

# Crypto-10 Сломанный стол зачарований

## Описание

Ты нашел в подземелье старый стол зачарований, но он работает неправильно! Вместо обычных зачарований он выдает зашифрованные руны.

Изучив механизм, ты понял: стол берет каждый символ заклинания, умножает его на уровень опыта игрока, добавляет количество лазурита в инвентаре, а результат ограничивает числом 256.

К счастью, на столе есть табличка: "Все заклинания начинаются с `vsosh{`" - видимо, это стандартная магическая формула.

Сможешь ли ты взломать этот странный стол зачарований и получить настоящее заклинание?

**Рекомендуемые утилиты:** python

**Цель работы:** Расшифровать сообщение и получить флаг

**Критерий оценки:** Предоставление правильного флага

## Решение

Рассмотрим алгоритм шифрования одного байта:

$$C \equiv (a \cdot M + b) \pmod{m}, \quad \gcd(a, m) = 1.$$

Предполагается, что флаг имеет вид `vsosh{...}` (известен префикс), шифртекст дан в байтах. Для байтов используем  $m = 256$ .

Пусть известны пары первых байт открытого текста и шифртекста из префикса `vsosh{` :

$$(M_0, C_0), (M_1, C_1), \dots$$

Для каждого известного байта выполняется линейное сравнение по модулю  $m$ :

$$\begin{cases} a M_0 + b \equiv C_0 \pmod{m} \\ a M_1 + b \equiv C_1 \pmod{m} \\ a M_2 + b \equiv C_2 \pmod{m} \end{cases} \quad (u.t.d.)$$

Исключим  $b$  из двух уравнений:

$$C_i - C_j \equiv a(M_i - M_j) \pmod{m}.$$

Если  $\gcd(M_i - M_j, m) = 1$ , то существует обратный элемент и:

$$a \equiv (C_i - C_j)(M_i - M_j)^{-1} \pmod{m}.$$

Далее находим  $b$  из любого уравнения:

$$b \equiv C_i - a M_i \pmod{m}.$$

Расшифровка любого байта:

$$M \equiv a^{-1}(C - b) \pmod{m},$$

где  $a^{-1}$  — элемент, обратимый к  $a$  по модулю  $m$ .

Далее расшифровываем весь текст по формуле:  $M \equiv a^{-1}(C - b) \pmod{m}$ .

## Скрипт

```
#!/usr/bin/env python3
```

```
from math import gcd
```

```
cipher = [551859375280905, 518930908428138, 475026285957782, 518930908428138, 398193196634659,
606740153368850, 1262159732110517, 442097819105015, 354288574164303, 299407796076358,
507954752810549, 540883219663316, 464050130340193, 1295088198963284, 518930908428138,
299407796076358, 518930908428138, 1306064354580873, 584787842133672, 518930908428138,
299407796076358, 1306064354580873, 299407796076358, 442097819105015, 1262159732110517,
529907064045727, 628692464604028]
```

```
m = 1478631250060842
```

```
known_text = b"vsosh{"
```

```
def recover_keys_from_known_prefix(C: list[int]) -> tuple[int, int]:
```

```
    for i in range(len(known_text)):
```

```
        for j in range(i + 1, len(known_text)):
```

```
            Mi = known_text[i]
```

```
            Mj = known_text[j]
```

```
            dM = (Mi - Mj) % m
```

```
            if gcd(dM, m) != 1:
```

```
                continue
```

```
            dC = (C[i] - C[j]) % m
```

```
            a = (dC * pow(dM, -1, m)) % m
```

```
            b = (C[i] - a * Mi) % m
```

```
            ok = True
```

```
            for k in range(len(known_text)):
```

```
                if (a * known_text[k] + b) % m != C[k] % m:
```

```
                    ok = False
```

```
                    break
```

```
            if ok:
```

```
                return a, b
```

```
def decrypt(C: list[int], a: int, b: int) -> bytes:
```

```
    ainv = pow(a, -1, m)
```

```
    M = [int((ainv * (c - b)) % m) for c in C]
```

```
    return bytes(M)
```

```
def main():
```

```
    a, b = recover_keys_from_known_prefix(cipher)
```

```
    pt = decrypt(cipher, a, b)
```

```
    print(pt.decode())
```

```
if __name__ == "__main__":  
    main()
```

## Флаг

vsosh{0ld\_run3s\_s4ys\_4\_l0t}