

Review

Formulas for roots modulo primes, from Fermat's Little Theorem

Euler's criterion for existence of roots modulo primes. Uses existence of **primitive roots** modulo primes.

Improved test for whether b is a primitive root modulo prime p .

Sun-Ze's theorem, congruences with **composite** moduli, using extended Euclid

To take roots modulo composites, *factoring* together with previous formulas suffices, via Sun-Ze and Euclid.

Conversely, square root oracle modulo $p \cdot q$ can be used to (probabilistically) take square roots modulo n .

Factoring versus primality testing

RSA is *feasible* because *primality testing* is *easy*.

RSA seems *secure* because *factoring* is *hard*.

Trial division is by far *not* the best method for attempting to factor large integers.

Currently, roughly speaking, in the worst-case scenario integers with more than 20 decimal digits take an hour to either prove prime or factor by trial division.

There are several much better factorization methods, although *none* of them is sufficiently fast to be able to factor 200-digit numbers.

Thus, for example, RSA with 1024-bit moduli still seems secure.

Pollard's rho (1976)

Try to beat the \sqrt{N} steps trial division needs to factor N .

Use the **birthday paradox**: if somewhat more than \sqrt{n} things are chosen from n (with replacement) then the probability is $> \frac{1}{2}$ that at least 2 choices are identical.

First try: Suppose that $N = p \cdot M$ with p prime and $p < \sqrt{N}$. If we choose more than \sqrt{p} integers x_i at random, then the probability is $> \frac{1}{2}$ that for some $i \neq j$

$$x_i = x_j \pmod{p}$$

so likely for **some** pair

$$\gcd(x_i - x_j, N) = \text{proper factor of } N$$

But we might have to compare

$$\sqrt{p} \cdot \sqrt{p} = p \sim \sqrt{N}$$

pairs, no better than trial division.

Remarks

The probability is roughly $\frac{1}{\sqrt{N/p}} \sim 0$ that $x_i = x_j \pmod N$ for the same specific indices i, j for which

$$x_i = x_j \pmod p$$

That is, it is likely that for *some* pair we have *only* the equality modulo p , not modulo the whole N . That is,

$$\gcd(x_i - x_j, N) = \mathbf{proper} \text{ factor of } N$$

rather than getting N as the *gcd*.

(We compute *gcds* quickly by the **Euclidean algorithm**, of course.)

Second try at Pollard's rho

Since we would have had trouble making a large number of truly random choices anyway, let's stipulate that we choose the x_i 's in a more structured way, in a sort of **random walk** in \mathbf{Z}/N . Let $f : \mathbf{Z}/N \rightarrow \mathbf{Z}/N$ be a deterministic '**random**' function, fix x_0 , and define

$$x_{i+1} = f(x_i)$$

Since f is deterministic

$$f(x_i) = x_j \implies$$

$$f(x_{i+1}) = f(f(x_i)) = f(x_j) = x_{j+1}$$

So if the walk enters a **cycle** it stays there. We use **Floyd's cycle-detection trick**:

Floyd's cycle-detection trick:

Fix x_o , define $y_o = x_o$, and define

$$x_{i+1} = f(x_i) \quad y_{i+1} = f(f(y_i))$$

so the y_i 's take the same walk but twice as fast.

Once the cycle is entered, the y 's walk one unit faster than the x 's, so in fewer additional steps than the cycle length, $x_j = y_j \pmod p$.

In summary: the initial walk plus cycle takes $\sqrt{p} \leq N^{1/4}$ steps, and another \sqrt{p} for the y 's to catch the x 's modulo p , so

$$2\sqrt{p} \leq 2N^{1/4} \quad \text{steps}$$

to find the factor p of N .

Implementation of Pollard's rho

To attempt to factor N :

Define function $f(x) = x^2 + 2 \% N$.

Take $x = y = 2$

Repeat:

replace x by $f(x)$

replace y by $f(f(y))$

compute $g = \gcd(x - y, N)$

stop if this gcd is $1 < g < N$

It is not easy to decide how long to let this loop run, but, using the birthday paradox as a guide, we might need $> N^{1/4}$ steps to find a factor.

Beware: *Failure to find a proper factor by Pollard's rho does not prove primality!*

Remarks

Again, as with other modern factorization attacks, failure to factor does *not* prove primality.

*Instead, before attempting to factor, run some **pseudoprime tests**. Only if a pseudoprime test proves a large integer composite should you attempt to factor it!*

Usually run Fermat base 2 first, since it's quick and simple.

Of course, pseudoprime tests are fast, and will detect compositeness quickly though not telling a factor.

And in real life one would use trial division for a little while to be sure to remove *small* proper factors.

More remarks

Of course, the function $f(x) = x^2 + 2 \pmod N$ is not a ‘random function’, but in practice it does seem to mix things around sufficiently to make Pollard’s rho work.

Note that there is some conflict between the implicit requirement that f be easily computable, and the need to have it be random.

And the requirement that f be *deterministic random* means that f always produces the same output for the same input (that’s the *deterministic* part), but mixes \mathbf{Z}/N well (that’s the *random* part).

A better word for this mixing property is **ergodic**.

Small examples of Pollard's rho

To factor $2491 = 47 \cdot 53$

Initialize $x = y = 2$

Replace x by $x^2 + 2 \pmod{2491} = 6$ and y by $(y^2 + 2)^2 + 2 \pmod{2491} = 38$. Compute $\gcd(6 - 38, 2491) = 1$.

Replace x by $x^2 + 2 \pmod{2491} = 38$, and y by $(y^2 + 2)^2 + 2 \pmod{2491} = 969$. Compute $\gcd(38 - 969, 2491) = 1$.

Replace x by $x^2 + 2 \pmod{2491} = 1446$, and y by $(y^2 + 2)^2 + 2 \pmod{2491} = 810$. Compute $\gcd(1446 - 810, 2491) = 53$. And 53 is a proper factor of 2491.

Factor $10403 = 101 \cdot 103$. Set $x = y = 2$.

Replace x by $x^2 + 2 \pmod{10403} = 6$, and y by $(y^2 + 2)^2 + 2 \pmod{10403} = 38$. Compute $\gcd(6 - 38, 10403) = 1$.

Replace x by $x^2 + 2 \pmod{10403} = 38$, and y by $(y^2 + 2)^2 + 2 \pmod{10403} = 10318$. Compute $\gcd(38 - 10318, 10403) = 1$.

Replace x by $x^2 + 2 \pmod{10403} = 1446$, and y by $(y^2 + 2)^2 + 2 \pmod{10403} = 6471$. Compute $\gcd(1446 - 6471, 10403) = 1$.

Replace x by $x^2 + 2 \pmod{10403} = 10318$, and y by $(y^2 + 2)^2 + 2 \pmod{10403} = 4926$. Compute $\gcd(10318 - 4926, 10403) = 1$.

Replace x by $x^2 + 2 \pmod{10403} = 7227$, and y by $(y^2 + 2)^2 + 2 \pmod{10403} = 4617$. Compute $\gcd(7227 - 4617, 10403) = 1$.

Replace x by $x^2 + 2 \pmod{10403} = 6471$, and y by $(y^2 + 2)^2 + 2 \pmod{10403} = 6883$. Compute $\gcd(6471 - 6883, 10403) = 103$ a proper factor.

Real examples of Pollard's rho

In less than 10 seconds total for this page:

2661 steps to find factor

10000103 of 100001220001957

14073 steps to find factor

100000007 of 100000130000000861

9630 steps to find factor

1000000103 of 10000001100000000721

(Even larger...) 129665 steps for factor

10000000019 of 100000001220000001957

162944 steps for factor

100000000103 of 100000000106000000000309

Yet larger...

89074 steps for

10000000000039 of
10000000000160000000004719

12 seconds, 584003 steps for

10000000000037 of
1000000000001660000000004773

2 minutes, 5751662 steps for

100000000000031 of
100000000000016400000000004123

*But still this is slowing down, and we're
nowhere near factoring a 200-digit number...*

Pollard's $p - 1$

This method can very often find proper factors which are primes p with $p - 1$ most likely to be a product of primes from a specified list called a *factor base*.

Primes p such that $p - 1$ has only relatively small prime factors are called **weak primes**. *If p is weak, then $n = p \cdot q$ can be easily factored even if p and q are large.*

We start from Fermat's Little Theorem: for prime p , for any b not divisible by p

$$b^{p-1} \equiv 1 \pmod{p}$$

That is, if p is a divisor of an integer n ,

$$p \mid \gcd(b^{p-1} - 1, n)$$

Unless we are unlucky, the whole n does *not* divide $b^{p-1} - 1$, so we have found a *proper* divisor.

For example, for prime $p = 1 + 2^t$ dividing n we have

$$t \sim \log_2 p \leq \log_2 n = \frac{\log n}{\log 2}$$

Let $T = \text{floor}(\log n / \log 2)$. Without knowing t , except that $T \geq t$, for any b not divisible by p

$$\begin{aligned} b^{2^T} &= (b^{2^t})^{2^{T-t}} = (b^{p-1})^{2^{T-t}} \\ &= 1^{2^{T-t}} = 1 \pmod{p} \end{aligned}$$

Similarly, if prime $p^1 + 2^s 3^r$ divides n , let

$$S = \text{floor}(\log n / \log 2) \geq s$$

$$T = \text{floor}(\log n / \log 3) \geq t$$

$$\begin{aligned} b^{2^S 3^T} &= (b^{2^s 3^t})^{2^{S-s} 3^{T-t}} = (b^{p-1})^{2^{S-s} 3^{T-t}} \\ &= 1^{2^{S-s} 3^{T-t}} = 1 \pmod{p} \end{aligned}$$

Pollard's $p - 1$ method (to factor n) with seed $b=3$ and factor base B says

For each prime q in the list B

compute $e = \text{floor}(\log n / \log q)$

replace b by $b^{q^e} \% n$

compute $\text{gcd}(n, b - 1)$

If this gcd is a proper factor of n , stop

Compute gcd by Euclid, and exponentials by fast modular exponentiation.

*If you fall out of the loop, it does **not** mean that n is prime.*

An integer n with prime factors from a specified list of primes B is called **B -smooth**.

The question of choice of factor base B to optimize this algorithm is non-trivial. If the factor base is too small many primes will be missed, but if the list is too long the runtime will be prohibitive.

Example: Try to find a prime factor p of 9991 such that $p - 1$ is $(2, 3)$ -smooth:

Take factor base (list of primes) $(2, 3)$, and initialize the seed $b = 3$.

Compute $13 = \text{floor}(\log(9991)/\log(2))$.
Replace $b = 3$ by

$$b = 3^{2^{13}} \% 9991 = 229$$

Compute $\text{gcd}(9991, 229 - 1) = 1$.

Compute $8 = \text{floor}(\log(9991)/\log(3))$.
Replace $b = 229$ by

$$b = 229^{3^8} \% 9991 = 3202$$

Compute $\text{gcd}(9991, 3202 - 1) = 97$.

Thus, we find the proper factor 97.

And $97 = 1 + 2^5 \cdot 3$.

Example: Try to find a prime factor p of $59431 = 577 \cdot 103$ such that $p - 1$ is $(2, 3)$ -smooth:

Compute $15 = \text{floor}(\log(59431)/\log(2))$.

Replace $b = 3$ by

$$b = 3^{2^{15}} \% 59431 = 53447$$

Compute $\text{gcd}(59431, 53447 - 1) = 1$.

Compute $10 = \text{floor}(\log(59431)/\log(3))$.

Replace $b = 53447$ by

$$b = 53447^{3^{10}} \% 59431 = 52508$$

Compute $\text{gcd}(59431, 52508 - 1) = 577$.

Thus, we find the proper factor 577.

And $577 = 1 + 2^6 \cdot 3^2$.

Ironically, sometimes if *all* the prime factors are weak, Pollard's $p - 1$ fails:

Example: Try to find a prime factor p of $55969 = 577 \cdot 97$ such that $p - 1$ is $(2, 3)$ -smooth:

Replace $b = 3$ by $b = 3^{2^{15}} \% 55969$.

This gives $b = 49408$. Compute $\gcd(55969, 49408 - 1) = 1$.

Compute $9 = \text{floor}(\log(55969)/\log(3))$.

Replace $b = 49408$ by $b = 49408^{3^9} \% 55969$.
This gives $b = 1$.

Compute $\gcd(55969, 1 - 1) = 55969$. No proper factor p with $p - 1$ smooth was found.

Remark: While an exponent which is itself a power of 2 easily fits into the fast modular exponentiation algorithm, powers of 3 or other primes are slightly more trouble.

But things still work: for example

$$b^{5^{12}} = (\dots (b^5)^5 \dots)^5$$

Thus, to compute $b^{5^t} \% n$

for $\ell = 1, \dots, t$:

let $c = b^2 \% n$

replace c by $c \cdot c \% n$

replace b by $b \cdot c \% n$

The final value of b is $b^{5^t} \% n$.

That is, the inner block is hard-coded to take the fifth power of b and reduces it modulo n . To take the 5^t power means to do this t times.

Definition: A **strong prime** is a prime p such that $p - 1$ is divisible by at least one *large* prime p_1 , meaning p_1 is near \sqrt{p} .

Further, for p to be *extra strong*, we might require that p_1 itself be strong, that is, divisible by a prime p_2 as large as $\sqrt{p_1}$.

For p to be *extra extra strong*, we might require that p_2 be strong, that is, divisible by a prime p_3 as large as $\sqrt{p_2}$.

Remark: In reality, if the factor base B is very small, it is simpler to test directly for divisors d of n with $d - 1$ B -smooth, rather than run Pollard's $p - 1$ test.

Remark: There is a similar test which finds prime factors p of n if $p + 1$ is *smooth*.

Remark: For *larger* large integers these issues seem to be of diminishing importance (?!) because a lower proportion of primes are weak.

More careful analysis of Pollard's rho

The birthday paradox heuristic that a prime factor p would be found in perhaps $2 \cdot \sqrt{p}$ is initially helpful, but in the end misleading.

The function $f : \mathbf{Z}/p \rightarrow \mathbf{Z}/p$, taken to be $f(x) = x^2 + 2 \% n$, might instead be viewed as a *permutation* (except that it's not a bijection, but never mind).

Then the number $\sim \sqrt{p}$ from the birthday paradox should be replaced by **the expected cycle length**, averaged over elements and permutations.

As a first approximation, we would hope that with a randomly chosen permutation f of p things, with a randomly chosen element j of the p things, that the cycle

$$j \quad f(j) \quad f^2(j) \quad f^3(j) \quad \dots$$

would be of length $\sim \sqrt{p}$.

This is *not* asking that the average *order* of a permutation of p things be $\sim \sqrt{p}$, since we are paying attention only to the trajectory of a single element.

And this still ignores the fact that f actually gives a permutation of \mathbf{Z}/n with p a secret divisor of n .

In the simple but relevant case that $n = p \cdot q$ with distinct primes p and q . By Sun-Ze's theorem, our $f(x) = x^2 + 2 \pmod{pq}$ has the peculiar property that it falls apart into a *cartesian product* permutation, meaning that under the bijection

$$j : x \pmod{pq} \rightarrow (x \pmod{p}, x \pmod{q})$$

the bijection f becomes a *pair* of permutations f_1 and f_2 of p things and of q things, respectively, in the sense that

$$j(f(x)) = (f_1(x) \pmod{p}, f_2(x) \pmod{q})$$

There are only $p! \cdot q! \ll (pq)!$ permutations of pq things that have this property of being cartesian products of permutations of p things and of q things.

So such a choice is *not* typical among all permutations of pq things, and thus anything we conclude about *typical* permutations of pq things may in fact typically *fail* in our special class.

Further, from the very outset our function $f : \mathbf{Z}/n \rightarrow \mathbf{Z}/n$ is not constrained to be of a sort which would *descend* to give a well-defined function modulo p at all! That is, we need *not* choose a function f such that

$$f(x) \% p = f(x \% p)$$

as *is* the case with polynomials such as $x^2 + 2$.

But the constraint that the function be easily computable will likely induce us to use a polynomial.

Watch out!

Linear polynomials $f(x) = ax + b \pmod n$ are *not* good to use in Pollard's rho, as one can verify by *linear algebra*. (Hint: Think about eigenvectors and eigenvalues. This is similar to the study of **linear congruential generators**.)

Nearly any polynomial of the form $f(x) = x^2 + c$ will work, but **not** $c = 0$ nor $c = 1$ nor $c = -1$.

Since *quadratic* polynomials seem to work, there's no practical need to use cubic or higher degree polynomials.

And $f(x) = x^2 + 2 \pmod p$ isn't injective, anyway, so the permutation model may still be too naive and fail to capture some key aspects.