密码算法入门指南

刘佳 编写

Email： 281033201@qq.com

2018.9.25

前言

密钥加解密现在是很常见的一种工具，我们可以很轻易的使用 RSA 或者 DES 来对目标对象进行加解密。最近听说天朝在推加密方式国产化，这篇文章的目的就是写给想在这个浪潮中分一杯羹的公司或者对加解密感兴趣的个人学习之用，希望读完此文后，读者也能做一个国产化的加解密算法。由于本人时间和能力有限，难免此文有不足与错误之处，烦请指正，谢谢。

正文

基本加解密手段

说到加解密，都会提到凯撒，因为似乎在西方世界，他是加解密的开山鼻祖。其实我们中国很早就有了加解密的手段，文人墨客中流行的藏头诗就算是一种，还有武侠小说中将武林秘籍分成上下部分开保管也算是加密的一种手段。加解密的宗旨就是让别人在不满足条件的情况下，无法获取信息。开宗明义，传统的加解密手段就是映射。对于字母而言，就是生产一个一一对应的映射规则，把有意义的单词转成无意思的单词，比如加密函数：

$$Y=(x+2)\%26$$

它把所有的英文字母往后移了两位。“Hello world”就变成了“Jgnnq yqtnf”。我想正常人应该是没有办法一眼看出来的吧。解密函数其实就是加密函数的逆运算：

$$x=(Y-2)\%26$$

中文的话似乎无法使用这个规则，但是我们知道计算机里保存的都是数字，所以还是可以利用上述函数来进行加解密的。

但是这样的规则比较容易破解，通用的破解方式就是利用统计规律，英文中 e 出现的次数最多（我记得福尔摩斯探案集里就用过这个方法），一篇很长的文章，统计一下密文字母的出现频率即可找到对应关系。

为了使统计方法失效，一般我们会用一种叫做密钥的东西来对数据进行加密，使用它的目的就是为了对每一个字母都进行单独的上述运算。我们可以理解成对每个字母都独立的加密，只有一个样本，无法得到统计特性，也就无法进行加密了。这种加密方法的数学形式可以写成：

$$Y=(X.+K).\%26$$

其中 X 是待加密的对象，是一个矩阵，K 是密钥，也是一个矩阵，由于矩阵中的每个元素都要单独加解密，使用了 matlab 中的元素操作符“.”来显示地表示。解密的表达式就不再累述，逆运算即可。

上述的做法对密钥的要求比较高，它要求密钥是随机生成的，举个极端的例子，还是对“Hello world”进行加密，如果我的 K 是“1111122222”，那么在统计 Y 的时候就会显露出某种统计特性，从而被企图破解的人利用。

知道了原理，我们可以试着做一个自己的密码算法，步骤如下：

1：找到一种映射关系 y=f(x,k)

2：确认 $x=f^{-1}(y)$是唯一的，如果不能，从新开始

另外要注意，k 要尽量随机。

我随便想了一个供参考

$Y1=(x_1+k_1)\%26$

$Y_n=Y_{n-1}+(x_n+k_n)\%26$ （n>1）

这个算法在之前介绍的算法上做了一个拓展，加入了一些关联信息，它的好处是可以防止别人破译非完整的信息，必须得到全部 Y，才可能破译。

交换密钥

我们设计了算法，就要告诉别人算法是怎样工作的，否则别人无法使用。保护加密不被破解的唯一手段，只剩下了保护密钥 K。如果是自己加密自己解密的话问题还不大。如果是 A 要加密后发给 B 看，B 需要解密，怎么保证 A 在给 B 密钥 K 的过程中不被别人盗取呢？答案是不给就行了。如果是不给，但是又要双方都知道，似乎是个难题，不过如果换一个思路，当“约定”无法实现，是否可以“协商”呢？因为只需要产生一个密钥，而密钥本身是没有限制的，什么都可以，只要你知我知，天知地知就行了，于是一个很自然的想法是根据双方的公开信息和自己的私密信息进行协商，产生一个共同的结果就行了。也就是说要找到一种数学形式，满足：

$$k=f2(f1(x1,s),x2)$$

其中 x1 和 x2 是 A 和 B 的私密信息，s 是公开信息，任何人可以知道，另外 x1，x2，s 都是向量，不局限于单个数字。流程如下：

1：A 和 B 都默默地思考出一个各自的 x，并铭记于心，无人知晓。

2：A 和 B 在大庭广众下讨论出一个 s，众人皆知。

3：A 把自己的 x1 和 s 做了一个 f1 运算，得到 t1。B 把自己的 x2 和 s 做了一个 f1 运算，得到 t2。双方在大庭广众之下交换了各自的 t1 和 t2。

4：A 拿 t2 和 x1 做了一个 f2 运算得到 k，B 拿 t1 和 x2 做了一个 f2 运算得到 k。

这时，双方都知道了 k，这样密钥协商完毕。整个问题的关键是找到运算 f1，使得第三者 C 在仅知道 s 和 t1，t2 的情况下无法知道 x1 或者 x2 是什么。即

t=f1(x,s)

是单向函数。y=power(x,2)就是一个单向函数，因为知道 y 后，无法确切知道 x 是多少，因为正负各占 50%。如果用这个函数来加密，我们需要尝试 2 次才能保证解密。一般来说 f1 和 f2 是一样的，因为这才能方便保证 A 和 B 计算

f2(f1(x1,s),x2)

后得到相同的 k。f 其实很容易找，上面说的平方函数就是一例。但是缺点也很明显，就是太容易破解了。至多尝试 2 次就能破解。取余是密码学中常用的运算，因为它是比较彻底的单向函数，比如 x%3=1，知道这个条件，x 的结果是无限的。因此我们在交换密钥时，一般使用

f=g^x mod p

来进行计算，其中 g 和 p 就是公开的 s 向量，特别的，当 p 很大时，

破解是相当困难的。我按前面所述的流程，用一个具体的例子来说明：

1：A 和 B 各自心中默念一个数字，A 是 x1=15，B 是 x2=13

2：A 和 B 协商了公开的 g=3 和 p=17

3：A 运算了 f，得到 f1=6，B 算了 f，得到 f2=12，并告诉了彼此运算结果

4：A 拿着 f2，和自己的 x1 做 f 运算，得到 f=（12^15）%17=10。B 拿着 f1，和自己的 x1 做 f 运算，得到 f=（15^13）%17=10。

完整的对称密钥算法

有了这个方法，密钥交换是没问题了，但是根据基础算法，我们需要每个明文都对应一个密钥，这样实际传输的数据是原来的两倍，这还不包含协商密钥时中间信息的交换。为了解决这个问题，有人想到是否可以只传输一次密钥，然后用这个密钥来生成后续的其它密钥呢？这个首先当然是可行的，函数

y=(x+1)%26

就能够实现，但是我之前说了，密钥为了不让人破解，必须要是随机的。显然上述的函数 y 很不随机。幸运的是，有人提供了一些伪随机数据的生成算法，这些方法，虽然不能生成完美的随机数据，但是可以满足需求，它们可以用很久才会出现模式重复。不过话又说回来，如果真的是随机了，仅使用一个密钥，想双方生成一致的数据序列也是不可能的了。具体伪随机算法不在本指南的讨论范围内，不过大多数计算机语言都提供了伪随机函数生成器，只需要将密钥作为种子传入即可。

新的问题

解决了上述的一切技术难题，对称密钥算法在很多时候是相当好用的。但是随着互联网的普及，数据传输只使用上述方法将会变得非常痛苦，想象一下，一个公司有 m 个人，如果这 m 个人之间要相互加密通信，那么这个公司总共需要保存 m*(m-1)个 x，这个成本是相当高的。

非对称密钥加密

为了解决这个问题，有个聪明人想到，为什么加解密要用相同的密钥呢，如果不一样不就行了。别人用 k1 进行加密，自己用 k2 进行解密，这样子，m 个人相互通信，只需要公开各自的 k1，自己保存一份 k2，那么 m 个人只需要保存 m 个密钥就行了。用数学描述就是

$$m=f2(f1(m,k1),k2)$$

说的再直白一些，就是找到两个不同的函数，一个用来加密，一个用来解密，他们的某种运算是一个常数。就像把一把打开的锁交给别人，别人锁上箱子后交还，由于钥匙只在自己手上，除了自己没人能够打开箱子。如果不考虑破解，满足这样条件的函数还是很普遍的。比如

f1=e^((m+k1)%26)

f2=(log(f1)+k2)%26

显然这个函数相当容易破解的，因为 k2=-k1。

如果读者能自己找到一套 f1 和 f2，并且仅知道 f1 和 k1，k2 很难计算的话，那么一个新的非对称加密算法就产生了。不过真的很难，我这里介绍一下 rsa 用到的方法，希望能抛砖引玉。

考虑之前的对称加密方法中用到的传递密钥的函数

f=c^d%n

目前数学上，只知道 f，d 和 n，要求得 c 是很难的。于是我们想到是不是可以找到函数满足

m= m^e^d%N （m^e^d%N= (m^e%N)^d%N）

其中 m 是消息，e 和 d 分别对应密钥 k1 和 k2，N 也是算法公开的一个随机数。现在的问题就是如何求 d。为了解决这个问题，我需要引入一些数学概念。

1：任何一个数都能分解为唯一的一组质数的乘积。比如 15=3*5，18=2*3*3 等待。要知道随便找两个质数相乘很简单，但是目前的数学要对一个数进行因数分解却相当困难，需要不断的试错。

2：欧拉的 phi(x)函数，作用是计算小于 x 的，且不和它有共享因数的数的个数。比如 phi(8)=4，因为 1 到 7 中，1、3、5、7 不和 8 共享因数，2、4、6 共享质数 2。要计算 phi 是相当困难的，原因上面刚说过。但是根据定义，phi(质数 x)却很好计算，它=x-1。因为质数不和任何数共享因数。

3：phi(A*B)=phi(A)*phi(B)，利用这个性质，如果 N=p1*p2，其中 p1 和 p2 是质数，那么 phi(n)=phi(p1)*phi(p2)=(p1-1)*(p2-1)

4：n 和 m 之间没有公因数，那么，(m^phi(n))%n=1%n=1，m^(k*phi(n))%n=1^k%n=1

因此

m*m^(k*phi(n))%n= (m^(k*phi(n)+1))%n=m %n

有

m^e*d%n=m%n

如果我们取 k*phi(n)+1=e*d，那么 d=(k*phi(n)+1)/e（k 的作用是保证 d 是一个整数，而不是除不尽），根据前面说的，只有当 n 是质数时，才容易求。模拟一下步骤如下：

1，A 随机选择两个质数 a,b，计算得到 n=a*b。得到 phi(n)。

2，选一个 e，e 必须不能和 phi（n）有共享因数的奇数。（随机再选个奇数的质数就行了），计算出 d。

3：分享 n 和 e。

4：B 用 m^e%n=t，对 m 加密后把 t 发送给 A

5：A 用 t^d%n=m，得到原信息 m

这种公开密钥算法的思想破解的关键是能求到 phi（n），但是知道 N 是由质数产生的，如果位数足够长，用事先计算好的查表法也要很久。在实际使用中，算法需要计算 m^n%t，在 m 或者 n 很大时，计算机一定会溢出，因此计算时需要使用额外的算法来实现。现在不论多少人通信，因为公钥 k1 是公开的，每个人一把钥匙就可以了。可是怎么知道加密的信息来自 A 而不是 B 呢，这又引申出另外一个话题，数字签名，不过不再这里讨论了，有兴趣的读者可以后续自己扩展学习。