3. Factorization. Primality testing.

A. Ushakov

MA503, September 15, 2021

Contents

Here we discuss some important details of RSA implementation.

- Simple primality test.
- Fermat primality test.
- Fermat pseudoprime.
- Carmichael numbers.
- Miller–Rabin primality test. Efficiency.
- Random prime generation.
- Shared primes.
- Pollard's $p - 1$ method to factor $N = pq$.
- Factorization of N via difference of squares.
- Pollard's rho algorithm.

Simple primality test

Primality problem *is an algorithmic question to decide if a given n is prime or not.*

A **primality test** *is an algorithm to solve primality problem.*

- There is a polynomial-time primality test, called **AKS-primality test**.
- There are even more efficient simple randomized tests, e.g., **Fermat test**, **Miller-Rabin test**, etc.

Lemma

An integer $n > 1$ is composite $\Leftrightarrow$ *n is divisible by some prime $p \leq \sqrt{n}$.*

$$\begin{aligned} n \text{ is composite} &\Leftrightarrow n = ab && \text{for some } 1 < a, b < n \\ &\Leftrightarrow \min(a, b) \mid n && \text{where } \min(a, b) \leq \sqrt{n} \end{aligned}$$

(Simple primality test)

Enumerate numbers $2, \ldots, \lfloor \sqrt{n} \rfloor$. If one number divides n, then output No. Otherwise, output Yes.

- This algorithm terminates very fast when n has a small divisor.
- It is very inefficient if n is prime.

Fermat primality test

Suppose that $\gcd(a, n) = 1$. By Fermat's little theorem

$$n \text{ is prime} \quad \Rightarrow \quad a^{n-1} \equiv_n 1.$$

(Contrapositive) $a^{n-1} \not\equiv_n 1 \quad \Rightarrow \quad n$ **is not prime**.

(Fermat primality test)

(1) *Generate a random* a *such that* $1 < a < n$.

(2) $a^{n-1} \not\equiv_n 1 \quad \Rightarrow \quad$ *output No.*

(3) *Otherwise output ProbablyYes.*

Fermat primality test can recognize composite n only. It does not solves the primality problem completely. In fact, it can fail to recognize a composite number too.

For instance, for $n = 6$ *pick a random* a *such that* $1 < a < 6$.

- *E.g., if* $a = 5$, *then* $a^{n-1} = 5^5 \equiv_6 5 \not\equiv_6 1$ *and on step (2) we output No.*

For instance, for $n = 35$ *pick a random* a *such that* $1 < a < 35$.

- *E.g., if* $a = 2$, *then* $2^{34} \equiv_{35} 9 \not\equiv_{35} 1$ *and on step (2) we output No.*
- *E.g., if* $a = 3$, *then* $3^{34} \equiv_{35} 4 \not\equiv_{35} 1$ *and on step (2) we output No.*

Fermat pseudoprime

Fermat primality test used with a is called **base-a test**.

Definition

A number $n > 2$ is called a **Fermat pseudoprime to base a** if

- n is composite;
- $a^{n-1} \equiv_n 1$ and, hence, base-a test fails to recognize that n is composite.

Fermat pseudoprime is a Fermat pseudoprime to base 2.

For instance, $341 = 31 \cdot 11$ is a Fermat pseudoprime to base 2 because the base-2 test fails for 341

$$2^{340} = (2^{10})^{34} = 1024^{34} \equiv_{341} 1^{34} = 1.$$

Theorem (no proof)

There are infinitely many base-2 pseudoprimes.

Idea: if n is pseudoprime, then $x = 2^n - 1$ is a pseudoprime too.

Fermat primality test can be performed in nearly-quadratic time $\widetilde{O}(\log^2(n))$.

Carmichael numbers

Q. Is it true that for any composite n there exists an appropriate base a such that the base-a test shows that n is indeed composite.
A. No. There are composite n that fail base-a test for every a. Such numbers are called **Carmichael numbers**.

Definition

A number $n > 2$ is called a **Carmichael number** if

- n is composite;
- $a^{n-1} \equiv_n 1$ for every $a \in \mathbb{Z}$.

A number $561 = 3 \cdot 11 \cdot 17$ is Carmichael because it satisfies the equality $a^{560} \equiv_{561} 1$ for every $a \in \mathbb{Z}$. Indeed, $a^{560} \equiv_{561} 1$ is equivalent to the system

$$\begin{cases} a^{560} \equiv 1 \mod 3 \\ a^{560} \equiv 1 \mod 11 \\ a^{560} \equiv 1 \mod 17 \end{cases}$$

which obviously holds (using $a^2 \equiv_3 1$, $a^{10} \equiv_{11} 1$, $a^{16} \equiv_{17} 1$).

Theorem (no proof)

There are infinitely many Carmichael numbers.

Miller–Rabin primality test

Later we will prove that 1 has exactly two square roots ± 1 modulo any prime p. Modulo a composite number it can have more than two square roots.

Proposition

Let p be an odd prime. Express $p-1$ as $p-1=2^k q$, where q is odd. Then for every a coprime with p one of the following conditions hold:

(1) $a^q \equiv_p 1$,

(2) $a^{2^i q} \equiv_p -1$ *for some* $0 \leq i \leq k-1$.

- Consider the list of powers $a^q, a^{2q}, a^{4q}, \ldots, a^{2^k q}$.
- The last number satisfies $a^{2^k q} = a^{p-1} \equiv_p 1$.
- Each previous number is a square root of the next number.
- Hence, for a prime modulus, the list of numbers must end with 1's

$$a^q, a^{2q}, a^{4q}, \ldots, \underbrace{a^{2^i q}, \ldots, a^{2^k q}}_{\text{ones}}.$$

- Case-I: If the whole list is the list of ones, then (1) holds.
- Case-II: Otherwise, the rightmost non-one must be -1 and so (2) holds.

(Miller–Rabin primality test for an odd n)

(1) *Generate a random a such that $1 < a < n$ satisfying $\gcd(a, n) = 1$.*

(2) *Compute q and k such that $n - 1 = 2^k q$.*

(3) *If $a^q \equiv_n 1$, then output ProbablyYes.*

(4) *If $a^{2^i q} \equiv_n -1$ for some $i = 0, \ldots, k-1$, then output ProbablyYes.*

(5) *Otherwise output No.*

Fermat base-a test outputs No $\Rightarrow$ $a^{n-1} \not\equiv_n 1$

$\Rightarrow$ the sequence of roots has no ± 1's

$\Rightarrow$ Miller-Rabin test mod-a outputs No.

Miller–Rabin testing is stronger than Fermat testing.

For a Carmichael number $n = 561$, choose $a = 2$, compute $n - 1 = 2^4 \cdot 35$ and a sequence of power

$$\begin{aligned} a^q = 2^{35} &\equiv_{561} 263 \\ 2^{2\cdot 35} &\equiv_{561} 166 \\ 2^{4\cdot 35} &\equiv_{561} 67 \\ 2^{8\cdot 35} &\equiv_{561} 1. \end{aligned}$$

Hence, the algorithm outputs No.

Miller–Rabin primality test: efficiency

Definition

A unit a modulo n is called a **Miller–Rabin witness** for the compositeness of n if Miller–Rabin algorithm base-a recognizes n as composite.

Proposition

Let n be an odd composite number. Then at least 75% of the numbers a between 1 and $n-1$ are Miller–Rabin witnesses for n.

Therefore, running the Miller–Rabin base-a algorithm on 10 randomly generated a's recognizes a composite number with probability at least

$$1 - \left(\tfrac{1}{4}\right)^{10} = 0.99999904632568359375.$$

This test has a tiny chance to give a wrong answer. Yet, it is preferred over the AKS test (that makes no mistakes) due to its efficiency.

Miller–Rabin primality test can be performed in nearly-quadratic time $\widetilde{O}(\log^2(n))$.

Generating a random prime

Definition

For $N \in \mathbb{N}$ define $\pi(N)$ to be the number of primes p between 1 and N.

$\pi(2) = 1$	$\pi(5) = 3$	$\pi(8) = 4$	$\pi(11) = 5$	$\pi(14) = 6$	$\pi(17) = 7$
$\pi(3) = 2$	$\pi(6) = 3$	$\pi(9) = 4$	$\pi(12) = 5$	$\pi(15) = 6$	$\pi(18) = 7$
$\pi(4) = 2$	$\pi(7) = 4$	$\pi(10) = 4$	$\pi(13) = 6$	$\pi(16) = 6$	$\pi(19) = 8$

Theorem (Prime number theorem)

$\frac{\pi(N)}{N/\ln(N)} \to 1$ *as* $N \to \infty$.

The chance that a randomly chosen number n between 1 and N is prime is

$$\frac{\pi(N)}{N} \approx \frac{1}{\ln(N)}.$$

Corollary

The chance to that a randomly chosen number $n \in [1, 2^m]$ *is prime is approximately* $\frac{1}{\ln(N)} \approx \frac{1}{\ln(2^m)} = \frac{1}{m\ln(2)}$.

Hence, a sequence of $2m\ln(2)$ random numbers has a prime with high probability. So, to generate a prime in $[1, 2^m]$ we can generate $2m\ln(2)$ random integers and test if one of them is prime.

Randomness is important!

Poorly generated RSA primes p and q lead to "easy attacks", like

N. Heninger, Z. Durumeric, E. Wustrow, J. Halderman, *Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices*. 2012.

Abstract. *RSA and DSA can fail catastrophically when used with malfunctioning random number generators, but the extent to which these problems arise in practice has never been comprehensively studied at Internet scale. We perform the largest ever network survey of TLS and SSH servers and present evidence that vulnerable keys are surprisingly widespread. We find that 0.75% of TLS certificates share keys due to insufficient entropy during key generation, and we suspect that another 1.70% come from the same faulty implementations and may be susceptible to compromise. Even more alarmingly, we are able to obtain RSA private keys for 0.50% of TLS hosts and 0.03% of SSH hosts, because their public keys shared nontrivial common factors due to entropy problems, and DSA private keys for 1.03% of SSH hosts, because of insufficient signature randomness.*

The authors analyzed a large collection of RSA public keys $\{(n_i, e_i)\}_i$ and realized that for significantly many pairs $i \neq j$

$$1 < \gcd(n_i, n_j) \neq n_i, n_j$$

and, hence, $\gcd(n_i, n_j)$ must be a prime shared by n_i and n_j.

Pollard's $p-1$ method to factor $N = pq$

For any $L \in \mathbb{N}$ and a randomly generated a coprime with N

$$(p-1) \mid L \text{ and } (q-1) \nmid L \quad \Rightarrow \quad L = i(p-1) = j(q-1) + k$$

$$\Rightarrow \begin{cases} a^L = a^{i(p-1)} = \left(a^{p-1}\right)^i \equiv 1^i = 1 & \mod p \\ a^L = a^{j(q-1)+k} = \left(a^{q-1}\right)^j a^k \equiv a^k & \mod q. \end{cases}$$

Since $k \neq 0$, it is "highly likely" that $a^k \not\equiv_q 1$, i.e., for most choices of a we have

$$p \mid a^L - 1 \text{ and } q \nmid a^L - 1.$$

which implies that $\gcd(N, a^L - 1) = p$.

Q. *But wait! How do we find L satisfying $(p-1) \mid L$ and $(q-1) \nmid L$?*

A. If $p-1$ is a product of small primes, then it divides $L = n!$ for some not-too-large n.

Q. *But then we have to compute $\gcd(N, a^{n!} - 1)$ and $a^{n!} - 1$ is huge! Can we do it?*

A. Yes, we can! Because

- By Euclidean lemma, $\gcd(N, a^{n!} - 1) = \gcd(N, a^{n!} - 1 \ \% \ N)$.
- $a^{(n+1)!} = a^{n! \cdot (n-1)} = \left(a^{n!}\right)^{n+1}$. Hence, we can compute $a^{n!} \ \% \ N$ using n modular exponentiations.

Pollard's $p-1$ algorithm

Pick a random a s.t. $\gcd(a, N) = 1$. For $n = 2, 3 \ldots$

- *compute $d = \gcd(N, a^{n!} - 1)$;*
- *if $1 < d < N$, then output d;*
- *if $d = N$, then output FAILURE.*

For instance, for $N = 13927189 = 3832 \cdot 3643$:

$$
\begin{aligned}
2^{9!} &\equiv_N 13867883 & \gcd(2^{9!} - 1, 13927189) &= 1,\\
2^{10!} &\equiv_N 5129508 & \gcd(2^{10!} - 1, 13927189) &= 1,\\
2^{11!} &\equiv_N 4405233 & \gcd(2^{11!} - 1, 13927189) &= 1,\\
2^{12!} &\equiv_N 6680550 & \gcd(2^{12!} - 1, 13927189) &= 1,\\
2^{13!} &\equiv_N 6161077 & \gcd(2^{13!} - 1, 13927189) &= 1,\\
2^{14!} &\equiv_N 879290 & \gcd(2^{14!} - 1, 13927189) &= 3823.
\end{aligned}
$$

Hence, $p = 3823$ is a factor in N. The method worked for $n = 14$ because $3823 - 1 = 2 \cdot 3 \cdot 7^2 \cdot 13$ which divides 14! and does not divide 13!. For the other factor $q = 3643$ we have $q - 1 = 2 \cdot 3 \cdot 607$.

$p - 1$ and $q - 1$ should not factor into small primes for RSA primes p and q.

Factorization of N via difference of squares

$$a^2 \equiv_N b^2 \quad \Rightarrow \quad N \mid a^2 - b^2 \quad \Rightarrow \quad (a-b)(a+b) = a^2 - b^2 = Nq \text{ for some } q \in \mathbb{Z}$$
$$\Rightarrow \quad \text{it is possible that } d = \gcd(a \pm b, N) \neq 1, N$$
$$\Rightarrow \quad d \text{ is a factor in } N.$$

Q. *Is it easy to find such a, b for a huge N?*

A. Unfortunately, no, it is not easy in general. We will try to find a, b using heuristics.

To find a, b we can attempt to generate sufficiently many pairs (a_i, c_i) such that $a_i^2 \equiv_N c_i$ and c_i **factors as a product of small primes**.

For instance, for $N = 914387$ we can get

$$\begin{aligned} 1869^2 &\equiv_N 750000 = 2^4 \quad 3 \quad 5^6 \\ 1909^2 &\equiv_N 901120 = 2^{14} \quad\quad 5 \quad 11 \\ 3387^2 &\equiv_N 499125 = \quad\quad 3 \quad 5^3 \quad 11^3. \end{aligned}$$

The numbers on the right are not squares, but their product is, and we get

$$9835^2 \equiv_N (1869 \cdot 1909 \cdot 3387)^2 \equiv_N 2^{18}3^2 5^{10} 11^4 = (2^9 \cdot 3 \cdot 5^5 \cdot 11^2)^2$$
$$= 580800000^2 \equiv_N 164255^2.$$

Hence, $a = 9835$ and $b = 164255$. Finally, $\gcd(914387, 9835 - 164255) = 1103$ which is a factor in N.

Factorization via difference of squares for N

- generate random a_i's;
- compute $c_i = a_i^2 \ \% \ N$;
- attempt to factor $c_i = p_1^{a_1} \ldots, p_k^{a_k}$ (called a **relation**) where $p_1, \ldots, p_k$ are "small primes";
- find a product of c_i's which is a square

$$c_1^{\varepsilon_1} \cdot \ldots \cdot c_k^{\varepsilon_k} = p_1^{2b_1} \ldots, p_k^{2b_k}.$$

- then $\left(\underbrace{a_1^{\varepsilon_1} \cdot \ldots \cdot a_k^{\varepsilon_k}}_{a} \right)^2 \equiv_N \left(\underbrace{p_1^{b_1} \ldots, p_k^{b_k}}_{b} \right)^2$.

Birthday problem

$1 - x < e^{-x}$ *for every* $x > 0$.

Consider $f(x) = e^{-x} - (1 - x)$ and notice the following:

- $f(0) = 0$;
- $f'(x) = 1 - e^{-x} > 0$ for every $x > 0$ and, hence, $f(x)$ is continuous and increasing.

Therefore, $f(x) > 0$ for every $x > 0$.

Let $x_1, \ldots, x_k$ *be random elements of a set of* n *elements. Then* $\Pr(x_i \neq x_j) < e^{-\frac{k(k-1)}{2n}}$.

$$\Pr(x_i \neq x_j, \quad \forall i \neq j) = \Pr(x_2 \neq x_1) \cdot \Pr(x_3 \neq x_1 \ \& \ x_3 \neq x_2) \cdot \ldots \cdot \Pr(x_k \neq x_1 \ \& \ \ldots \ x_k \neq x_{k-1})$$

$$= \tfrac{n-1}{n} \cdot \tfrac{n-2}{n} \cdot \ldots \cdot \tfrac{n-(k-1)}{n} = \left(1 - \tfrac{1}{n}\right) \cdot \left(1 - \tfrac{2}{n}\right) \cdot \ldots \cdot \left(1 - \tfrac{k-1}{n}\right)$$

$$< e^{-\frac{1}{n}} e^{-\frac{2}{n}} \ldots e^{-\frac{k-1}{n}} = e^{-\frac{k(k-1)}{2n}}.$$

$k > \sqrt{2n\ln(2)} \quad \Rightarrow \quad e^{-\frac{k(k-1)}{2n}} < \frac{1}{2}$.

$$e^{-\frac{k(k-1)}{2n}} < \tfrac{1}{2} \quad \Leftrightarrow \quad -\tfrac{k(k-1)}{2n} < \ln(\tfrac{1}{2}) = -\ln(2) \quad \Leftrightarrow \quad k(k-1) > 2n\ln(2)$$

$$\Leftarrow \quad k = 1 + \sqrt{2\ln(2)n} \approx 1 + 1.177\sqrt{n} = O(\sqrt{n}).$$

$k > \sqrt{2n\ln(2)} \quad \Rightarrow \quad \Pr(x_i = x_j \text{ for some } i \neq j) = 1 - \Pr(x_i \neq x_j) > 1 - e^{-\frac{k(k-1)}{2n}} > \frac{1}{2}$.

Conclusion. A randomly chosen sequence $x_1, \ldots, x_k$ of numbers $x_i \in \{1, \ldots, n\}$ has a collision with probability at least $\frac{1}{2}$ if $k > \sqrt{2n\ln(2)}$.

Pollard's rho algorithm (general purpose factoring)

Suppose that p is the least prime factor in N. Then for any $x, y \in \mathbb{N}$ we have

$$\begin{array}{l} x \equiv_p y \\ x \not\equiv_N y \end{array} \quad \Rightarrow \quad \begin{array}{l} p \mid x - y \\ N \nmid x - y \end{array} \quad \Rightarrow \quad \gcd(x - y, N) \text{ is a non-trivial factor of } N.$$

How can we find such x, y?

Birthday problem: *Generate random numbers $x_1, \ldots, x_k \in [0, N-1]$. If $k = O(\sqrt{p}) = O(\sqrt[4]{N})$, then some $x = x_i, y = x_j$ satisfy the assumption above, with probability at least $\frac{1}{2}$.*

Unfortunately, it would require to check $k^2 = O(\sqrt{N})$ pairs x_i, x_j to find one satisfying $\gcd(x_i - x_j, N) > 1$. So, we will do a trick. We will generate $x_1, x_2, \ldots, x_{2k}$ in a special way which will require testing only k pairs.

For any polynomial $f(x) = c_n x^n + \ldots + c_1 x + c_0$ and a modulus $N \in \mathbb{N}$

$$a \equiv_N b \quad \Rightarrow \quad \begin{array}{ccccc} c_n a^n & + \ldots + & c_1 a & + c_0 & = f(a) \\ \| & & \| & \| & \| \\ c_n b^n & + \ldots + & c_1 b & + c_0 & = f(b). \end{array}$$

Fix a polynomial $f(x)$ and generate a sequence $x_1, x_2, \ldots$ defined by

- x_1 is randomly chosen in $[1, N-1]$, and
- $x_{i+1} = f(x_i) \,\%\, N$.

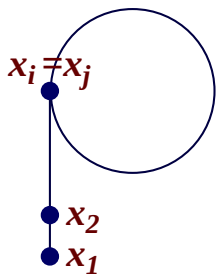
The sequence $x_1, x_2, \ldots$ is not random and does not satisfy assumptions of the birthday problem. Yet, it works in practice and is called **pseudo-random**.

The function f here plays the role of a pseudo-random number generator.

Pollard's rho algorithm

Observe that the sequence $x_1, x_2, x_3, \ldots$

- has a collision (because there are finitely many numbers modulo N)
- is periodic.



$$
\begin{aligned}
x_i \equiv_p x_j \quad &\Rightarrow \quad x_{i+1} = f(x_i) \equiv_p f(x_j) = x_{j+1} \\
&\Rightarrow \quad x_{i+2} = f(x_{i+1}) \equiv_p f(x_{j+1}) = x_{j+2} \\
&\Rightarrow \quad x_{i+3} = f(x_{i+2}) \equiv_p f(x_{j+2}) = x_{j+3} \\
&\Rightarrow \quad \text{etc.}
\end{aligned}
$$

Proposition

If $x_i \equiv_p x_j$, then $x_{i'} \equiv_p x_{2i'}$ for some $1 \leq i' < j$.

Hence, $j - i$ is a period for the sequence and we can choose $i' = q(j - i) \in [i, j)$.

(The algorithm)

Use the polynomial function $f(x) = x^2 + 1$ to generate $x_1, x_2, \ldots, x_{2\sqrt[4]{N}}$ and if $\gcd(x_{2i} - x_i, N) = d > 1$, then output d.

Pollard's rho algorithm: example

For instance, for $N = 8051$

$$x_1 = 5 \qquad x_3 = 677 \qquad x_5 = 2839 \qquad x_7 = 1848$$
$$x_2 = 26 \qquad x_4 = 7474 \qquad x_6 = 871 \qquad x_8 = 1481$$

$$\begin{aligned}
\gcd(x_2 - x_1, 8051) &= \gcd(26 - 5, 8051) = 1,\\
\gcd(x_4 - x_2, 8051) &= \gcd(7474 - 26, 8051) = 1,\\
\gcd(x_6 - x_3, 8051) &= \gcd(871 - 677, 8051) = 97,\\
\gcd(x_8 - x_4, 8051) &= \gcd(1481 - 7474, 8051) = 1.
\end{aligned}$$

We could stop the process after the 3rd step.