Solutions to Daily Due Sep 18 (AI generated)

Math 4440/5440, Coding and Cryptography, Fall 2026

Generated by Claude for Katherine E. Stange
(all computations computer-verified)

Problem 2: how $x^2 \equiv y^2 \pmod{n}$ with $x \not\equiv \pm y$ helps you factor

The idea

Suppose $x^2 \equiv y^2 \pmod{n}$. Then n divides the difference of squares,

$$x^2 - y^2 = (x-y)(x+y).$$

So every prime factor of n divides $(x-y)(x+y)$, and hence divides $x-y$ or $x+y$. There are three possibilities.

- All of n divides $x-y$; that is, $x \equiv y \pmod{n}$.
- All of n divides $x+y$; that is, $x \equiv -y \pmod{n}$.
- Otherwise, n's prime factors are *split* between the two factors: some divide $x-y$ and the rest divide $x+y$, with neither getting all of them.

The hypothesis $x \not\equiv \pm y \pmod{n}$ rules out the first two cases. So we are in the third case, and then

$$\gcd(x-y,\, n)$$

is a factor of n which is neither 1 nor n: a **nontrivial factor**. (So is $\gcd(x+y, n)$, and in fact for $n = pq$ the two gcds are the two prime factors.) Computing a gcd is fast — the Euclidean algorithm — so the whole difficulty of factoring n has been moved into the difficulty of *finding* such a pair x, y.

An example

Take $n = 91$. Note $10^2 = 100 \equiv 9 = 3^2 \pmod{91}$, while $10 \not\equiv \pm 3 \pmod{91}$. Then

$$\gcd(10-3,\, 91) = \gcd(7, 91) = 7, \qquad \gcd(10+3,\, 91) = \gcd(13, 91) = 13,$$

and indeed $91 = 7 \cdot 13$.

A non-example, to see why the condition $x \not\equiv \pm y$ is needed: $12^2 \equiv 79^2 \pmod{91}$ is true, but $79 \equiv -12$, and $\gcd(12-79, 91) = \gcd(-67, 91) = 1$ tells us nothing.

Why this is the engine of modern factoring

Every method in the "squares" family — Fermat factoring, Pollard's methods, the quadratic sieve, the number field sieve — is a way of manufacturing a congruence of squares $x^2 \equiv y^2 \pmod{n}$. Fermat factoring looks for it directly ($x^2 - n$ a perfect square). The quadratic sieve manufactures it by multiplying together many relations whose prime factorizations combine into a square. Once you have the congruence, a single gcd finishes the job.

It is also exactly why a random such pair is useful: if $n = pq$ and x is a random square root of a square y^2, there are four square roots, two of which ($\pm y$) are useless and two of which factor n. So each random congruence of squares factors n with probability about $1/2$.

Problem 3: patterns in the dynamical portraits modulo n

Here are the patterns that show up in the *Modular Arithmetic Playground* as you step the modulus with the arrow keys, and the reasons behind them. "In-degree" means the number of arrows pointing *into* a node.

Squaring, $x \mapsto x^2$: count the arrows coming in

For a **prime** modulus p (odd), every nonzero residue has exactly 0 or 2 incoming arrows: a nonzero square a has exactly two square roots $\pm\sqrt{a}$, and a non-square has none. Exactly half of the $p - 1$ nonzero residues are squares. So the picture is made of **binary** trees: nodes either are childless or have exactly two children, hanging off the cycles.

For n with two distinct odd prime factors, say $n = pq$, the Chinese remainder theorem says a square root modulo n is a pair of square roots, one modulo p and one modulo q. So a unit which is a square has $2 \times 2 = \mathbf{4}$ square roots, and the trees are **4-ary**. In general, a unit square modulo n has 2^k square roots where k is the number of distinct odd primes dividing n (with a correction for powers of 2; $8 \mid n$ doubles it again). Verified by machine:

n	factorization	in-degrees of $x \mapsto x^2$ on units	fixed points of $x \mapsto x^2$
$13, 17$	prime	2	2
$15, 21$	$3 \cdot 5,\ 3 \cdot 7$	4	4
45	$3^2 \cdot 5$	4	4
105	$3 \cdot 5 \cdot 7$	8	8

So: branching factor 2 means one prime (a prime or a prime power); branching factor $4, 8, 16, \ldots$ means $2, 3, 4, \ldots$ distinct prime factors. The picture literally counts $\omega(n)$, the number of distinct prime factors, without factoring n — but only because we can *see* the whole portrait, which requires listing all n residues. That is exponential work in the input size $\log n$, which is why this is a nice picture and not a factoring algorithm.

The same fact in other clothes: fixed points and square roots of 1

The fixed points of $x \mapsto x^2$ are the solutions of $x(x-1) \equiv 0$, and by the Chinese remainder theorem there are 2^k of them (k = number of distinct prime factors): modulo each prime power, x must be 0 or 1. For $n = 15$ they are $0, 1, 6, 10$. Likewise the solutions of $x^2 \equiv 1$ number 2^k (up to the power-of-two correction): for a prime there are only the boring ± 1, and **any extra square root**

of 1 is a certificate that n is composite — indeed $\gcd(x-1, n)$ then factors n. This is precisely the observation behind the Miller-Rabin test, and it is the same phenomenon as Problem 2 with $y = 1$.

Multiplication by a fixed a, $x \mapsto ax$

If $\gcd(a, n) = 1$ this map is a bijection, so the portrait is *all cycles, no trees*, and every cycle has the same length, namely the multiplicative order of a modulo n. The number of cycles is $n/\mathrm{ord}(a)$ counting the fixed point 0 separately. When p is prime and a is a primitive root, there is exactly one big cycle through all $p-1$ units — a striking picture that never happens for composite n unless the unit group is cyclic. If $\gcd(a, n) > 1$, the map is not injective and tails appear.

Can you tell whether n is prime from the picture?

Yes, in several ways, all versions of the same idea:

- under $x \mapsto x^2$, branching factor exactly 2 everywhere (equivalently: only two square roots of 1, only two fixed points);
- under $x \mapsto x^2$, the units form trees over cycles with exactly $(p-1)/2$ squares;
- under $x \mapsto ax$ for a primitive root a, one single cycle containing every nonzero residue.

A composite n with several prime factors breaks each of these: more square roots, more fixed points, more cycles, higher branching. Prime powers p^k are the sneaky case — they have only two square roots of 1, so they *look* prime by the branching test; what gives them away is that the non-units (multiples of p) form their own structure, visible as a separate family of nodes that collapse to 0.

Two things are worth saying about the usefulness of all this. First, everything visible in the portrait is a statement about $(\mathbb{Z}/n\mathbb{Z})^*$ that we could also have computed algebraically. Second, and more importantly, drawing the portrait takes n steps, which is exponential in the size $\log n$ of the input — so this is a way of *seeing* the theorems that primality tests are built on, not a practical test. The practical tests (Fermat, Miller-Rabin) sample one or two points of this picture with a fast modular exponentiation instead of drawing all of it.