公钥私钥体系

- 非对称加密

攻击方式

- 对于对称加密来说 : 一个安全的对称加密是保证了完整性和机密性的(AE), 所以攻击者是不能自己创造出新的密文(即**所有的密文都是根据挑战者返回的, 而不能自己创造**)进行攻击
- 对于非对称加密来说 : 攻击者有pk , 从而可以创造新的密文, 所以需要直接考虑CCA安全

体系构造的函数

- G() 生成一个pk 和 sk
- E(pk, m) 利用 pk 加密
- D(sk, c) 利用 sk 解密得出m

构造CCA安全级别的公钥私钥体系

1. TDF (Trapdoor functions)

- 定义于 (G, F , F^{-1})
- G用于生成 sk, pk , 且 F^{-1} 利用 sk 解密的时候, 不能没有sk
- **知道输入参数求输出很容易, 但是知道输出求输入参数则很困难(单向性)**

2. public-key encryption from TDF

基于三个函数体系

- TDF : (G, F , F^{-1})
- (Es, Ds) 对称加密的AE体系 : (K, M, C)
- hash 函数

首先生成sk 和 pk 是基于TDF 的 G() 函数

E(pk, m) :

$x \xleftarrow{R} X$, $y \longleftarrow F(pk, x)$

$k \longleftarrow H(x)$, $c \longleftarrow E_s(k, m)$

output (y, c)

D(sk, (y,c)) :

$x \longleftarrow F^{-1}(sk, y)$,

$k \longleftarrow H(x)$, $m \longleftarrow D_s(k, c)$

output m

加密方 :

- 随机生成一个x , 然后给TDF 的 y = F(pk, x)函数加密
- 给这个x做hash 得到一个key , 然后用这个key给对称AE加密的c = Es(k, m) 加密 messages
- 发出 (y, c)

解密方 :

- 利用TDF 的 F^{-1} (sk, y) 解密出 x
- 对x做hash 得到 key
- 利用key作为对称key解密 D(c, key) 得到message

结论:

如果 TDF 安全 , (Es Ds) is AE , Hash函数is a “random oracle” : 这个模式就是CCA 模式的安全

- **不要用 TDF 的方式去直接加密message , 这样连语义安全都不是**
- 基于Trapdoor functions Schemes : IOS standard , OAEP + , ...

基于RSA的TDF加密模式的公私钥加密体系

- 基于RSA的TDF基于算法为:
  - RSA.TDF 中的G() : 随机生成大的素数p q let N = p * q , 在N中找到 e, d 使得e, d互为逆元即 e*d = 1 In N

    Let **pk = (N, e) ; sk = (N, d)**
  - 加密 : F(pk, x) = x^e In N
  - 解密 : F^{-1}(sk, x^e) = x^{ed} = $x^{K*\phi(N)+1}$ = $x^{K*\phi(N)}$ * x = x
- 所以基于RSA的机密体系 (G, E, D) 被定义为
  - G() 为RSA.TDF 的输出 : (pk, sk) 即分别为 ((N, e), (N, d))
  - E(pk, m) = E((N, e), m)为 :
    - 选择一个随机的x in N
    - 对x进行TDF的加密 即为 RSA(x) = x^e
    - 对x进行hash作为key
    - 对message 用 可以加密 => Es(key, message)
    - output (x^e , Es(key, message))
  - D(sk ,c) = E((N, d), c)为 :
    - 根据F^{-1}(sk, x^e) 解密 得到 x
    - 对x做Hash 函数 得到 key
    - 利用 Ds (key, ciphertext) 得到 message
- **同样, 别利用RSA.TDF直接加密明文, 这是不安全的, 甚至连语义安全都达不到!!!**

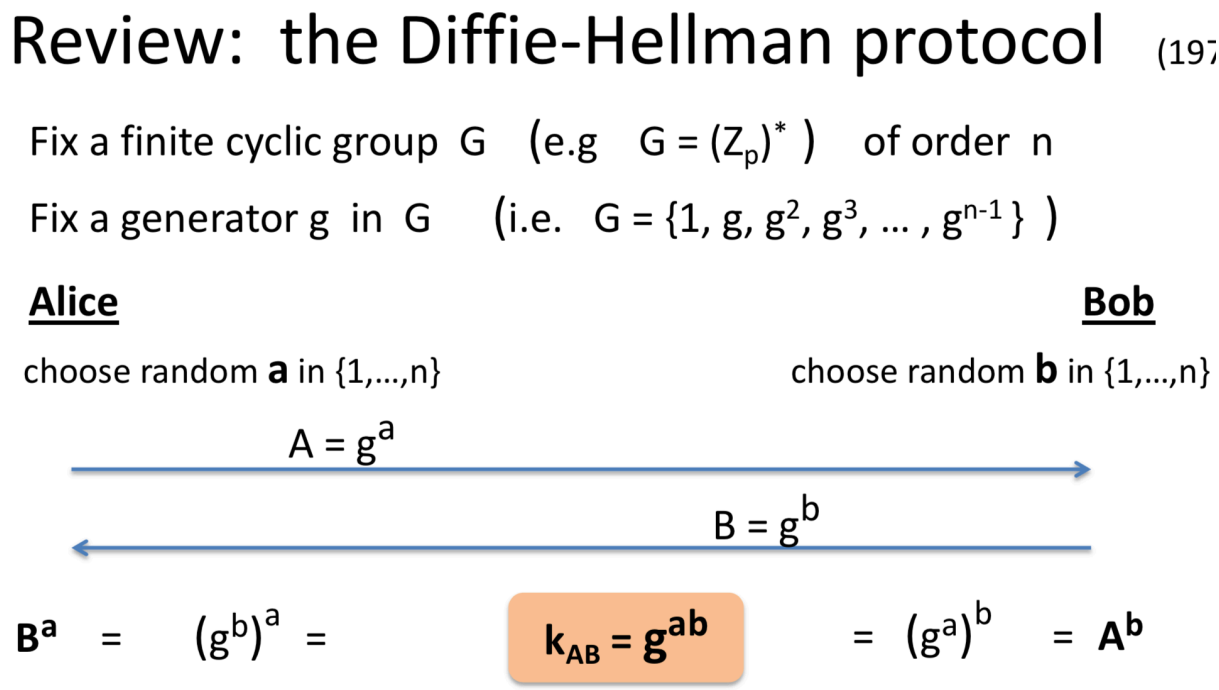
Publick key system from Diffie-Hellman protocol

- 第二种构造CCA安全级别的公私钥匙体系的方法
- Different from based on TDF
- Schemes : **Elgamal encryption , variants**

- 安全级别 : CCA

Elgamal based on Diffie - Hellman build a publick system

- Diffie - Hellman :



- Elgamal :

We construct a pub-key enc. system (Gen, E, D):

- Key generation Gen:
  - choose random generator g in G and random a in Z_n
  - output sk = a , pk = (g, h=g^a)

Dan Boneh

The ElGamal system (a modern view)

- G: finite cyclic group of order n
- (E_s, D_s) : symmetric auth. encryption defined over (K,M,C)
- H: $G^2 \longrightarrow K$ a hash function

E(pk=(g,h), m) :

b $\xleftarrow{R}$ Z_n , u $\longleftarrow$ g^b , v $\longleftarrow$ h^b

k $\longleftarrow$ H(u,v) , c $\longleftarrow$ E_s(k, m)

output (u, c)

D(sk=a, (u,c)) :

v $\longleftarrow$ u^a

k $\longleftarrow$ H(u,v) , m $\longleftarrow$ D_s(k, c)

output m

Dan Boneh

- Gen()函数 , 输出sk, 和 pk , 其中 sk 为族中随机选择的 a , pk 为 (g, g^a) **(攻击者知道g, g^a, 求出a很困难)**
- 加密函数 E(PK, m) = E((g, g^a), message) :
  - 首先在族中随机选择一个b
  - 计算 u = g^b , v = g^{ab}
  - 计算哈希值 H(u, v) = key
  - 利用key 进行对称机密 => ciphertext = Es(key, message)
  - output (u, ciphertext)
- 解密函数 D(sk, (u, ciphertext))
  - u = g^b , 计算出 v= g^{b*sk} = g^{ab}
  - Hash (u, v) = key
  - 对称解密 D(key, ciphertext) = message

- Elgamal 基于困难问题 CDH : **已知 g, g^a , g^b 求出 g^{ab} 很困难**