通 信 安 全

L6—非对称加密算法

- 教师：崔爱娇

- 编号：ELEC3019
- 学时：32学时

概 要

非对称加密算法

- 加密算法
- 非对称加密算法
- RSA加密算法
- 其他非对称加密算法

概 要

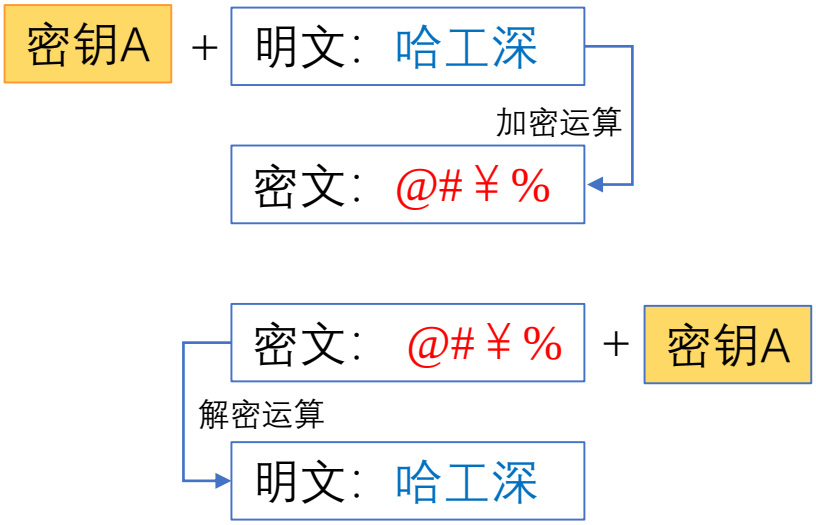
非对称加密算法

- 加密算法
- 非对称加密算法
- RSA加密算法
- 其他非对称加密算法

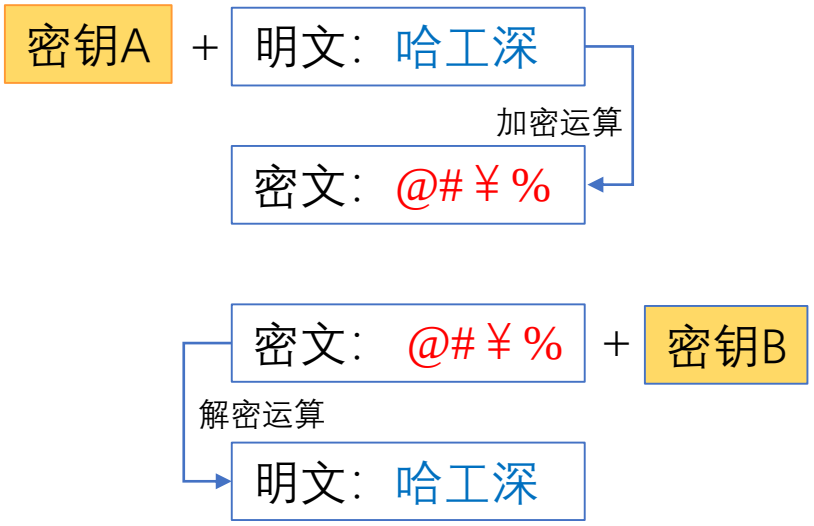
加密算法

加密：原来为明文的文件或数据按某种算法进行处理，使其成为不可读的一段代码为“密文”，使其只能在输入相应的密钥之后才能显示出原容，通过这样的途径来达到保护数据不被非法人窃取、阅读的目的。 [1]

对称加密算法



非对称加密算法



[1] https://baike.baidu.com/item/%E5%8A%A0%E5%AF%86%E7%AE%97%E6%B3%95/2816213?fr=aladdin.

概 要

非对称加密算法

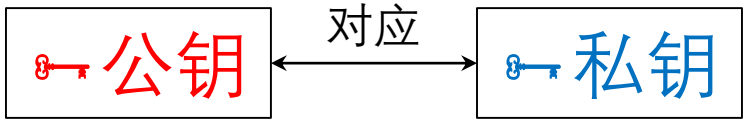
- 加密算法
- 非对称加密算法
- RSA加密算法
- ECC加密算法

非对称加密

W.Diffie和M.Hellman 1976年在IEEE Trans. on Information刊物上发表了“New Direction in Cryptography”文章，提出了“非对称密码体制即公开密钥密码体制”的概念，开创了密码学研究的新方向。

非对称加密算法需要两个密钥：公开密钥（public key:简称公钥）和私有密钥（private key:简称私钥）

公钥私钥的使用原则

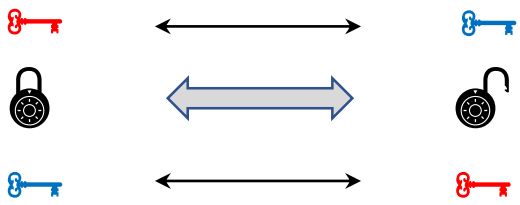


① 每一个公钥都对应一个私钥。

② 大家都知道的是公钥，只有自己知道的是私钥。

③ 用一个密钥加密数据，只有对应的密钥才可以解密。

④ 如果一个密钥可以解密数据，则该数据必然被对应的密钥加密。



公钥加密 主要应用 公钥认证

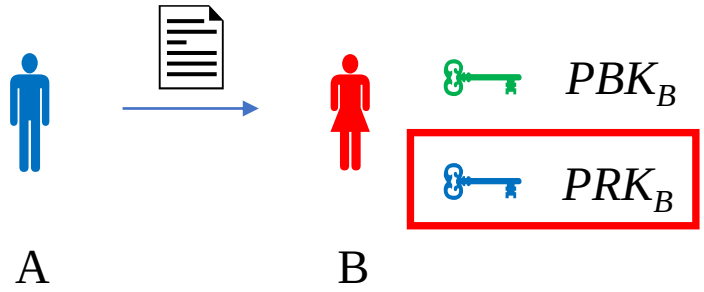
非对称加密与对称加密

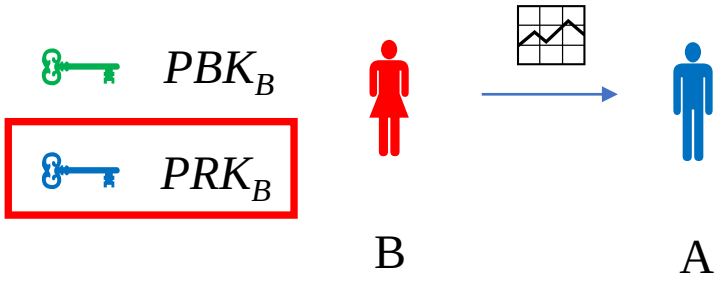
- 非对称加密算法强度复杂、安全性依赖于算法与密钥；
- 但是由于其算法复杂，而使得加密解密速度没有对称加密解密的速度快。
- 对称密码体制中只有一种密钥，并且是非公开的，如果要解密就得让对方知道密钥。所以保证其安全性就是保证密钥的安全
- 而非对称密钥体制有两种密钥，其中一个是公开的，这样就可以不需要像对称密码那样传输对方的密钥了。这样安全性就大了很多。

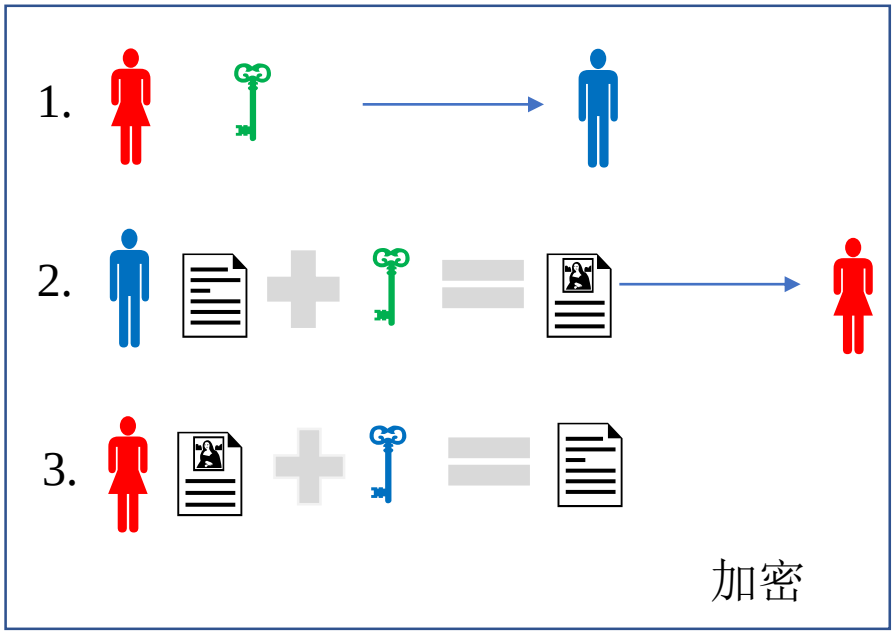
非对称加密体系不要求通信双方事先传递密钥或有任何约定就能完成保密通信，并且密钥管理方便，可实现防止假冒和抵赖，因此，更适合网络通信中的保密通信要求。

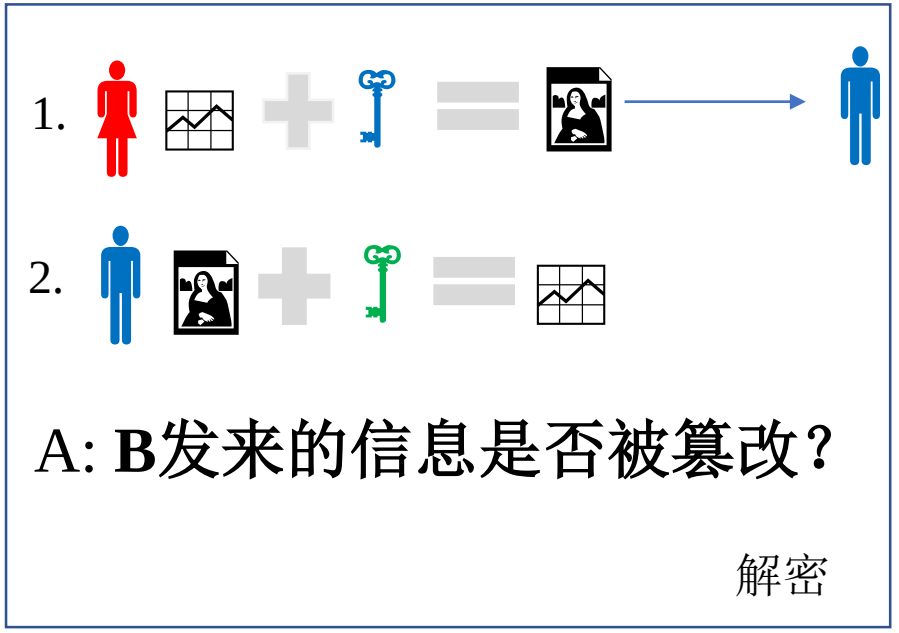
公钥加密解密

加密的目的？？ 不希望第三者看到当前两个通讯用户的通讯内容。



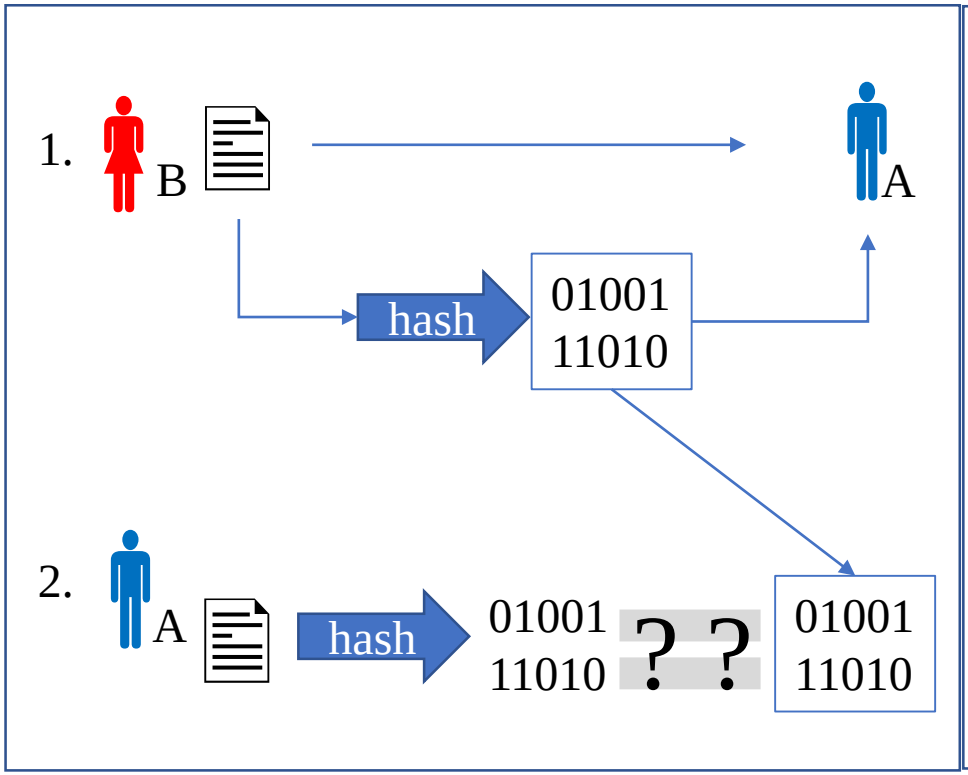


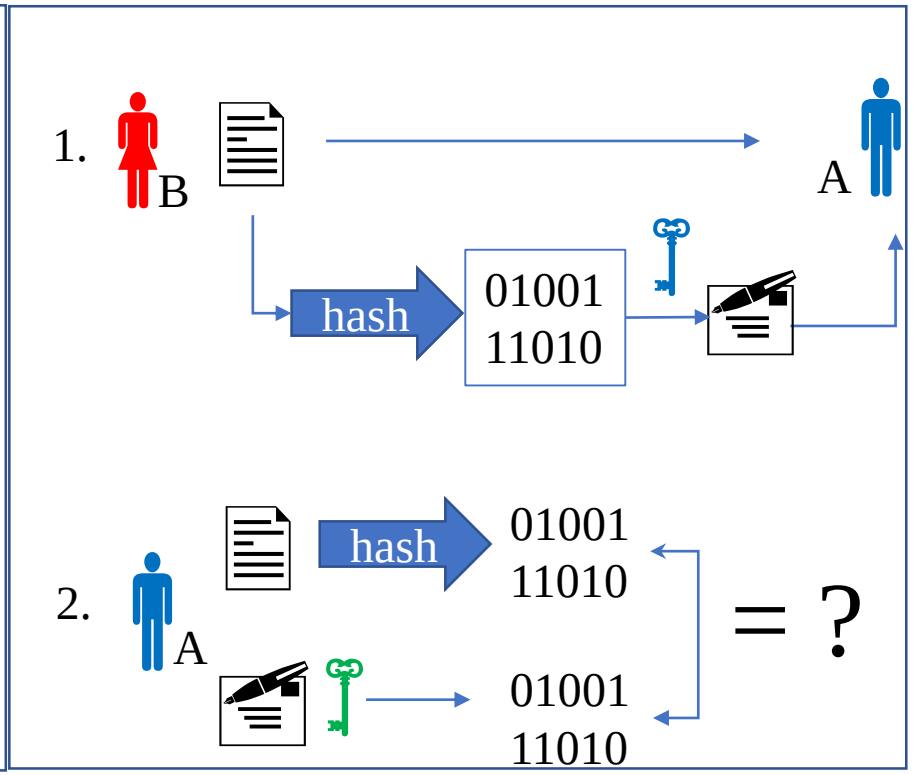




公钥认证

数字签名：是指可以添加到文件的电子安全标记。使用它可以验证文件的发行者以及帮助验证文件自被数字签名后是否发生更改。





同时篡改信息内容和hash值？？

B的信息未被修改过！！

概 要

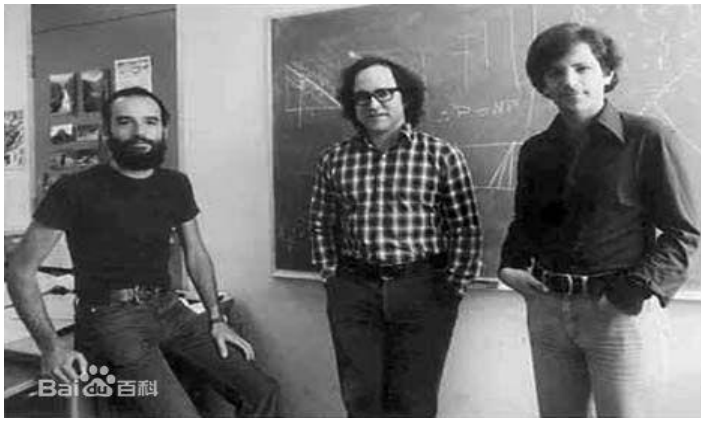
非对称加密算法

- 加密算法
- 非对称加密算法
- RSA加密算法
- 其他非对称加密算法

RSA公钥算法

RSA加密算法：是一种非对称加密算法。RSA是1977年由罗纳德·李维斯特（Ron Rivest）、阿迪·萨莫尔（Adi Shamir）和伦纳德·阿德曼（Leonard Adleman）一起提出的。2017年被普遍认为是最优秀的公钥方案之一

对极大整数做因数分解的难度决定了RSA算法的可靠性



- **RSA密钥对的生成：**

1) 选取两个大素数p，q；

2) 计算模数$n=pq$；

3) 随机选取加密密钥e，使得e与$(p-1)(q-1)$ 互素；

4) 计算d，$d*e \bmod (p-1)(q-1) = 1$；

5) 公开密钥为：（e，n），私人密钥为：d。

RSA公钥算法

RSA密钥对的生成：

1) 选取两个大素数p，q；　（p =17, q = 19）

2) 计算模数n＝pq；　　(n=323)

3) 随机选取加密密钥e，使得e与(p-1)(q-1) 互素；

lcm(p-1)(q-1) =144, 5,7,11,13,17,19,23,25,29,31...,　e = 5

4) 计算d，d*e mod (p-1)(q-1) =1；　d = 29

5) 公开密钥为：（e，n），私人密钥为：d。（e，n）= （5, 323）

RSA加密解密，签名验证

公开密钥为:(e，n) = (5, 323); 私人密钥为: d = 29; 明文$m < n$

- 加密：$c = m^e \bmod n$　　（m:明文，c:密文）
- 解密：$m = c^d \bmod n$　　（m:明文，c:密文）

m = 123, $c = 123^5 \bmod 323 = 225$;

d = 29, $m = 225^{29} \bmod 323 = 123$;

- 签名：$s = m^d \bmod n$　（m：明文，c：密文）
- 验证：$m = s^e \bmod n$　（m：明文，c：密文）

RSA算法性质

◆由前可知，RSA的实现实际上就是大数的模指数运算，为了安全，模数n的长度一般大于等于1024比特，而且，要求生成n的大素数p和q是等长的。

◆由于大数模幂运算要用到大量的乘法和除法，乘法和除法是十分耗时的运算，比起密钥长度仅为56位（不带奇偶校验位），而且运算主要是由置换、异或、移位组成的DES来说，RSA的运算速度实在是慢得很。与DES相比，硬件实现时，RSA比DES慢1000倍；软件实现时，RSA比DES慢100倍。

概 要

非对称加密算法

- 加密算法
- 非对称加密算法
- RSA加密算法
- 其他非对称加密算法

椭圆加密算法

公钥密码体制根据其所依据的难题一般分为三类：大素数分解问题类、离散对数问题类、椭圆曲线类。有时也把椭圆曲线类归为离散对数类

Koblitz

- ECC也叫椭圆加密算法，由Koblitz和Miller两人于1985年提出。
- ECC加密算法是一种公钥加密技术，以椭圆曲线理论为基础。
- 利用有限域上椭圆曲线的点构成的Abel群离散对数难解性，实现加密、解密和数字签名。

椭圆加密算法

- ECC的主要优势是在某些情况下它比其他的方法使用更小的密钥——比如RSA加密算法——提供相当的或更高等级的安全。
- 与传统的基于大质数因子分解困难性的加密方法不同，ECC通过椭圆曲线方程式的性质产生密钥。
- ECC 164位的密钥产生的一个安全级相当于RSA 1024位密钥提供的保密强度，而且计算量较小，处理速度更快，存储空间和传输带宽占用较少。
- 目前我国 **居民二代身份证** 正在使用 256 位的椭圆曲线密码，虚拟 货币 **比特币** 也选择ECC作为加密算法。

椭圆加密算法

考虑$K=kG$，其中K、G为椭圆曲线$E_p(a,b)$上的点，n为G的阶（$nG=O\infty$），k为小于n的整数。则给定k和G，根据加法法则，计算K很容易但反过来，给定K和G，求k就非常困难。因为实际使用中的ECC原则上把p取得相当大，n也相当大，要把n个解点逐一算出来列成上表是不可能的。这就是椭圆曲线加密算法的数学依据。

点G称为基点（base point）

k（$k < n$）为私有密钥（private key）

K为公开密钥（public key)

ECC保密通信

1.Alice选定一条椭圆曲线E，并取椭圆曲线上一点作为基点G
假设选定$E_{29}(4,20)$，基点$G(13,23)$, 基点G的阶数n=37；

2. Alice选择一个私有密钥k（$k < n$），并生成公开密钥$K = kG$
比如25, $K = kG = 25G = (14,6)$）；

3. Alice将E和点K、G传给Bob

4. Bob收到信息后，将待传输的明文编码到一点M，并产生一个随机整数r（$r < n$, n为G的阶数） 假设r=6 要加密的信息为3, 因为M也要在$E_{29}(4,20)$ 所以M=(3,28)

ECC保密通信

5. Bob计算点$C_1=M+rK$和$C_2=rG$, $C_1= M+6K= M+6*25*G=M+2G=(3,28)+(27,27)=(6,12)$, $C_2=6G=(5,7)$

6. Bob将C_1, C_2传给Alice

7. Alice收到信息后，计算C_1-kC_2，结果就应该是点M，C_1-kC_2 =(6,12)-25C_2 =(6,12)-25*6G =(6,12)-2G =(6,12)-(27,27) =(6,12)+(27,2) =(3,28)

ECC vs. RSA

优点

- 安全性能更高
- 160位ECC与1024位RSA、DSA有相同的安全强度
- 处理速度更快
- 在私钥的处理速度上，ECC远 比RSA、DSA快得多
- 带宽要求更低
- 存储空间更小
- ECC的密钥尺寸和系统参数与RSA、DSA相比要小得多

缺点

- 设计困难，实现复杂
- 如果序列号设计过短，那么安全性并没有想象中的完善

概 要

非对称加密算法

- 加密算法
- 非对称加密算法
- RSA加密算法
- 其他非对称加密算法

Diffie-Hellman密钥交换算法

- 一种确保共享KEY安全穿越不安全网络的方法
- 这个机制的巧妙在于需要安全通信的双方可以用这个方法确定对称密钥
- 是Ralph Merkle最初设计并以Whitfield Diffie和Martin Hellman命名的第一个公钥协议之一。
- DH是在密码领域实现的公钥交换最早的实际例子之一
- Diffie-Hellman密钥交换算法的有效性依赖于计算离散对数的难度

DH密钥交换算法

离散对数：定义素数p的原始根是生成1—(p-1)之间所有数的一个数，设a为p的原始根，则：a mod p， a^2 mod p,……，a^{p-1} mod p 是各不相同的整数，且以某种排列方式组成了1到p-1的所有整数。对于任意数b及素数p的原始根a，可以找到一个唯一的指数i，满足：$b=a^i$ mod p, 其中 $0 \leq i \leq p-1$，那么指数i称为b的以a为基数的模p的离散对数。

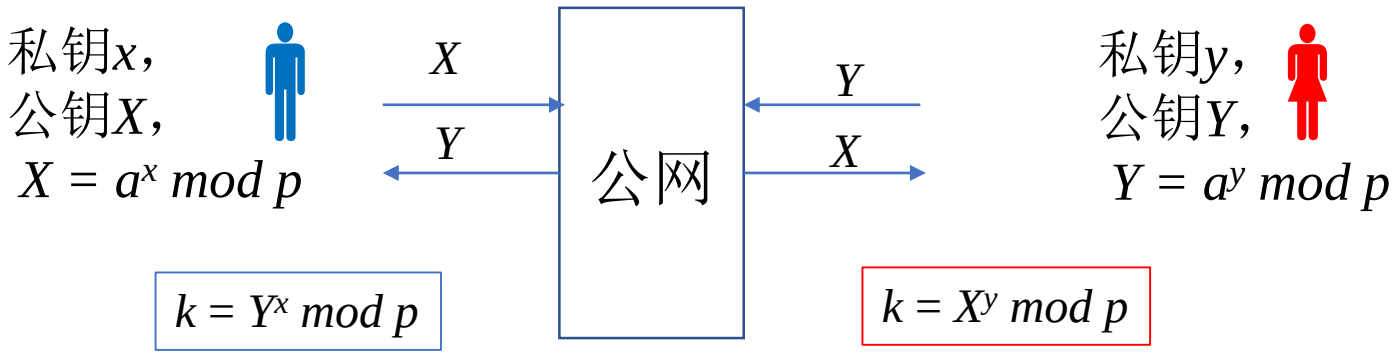
Diffie-Hellman算法的有效性依赖于计算离散对数的难度，其含义是：当已知大素数p和它的一个原根a后，对于给定的b，要计算出i 被认为是很困难的，而给定i 计算b却相对容易。

DH密钥交换算法

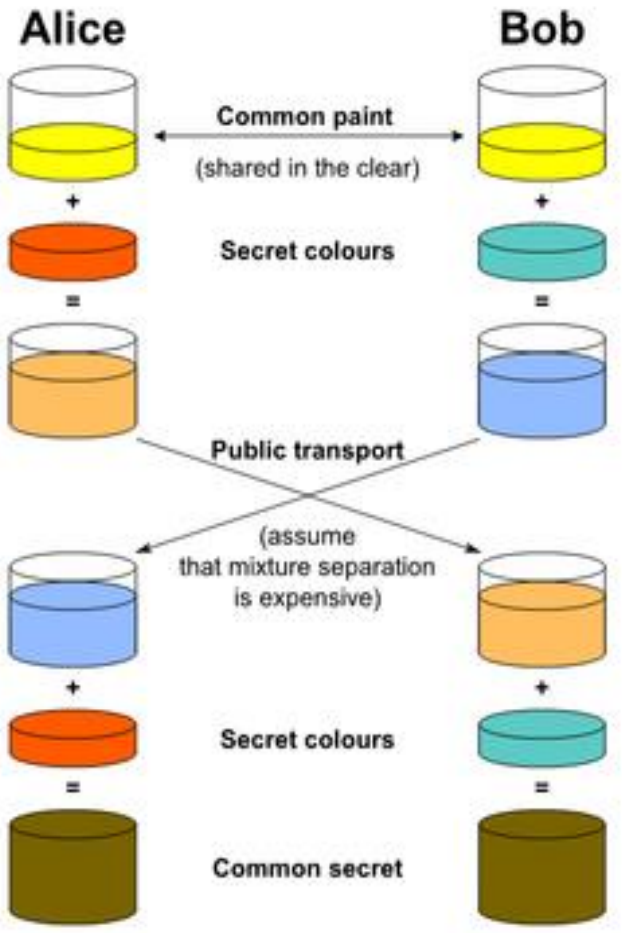
假设网络上有两个用户A和B，彼此之间协商共同的密码，假设交换密钥的值为k

- A和B事先约好大素数p和它的原始根a；
- A随机产生一个数x，计算$X = a^x \bmod p$，然后把X发给B;
- B随机产生一个数y，计算$Y = a^y \bmod p$，然后把Y发给A;
- A计算$k = Y^x \bmod p$;
- B计算$k' = X^y \bmod p$;

因为：$k = Y^x \bmod p = (a^y)^x \bmod p = (a^x)^y \bmod p = X^y \bmod p = k'$



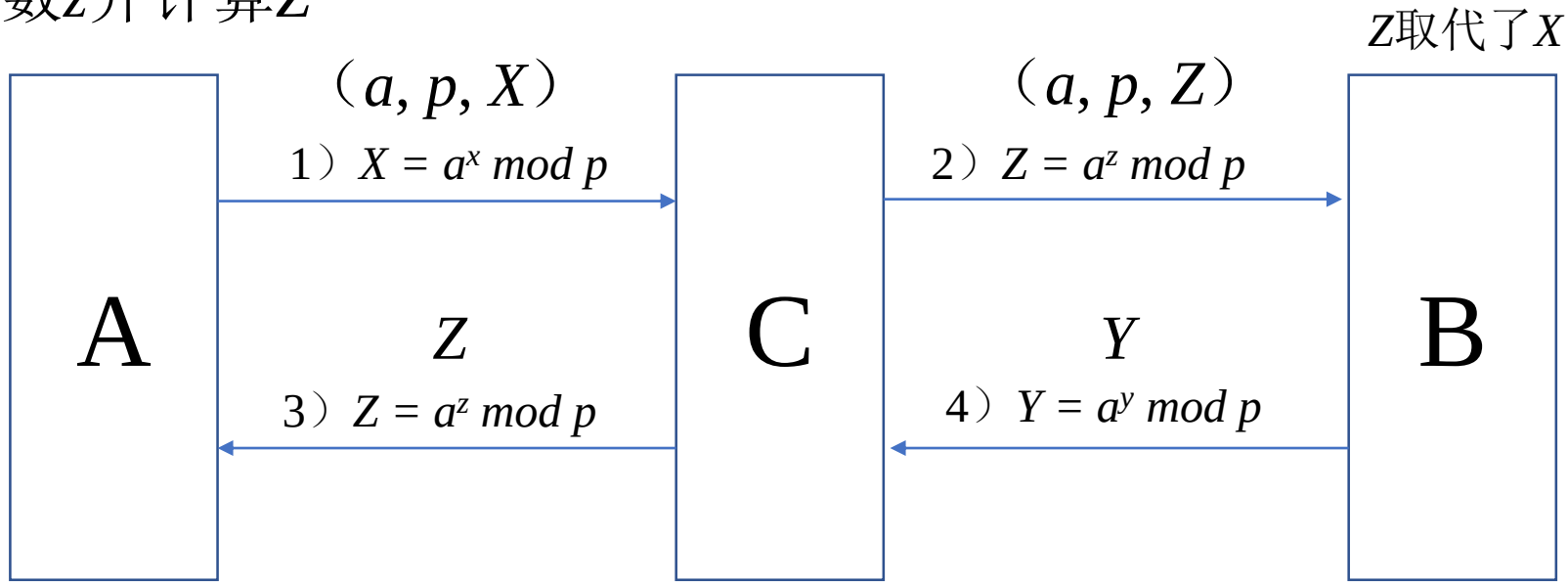
DH的优点



不安全网络上的窃听者只能得到a、p、X、Y，除非能计算离散对数x和y，否则将无法得到密钥k，但对于大素数p，计算离散对数是十分困难的，因此k为用户A和B独立计算出的密钥。

DH的缺点

尽管Diffie-Hellman算法十分巧妙，但它也存在一个问题：当B得到一个三元组时，他怎么知道这是来自于A而不是网络攻击者C呢?他无法知道。不幸的是，C可以利用这一点来欺骗A和B，如图所示。当A和B分别选择X和Y时，C也选择了自己的随机数z并计算Z



$k = a^{xz} \bmod p$

即使事后A和B知道有人攻击，但也无法确定是谁，因为网络上的任何人都可以发起中间人攻击。

$k = a^{yz} \bmod p$

DH分析

该方法增加了密钥的安全性，因为从头到尾密钥没有出现在网络上。但是问题是，如果a、p、X、Y取值较大也增加了彼此双方计算余数的难度，mod运算结果将会出现误差，因此需要专门编程进行大整数取余运算来实现或用专门的逻辑计算电路来实现。

同公开密钥加密算法相比，对于密钥的管理相对复杂一些，因为需要进行相关计算。而公开密钥加密算法直接传递的就是加密密钥，直接解密就可以了，但是一旦私有密钥丢失也面临巨大的泄密风险。因此为确保数据安全，应加强私有密钥的管理，防止私有密钥泄露造成不必要的损失。

Diffie-Hellman-Merkle密钥交换

2002年，Hellman提出这个算法被称为**Diffie-Hellman-Merkle密钥交换，**以认识Ralph Merkle对公钥密码学发明的贡献（Hellman，2002），他写道：

这个系统已经被称为Diffie-Hellman密钥交换。这个系统最初是由迪菲和我在一篇论文中描述的，它是一个公开密钥分发系统，是由Merkle开发的一个概念，因此如果名字与它关联，应该被称为“Diffie-Hellman-Merkle密钥交换”。我希望这个小小的讲坛可以帮助我们认识到默克尔对公钥密码学发明的平等贡献。

概 要

非对称加密算法

- 加密算法
- 非对称加密算法
- RSA加密算法
- 其他非对称加密算法

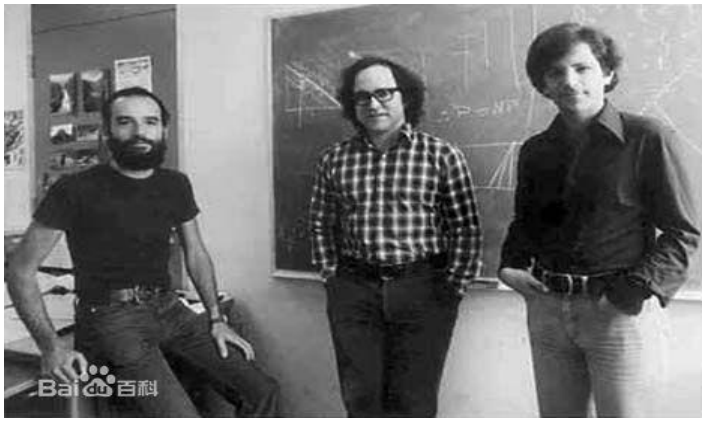
35年的预言

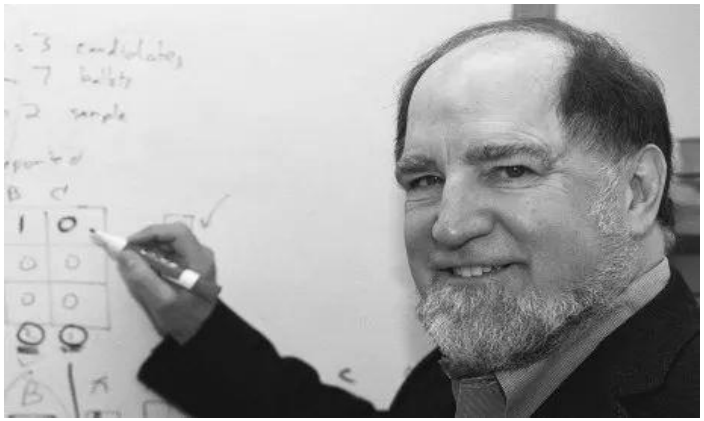
1. 1994 年 4 月，麻省理工学院计算机科学实验室成立 35 周年的庆祝活动。

2.时任实验室主任 Michael Dertouzos 设计了一个“创新成果时间胶囊“，他将一系列计算机领军人物的创新成果收录其中，准备在35年后再取出来，作为实验室成立 70 周年的献礼工程

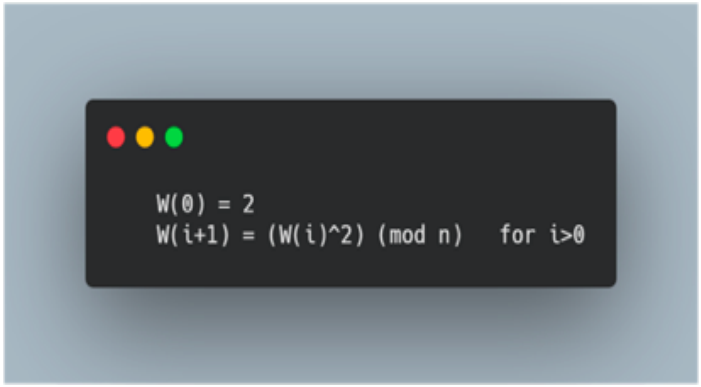
35年的预言

这个时间胶囊可以说是一个早期计算机历史的博物馆，它里面包含由微软创始人比尔·盖茨和图灵奖得主、万维网之父Tim Berners-Lee爵士等计算机领军人物捐献的 50 件计算机历史上伟大的藏品





3. 他们为时间胶囊设计了一把“密码锁“，也就是一道密码学难题。使得密码学难题至少需要 35 年时间来破解，这个难题就是要找到运行近 80 万亿次平方操作的结果



3年半的真实

4. Fabrot是在 2015 年才偶然发现了这个密码学难题,所以 Fabrot 立即着手去做，他在家里的台式计算机上专门分出一个 CPU 内核用于运行平方计算，在此期间除了他去度假或是家里停电，Fabrot 的电脑一直在全天候运行.

三年半之后， Fabrot 终于完成了大约 80 万亿的平方计算，得到了密码学难题的结果.

Cryptophage （直译为：加密噬菌体）的项目，该项目主攻的目标是硬件，目标是使用专门的硬件来解决麻省理工学院提出的密码学难题。

Fabrot

我相信自己可以做到，同时我也知道如果我告诉其他人，他们可能会使用更强大的 **CPU** 来超越我。

作业

1. 请以实际数字为例，找出RSA密钥对；
2. 请给出RSA算法中，加密运算和解密运算过程
3. 浅谈自己是如何面临挑战的。