

STEP 1 (ANALYZE)

We are provided with an obfuscated chal.py file and an output.txt which contains the length of the secret in bytes and 15 shares.

From the description and name of the challenge we can safely assume that this challenge is about Shamir's Shared Secret technique which we can confirm from the chal.py and where they print the shares as `print("Shares from Shamirs Shared Secrets (x, y, mod):\n")`.

Shamir's Shared Secret technique basically breaks a secret into multiple shares and the secret can only be recovered after you have the minimum number of shares. From the chal.py file we see that the minimum shares required for the challenge are 7. The output file gives us 15 shares from 2 different modulus p1 and p2 which alternate during the generation of shares.

STEP 2 (SEGREGATING THEN RECOVERING THE SECRET)

So now to solve it we can start by segregating the shares to their respective moduli.

To recover the secret for each modulus we can run an interpolation algorithm like Lagrange interpolation from which we will recover secret%p1 and secret%p2.

To recover the secret from secret%p1 and secret%p2 we can run an algorithm like CRT (Chinese Remainder Theorem) which will recover the secret as bytes which is the flag as indicated by the description. Now that we know what is to be done we can do it through scripting:

```
from sympy import mod_inverse

# ---- SHARES ----
shares = [
    (1, 119851754009690660916, 144617414729854976017),
    (2, 37991115101738800321, 144617414729854976027),
    (3, 69163871120902756702, 144617414729854976017),
    (4, 69544837930006065004, 144617414729854976027),
    (5, 76659971158768438387, 144617414729854976017),
    (6, 43868527434343164463, 144617414729854976027),
    (7, 115595609094483204434, 144617414729854976017),
    (8, 3210196247551954359, 144617414729854976027),
    (9, 143590095238310798554, 144617414729854976017),
    (10, 137009597012602961873, 144617414729854976027),
    (11, 127577839959426928703, 144617414729854976017),
    (12, 14575911154724053845, 144617414729854976027),
```

```

    (13, 102924801761451620259, 144617414729854976017),
    (14, 88523782506687773742, 144617414729854976027),
    (15, 43163861931617331069, 144617414729854976017)
]

# ---- SEGREGATE BY MODULUS ----
# can be segregated manually as well
p1 = 144617414729854976017
p2 = 144617414729854976027
shares_p1 = [(x, y) for x, y, mod in shares if mod == p1]
shares_p2 = [(x, y) for x, y, mod in shares if mod == p2]

# The scripts for these algorithms can easily be found in other
# writeups about Shamir's Shared Secret.
# ---- LAGRANGE INTERPOLATION ----
def lagrange_interpolate_zero(points, mod):
    total = 0
    for i, (x_i, y_i) in enumerate(points):
        num, den = 1, 1
        for j, (x_j, _) in enumerate(points):
            if i != j:
                num = (num * (-x_j)) % mod
                den = (den * (x_i - x_j)) % mod
        total = (total + y_i * num * mod_inverse(den, mod)) % mod
    return total

s_p1 = lagrange_interpolate_zero(shares_p1[:7], p1)
s_p2 = lagrange_interpolate_zero(shares_p2[:7], p2)
print(f"Secret mod p1: {s_p1}")
print(f"Secret mod p2: {s_p2}")

# ---- CRT COMBINATION ----
N = p1 * p2
M1, M2 = N // p1, N // p2
inv1, inv2 = mod_inverse(M1, p1), mod_inverse(M2, p2)
base = (s_p1 * M1 * inv1 + s_p2 * M2 * inv2) % N
print(f"Flag: {base}")

```

The output obtained from this script is:

```
Secret mod p1: 120691705935694332047
```

Secret mod p2: 55266033081063690396

Flag: 15586106816722605912563682414346023421575

Thus the flag is "DJSISACA{15586106816722605912563682414346023421575}"