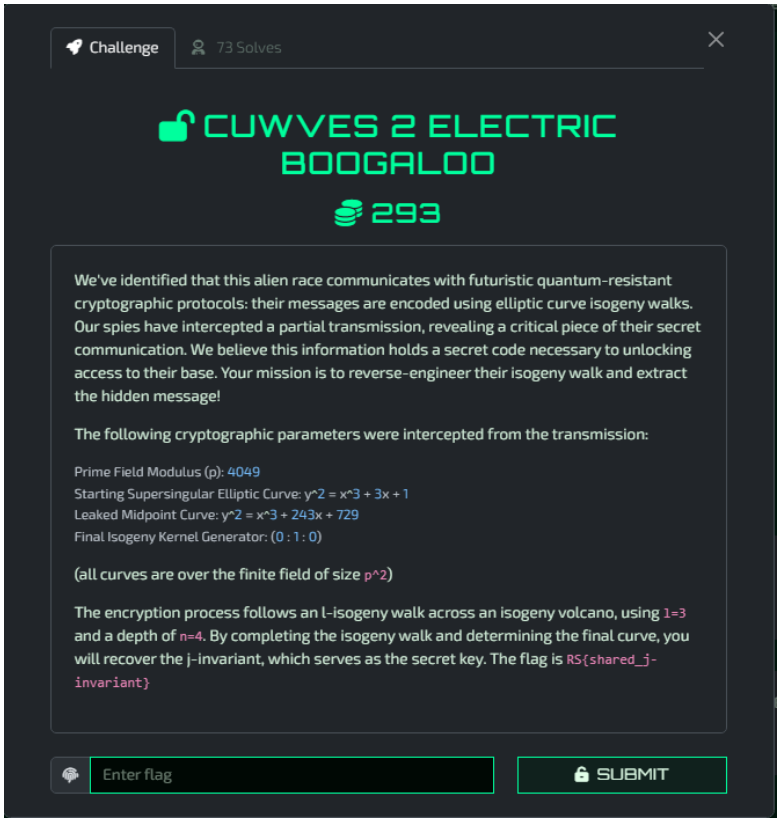
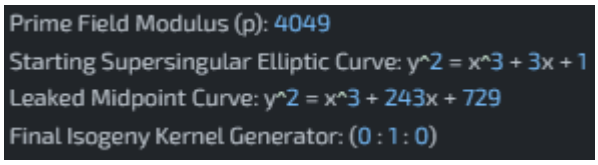


A **lliptic curve cryptography (ECC)** and **isogeny-based cryptography** that are it involves **isogeny walks**, a method for creating cryptographic protocols resistant to quantum attacks.

in this challenge, we need to reverse-engineer an encryption process that uses isogeny walks on elliptic curves, extract the j-invariant (a key element in elliptic curve cryptography), and then determine the secret key embedded in the cryptographic parameters

Given parameters that were intercepted from the transmission



We need to determine the final curve to recover the j-invariant.

Flag Format: RS{shared_j_invariant}

Encryption Process

1. **Define Field and Curves:**
   - Prime field modulus $p = 4049$ and finite field $\mathbb{F}_{p^2}$ are defined.
   - Starting curve E_0 and midpoint curve E_{mid} are set.
2. **Isogeny Walk:**
   - Apply an isogeny walk with degree $l = 3$ and depth $n = 4$.
   - The final curve is the midpoint curve E_{mid}.
3. **Compute j-Invariant:**
   - The j-invariant of the midpoint curve E_{mid} is calculated.
4. **Output Flag:**
   - Print the flag: `RS{<j_invariant_value>}`.

(asked chatgpt to explain the encryption process because I don't know how to explain it in a short way hehe)

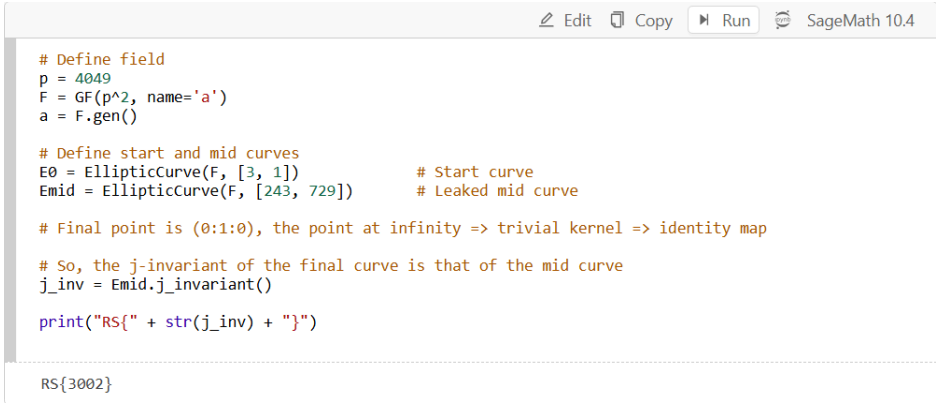
Python Script (use sagemath)

```
p = 4049
F = GF(p^2, name='a')
a = F.gen()

E0 = EllipticCurve(F, [3, 1])
Emid = EllipticCurve(F, [243, 729])

j_inv = Emid.j_invariant()

print("RS{" + str(j_inv) + "}")
```



RS{3002} is the secret key (j-invariant).