

CHALLENGE NAME: [Forgotten Cipher]   
DEV : [Dhanashri Bodke]

CATEGORY: [Cryptography]   
LEVEL: [Medium]

2025

CHALLENGE NAME: [Forgotten Cipher]

**Challenge Description**:

The VIC cipher, a Cold War-era encryption, used evolving keys and transformations to secure messages. Inspired by this, the given cipher dynamically modifies its key and applies subtle bitwise transformations to obscure the text.XOR based transforms are used with base key as a single digit and bitwise shifts follow certain index rules.Reversing the process requires careful observation—can you unravel the sequence and restore the original message?

File :- info.txt

Encrypted Message :- 0d4ac648a2f0bee7bccf0231c35e13ba7bc93a2d8f7d9498885e3f4998

Key Evolution Formula :- K(n) = [ K(n−1) × 3 + index ] mod 256

Hint Provided :- Evolve the key dynamically with position, XOR each character, then apply bitwise shifts of 2—left for even index, right for odd.

#### **Solution:**

#### ****Understanding the Encryption****

The encryption process follows three key transformations:

1. **Key Evolution**: The key evolves dynamically using the formula:K(n)=(K(n−1)×3+index)mod  256K(n)
2. This means each character in the text is encrypted using a different key value.
3. **XOR Operation**: Each character is XORed with its respective key.
4. **Bitwise Shifts**:
   1. If the index is **even**, the character undergoes a **left circular shift by 2** bits.
   2. If the index is **odd**, the character undergoes a **right circular shift by 2** bits.

By combining these steps, the original text is transformed into an encrypted byte sequence that appears random.

#### ****Reversing the Encryption (Decryption Process)****

To decrypt the text, we need to **reverse each step in the encryption process**:

1. **Recalculate the key** sequence using the same key evolution formula.
2. **Reverse the bitwise shifts**:
   1. For **even indices**, apply a **right circular shift by 2** bits.
   2. For **odd indices**, apply a **left circular shift by 2** bits.
3. **Reverse the XOR operation** using the recalculated key to recover the original characters

Python Code For Decryption :-

def decrypt(encrypted\_bytes, base\_key=7):

decrypted\_chars = []

key = base\_key

for i, enc\_char in enumerate(encrypted\_bytes):

key = (key \* 3 + i) % 256 # Recalculate evolving key

# Reverse alternating bitwise shifts

if i % 2 == 0:

enc\_char = ((enc\_char >> 2) & 0xFF) | ((enc\_char & 0x3) << 6) # Reverse left shift

else:

enc\_char = ((enc\_char << 2) & 0xFF) | (enc\_char >> 6) # Reverse right shift

# Reverse XOR operation

decrypted\_char = enc\_char ^ key

decrypted\_chars.append(chr(decrypted\_char))

return ''.join(decrypted\_chars)

encrypted\_data = bytes.fromhex("<hex value of encrypted flag>") # Replace with actual hex

flag = decrypt(encrypted\_data)

print("Decrypted Flag:", flag)

**Flag: VishwaCTF{VIC\_Decoded\_113510}**

DEV : [Dhanashri Bodke]

DEV : [Dhanashri Bodke]



**1**