Solutions to Daily Due Oct 7 (AI generated)

Math 4440/5440, Coding and Cryptography, Fall 2026

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

Conventions as in lecture and on the Pollard Rho Demo page: iterate $f(x) = x^2 + 1 \pmod{n}$ from x_0. The tortoise is x_i (one step per turn) and the rabbit is x_{2i} (two steps per turn). At step i compute $\gcd(x_{2i} - x_i, n)$: if it is 1, keep going; if it is a proper factor, done; if it is n, the method failed for this x_0 and c.

Problem 1: Pollard's ρ on $n = 8051$

The sequence $x_0, x_1, x_2, \ldots$ modulo 8051 begins

$$2,\ 5,\ 26,\ 677,\ 7474,\ 2839,\ 871,\ 1848,\ 1481,\ \ldots$$

(each term is the square of the previous one, plus 1, reduced mod 8051; e.g. $26^2 + 1 = 677$ and $677^2 + 1 = 458330 \equiv 7474$).

step i	1	2	3
tortoise x_i	5	26	677
rabbit x_{2i}	26	7474	871
$x_{2i} - x_i$	21	7448	194
$\gcd(x_{2i} - x_i, n)$	1	1	**97**

At step 3, $\gcd(194, 8051) = 97$, so

$$8051 = 97 \cdot 83.$$

Why 97 and not 83? Reduce the same sequence modulo each prime factor separately (this is what the demo shows "behind the scenes").

Modulo 97: $2 \to 5 \to 26 \to 95 \to 5 \to 26 \to 95 \to \cdots$. The tail has length 1 (just $x_0 = 2$) and then the sequence cycles with period 3. The rabbit laps the tortoise as soon as both are on the cycle and the gap $2i - i = i$ is a multiple of the period: that happens at step 3.

Modulo 83: $2 \to 5 \to 26 \to 13 \to 4 \to 17 \to 41 \to 22 \to 70 \to 4 \to \cdots$. The tail has length 4 and the cycle has period 5, so tortoise and rabbit first agree modulo 83 at step 5.

The gcd reports whichever prime's ρ closes up first. Modulo 97 the picture is a short tail and a 3-cycle; modulo 83 it is a longer tail and a 5-cycle, so 97 wins at step 3, when $x_6 - x_3 = 194 = 2 \cdot 97$ is divisible by 97 but not by 83. (Had both primes closed up at the same step, the gcd would have been 8051 itself: a failure, like the 299 preset.)

$n = 10403$**.** Nine steps are needed:

i	1	2	3	4	5	6	7	8	9
x_i	5	26	677	598	3903	3418	156	3531	5168
x_{2i}	26	598	3418	3531	3724	9812	9970	3682	7087
gcd	1	1	1	1	1	1	1	1	**101**

so $10403 = 101 \cdot 103$. Modulo 101 the sequence has tail 8 and period 9 (first meeting at step 9); modulo 103 it has tail 3 and period 14 (first meeting at step 14).

$n = 455459$**.** Again nine steps:

i	1	2	3	4	5	6	7	8	9
x_i	5	26	677	2871	44380	179685	121634	155260	44567
x_{2i}	26	2871	179685	155260	416250	43670	164403	247944	68343
gcd	1	1	1	1	1	1	1	1	**743**

so $455459 = 743 \cdot 613$. Modulo 743: tail 6, period 9, meeting at step 9. Modulo 613 the sequence falls into a *fixed point* (period 1) but only after a tail of length 33, so 613 would not have been detected until step 33.

Problem 2: why about $\sqrt{p}$ steps?

Fix a prime factor p of n. Reduced modulo p, the values $x_0, x_1, x_2, \ldots$ behave like random elements of $\{0, 1, \ldots, p-1\}$ (the function $x^2 + 1$ scrambles them well enough). By the birthday paradox, among about $\sqrt{p}$ random values from a set of size p we expect a repeat (more precisely about $\sqrt{\pi p/2} \approx 1.25\sqrt{p}$): that is, $x_i \equiv x_j \pmod{p}$ for some $i < j$ with j around $\sqrt{p}$. From then on the sequence modulo p is periodic with period $j - i$, so the ρ modulo p has tail plus cycle of total length about $\sqrt{p}$. The tortoise-and-rabbit trick (Floyd's cycle detection) notices this periodicity within a small constant multiple of that many steps, and at that moment $p \mid x_{2i} - x_i$, so the gcd with n reveals p (unless the other prime factors happen to divide the difference too).

Note the method never needed p: it only computes modulo n and takes gcds. The running time is governed by the *smallest* prime factor of n.

RSA sizes. If $n = pq$ with p and q both around 100 digits, then $p \approx 10^{100}$ and

$$\sqrt{p} \approx 10^{50}$$

steps. Each step is cheap, but 10^{50} steps is far beyond any computer (a billion steps per second for the age of the universe is only about 10^{27}). So Pollard's ρ is wonderful for shaking small factors out of a number, and useless against a properly chosen RSA modulus; factoring those needs the sieve-type methods we have seen, whose runtime grows much more slowly than $\sqrt{p}$.