1. RC4 算法流程

RC4 主要包括 密钥调度算法（KSA） 和 伪随机生成算法（PRGA） 两部分。

(1) 密钥调度算法（KSA）

作用：用密钥初始化 S 盒（S-Box）（256 字节的置换表）。

步骤：

初始化 S[0] 到 S[255] 为 0, 1, 2, ..., 255

用密钥 K（可变长度，通常 1~256 字节）打乱 S 盒：

python

```python
j = 0
for i in range(256):
    j = (j + S[i] + K[i % keylen]) % 256
    swap(S[i], S[j])
```

(2) 伪随机生成算法（PRGA）

作用：生成伪随机字节流（与明文异或加密）。

步骤：

初始化 i = 0, j = 0。

对每个字节：

python

```python
i = (i + 1) % 256
j = (j + S[i]) % 256
swap(S[i], S[j])
K = S[(S[i] + S[j]) % 256]  # 生成 1 字节密钥流
```

加密/解密：`Ciphertext = Plaintext ^ K`（异或操作）。

2. RC4 代码实现（Python）

python

```python
def rc4(key, data):
    # 1. KSA (秘钥调度算法)
    S = list(range(256))
    j = 0
    for i in range(256):
        j = (j + S[i] + key[i % len(key)]) % 256
        S[i], S[j] = S[j], S[i]

    # 2. PRGA (伪随机生成算法)
```

```
    i = j = 0
    result = []
    for byte in data:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        K = S[(S[i] + S[j]) % 256]
        result.append(byte ^ K)  # 异或加密/解密
    return bytes(result)
# 测试
key = b"SecretKey"
plaintext = b"Hello, RC4!"
ciphertext = rc4(key, plaintext)
decrypted = rc4(key, ciphertext)
print("Plaintext:", plaintext)print("Ciphertext:", ciphertext)print
("Decrypted:", decrypted)
```

输出：

```
Plaintext: b'Hello, RC4!'
Ciphertext: b'\xd3\x8a\xf9\x1f\xc5\x8d\x1d\xb9\xd3\x0e'
Decrypted: b'Hello, RC4!'
```

3. RC4 的安全性

RC4 曾经广泛使用，但已被证明不安全，主要漏洞：
初始字节偏差（Initial Bytes Bias）
前几个字节的密钥流有统计偏差，可被攻击者利用。

WEP 协议中的弱点
在 WEP 中，IV（初始化向量）太短，导致密钥重用攻击。
已被破解
2015 年，RFC 7465 禁止在 TLS 中使用 RC4。