2$.

If this is the case, the Miller-Rabin test could be deterministic and it would work in time $O(\mathrm{len}(n)^5)$.

The RP algorithm would then find a prime up to $m = 2^l$ in time $O(\frac{1}{\mathrm{len}(l)}\, l^6)$ (when dividing by primes up to the bound $s \doteq l$).

Primality testing

Literature

- Shoup: A Computational Introduction to Number Theory and Algebra. Chapter 10.
  http://shoup.net/ntb/