4 Challenge 4: Fixing the Fault

4.1 Problem Understanding

The encryption scheme takes a plaintext byte sequence $m = m_1 m_2 \ldots m_n$, interprets it as a base-256 number, and then converts this number into base-255 digits $p_1 p_2 \ldots p_{n'}$, where each digit $p_i \in [0, 254]$. The ciphertext digits c_i are computed as:

$$c_i = (p_i + k_i - 1) \bmod 255$$

where $k_i \in [1, 255]$ are the key bytes. The ciphertext digits are then converted back from base-255 to base-256 bytes, producing the encrypted message. This scheme ensures perfect secrecy as the key digits k_i act as a one-time pad on base-255 digits.

4.2 Approach

To decrypt, we reverse the process:

1. Convert the ciphertext bytes into an integer and then to base-255 digits c_i.
2. Use the key digits k_i to recover the original base-255 plaintext digits by computing:

$$p_i = (c_i - k_i + 1) \bmod 255$$

3. Convert the recovered base-255 digits p_i back to an integer and then to the original base-256 plaintext bytes.

This process reverses the encryption step and recovers the original plaintext perfectly due to the perfect secrecy properties of the scheme.

4.3 Implementation

```python
def bytes_to_int(b):
    return int.from_bytes(b, byteorder='big')

def int_to_bytes(x, length):
    return x.to_bytes(length, byteorder='big')

def int_to_base(n, base):
    digits = []
    while n > 0:
        digits.append(n % base)
        n //= base
    digits.reverse()
    return digits if digits else [0]

def base_to_int(digits, base):
    n = 0
    for d in digits:
        n = n * base + d
    return n

def decrypt(ciphertext_bytes, key_bytes):
    # Converting ciphertext to integer
    c_int = bytes_to_int(ciphertext_bytes)

    # Converting ciphertext integer to base-255 digits
    c_digits = int_to_base(c_int, 255)
    key_digits = list(key_bytes[:len(c_digits)])

    # Recover plaintext digits p_i = (c_i - k_i + 1) mod 255
    p_digits = [(c - k + 1)%255 for c, k in zip(c_digits, key_digits)]

    # Converting plaintext digits back to integer
    p_int = base_to_int(p_digits, 255)

    # Converting integer back to bytes
    # Assuming plaintext length approx ciphertext length
    plaintext_len = len(ciphertext_bytes)
    plaintext = int_to_bytes(p_int, plaintext_len)

    return plaintext

if __name__ == "__main__":
    with open("ciphertext.enc", "rb") as f:
        ciphertext = f.read()
    with open("keyfile", "rb") as f:
        key = f.read()

    plaintext = decrypt(ciphertext, key)
    print("Recovered plaintext:")
    print(plaintext)
```

4.4 Output and Flag

Flag: cs409{r4d1x_ch4ng1ng_f0r_th3_w1n!}