Write-up Problèmeuh

Nuliel

Problèmeuh is a crypto challenge from FCSC 2025. The goal is to solve a system of equations, with both linear and quadratic equations.

1 Problem statement

Here is a nice and small system to solve.

And the python script attached:

```python
import sys
from hashlib import sha256
sys.set_int_max_str_digits(31337)
try:
    a, b, c, x, y = [ int(input(f"{x} = ")) for x in "abcxy" ]
    assert a > 0
    assert a == 487 * c
    assert 159 * a == 485 * b
    assert x ** 2 == a + b
    assert y * (3 * y - 1) == 2 * b
    h = sha256(str(a).encode()).hexdigest()
    print(f"FCSC{{{h}}}")
except:
    print("Nope!")
```

2 Solution

We have this system of equations:

$$\begin{cases} a = 487c \\ 159a = 485b \\ x^2 = a + b \\ y(3y-1) = 2b \end{cases}$$

2.1 Two first equations

We multiply the first equation by 159:

$$\begin{cases} 159a = 159 \cdot 487c \\ 159a = 485b \end{cases}$$

So we have

$$159 \cdot 487c = 485b$$

As 159, 485 and 487 are coprime, we must have

- 159 and 487 in the factors of b
- 485 in the factors of c

From this fact, we can express a, b and c in function of only one unknown k:

$$\begin{cases} b = 159 \cdot 487k \\ c = 485k \\ a = 487 \cdot 485k \end{cases}$$

2.2 Third equation

We can replace, develop and factor in this equation:

$$\begin{aligned} x^2 &= a + b \\ &= k \cdot (487 \cdot 485 + 159 \cdot 487) \\ &= k \cdot (2^2 \cdot 7 \cdot 23 \cdot 487) \end{aligned}$$

x^2 is obviously a square number, so each prime factor must appear at least two times (precisely an even number of times). To compensate, k must contain the factors 7, 23 and 487, so $k = 7 \cdot 23 \cdot 487k'$, with k' a square number.

2.3 Last equation

$$\begin{aligned} y(3y-1) &= 2b \\ 3y^2 - y &= 2 \cdot 159 \cdot 487k \\ 3y^2 - y &= 2 \cdot 7 \cdot 23 \cdot 159 \cdot 487^2 k' \end{aligned}$$

We have an equation of degree two like this one:

$$\begin{aligned} &Ay^2 + By + C = 0 \\ &A = 3 \\ &B = -1 \\ &C = -2 \cdot 7 \cdot 23 \cdot 159 \cdot 487^2 k' \end{aligned}$$

So we can compute the discriminant

$$\begin{aligned}\Delta &= B^2 - 4AC \\ &= (-1)^2 - 4 \cdot 3 \cdot (-2) \cdot 7 \cdot 23 \cdot 159 \cdot 487^2 k' \\ &= 1 + (2)^3 \cdot 3 \cdot 7 \cdot 23 \cdot 159 \cdot 487^2 k'\end{aligned}$$

We know that there exists a solution (because this challenge can be solved), so Δ must be positive, and must be a square number. Recall that k' is also a square number.
This equation is of form

$$X^2 - D \cdot Y^2 = 1$$

with $X = \sqrt{\Delta}$ and $Y = \sqrt{k'}$ so it's a Pell-Fermat equation. We can use sympy to solve the Pell-Fermat equation and get the flag:

```
import sys
from hashlib import sha256
from math import isqrt
from sympy.solvers.diophantine.diophantine import diop_DN
sys.set_int_max_str_digits(31337)

# sqrt_delta**2 - D * (sqrt_k')**2 = 1

D = 12 * 2*7*23*159*487**2

# solve Pell equation
l = diop_DN(D, 1)
# get the result
sqrt_delta, sqrt_k_prime = l[0][0], l[0][1]

# evaluate all unknowns
k_prime = sqrt_k_prime**2
k = 7*23*487*k_prime
a = 487*485*k
b = 159*487*k
c = 485 * k
x = isqrt(a+b)
y = (1 + sqrt_delta) // 6

# time to verify each equation
assert a > 0
assert a == 487 * c
assert 159 * a == 485 * b
```

```python
assert x ** 2 == a + b
assert y * (3 * y - 1) == 2 * b

# and get the flag
h = sha256(str(a).encode()).hexdigest()
print(f"FCSC{{{h}}}")
```