非对称加密

- 对称加密：对称加密的意思是加密和解密使用同一套密码
- 非对称加密：非对称加密的加密密钥和解密密钥是不同的，涉及两个密钥，分别称为公开密钥（public key）和私有密钥（private key）。公开密钥与私有密钥是成对出现的，如果公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。
- 一个很重要的函数：**欧拉函数**，很多加密算法都是基于这个函数的，用来求一个数的原根个数的

  欧拉函数描述的问题是，任意给定正整数n，在小于等于n的正整数之中，有多少个与n构成互质关系？这个函数一般表示为ϕ(n)。

  正整数m的原根的个数为 $\varphi(\varphi(m))$ 。
- 另一个一个概念：**原根**，如果*a*是素数*p*的一个原根，那么数值：

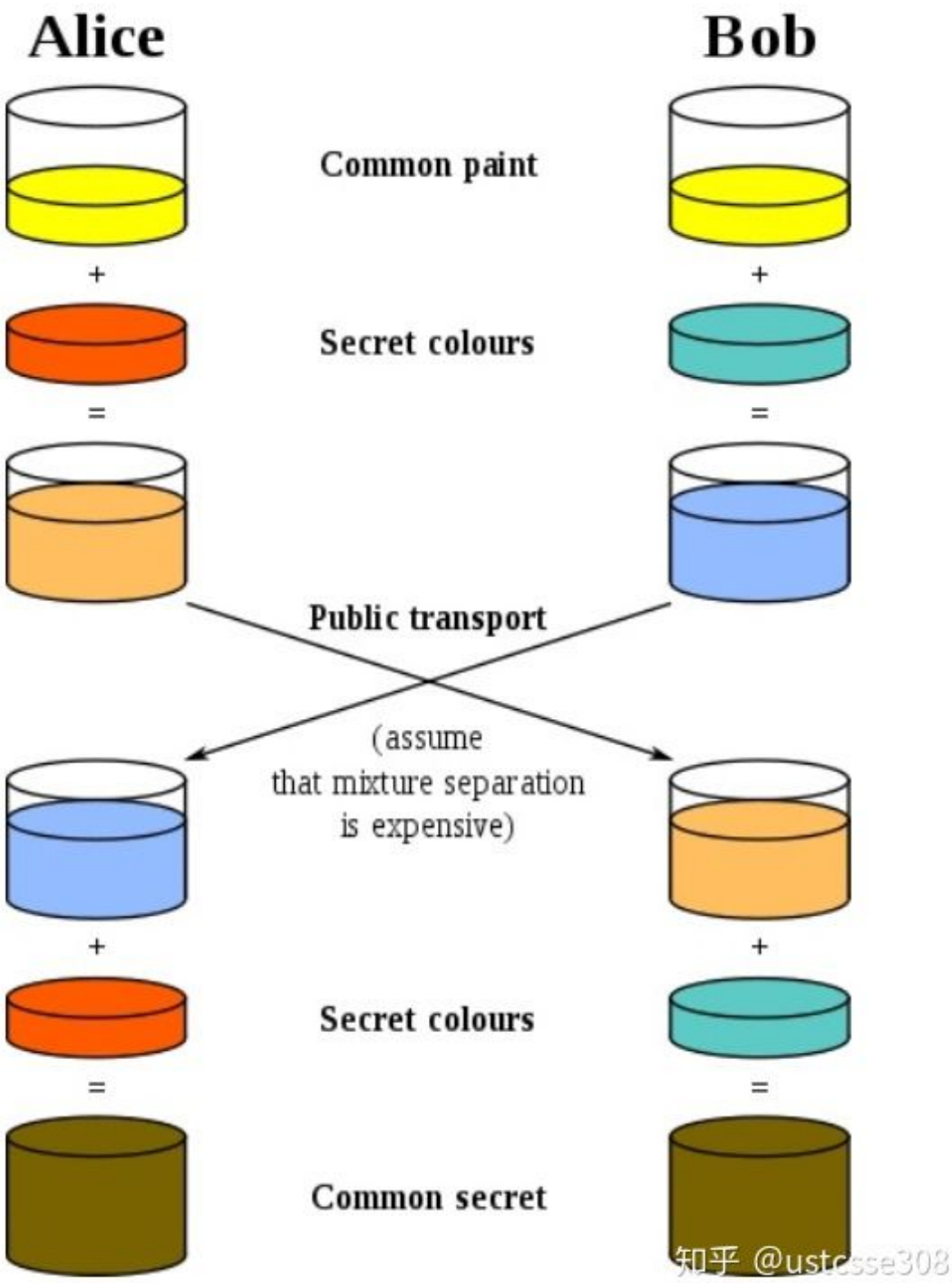
  *a*mod*p*，*a^2* mod*p*，...，*a^(p-1)* mod*p*

  是各不相同的整数，且以某种排列方式组成了从*1*到*p-1*的所有整数。

  例如下面的例子：3是7的原根

$$
\begin{aligned}
3^1 &= 3 = 3^0 \times 3 \equiv 1 \times 3 = 3 \equiv 3 \pmod 7 \\
3^2 &= 9 = 3^1 \times 3 \equiv 3 \times 3 = 9 \equiv 2 \pmod 7 \\
3^3 &= 27 = 3^2 \times 3 \equiv 2 \times 3 = 6 \equiv 6 \pmod 7 \\
3^4 &= 81 = 3^3 \times 3 \equiv 6 \times 3 = 18 \equiv 4 \pmod 7 \\
3^5 &= 243 = 3^4 \times 3 \equiv 4 \times 3 = 12 \equiv 5 \pmod 7 \\
3^6 &= 729 = 3^5 \times 3 \equiv 5 \times 3 = 15 \equiv 1 \pmod 7 \\
3^7 &= 2187 = 3^6 \times 3 \equiv 1 \times 3 = 3 \equiv 3 \pmod 7
\end{aligned}
$$

- 非对称加密的起源Diffie-Hellman密匙交换算法



DH协商过程，Alice和Bob首先挑选一个颜色（黄色），这个颜色是可以公开的（每次通信不同）；然后再各自挑选一个秘密的颜色（Alice橙色，Bob青色）。然后Alice和Bob各自将自己的秘密颜色和黄色进行混合，得到了另外的两个颜色（Alice橙褐色，Bob淡蓝色）。Alice和Bob分别将自己的颜色发给对方。Alice和Bob在收到对方发来的颜色后，再分别和自己的颜色相混合，此时，两人得到了一个相同的颜色（黄褐色）。

在这个过程中，攻击者Eve可以一直监听网络，并且获得Alice和Bob在网络上交换的所有信息。也即，可以获得黄色、橙褐色、淡蓝色这些信息。阻止Eve获得最终的黄褐色的是Alice和Bob分别挑选的秘密颜色，橙色和青色。也即，需要能够证明，即使Eve得到了黄色和橙褐色，Eve也不能推导出Alice的秘密颜色。

在数学上，DH算法的有效性依赖于计算离散对数的难度。也即，当已知大素数 和它的一个原根（primitive root） (**这里使用原根的目的是保证后面`g^amod(p)`的时候结果分布空间足够大，在下面有详细说明)，对于给定的 ，要计算指数 ，是非常困难的（暴力破解），而给定 ，计算 却很容易。

再用一个具体的例子来解释DH。

1. Alice和Bob通过交流，决定选择素数 以及原根 。
2. Alice选择了一个秘密整数 ，Bob选择了秘密整数 。
3. Alice和bob分别使用 和 计算出A和B，其中

4. Alice和Bob分别将这两个数字A=4和B=10通过网络发送给对方。
5. Alice和Bob收到B和A之后，分别计算：

6. 现在Alice和Bob拥有了一个共同的密钥18。而且这个密钥从来没有在网络上传输过。

为什么Alice和Bob可以获得共同的密钥呢?

那现在看一下，DH算法可以运行的关键是什么?

即使攻击者Eve可以获得23、5、A和B，她仍然不能得到Alice和Bob的秘密数字4和3。也即，即使知道 这个计算过程中的底数5，模数23和结果10，她依然不能得到指数3。这个就是DH算法所依赖的计算离散对数的难度。（证明计算离散对数很难超纲）

当然我们这里所举的例子非常简单，在实际使用中，必须使用很大的 。如果 长度为300位， 和长度为100位，那基本就安全了

使用原根的原因：

首先需要明确，对于下面这个公式而言，任何情况下都是成立的：

那么为什么要使用原根而不是使用和p互质的其他数呢?

这里主要涉及的问题是，破解的难度。

如果使用的 不是 的原根，那么 的所有指数只能生成 小于 的整数的一个子集。那么对于攻击者而言，此时即使是暴力破解，需要计算的也只是小于 的整数的子集，而不是小于 的整数全部。

譬如，对于素数13而言，3不是它的原根，(3^1=3, 3^2=9, 3^3=1)mod（13），所以生成的模数只有3个（order），等效的指数也只有3个，降低了攻击者暴力破解的难度。

例如，在DH系统中，如果选择 =3，Alice选择了 ，那么对于攻击者Eve而言，本来她需要尝试到7（遍历小于13的全部数字）才能得到 ；但是实际上，Eve只需要知道 就足够了。因为最终的

所以大大降低了DH算法的破解难度。

- DH算法的弊端

DH算法并没有对双方身份进行验证。当Alice和Bob希望进行通信时，Eve可以很容易地向Alice冒充自己是Bob，以及向Bob冒充自己是Alice，然后分别和Alice和Bob建立公共的对称密钥。然后，Alice到Bob的通信都会通过Eve先使用自己与Alice建立的密钥先解密，获得明文信息之后，再用Eve与Bob建立的密钥加密，传给Bob。Bob到Alice的通信亦然。这样，Alice和Bob会以为自己和对方的通信是加密的，从

而是安全的，但是它们的通信会经过Eve加解密一遍。Eve在Alice和Bob之间，拦截他们的通信，并且维持通信，就称为中间人攻击。