**Walkthrough: Elliptic Curve Cryptography (ECC) Anamoly Challenge**

**Challenge Overview**

**Objective**: This challenge is based on **Elliptic Curve Cryptography (ECC)**, requiring participants to determine the secret scalar k used to derive a public point.

The process involves:

1. Generating an ECC curve with a prime p, and parameters a and b.
2. Finding a valid **base point** on the curve.
3. Generating a **secret scalar** k and computing the public point.
4. **Obfuscating** the public point using SHA-256 and Base64 encoding.
5. Reversing the process to find k and retrieve the flag.

--------------------------------------------------------------------------------------------------------------------------------------

**Step 1: Understanding the Given Code**

* The script first generates an **elliptic curve** with parameters (p, a, b).
* It searches for a valid **base point** G where y² = x³ + ax + b (mod p).
* A **secret scalar** k is selected randomly.
* The public point is calculated as:
  + - * + P=k×G
* The public point (x, y) is **obfuscated** using SHA-256-based XOR masking and Base64 encoding.

--------------------------------------------------------------------------------------------------------------------------------------

**Step 2: Extracting the Masked Public Point**

* The script provides the **masked public point** encoded in **Base64**.
* Decode the Base64 output to retrieve (masked\_x, masked\_y).

--------------------------------------------------------------------------------------------------------------------------------------

**Step 3: Recovering the Original Public Point**

* Compute the **masking key** from SHA-256:

mask\_key = sha256(b"ai\_resistant\_challenge").digest()

* XOR **masked\_x** and **masked\_y** with the respective parts of mask\_key to recover the original public point (x, y).

x = masked\_x ^ int.from\_bytes(mask\_key[:16], 'big')

y = masked\_y ^ int.from\_bytes(mask\_key[16:], 'big')

--------------------------------------------------------------------------------------------------------------------------------------

**Step 4: Recovering the Secret Scalar k**

* Since ECC uses scalar multiplication:

P=k×G

* To find k, perform a brute-force search for a small k:

for k in range(1, p):

if curve.scalar\_multiplication(k, base\_point) == public\_point:

print(f"Secret Scalar Found: {k}")

break

--------------------------------------------------------------------------------------------------------------------------------------

**Step 5: Extracting the Flag**

Once k is found, the flag format is:

**flag{ecc\_secret\_<k>}**

Print the recovered flag:

**flag = f"flag{{ecc\_secret\_{k}}}"**

**print("Recovered Flag:", flag)**

--------------------------------------------------------------------------------------------------------------------------------------**Conclusion**

This challenge tests knowledge of **elliptic curve operations, masking techniques, and brute-force key recovery**. By reversing the obfuscation and iterating for k, we retrieve the flag.

**Happy CTF Solving!**