# 密码学中的CATCH-22问题

**在保密通信中，通信双方在进行秘密通信之前，他们首先得共享密钥，即需要首先进行一次分发密钥的秘密通信，但“分发密钥的秘密通信”是否是安全的不能得到证明**

**似乎进行“秘密的分发密钥”是不可避免的，两千年来，它一直被认为是密码学的一个定理。**

对称密码体制的缺陷：

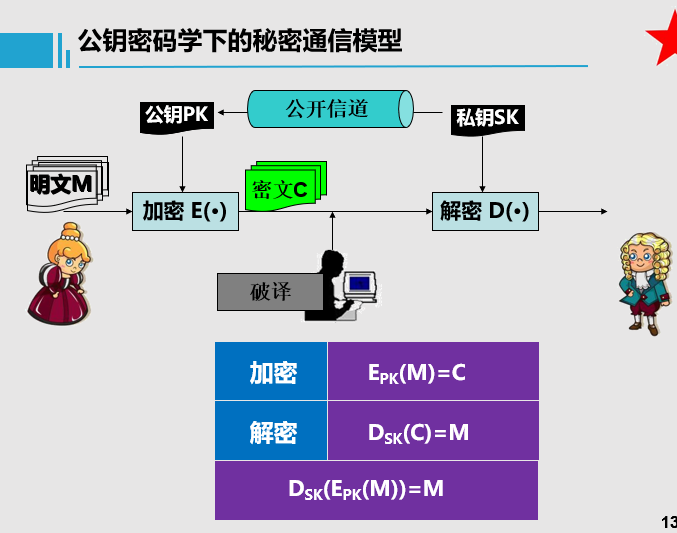
* 密钥管理困难（n个用户互相通信，系统中共有n(n-1)/2个密钥。如n=100时，共4,995个密钥。密钥爆炸。）
* 无法实现非否认（因为解密者也可以加密）
* 密钥分发困难（与 “CATCH-22 问题”有关(如何安全地分发密钥)密钥分发不仅要耗费巨大的成本，而且也很容易成为保密通信中的薄弱环节！）

分发管理困难，无法实现非否认

## 公钥密码学

* **基本思想：**
  + Alice 有一个 加密密钥 PK，以及一个解密密钥 SK
  + PK 公开，SK 保密（要求 PK 的公开不能影响 SK 的安全）
  + Bob 要向 Alice 发送明文 M 时，可先查 Alice 的加密密钥 PK，并用 PK 加密得密文 C
  + Alice 收到 C 后，用只有她自己才掌握的解密密钥 SK 解密 C 得到 M。
  + A有加密密钥和解密密钥，B通过加密密钥得到密文，A收到密文后只能用自己掌握的解密密钥来解密密文
* **特点：**
* 加密与解密由不同的密钥完成
* 公钥: 用于加密，任何人都可以知道(包括攻击者)
* 私钥: 用于解密，必须保密
* 加解密的非对称性，故而又称 **非对称密码**

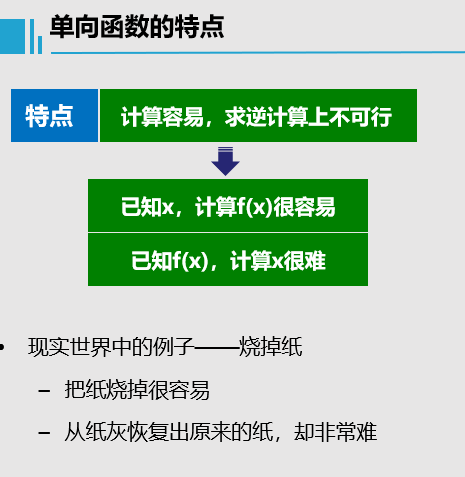
**公钥密码学：加密密钥公开，解密密钥私有。故又称非对称密钥**



**Q: 从未见过面的两个人如何实现秘密通信？**

* 公钥密码的提出，首先是解决 **从未见过面的两个人如何实现秘密通信**的问题。
  1. 实质是，如何在公开信道上(肯定不安全)，实现对称密钥的秘密分发。
* Diffie和Hellman提出的“Diffie-Hellman密钥交换协议”正是为解决这一问题
* 他俩设想公钥加密的可能性，但没能提出解决方案

**单向函数的概念是公钥密码学的中心概念**



单向函数计算容易求解难

**Q: 单向函数在密码学中很有用，但它们是否存在呢？**

**没有人证明单向函数是否真的存在，也没有实际的证据能够构造出真正的单向函数**

**但是，有很多函数看起来像单向函数**，**我们能有效地计算它们，且至今还不知道有什么办法能容易地求出它们的逆**

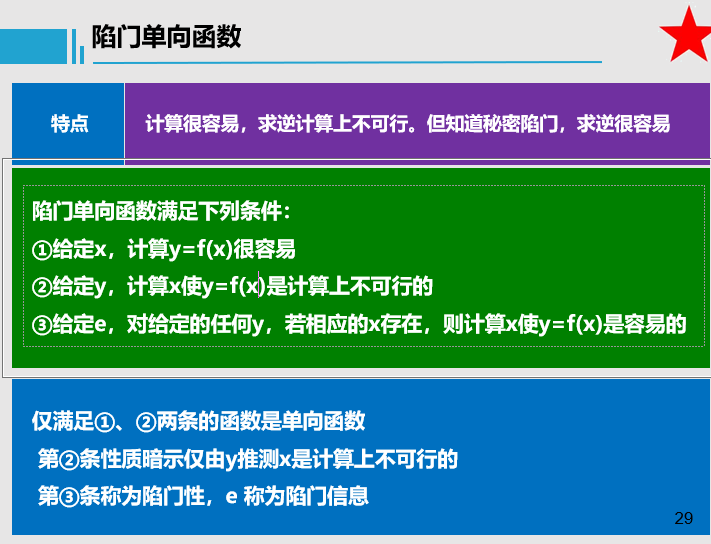
有很多函数像单向函数但是**也没有实际的证据能够构造出真正的单向函数**

## 单向函数的作用

作用：**单向函数不能用于加密，因为用单向函数加密的明文，没人能解开**

**陷门单向函数**

特点：**计算很容易，求逆计算上不可行。但知道秘密陷门，求逆很容易**



**注意 不是所有的困难问题都能转化成密码体制**

* **一个困难问题可以转化为密码体制必须满足以下条件：**
  1. 能将陷门信息 **成功且安全** 地嵌入到该问题中，使得只有用该陷门信息才可以有效求解

困难问题转化为密码体制必须满足能将陷门信息成功且安全的嵌入到该问题中，使得只有用该陷门信息才可以有效求解。

候选的单向函数：

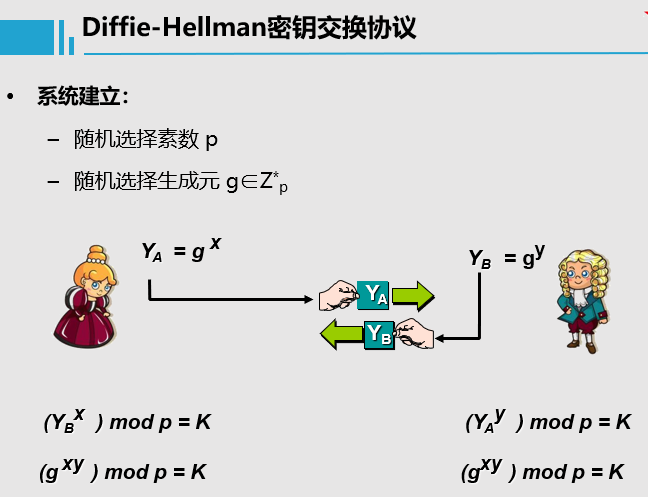
* 大整数分解问题（已知n是两个大p和q的乘积对n进行分解求出p和q）
* 离散对数问题
* Diffie-Hellman问题（
* 给定 (p, g, gx, gy)，p 是素数，g∈Zp\* 是生成元
* 计算Diffie-Hellman问题 (CDH)
* 计算gxy
* 判定Diffie-Hellman问题 (DDH)
* 给定T∈Zp\* ，判断 T 是否等于 gxy）

它们是否真是单向函数，并未得到证明。但到目前为止，还没找到多项式时间算法能解决这些问题

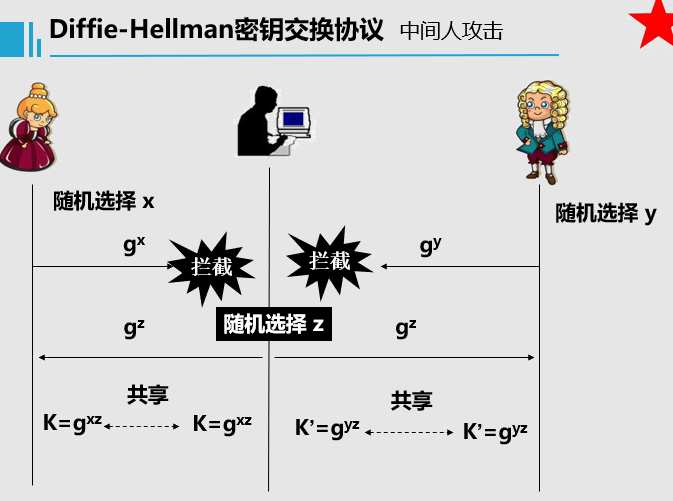
他们是困难的只是一个假设（困难假设）

**解决从未见过面的两个人如何实现秘密通信**

* 如何在公开信道上(肯定不安全)，实现对称密钥的秘密分发。
* Diffie-Hellman密钥交换协议正是为解决这一问题而提出的
  1. 基于的困难问题被称为Diffie-Hellman问题



* 注意
  1. 该方案不能用于交换任意实际的信息
  2. 仅允许安全地共享一个密钥，用于后续通讯中使用对称密码体制
* **目前，p的长度至少应为512比特**



**如何防止中间人攻击**

* 需要对传送的信息进行**认证**保护，以使通信双方知道到底在与谁进行密钥交换

RSA加密方案

1.流程

1. 系统建立
   * 随机选择大素数p、q，计算n＝pq
   * 随机选取e <ø(n)，且gcd(e, ø(n))=1
   * 计算d，使 ed≡1 (mod ø(n))
   * (e，n)为公钥，d 为私钥
2. 加密：c＝me mod n, m∈Z\*n
3. 解密：m＝cd mod n

2.解密过程的正确性

* 根据欧拉定理有:
  + mkø(n)＋1  = m mod n
* 在RSA中
  + 有 e 和 d 满足 ed ≡ 1 (mod ø(n))
  + 因此，存在k使得 ed = 1+ kø(n)
* 因此  
  cd ≡ (me)d ≡ m1+kø(n) ≡ m (mod n)
* **欧拉定理** 设n≥2，如果gcd(a,n)=1，则a*φ*(n) ≡ 1 (mod n)

**RSA加密方案** RSA中的计算问题

1. **如何产生两个大素数？**
   * 素性检测
2. **如何计算d？**
   * d满足 ed≡1 (mod ø(n))，且gcd(e, ø(n))=1
   * 实际上，d和e在模ø(n)下互为逆元
   * 用扩展的欧几里得算法计算
3. **加解密**
   * 实际都是对一个整数的幂运算，再求模
   * 使用 乘法链算法 可以降低计算复杂度

**RSA加密方案** 安全性

1. **分解模数n**
   * 理论上，RSA的安全性基于分解大整数是困难的这一假设
     + 如果模数n被成功地分解为p和q，则可立即计算出*φ*(n),从而能确定e在*φ*(n)下的乘法逆元d，便可成功破译（解密密文）
   * 但数学上至今未能证明分解n就是攻击RSA的最佳方法，也未证明分解大整数就是NP问题。
   * 随着人类计算能力的不断提高，原来被认为是不可能分解的大数已被成功分解
2. **目前，n的长度至少介于1024到2048比特之间**

**RSA加密方案** 对RSA的攻击举例——共模攻击

**为方便起见，可能给每个用户相同的模数n，不同的加解密密钥，但这样做是不安全的**

设两个用户的公钥分别为e1和e2，且gcd(e1,e2)=1，明文是m，密文分别是

c1=me1 (mod n), c2=me2 (mod n)

**攻击者截获c1和c2后，可以这么做：**

* 1. 计算两个整数r和t,满足

re1 + te2 = 1 (扩展的欧几里得算法)，其中一个必为负数，设为r

* 1. 计算c1在模n下的逆元c1-1
  2. 恢复明文m如下

(c1-1)-rc2t ≡ c1rc2t ≡ mre1 + te2 ≡ m (mod n)

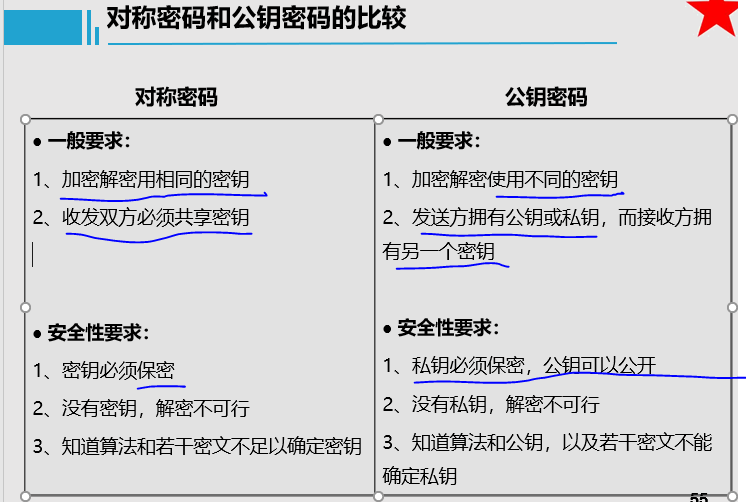
RSA加密方案 参数选择

* **为保证算法安全性，要求p和q满足以下要求**
  1. p、q不能相同，同时既不要太接近，又不能差别太大
  2. p、q的长度应该差不多(512位)
  3. p、q是安全素数
     + **安全素数：**形如p=2p1+1的素数, p1也是素数
     + **目的：**确保p-1、q-1都有大的素因子，以增加猜测*φ*(n)的难度
  4. gcd(p-1,q-1)应当小
* 另外，d要大于1/3n1/4，否则易遭到连分式算法(Wiener’s attack)的攻击
* e的选择也要注意
  + 一般选 e=3、17、65537(216+1)，二进制表示只有两个1
  + 加密前先用随机比特填充m，使之与n有相同的长度。PEM、PGP等软件都是这么做的。
  + 这样，即使多个用户使用相同的e，使用这三个值也没有任何安全问题，且可提高计算速度

**RSA加密方案** 缺点

* **安全性问题**
  + 还无法证明对RSA的破译是否等同于大整数分解问题
  + 只是到目前为止，RSA似乎是安全的
* **速度问题**
  + 增大p和q将使计算开销指数级增长
* **选择合适的p、q对于普通用户来说比较困难**
  + 因为他们没有相关的知识。

**对称密码和公钥密码的比较**

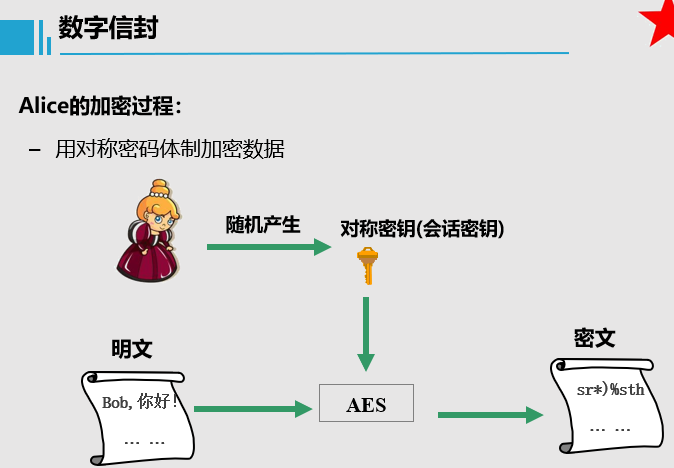


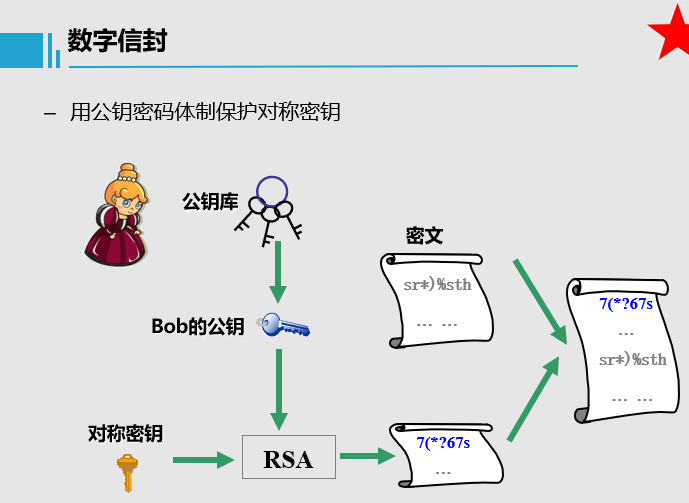
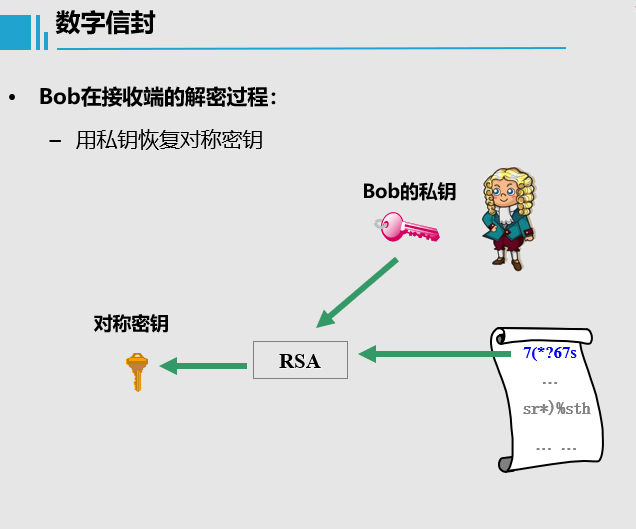
**公钥密码小结**

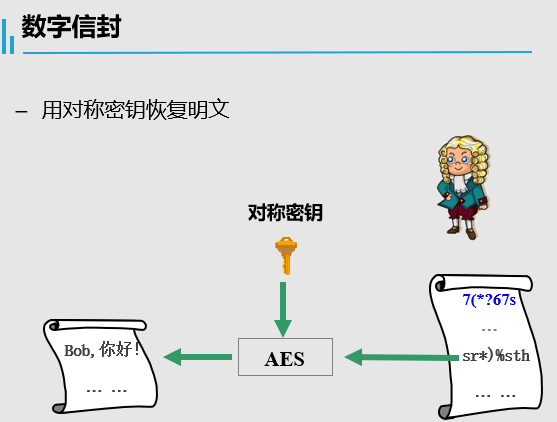
* **缺点**
  + 计算速度慢
  + 密钥长
  + 应用历史短
* **误区**
  + 公钥密码更安全
  + 对称密码体制已经过时
    - RSA最快的情况也比DES慢上100倍，速度一直是公钥密码体制的缺陷。
  + 公钥分发十分简单

数字信封

* **对称密码、公钥密码各有长短**
  + 对称密码体制：高效，适用于加密大量数据，但密钥分发困难
  + 公钥密码体制：效率低，不适于加密大量数据，但灵活
* **数字信封(Digital Envelope)**
  + 目的：取长补短，利用对称密钥加密数据，用接收者公钥保护该对称密钥





**数字信封** 总结  
**发送方**

* 1. 随机生成对称密钥, 用该密钥加密明文
  2. 用接收方的公钥加密该对称密钥
  3. 将①、②的结果发送给接收方
* **接收方**
  1. 用自己的私钥恢复对称密钥
  2. 用该对称密钥解密密文
* 考虑数据完整性的话，还应加入完整性技术