Crypto_P2

对称密码

- 一次一密
- 流密码
- 块密码
  - DES
  - AES
- 加密模式

一次一密

选择和明文二进制等长的密钥，然后 `C = P xor K`。

无条件安全

如果攻击者对一次一密暴力破解，当他遍历所有密钥的同时也遍历了等长度的所有可能信息，攻击者无法从中识别出明文。

密钥配送问题

如果存在安全的方式交换和明文等长的密钥，为什么不直接交换明文?

密钥不可复用

```
C1 = P1 xor Key
C2 = P2 xor Key
C1 xor C2 = P1 xor Key xor P2 xor Key
C1 xor C2 = P1 xor P2
P2 = P1 xor C1 xor C2 (已知一对明文密文)
```

比特翻转攻击

```
A借B 1000 元
1 对应明文 '00110001'
假设此段key为 '10000111'
对应密文：'10110110'
攻击者篡改其中一个bit，新密文为：'10111110'
解密前没有检查是否被篡改，直接解密，得到：'00111001'，对应字符 9
明文变成了： A借B 9000 元
所以，需要保证密文的完整性。
```

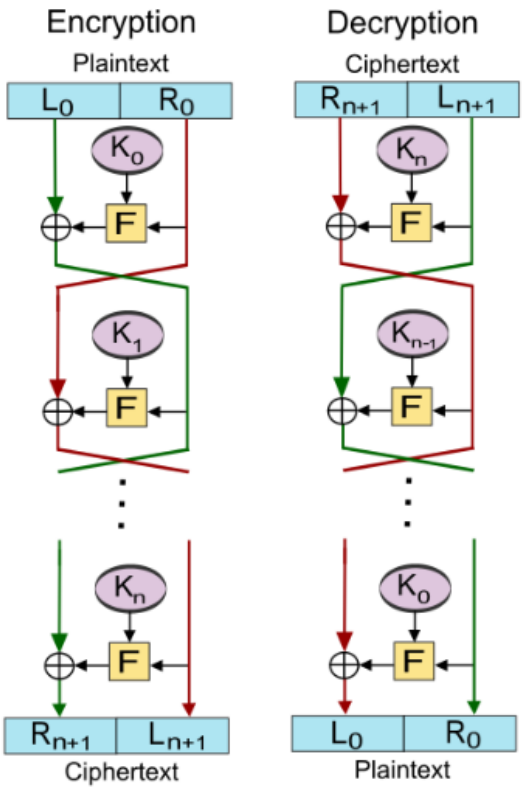
流密码

- 延续一次一密的思路，如果可以使用较短的密钥生成较长的随机比特流，就可以使用这样的比特流进行加密
- 因为密钥长度远小于比特流长度，所以可以使用安全的方式交换

DES

feistel结构

- 代替和置换 - substiution&permutation
- 混淆和扩散 - confusion&diffusion (对于DES 分别由 S-box 和 P-box 提供)
- L(i+1), R(i+1) = R(i), L(i) xor f(R(i), ki)
- 最后一轮结束后再次交换（也可以认为最后一轮不交换）



针对DES的攻击

- 有效密钥长度 56
- 线性分析 2^43
- 3DES
  - EDE 兼容 DES
  - 中间相遇攻击 安全强度

```
加密：
已知P
t1 = DESEnc(P, K1)
t2 = DESDec(t1, K2)
C  = DESEnc(t2, K3)

解密：
已知C
t2 = DESDec( C, K3)
t1 = DESEnc(t1, K2)
P  = DESDec(t1, K1)

兼容DES：
已知P
t1 = DESEnc(P, K1)
P  = DESDec(t1, K1)
C  = DESEnc(P, K1)
```

- 1 DES
  - 遍历所有2^56种可能的密钥，E(P, [Ki]) == C
- 3 DES
  - 遍历2^(56*3)种， E(E(E(P, [K1i]), [K2i]), [K3i]) == C
  - 中间相遇攻击：
    - 遍历所有2^56种可能的密钥， C1 = E(P, [K1i]) , 得到2^56种C1,存进hash tbl
    - 遍历2^(56*2)种可能的密钥，D(D(C, [K3i]), [K2i]) in hash tbl, K1,K2,K3

```
集合 A 存在若干数字，要求找出 a，b 两个数字，a + b == c。

遍历 A 的同时，把 c - a 存入哈希表，再次遍历 A，如果发现 b 在哈希表中，则找到一对满足条件的 a，b。
```

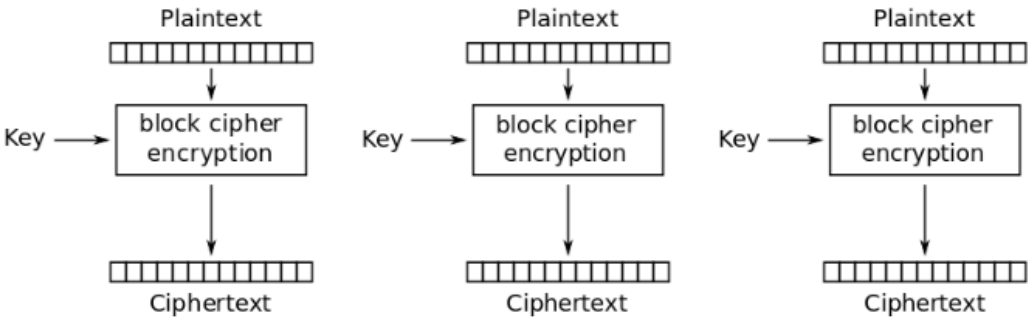
AES

- 高级加密标准
- 支持 128,192, 256 三种长度的密钥

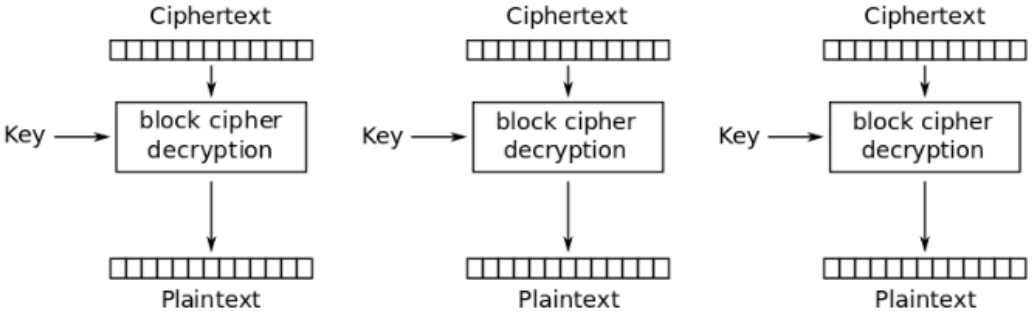
加密模式

- 用于块密码
- P = P1 | P2 | P3 ...
- C = C1 | C2 | C3...

ECB



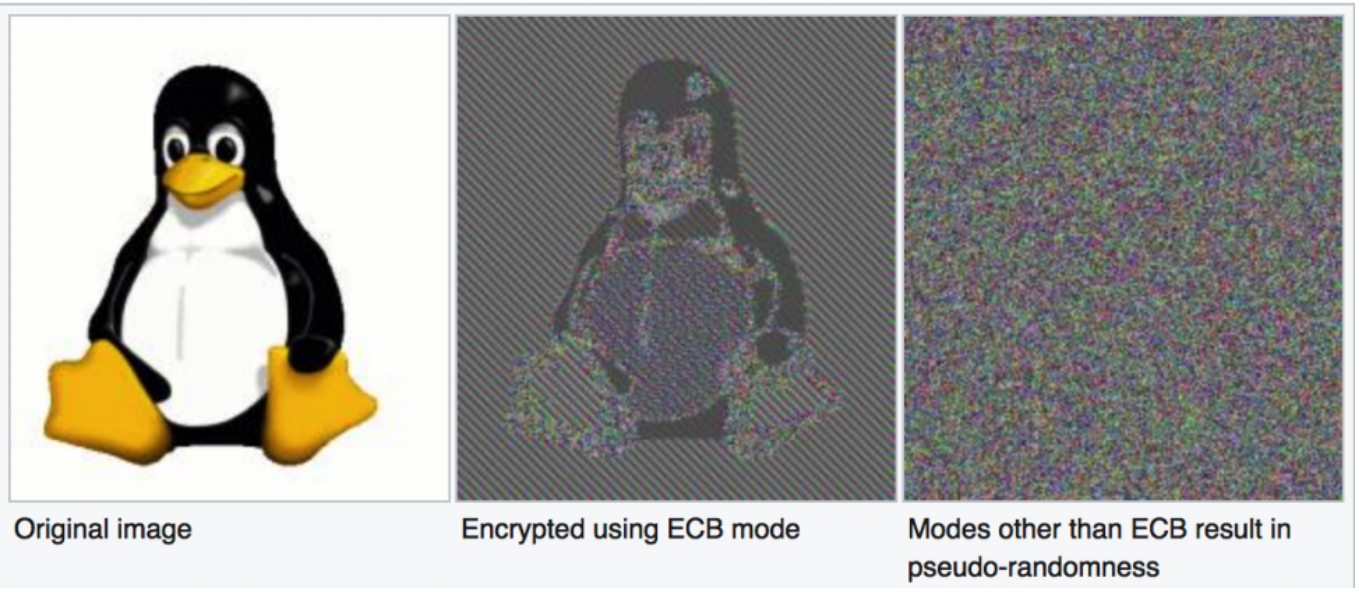
Electronic Codebook (ECB) mode encryption



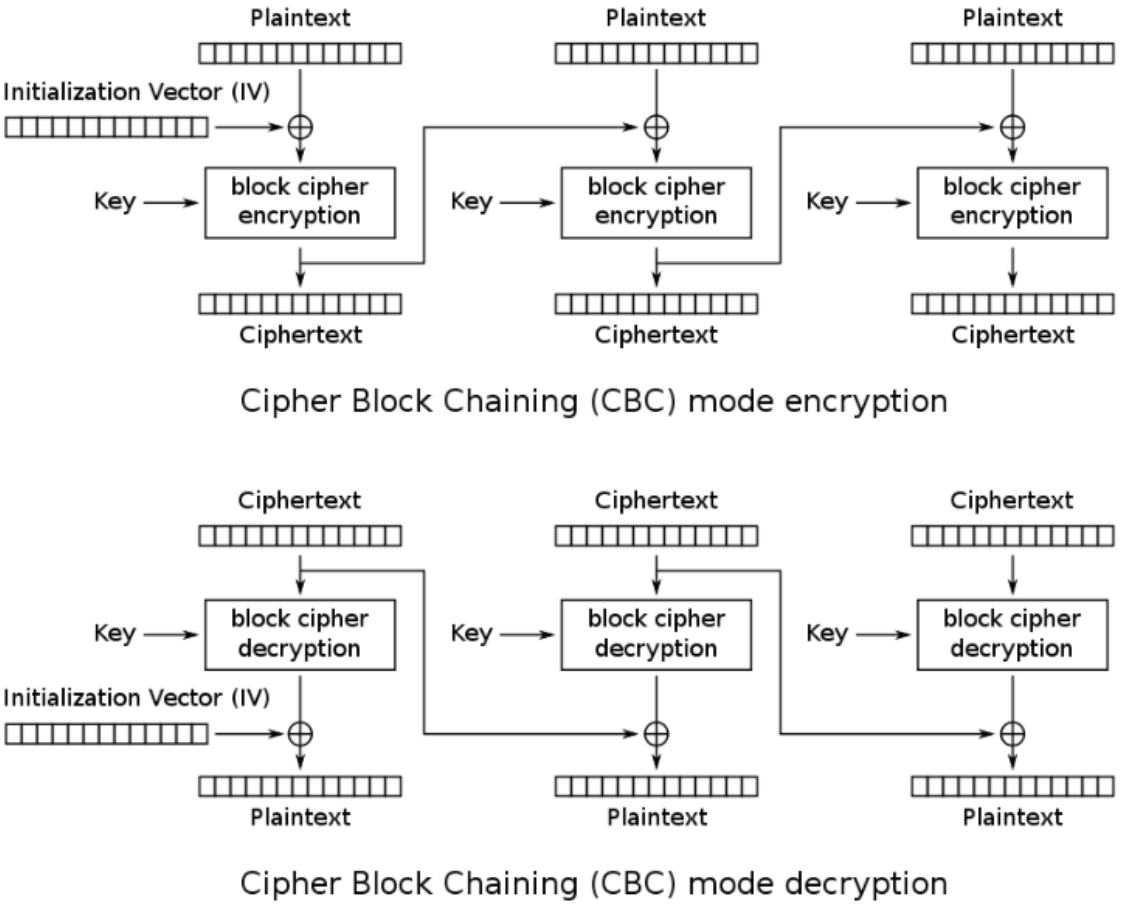
Electronic Codebook (ECB) mode decryption

- 简单直观
- C1 = E(P1, k)

- P1 = D(C1, k)
- 密文的一个比特错误，导致一个明文分组错误，但是不会影响其他明文分组。
- 加解密都可以并行。
- 对于相同的明文，会被加密成相同的密文。



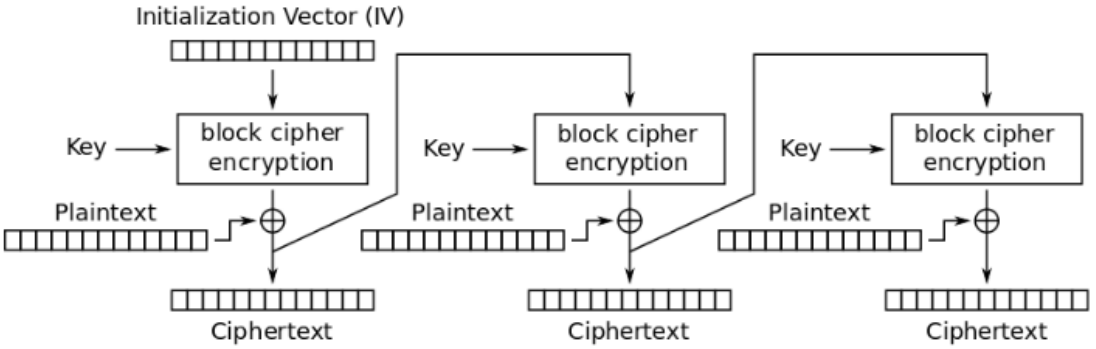
CBC



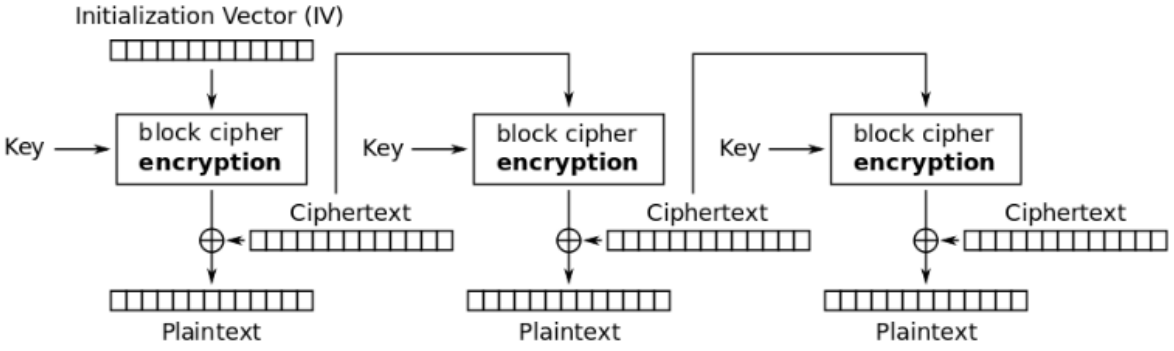
- 借用上一个分组的加密结果，获取不同的明文
- C[i] = E((P[i] xor C[i-1]), k)
- P[i] = D(C[i], k) xor C[i-1]
- IV

    - 充当C[0]
    - 避免第一个分组每次都一样
- 密文的一个比特错误，对应明文分组错误，下一个明文分组的一个比特错误，后续分组不受影响。
    - 假设C[2] 一个比特错误
    - P[2] = D(C[2], k) xor C[1] 整个分组错误。
    - P[3] = D(C[3], k) xor C[2] 一个比特错误。
    - P[4] = D(C[4], k) xor C[3] 后续分组不受影响。
- 计算:
    - 加密： C[i] = E((P[i] xor C[i-1]), k), 计算一个分组的密文，依赖前一个分组的密文。不能并行。
    - 解密：P[i] = D(C[i], k) xor C[i-1], 需要两个分组的密文，都是已知的，所以可以并行解密。
- 比特翻转攻击
    - 和流密码或一次一密类似，攻击者可以翻转密文的某个比特，使得下一个分组对应明文的一个比特翻转。
        - xxxxxxxxxxxxxxxxxxxxx username = xxx, isadmin = 0
        - doihfiljfo;sdjvifjooooooo1 username = xxx, isadmin = 1
        - 因为会影响前一个明文分组，所以比较容易识别
- Padding Oracle Attack *
    -

CFB



Cipher Feedback (CFB) mode encryption

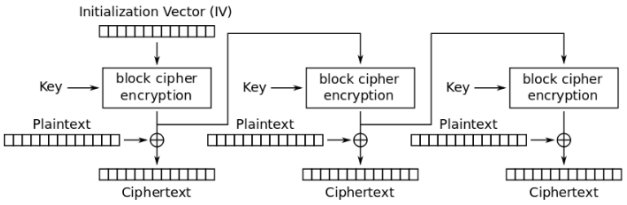


Cipher Feedback (CFB) mode decryption

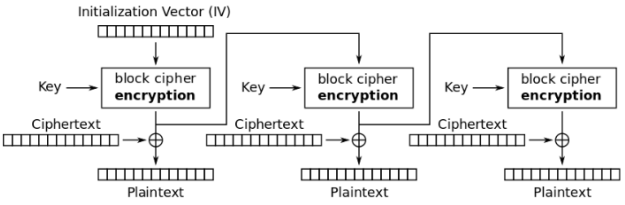
- C[i] = P[i] xor E(C[i-1], k)
- P[i] = C[i] xor E(C[i-1], k)
- 密文的一个比特错误，对应明文分组一个比特错误，下一个明文分组错误，后续分组不受影响
    - 假设C[2] 一个比特错误

    - P[2] = C[2] xor E(C[1], k) 一个比特错误
    - P[3] = C[3] xor E(C[2], k) 整个分组错误
    - P[4] = C[4] xor E(C[3], k)后续分组不受影响。
- 计算：
    - 加密：C[i] = P[i] xor E(C[i-1], k)，计算一个分组的密文，依赖前一个分组的密文。不能并行。
    - 解密：P[i] = C[i] xor E(C[i-1], k)，需要两个分组的密文，都是已知的，所以可以并行解密。

OFB



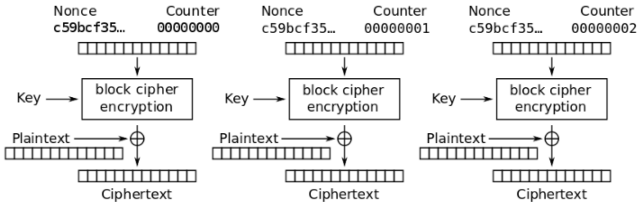
Output Feedback (OFB) mode encryption



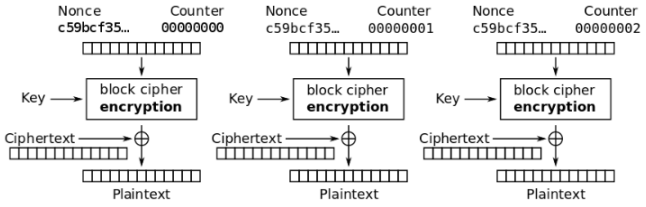
Output Feedback (OFB) mode decryption

- C[i] = P[i] xor E^i(IV, k) # E^i 指连续加密i次
- P[i] = C[i] xor E^i(IV, k)
- 密文的一个比特错误，对应明文分组一个比特错误，不影响其他分组
    - 假设C[2] 一个比特错误
    - P[2] = C[2] xor E^2(IV, k) 一个比特错误
    - P[3] = C[3] xor E^3(IV, k) 后续分组均无影响
- 计算：
    - 加密：C[i] = P[i] xor E^i(IV, k) ，虽然异或很快且可以并行，但是需要生成密钥流，问题在于密钥流每生成一个分组，都需要前一个密钥分组，所以这一步不可并行
    - 解密：同加密
    - 密钥流产生和明文无关，所以加密方可以预先计算
- 比特翻转攻击：流密码的特性
- IV 不能重复

CTR



Counter (CTR) mode encryption



Counter (CTR) mode decryption

- C[i] = P[i] xor E(Counter[i], k)
- P[i] = C[i] xor E(Counter[i], k)
- 密文的一个比特错误，对应明文分组一个比特错误，不影响其他分组
    - 假设C[2] 一个比特错误
    - P[2] = C[2] xor E(Counter[2], k)一个比特错误
    - P[3] = C[3] xor E(Counter[3], k) 后续分组均无影响
- 计算：
    - 加密：C[i] = P[i] xor E(Counter[i], k)， 可以并行
    - 解密：P[i] = C[i] xor E(Counter[i], k)， 可以并行
    - 密钥流产生和明文无关，所以加密方可以预先计算
- 比特翻转攻击：流密码的特性
- IV 不能重复，否则出现密钥流重复