Name: Sharing is Caring

Description: My friend Shamir has used this special encoding technique to protect the precious Flag

Can you help me decode it?

Difficulty: Medium

Author: Revak Pandkar

Writeup:

From the description we get a hint about the Shamir Secret Sharing which is used to divide a secret into different shares and we can get the secret with a particular number of shares.

Looking this up on google we can find the Wikipedia article about this. We can also see that Lagrange Interpolation is used to get the original polynomial that was used to encode the secret.

As we can see from the encrypt function, this encoding is further strengthened by using random primes and operating in the Finite Field.

Learning about this encoding in detail we can create a decrypt function which is used to get the Flag.

We can see that in the values given each row has a different list of shares with the particular prime associated with it.

Each tuple in the list of shares consists of two values: xi and yi, which represent the x-coordinate and y-coordinate of a point on a polynomial curve.

The secret that we will finally get from the first row will be ASCII value of the first character of the flag.

Each character of the flag has been encoded in this way.

Decrypt Code:

```
from sympy import mod_inverse


def recover_secret(shares, prime):
    k = len(shares)
    secret = 0
    for i in range(k):
        xi, yi = shares[i]
        prod = yi
        for j in range(k):
            if i != j:
                xj, yj = shares[j]
                prod *= (0 - xj) * mod_inverse(xi - xj, prime)
        secret += prod
    return secret % prime



k = 3

n = 5


list_of_shares = [[[(1, 76), (2, 31), (3, 58), (4, 50), (5, 7)], 107],
                  [[(1, 9), (2, 146), (3, 170), (4, 81), (5, 52)], 173],
                  [[(1, 48), (2, 102), (3, 86), (4, 0), (5, 35)], 191],
                  [[(1, 12), (2, 44), (3, 1), (4, 82), (5, 88)], 199],
```

[[(1, 50), (2, 103), (3, 51), (4, 121), (5, 86)], 227],

[[(1, 13), (2, 110), (3, 74), (4, 62), (5, 74)], 157],

[[(1, 56), (2, 6), (3, 44), (4, 43), (5, 3)], 127],

[[(1, 148), (2, 123), (3, 9), (4, 104), (5, 110)], 149],

[[(1, 77), (2, 85), (3, 94), (4, 104), (5, 115)], 127],

[[(1, 235), (2, 20), (3, 195), (4, 43), (5, 42)], 239],

[[(1, 118), (2, 138), (3, 168), (4, 208), (5, 47)], 211],

[[(1, 36), (2, 81), (3, 90), (4, 63), (5, 0)], 97],

[[(1, 128), (2, 19), (3, 50), (4, 84), (5, 121)], 137],

[[(1, 106), (2, 138), (3, 31), (4, 143), (5, 116)], 179],

[[(1, 91), (2, 18), (3, 27), (4, 21), (5, 0)], 97],

[[(1, 116), (2, 98), (3, 56), (4, 121), (5, 31)], 131],

[[(1, 17), (2, 48), (3, 69), (4, 80), (5, 81)], 127],

[[(1, 36), (2, 36), (3, 51), (4, 20), (5, 4)], 61],

[[(1, 88), (2, 84), (3, 83), (4, 85), (5, 90)], 127],

[[(1, 104), (2, 84), (3, 42), (4, 85), (5, 106)], 107],

[[(1, 50), (2, 68), (3, 13), (4, 63), (5, 40)], 89],

[[(1, 60), (2, 105), (3, 122), (4, 111), (5, 72)], 127],

[[(1, 61), (2, 119), (3, 112), (4, 40), (5, 60)], 157],

[[(1, 92), (2, 0), (3, 63), (4, 58), (5, 208)], 223],

[[(1, 151), (2, 80), (3, 48), (4, 55), (5, 101)], 157],

[[(1, 12), (2, 10), (3, 45), (4, 58), (5, 49)], 59],

[[(1, 133), (2, 104), (3, 8), (4, 119), (5, 26)], 137],

[[(1, 14), (2, 24), (3, 149), (4, 7), (5, 171)], 191],

[[(1, 12), (2, 54), (3, 86), (4, 19), (5, 31)], 89],

[[(1, 9), (2, 70), (3, 82), (4, 45), (5, 170)], 211],

[[(1, 33), (2, 35), (3, 2), (4, 8), (5, 16)], 37],

[[(1, 13), (2, 204), (3, 65), (4, 18), (5, 63)], 211]]

```
for share,prime in list_of_shares:
    flag = recover_secret(share[:k],prime)
    print(chr(flag),end='')
```

The decrypt function first initializes the secret to 0. Then it loops over the list of shares and stores the x,y co-oridnates in xi and yi respectively. Then an inner for loop is used to loop over the other co-ordinates and the prod value is calculated based on the Lagrange Interpolation technique. But here an extra mod inverse function is used which helps the answer to stay within the finite field of the prime. This value of prod is added to the secret value and the same process is repeated for the other shares in the list. Finally, the secret variable is returned modulo prime. This is done to ensure that the result is within the range of the finite field of integers modulo prime.

Flag: VishwaCTF{l4gr4ng3_f0r_th3_w1n!}