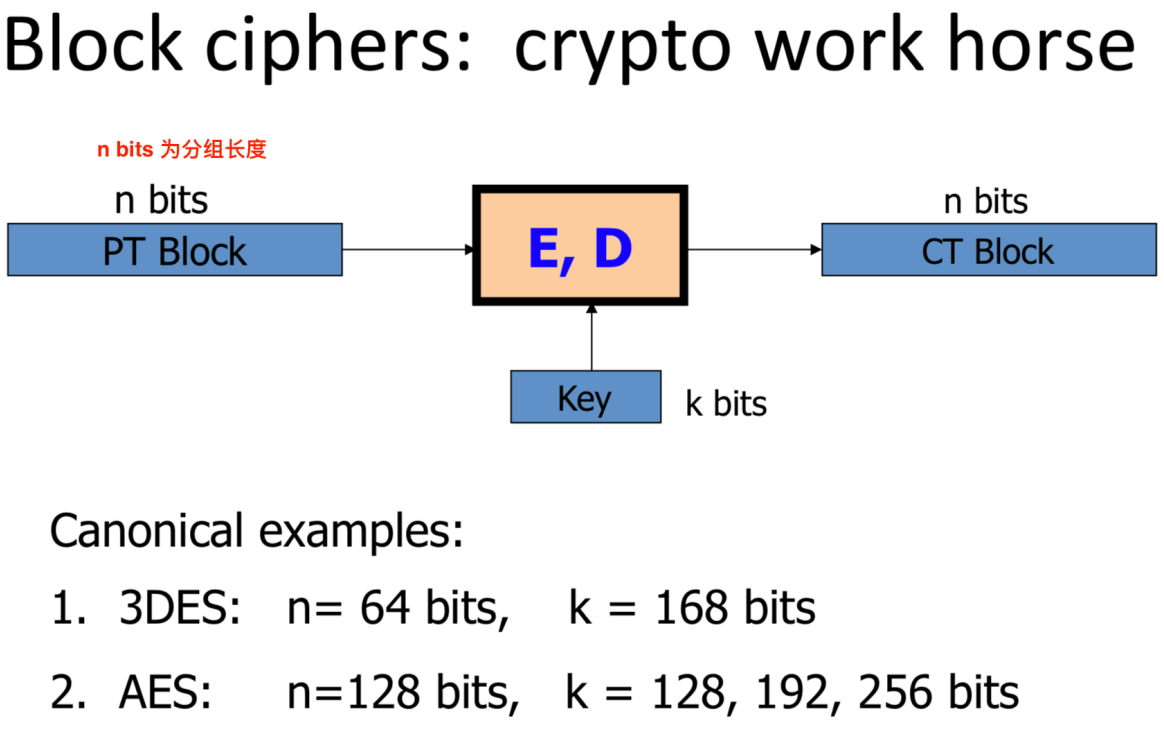
分组密码

定义



- n 为 block size ; k 为key size

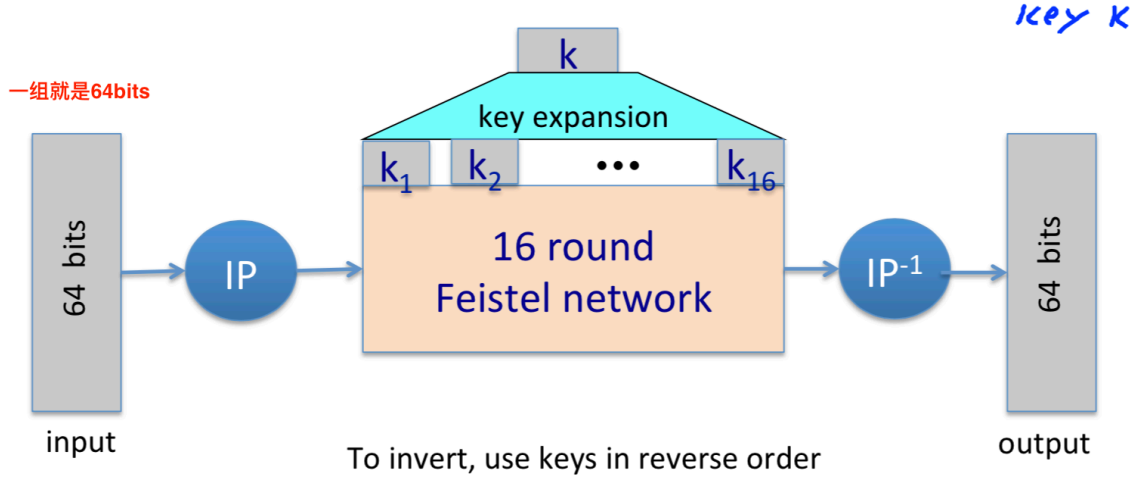
real - wrold block cipher

DES

- key的扩展

$$f_1, \ldots, f_{16}: \{0,1\}^{32} \longrightarrow \{0,1\}^{32} \quad , \quad f_i(x) = F(k_i, x)$$

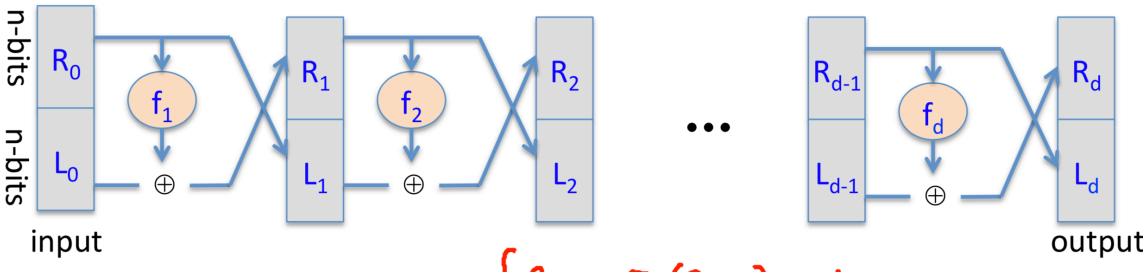
from key K

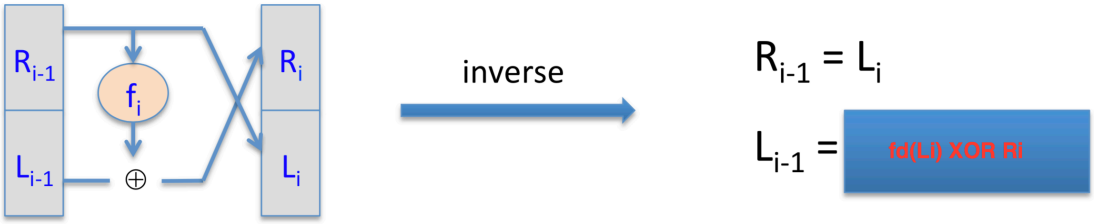


在DES中, 一个Feistel Network有16组 f_i ,这16和生成的key对应了这些 f_i

这一个过程本质上就相当于一个block cipher 的一个组和一个密钥匙的加密函数 E(block[i], key), 组大小为 64 bits, key大小为48bits, 再经过S-BOX 之后生成一个32bits的key

- core idea : Feistel Network



- f1为一个 nbits 到 nbits 映射
- 整个即为了**得到 2n bits 到 2n bits 的映射 , 其实是PRF到PRP的扩展, fi 不可逆, 但是整个结构可逆**
- f 1...d 都不用可逆, 因为 这个网络结构就是可逆的
- 

- 注意fi 接受的为两个参数 , 一个为key , 一个即为序列, **且一般每一轮的key都不一样**
- 理论上, 在fi为保证安全的PRF的情况下, 只需要三轮就可以保证这个PRP是安全的(三轮用的密钥不同)
- **在DES结构中, 这个有16轮, 为了防止不安全的PRF**
- **传统的DES, key = 56 bits, 将64bits 的block 映射到另一个 64bits**
- |key| = 56bits 的 DES 很容易就被穷搜法给攻破

3DES

- DES的三倍扩展
- def : E((k1, k2, k3), m) = E(k1, E(k2, E(k1, m)))

- 密钥增长三倍 key = 168bits, 速度减慢三倍. **In fact : simple apack in time ≈2^{118}**
- **DES攻击的目标就是 , 给定少量的(Mi , Ci)对, 求的中间的密钥key, 此时只知道Mi , Ci . 不知道key, 但是可以通过穷举法, 一个个尝试key加密M和C做对比, 最后可以得出正确的key**
- 不用2DES的原因是: 可以利用 meet in the middle (以空间换时间) 的方法 : 最后结果时间复杂度和DES是一样的
- 3DES 的 meet in the middle attack , 解释了为什么2DES 的时间复杂度其实的 2^{56} : 2DES 即为 E((k1, k2), message) => E(k1, E(k2, m))

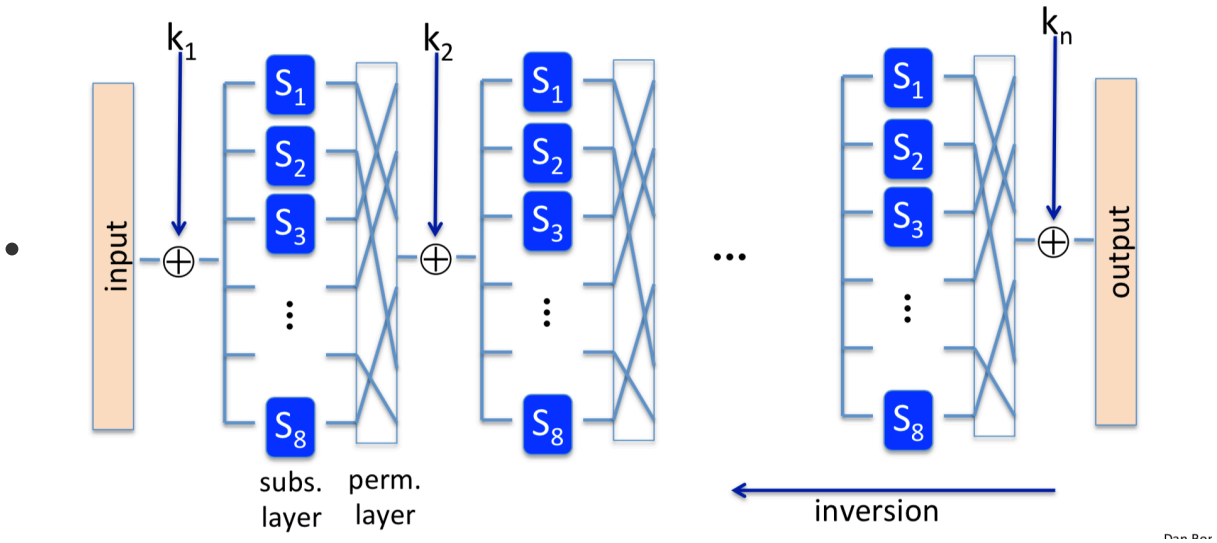
  - stpe 1 : 尝试2^{56} 个 k2 ,分别和m 做加密操作 , 建立一个table :

k2	E(k2, m)
{0...0}	output
{0...1}	output
...	...

  - step 2 : 尝试2^{56} 个 k1 , 分别对c 做解密操作 ,得到的值 和table中的output做比对, 如果匹配了, 则此时遍找到了(k1, k2)
  - 时间复杂度 : $2^{56}+2^{56} = 2^{56}$
  - 空间复杂度 : 2^{56}
- 同理 3DES 也可以用meet in the middle attack 成功把时间复杂度下降到2^{118}

AES

- block size : 128bits , key size : 128 , 192 , 256 bits
- **Core idea : Subs-Perm network (SPN)** , not same as DES which is Feistel Network



Attack types

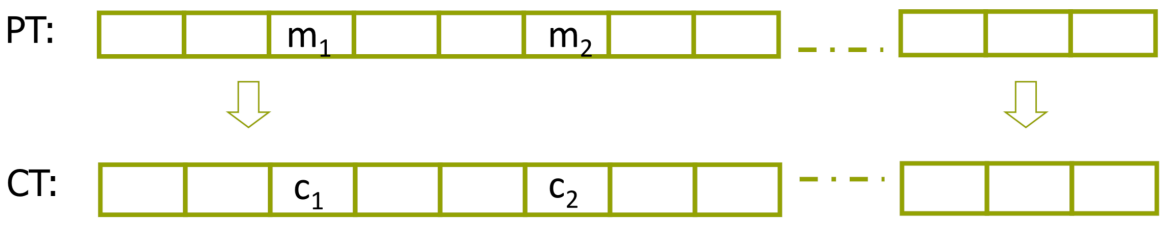
- Side channel apacks (侧信道攻击, 物理手段)
- Fault apacks
- Linear and differential apacks
- Linear apacks
- Quantum apacks

加密模式

ECB

- 不安全
- 将message 切分为和密钥 key 相同的长度做映射

Electronic Code Book (ECB): 不安全的模式

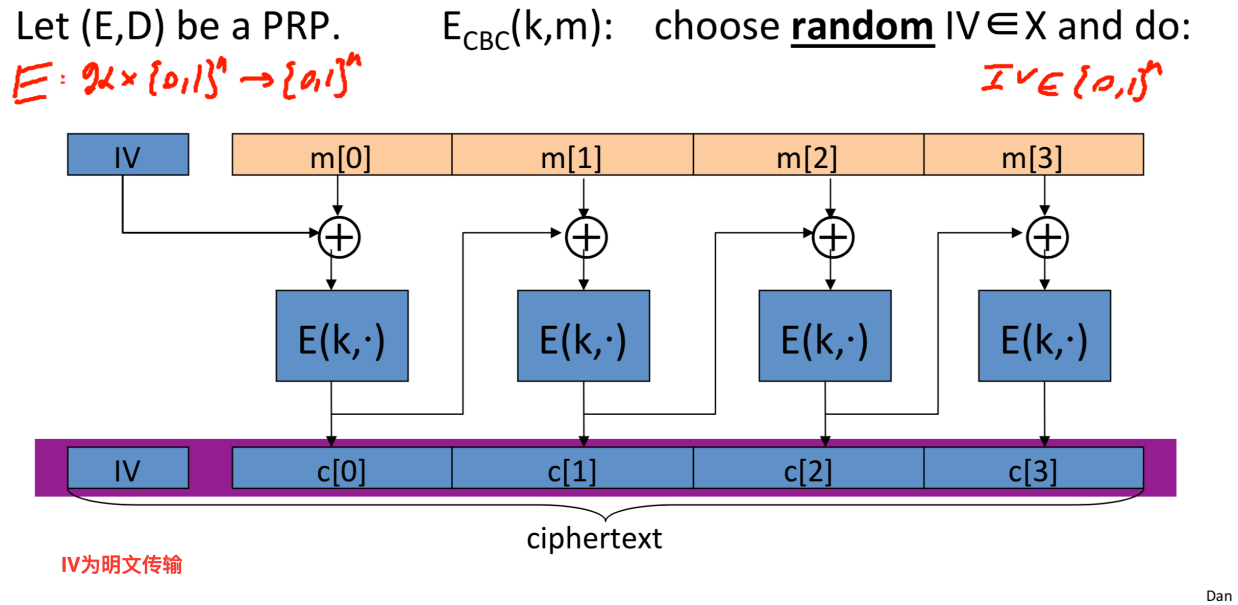


if m1 = m2 then c1 - c2

- 连 one - time key 的 语义安全 都不满足
  - e.g. m0 = "hello hello" m1= "hello world" , 就明显可以区分实验一还是实验二

CBC

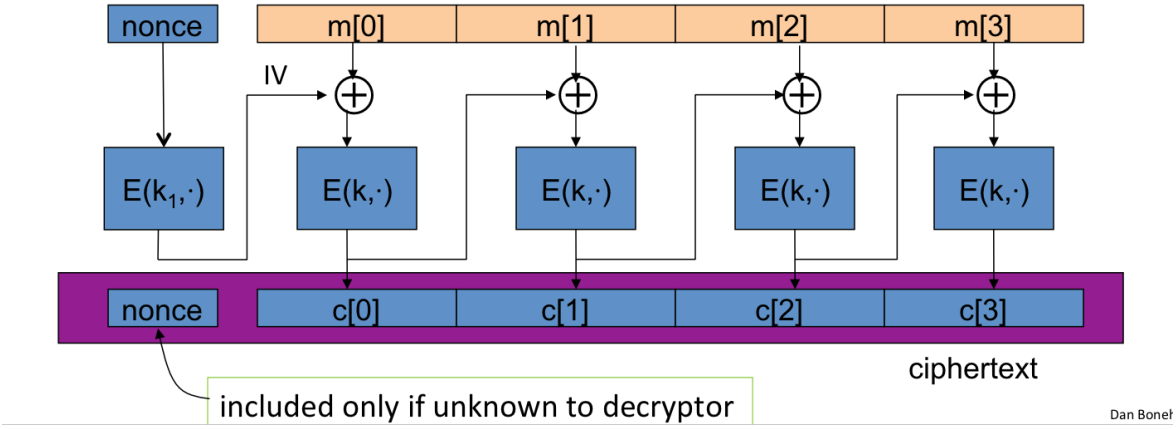
- use PRP
- 构造方法 :
  - 加密函数 : E(M, key)
  - 随机选择一个IV, 有一个对称的加密函数 E_{tiny}(m, k), 且这些小的加密函数都是一样的. 负责加密每一块的消息 message
  - 加密的时候 :
    - E_{tiny}(IV XOR m[0], key) 形成密文的第一块 C[0]
    - E_{tiny}(c[0] XOR m[1], key) 形成密文的第二块 C[1]
    - .....
  - 传输的时候, **IV为明文传输**, 一般放于密文的头部
  - 解密函数 : D(C, key)
  - 先提取出密文消息首部的IV, 且有一个和加密函数对应的解密函数 D_{tiny} (m, k)
  - 解密的时候 :
    - D_{tiny} (c[0], k) 得到 IV XOR m[0] , 再异或 IV 得到了 m[0]
    - D_{tiny} (c[1], k) 得到 c[0] XOR m[1] , 再异或 c[0] 的动了m[1]



- 如果CBC模式的IV是可以预测的, 那么CBC模式将不是CPA安全的
-
- 变化1 : 基于nonse 的CBC模式 , **注意有两个key**

  - Cipher block chaining with unique nonce: key = (k,k_1)

    unique nonce means: (key, n) pair is used for only one message

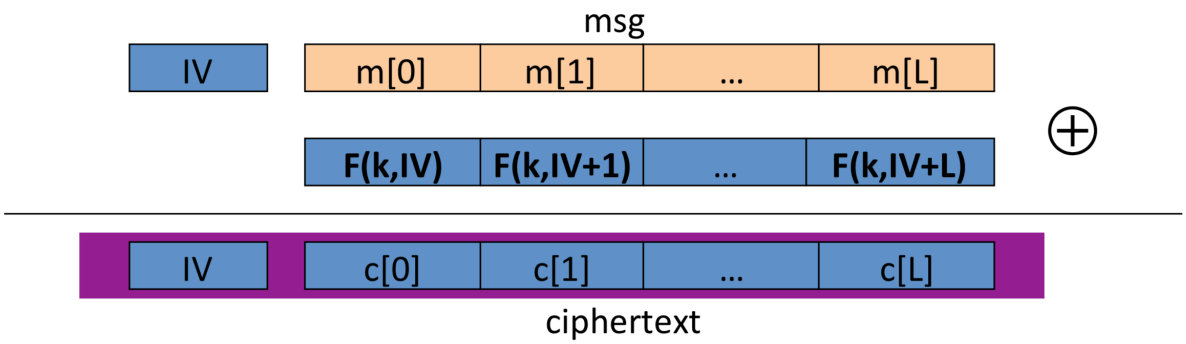


CTR

- 不同于CBC , CTR是并行的
- use PRF

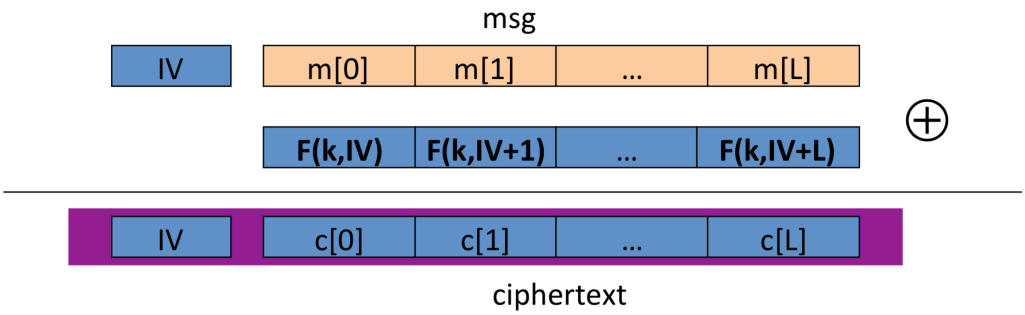
E(k,m): choose a random $IV \in \{0,1\}^n$ and do:

并行的

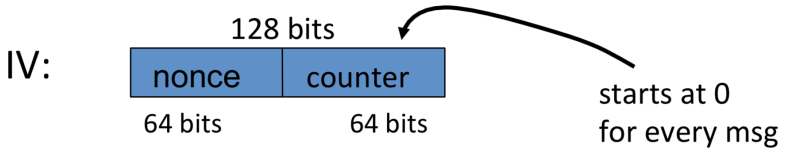


- 基于nonse 构造的, CTR

Construction 2 : nonce ctr-mode



To ensure F(k,x) is never used more than once, choose IV as:



Dan Bc

这里的IV有64bits取自于nonce