

Crypto-9 Тайна заброшенной шахты

Описание

Шахтер! В глубинах заброшенной алмазной шахты обнаружен древний сундук с загадочной табличкой. Нам известно лишь, что послание начиналось с 'vsosh{'. Помоги расшифровать руны и найти путь к сокровищам!

Рекомендуемые утилиты: python

Цель работы: Расшифровать сообщение и получить флаг

Критерий оценки: Предоставление правильного флага

Решение

Алгоритм шифрования: XOR + линейная система по блокам из 2 байт

Флаг имеет вид `vsosh{...}`. Шифрование выполняется блочно по 2 байта, один и тот же ключ $k = (k_1, k_2)$, где k - случайное простое число.

Для каждого байта внутри блока вычисляется линейная комбинация коэффициентов и ключа, затем применяется XOR:

$$c_i = m_i \oplus (a_i \cdot k_1 + b_i \cdot k_2)$$

Коэффициенты (a_i, b_i) даны для каждой позиции в блоке. Сообщение режется на блоки по 2 байта и шифруется одинаково.

Зная префикс `vsosh{` (первые 2 байта уже достаточны), можно восстановить ключ k , а затем расшифровать весь флаг.

Алгоритм решения:

- Разбить шифртекст на блоки по 2 байта.
- Первые 2 байта флага — это `b"vs"`.
- Для каждого байта посчитать $t_i = m_i \oplus c_i$, где m_i — известный байт открытого текста, c_i — соответствующий байт шифртекста.
- Коэффициенты: пусть внутри каждого блока для позиций 0, 1 заданы коэффициенты $(a_0, b_0), (a_1, b_1)$. Далее выписываем систему линейных уравнений по модулю 256 для первых двух позиций.

Для позиций 0..1:

$$\begin{cases} a_0 k_1 + b_0 k_2 \equiv t_0 \\ a_1 k_1 + b_1 k_2 \equiv t_1 \end{cases}$$

Где $t_i = m_i \oplus c_i$ и m_0, m_1 — байты `v, s`.

Для нахождения ключа надо решить систему для неизвестных k_1, k_2

После нахождения ключа расшифровка всех блоков тривиальна: $m_i = c_i \oplus (a_i \cdot k_1 + b_i \cdot k_2)$.

Скрипт

```
#!/usr/bin/env python3
from sympy import symbols, Eq, solve

coeffs = [(1907887, 1895415), (1896187, 1192784)]
cipertext = [3390776289007066558402, 2906881732666172902516, 3390776289007066558427,
```

```
2906881732666172902516, 3390776289007066558428, 2906881732666172902524,
3390776289007066558407, 2906881732666172902515, 3390776289007066558406,
2906881732666172902451, 3390776289007066558426, 2906881732666172902496,
3390776289007066558343, 2906881732666172902488, 3390776289007066558423,
2906881732666172902511, 3390776289007066558343, 2906881732666172902516,
3390776289007066558400, 2906881732666172902488, 3390776289007066558403,
2906881732666172902454, 3390776289007066558400, 2906881732666172902511,
3390776289007066558443, 2906881732666172902515, 3390776289007066558406,
2906881732666172902452, 3390776289007066558336, 2906881732666172902516,
3390776289007066558401, 2906881732666172902517, 3390776289007066558343,
2906881732666172902516, 3390776289007066558409]
```

```
KNOWN_PREFIX = b"vsosh{"
```

```
def get_key(coeffs, ct):
    (a0, b0), (a1, b1) = coeffs
    k1, k2 = symbols("k1 k2")
    eq1 = Eq((a0 * k1 + b0 * k2), (ct[0] ^ KNOWN_PREFIX[0]))
    eq2 = Eq((a1 * k1 + b1 * k2), (ct[1] ^ KNOWN_PREFIX[1]))
    sol = solve((eq1, eq2), (k1, k2))
    k1_val = sol[k1]
    k2_val = sol[k2]
    return (k1_val, k2_val)
```

```
def decrypt(coeffs, ct, k):
    k1, k2 = k
    out = bytearray()
    for i, cval in enumerate(ct):
        a, b = coeffs[i % 2]
        mask = (a * k1 + b * k2)
        out.append(cval ^ mask)
    return bytes(out)
```

```
def main():
    k = get_key(coeffs, ciphertext)
    pt = decrypt(coeffs, ciphertext, k)
    print(pt.decode())
```

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

Флаг

vsosh{str4ng3_ch3st_w1th_tr34sur3s}