

Number Theory

Aayush Bajaj

July 10, 2026

Contents

1	divisibility and the euclidean algorithm	1
1.1	euclid's algorithm	2
1.2	the extended algorithm	2
2	bezout's identity	2
3	the fundamental theorem of arithmetic	3
4	modular arithmetic	3
4.1	the chinese remainder theorem	3
5	fermat and euler	4
6	primality testing	4
7	RSA: the payoff	5
7.1	the scheme	5
7.2	why decryption works	5
7.3	a toy run, honestly computed	5
8	see also	7

number theory studies the integers on their own terms — divisibility, primes, remainders — and for two millennia it was the purest of pure mathematics, prized precisely because it was useless. then in 1977 the RSA cryptosystem turned euclid, fermat and euler into load-bearing internet infrastructure. this page walks the classical results in dependency order and ends at that payoff (Epp, Susanna S., 2019).

hardy, 1940: 'no one has yet discovered any warlike purpose to be served by the theory of numbers'. he was off by about 37 years

1 divisibility and the euclidean algorithm

for integers a, b , say $d \mid a$ (“ d divides a ”) if $a = dk$ for some integer k . the bedrock fact is the **division algorithm**: for any a and $b > 0$ there exist *unique* q, r with

$$a = qb + r, \quad 0 \leq r < b. \quad (1)$$

(existence: take $r = a - qb$ minimal non-negative — the well-ordering principle; uniqueness: two candidate remainders differ by a multiple of b yet lie in $[0, b)$.) the **greatest common divisor** $\gcd(a, b)$ is the largest integer dividing both.

1.1 euclid's algorithm

theorem (euclid, ~300 BCE). if $a = qb + r$ then $\gcd(a, b) = \gcd(b, r)$.

proof. if $d \mid a$ and $d \mid b$ then $d \mid a - qb = r$; if $d \mid b$ and $d \mid r$ then $d \mid qb + r = a$. so the pairs (a, b) and (b, r) have exactly the same common divisors, hence the same greatest one. ■

iterate: replace (a, b) by $(b, a \bmod b)$ until the remainder is 0; the last nonzero remainder is the gcd. **termination:** the remainders form a strictly decreasing sequence of non-negative integers, so by well-ordering they hit 0. the run for $\gcd(240, 46)$:

$$\begin{aligned} 240 &= 5 \cdot 46 + 10 \\ 46 &= 4 \cdot 10 + 6 \\ 10 &= 1 \cdot 6 + 4 \\ 6 &= 1 \cdot 4 + 2 \\ 4 &= 2 \cdot 2 + 0 \quad \Rightarrow \gcd = 2. \end{aligned}$$

the worst case is consecutive fibonacci numbers — every quotient is 1, nothing shrinks fast — and lamé's theorem turns that observation into the bound: the number of steps is $O(\log \min(a, b))$ (Graham, Ronald L. and Knuth, Donald E. and Patashnik, Oren, 1994).

1.2 the extended algorithm

running the same ladder *backwards* expresses the gcd as an integer combination. from the run above:

$$\begin{aligned} 2 &= 6 - 1 \cdot 4 = 6 - (10 - 6) = 2 \cdot 6 - 10 \\ &= 2(46 - 4 \cdot 10) - 10 = 2 \cdot 46 - 9 \cdot 10 \\ &= 2 \cdot 46 - 9(240 - 5 \cdot 46) = 47 \cdot 46 - 9 \cdot 240. \end{aligned}$$

so $240 \cdot (-9) + 46 \cdot 47 = 2$ — machine-checked below. the recursive implementation carries this as an invariant: if $\gcd(b, a \bmod b) = bx' + (a \bmod b)y'$, substitute $a \bmod b = a - \lfloor a/b \rfloor b$ to get coefficients for (a, b) .

2 bezout's identity

theorem. for integers a, b not both zero, there exist integers x, y with $ax + by = \gcd(a, b)$; moreover $\gcd(a, b)$ is the *smallest positive* integer of that form.

proof. let $S = \{ax + by : x, y \in \mathbb{Z}\} \cap \mathbb{Z}^+$. S is non-empty ($a^2 + b^2 \in S$), so by well-ordering it has a least element $g = ax_0 + by_0$. divide: $a = qg + r$ with $0 \leq r < g$; then $r = a(1 - qx_0) + b(-qy_0)$ is of the same form, and $r > 0$ would contradict minimality — so $r = 0$ and $g \mid a$. symmetrically $g \mid b$. finally any common divisor d of a and b divides $ax_0 + by_0 = g$, so g is the greatest. ■

two consequences that carry the rest of the page:

- **modular inverses:** if $\gcd(a, n) = 1$ then $ax + ny = 1$ gives $ax \equiv 1 \pmod{n}$ — the extended algorithm *computes* $a^{-1} \bmod n$.
- **euclid's lemma:** if p is prime and $p \mid ab$, then $p \mid a$ or $p \mid b$. (if $p \nmid a$ then $\gcd(p, a) = 1$, so $1 = px + ay$, so $b = pbx + aby$ — both terms divisible by p .)

3 the fundamental theorem of arithmetic

theorem. every integer $n \geq 2$ factors into primes, uniquely up to order.

proof sketch. **existence** is strong induction (done on the **logic** page): n prime is done; otherwise $n = ab$ with both factors smaller, and each factors by hypothesis.

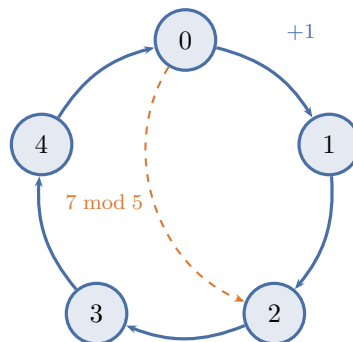
uniqueness: suppose $p_1 p_2 \cdots p_s = q_1 q_2 \cdots q_t$ are two prime factorisations of the same number. p_1 divides the right-hand side, so by euclid's lemma (applied repeatedly) p_1 divides some q_j , and primality forces $p_1 = q_j$. cancel and recurse; a minimal counterexample dies of embarrassment. ■

uniqueness is not free — it genuinely needs euclid's lemma, and in rings without it (e.g. $\mathbb{Z}[\sqrt{-5}]$, where $6 = 2 \cdot 3 = (1 + \sqrt{-5})(1 - \sqrt{-5})$) factorisation exists but is not unique.

*the repair job — ideals —
invented modern algebra. good
bug*

4 modular arithmetic

fix $n \geq 1$. write $a \equiv b \pmod{n}$ iff $n \mid a - b$. congruence is an equivalence relation compatible with arithmetic: if $a \equiv a'$ and $b \equiv b'$ then $a + b \equiv a' + b'$ and $ab \equiv a'b'$ — so the n residue classes $\mathbb{Z}/n\mathbb{Z} = \{0, 1, \dots, n-1\}$ form a ring. the mental picture is a clock:



$\mathbb{Z}/5\mathbb{Z}$: five residue classes, arithmetic on a cycle

Figure 1: arithmetic modulo 5 lives on a directed cycle: the successor map $x \mapsto x + 1$ walks it, and $7 = 5 + 2$ lands on 2.

division is the subtle operation: a is invertible mod n iff $\gcd(a, n) = 1$ (bezout, both directions). the invertible residues form the **unit group** $(\mathbb{Z}/n\mathbb{Z})^\times$, of size $\varphi(n)$ — euler's totient, counting $1 \leq a \leq n$ coprime to n .

4.1 the chinese remainder theorem

theorem (CRT). if $n_1, \dots, n_k$ are pairwise coprime, the system $x \equiv a_i \pmod{n_i}$, $i = 1, \dots, k$, has a solution, unique modulo $N = n_1 n_2 \cdots n_k$.

construction. let $N_i = N/n_i$. since $\gcd(N_i, n_i) = 1$, the extended euclidean algorithm gives $M_i = N_i^{-1} \bmod n_i$. then

$$x \equiv \sum_{i=1}^k a_i N_i M_i \pmod{N} \quad (2)$$

works: modulo n_i , every term with $j \neq i$ vanishes ($n_i \mid N_j$), and the surviving term is $a_i N_i M_i \equiv a_i \cdot 1$. uniqueness: two solutions differ by a multiple of every n_i , hence of N by pairwise coprimality. ■

worked example (sun tzu, 3rd century): find x with $x \equiv 2 \pmod{3}$, $x \equiv 3 \pmod{5}$, $x \equiv 2 \pmod{7}$. here $N = 105$:

i	a_i	n_i	$N_i = N/n_i$	$N_i \bmod n_i$	$M_i = N_i^{-1} \bmod n_i$	$a_i N_i M_i$
1	2	3	35	2	2	140
2	3	5	21	1	1	63
3	2	7	15	1	1	30

sum = 233 $\equiv 23 \pmod{105}$. check: $23 = 7 \cdot 3 + 2 = 4 \cdot 5 + 3 = 3 \cdot 7 + 2$. structurally, CRT says the ring $\mathbb{Z}/N\mathbb{Z}$ is a product of the rings $\mathbb{Z}/n_i\mathbb{Z}$ — one big clock is k small independent clocks.

this is also why the totient is multiplicative: units of a product ring are pairs of units

5 fermat and euler

fermat's little theorem. if p is prime and $p \nmid a$, then $a^{p-1} \equiv 1 \pmod{p}$.

proof. multiplication by a is injective on $\{1, 2, \dots, p-1\} \bmod p$ (cancel a : it is invertible), and an injection of a finite set into itself is a bijection — so $a \cdot 1, a \cdot 2, \dots, a \cdot (p-1)$ is a rearrangement of $1, 2, \dots, p-1$. multiply everything together:

$$a^{p-1} (p-1)! \equiv (p-1)! \pmod{p}, \quad (3)$$

and $(p-1)!$ is coprime to p , so cancel it. ■

euler's theorem. if $\gcd(a, n) = 1$, then $a^{\varphi(n)} \equiv 1 \pmod{n}$.

same proof, run over the unit group instead: multiplication by a permutes the $\varphi(n)$ units, multiply them all, cancel the product.¹ the totient itself is computable from the factorisation: $\varphi(p^k) = p^k - p^{k-1}$ (count everything, exclude the multiples of p), and φ is multiplicative on coprime arguments by CRT, giving

$$\varphi(n) = n \prod_{p \mid n} \left(1 - \frac{1}{p}\right). \quad (4)$$

for RSA the case that matters: $n = pq$ with distinct primes gives $\varphi(n) = (p-1)(q-1)$.

6 primality testing

fermat's theorem is a compositeness detector: find a base a with $a^{n-1} \not\equiv 1 \pmod{n}$ and n is certifiably composite — no factor exhibited, the certificate is pure arithmetic. the converse fails, and not just occasionally:

¹the group-theoretic one-liner: in a finite group every element's order divides the group order (lagrange), and $|(\mathbb{Z}/n\mathbb{Z})^\times| = \varphi(n)$. fermat is the case $n = p$, where the unit group has order $p-1$.

- **fermat test**: pick random a , check $a^{n-1} \equiv 1 \pmod{n}$. composites usually fail it, but the **carmichael numbers** — $561 = 3 \cdot 11 \cdot 17$ is the smallest — satisfy $a^{n-1} \equiv 1$ for *every* a coprime to n . there are infinitely many of them (alford–granville–pomerance, 1994), so the fermat test alone is never enough.
- **miller–rabin** (sketch): write $n - 1 = 2^s d$ with d odd, and inspect the chain $a^d, a^{2d}, a^{4d}, \dots, a^{2^s d} \pmod{n}$. if n were prime, the only square roots of 1 would be ± 1 , so the chain could only reach 1 starting from $a^d = 1$ or stepping through a -1 . a base whose chain reaches 1 any other way — via a nontrivial square root of 1 — is a **witness** that n is composite. for odd composite n , at least $3/4$ of bases are witnesses, so k random rounds are fooled with probability at most 4^{-k} . this (plus a fixed deterministic base set for word-sized n) is what real libraries run.

7 RSA: the payoff

everything above assembles into public-key cryptography. the asymmetry: multiplying two primes is instant; factoring the product back apart is — as far as public knowledge goes — hard.

7.1 the scheme

- **keygen**: choose large distinct primes p, q (found by sampling odd numbers and miller–rabin-testing them). set $n = pq$ and $\varphi = (p-1)(q-1)$. pick e with $\gcd(e, \varphi) = 1$, and compute $d = e^{-1} \bmod \varphi$ — extended euclid again. **publish** (n, e) ; **keep** d secret (and destroy p, q, φ).
- **encrypt** a message encoded as an integer $0 \leq m < n$: $c = m^e \bmod n$. anyone can — the key is public.
- **decrypt**: $m = c^d \bmod n$. only the holder of d can.

both operations are modular exponentiation — $O(\log \text{exponent})$ multiplications by repeated squaring — so the entire scheme runs on this page’s toolkit.

7.2 why decryption works

$ed \equiv 1 \pmod{\varphi}$ means $ed = 1 + k(p-1)(q-1)$ for some integer k . claim: $m^{ed} \equiv m \pmod{n}$ for *every* m , not just the units. check modulo p and q separately and let CRT glue the results:

- if $p \nmid m$: $m^{ed} = m \cdot (m^{p-1})^{k(q-1)} \equiv m \cdot 1 \pmod{p}$ by fermat’s little theorem.
- if $p \mid m$: both sides are $\equiv 0 \pmod{p}$.

the same argument runs mod q ; since $p \neq q$ are primes, CRT lifts $m^{ed} \equiv m$ from both small moduli to their product n . ■

7.3 a toy run, honestly computed

with $p = 61$, $q = 53$: $n = 3233$, $\varphi = 3120$, $e = 17$, and extended euclid gives $d = 2753$ (indeed $17 \cdot 2753 = 46801 = 15 \cdot 3120 + 1$). encrypting $m = 65$:

$$c = 65^{17} \bmod 3233 = 2790, \quad 2790^{2753} \bmod 3233 = 65. \quad (5)$$

textbooks often invoke euler’s theorem directly, which silently assumes the message is coprime to n — the fermat-plus-CRT route covers every message

the code below performs exactly these computations. what makes the toy a toy: $n = 3233$ factors by trial division in microseconds, and deterministic encryption of a tiny message space is guessable — real deployments use 2048-bit-plus moduli, randomised padding (OAEP), and hybrid schemes where RSA only wraps a symmetric key. one honest caveat while we are here: recovering the private exponent d from (n, e) is provably as hard as factoring n , but breaking a single ciphertext (the “RSA problem”) is *not* known to be equivalent to factoring.

```

1  # --- extended euclid: returns (g, x, y) with a*x + b*y = g = gcd(a, b) -----
2  def egcd(a, b):
3      if b == 0:
4          return a, 1, 0
5      g, x, y = egcd(b, a % b)
6      return g, y, x - (a // b) * y      # unwind: g = b*x' + (a mod b)*y'
7
8  def modinv(a, m):
9      g, x, _ = egcd(a, m)
10     assert g == 1, "not invertible"
11     return x % m
12
13 g, x, y = egcd(240, 46)
14 print(f"egcd(240, 46): gcd = {g}, 240*({x}) + 46*({y}) = {240*x + 46*y}")
15
16 # --- modular exponentiation by repeated squaring -----
17 def modexp(b, e, m):
18     r = 1
19     b %= m
20     while e:
21         if e & 1:
22             r = r * b % m
23         b = b * b % m
24         e >>= 1
25     return r
26
27 print(f"modexp(7, 128, 13) = {modexp(7, 128, 13)} (check: {pow(7, 128, 13)})")
28 print(f"fermat: 2^560 mod 561 = {modexp(2, 560, 561)} <- 561 = 3*11*17 lies (carmichael)")
29 print(f"5^560 mod 561 = {modexp(5, 560, 561)}")
30
31 # --- toy RSA -----
32 p, q = 61, 53
33 n, phi = p * q, (p - 1) * (q - 1)
34 e = 17      # public exponent, coprime to phi
35 d = modinv(e, phi)      # private exponent: e*d = 1 mod phi
36 print(f"\nkeygen: p={p} q={q} n={n} phi={phi} e={e} d={d} (e*d mod phi = {e*d % phi})")
37
38 m = 65      # the message, as a number < n
39 c = modexp(m, e, n)      # encrypt with the PUBLIC key (e, n)
40 m2 = modexp(c, d, n)      # decrypt with the PRIVATE key (d, n)
41 print(f"encrypt: {m}^{{e}} mod {n} = {c}")
42 print(f"decrypt: {c}^{{d}} mod {n} = {m2} round trip {'ok' if m2 == m else 'BROKEN'}")

```

```

egcd(240, 46): gcd = 2, 240*(-9) + 46*(47) = 2
modexp(7, 128, 13) = 3 (check: 3)
fermat: 2^560 mod 561 = 1 <- 561 = 3*11*17 lies (carmichael)
5^560 mod 561 = 1

```

```

keygen: p=61 q=53 n=3233 phi=3120 e=17 d=2753 (e*d mod phi = 1)
encrypt: 65^17 mod 3233 = 2790

```

decrypt: $2790^{2753} \bmod 3233 = 65$ round trip ok

note the carmichael lines: 561 passes the fermat test at bases 2 and 5 (and every other coprime base) while being flagrantly composite — exactly why miller–rabin exists.

8 see also

- [logic](#) — induction and the well-ordering principle, load-bearing under every proof here
- [sets, functions and sequences](#) — equivalence relations, of which congruence is the star example
- [combinatorics](#) — counting arguments, totient-style
- [classical algorithms](#) — repeated squaring as the exemplar of divide-and-conquer

references

Epp, Susanna S. (2019). *Discrete Mathematics with Applications*, Cengage Learning.

Graham, Ronald L. and Knuth, Donald E. and Patashnik, Oren (1994). *Concrete Mathematics: A Foundation for Computer Science*, Addison-Wesley.