Solutions to Daily Due Oct 19 (AI generated)

Math 4440/5440, Coding and Cryptography, Fall 2026

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

Problem 1 is a full worked pass through the Elliptic Curve El Gamal ciphertext chain, including the "challenge for fun". Problem 4 was a link to read, with nothing to solve.

Problem 1: the Elliptic Curve El Gamal chain

The parameters

$$E : y^2 = x^3 + x^2 + x + 1, \qquad p = 12345678910123456789 1027,$$

$$P = (3,\ 11655832467975276266127), \qquad N = \mathrm{ord}(P) = 617283945509492876147 31.$$

Here p is prime and P is on E. $NP = \infty$, and $(N/q)P \neq \infty$ for each prime $q \mid N$, so the order is exactly N. Since $N \approx p/2$, Hasse's bound leaves only one multiple of N in the possible range, so $\#E = 2N$.

In Sage:

```
p = 12345678910123456789 1027
E = EllipticCurve(GF(p), [0,1,0,1,1])
P = E([3,11655832467975276266127,1])
```

(3) Bob's key pair

Bob picks a private integer s (any number less than N) and publishes $B = sP$:

$$s = 314159265358979323846,$$

$$B = sP = (7138157337112451311969,\ 784653316740386512827).$$

The public key is a *point*; the private key is just an integer.

(4) The message as an integer

Alice answers "Best hallowe'en costume?" with `ghost`. Text to Integer reads the letters as digits base 255:

$$m = \texttt{ghost} \mapsto 437241524591,$$

and with three digits of padding the candidate x-coordinate is 437241524591000.

(5) Placing the message on the curve

We need y with $y^2 = x^3 + x^2 + x + 1$, so $x^3 + x^2 + x + 1$ must be a square modulo p. About half of all residues are squares, so we just try padding values in turn:

padding	x	is $x^3 + x^2 + x + 1$ a square mod p?
`000`	437241524591000	no
`001`	437241524591001	no
`002`	437241524591002	no
`003`	437241524591003	yes

So the message point is

$$M = (437241524591003,\ 55597716160402931650709).$$

(Since $p \equiv 3 \pmod 4$, the square root can even be written down directly: $y = a^{(p+1)/4} \bmod p$, from §5.7.2. In Sage, `sqrt(Mod(a,p))` does it.)

(6) Alice encrypts to Bob

Alice picks a fresh ephemeral integer k (new for every message) and computes

$$k = 271828182845904523536,$$

$$C_1 = kP = (3115343649191261501520,\ 25373702994261806320235),$$

$$C_2 = M + kB = (58069558590384761528449,\ 12146173524543821435623 4).$$

She posts the ciphertext (C_1, C_2), two points, on `#ciphertexts`, @mentioning Bob.

(7) Bob decrypts

Bob uses his private key to compute the shared point

$$sC_1 = (49700571508132054021345,\ 36158355629085077756485),$$

which is the same as Alice's kB, because both equal skP. Then

$$C_2 - sC_1 = M + kB - skP = M = (437241524591003,\ \ldots).$$

Drop the last three digits of the x-coordinate: 437241524591, and Integer to Text gives `ghost`. In Sage: `M = C2 - s*C1; M[0]`.

(10) Challenge: break someone else's message

To read a message to Bob, it is enough to find his private key s from $B = sP$, which is the ECDLP. Following the hint, factor the order of P:

$$N = 3 \cdot 40197392597 \cdot 511877268341.$$

Its largest prime factor is only about 5×10^{11}, so this ECDLP falls to **Pohlig–Hellman**, just as the multiplicative El Gamal chain did. For each prime $q \mid N$:

1. Multiply by N/q: $P_q = (N/q)P$ has order q, and $B_q = (N/q)B = (s \bmod q)P_q$.

2. Solve the small ECDLP $B_q = d_q P_q$ for $d_q = s \bmod q$, by baby-step giant-step in about $\sqrt{q}$ steps ($\leq 7.2 \times 10^5$ here).

3. Glue the answers together with the Chinese Remainder Theorem.

On Bob's key this gives

$$s \equiv 1 \pmod 3, \qquad s \equiv 10520454682 \pmod{40197392597},$$

$$s \equiv 409907864852 \pmod{511877268341},$$

and CRT returns $s = 314159265358979323846$, Bob's private key. This took about 8 seconds in plain Python; Sage's `discrete_log(B, P, N, operation='+')` (or `P.discrete_log(B)`) does all of it at once. You could also attack the ephemeral key: solve $C_1 = kP$ for k, then $M = C_2 - kB$.

Lesson: as in the multiplicative setting, the security of the ECDLP depends on the size of the *largest prime factor* of the order of P, not on the size of p. Real systems use a point of prime order $N \approx p$, so the best attack takes about $\sqrt{p}$ steps.

Problem 2: why Pollard $p-1$ works

Suppose $n = pq$, and suppose $p-1$ is *smooth*: all its prime-power factors are small. Then once B is large enough, $p-1 \mid B!$, say $B! = (p-1)t$. By Fermat's little theorem (taking $p \nmid a$),

$$a^{B!} = \left(a^{p-1}\right)^t \equiv 1^t = 1 \pmod p,$$

so $p \mid a^{B!} - 1$. Also $p \mid n$, so $p \mid \gcd(a^{B!} - 1, n)$.

The gcd is a *nontrivial* factor as long as $q \nmid a^{B!} - 1$. That happens, for example, if $q-1$ has a large prime factor that doesn't divide $B!$ yet. So the method works when one prime factor p of n has $p-1$ smooth and the other doesn't. We never need to know p: the factorial $B!$ "tries every small $p-1$ at once".

If both $p-1$ and $q-1$ divide $B!$, the gcd becomes n and we've gone too far. If neither $p-1$ nor $q-1$ is smooth, the method fails. This is why RSA primes are sometimes chosen with $p-1$ having a large prime factor.

Problem 3: a Sage loop for $n = 16637$

```
n = 16637
a = Mod(2,n)
for B in range(2,30):
    a = a^B                  # now a = 2^(B!) mod n
    g = gcd(ZZ(a)-1, n)
    print(B, a, g)
    if g > 1:
        print("factor:", g, n//g)
        break
```

Raising to the power B at step B turns $a^{(B-1)!}$ into $a^{B!}$:

B	$2^{B!} \bmod 16637$	$\gcd(2^{B!} - 1, 16637)$
2	4	1
3	64	1
4	7120	1
5	6479	1
6	3112	1
7	636	**127**

So

$$16637 = 127 \cdot 131.$$

Why $B = 7$: $127 - 1 = 126 = 2 \cdot 3^2 \cdot 7$, which divides $7! = 5040$ (since $7!$ contains 3 and 6, it has 3^2) but not $6! = 720$, which has no factor of 7. Meanwhile $131 - 1 = 130 = 2 \cdot 5 \cdot 13$ needs $B \geq 13$. Indeed, if you keep going, $2^{13!} \equiv 1 \pmod{16637}$ and the gcd jumps to 16637 itself, which is trivial.