一种新的数据加密技术

[作者] 潘晓中，孙军，杨晓元，王法能
[单位] 武警工程学院电子技术系
[摘要] 该文简要介绍了数据加密的一般方法及基于公钥加密算法的方法与步骤，较为详细介绍了多步加密算法的原理与算法。
[关键词] 加密算法，密钥，多步加密

我们处在一个信息时代，人们需要一种强有力的安全措施来保护机密数据不被他人窃取或篡改。数据加密与解密从宏观上讲是非常简单的。加密与解密的一些方法是非常直接的，很容易掌握。因此，可以很方便地对机密数据进行加密和解密。

1、数据加密方法

在传统上，我们有几种方法来加密数据流。所有这些方法都可以用软件很容易的实现，但是当我们只知道密文的时候，是不容易破译这些加密算法的（当同时有原文和密文时，破译加密算法虽然也不是很容易，但已经是可能的了）。最好的加密算法对系统性能几乎没有影响，并且还可以带来其他内在的优点。例如，大家都知道的 pkzip，它既压缩数据又加密数据。又如，dbms 的一些软件包总是包含一些加密方法以使复制文件这一功能对一些敏感数据是无效的，或者需要用户的密码。所有这些加密算法都要有高效的加密和解密能力。

在所有的加密算法中最简单的一种就是“ 置换表 ”算法，这种算法也能很好达到加密的需要。每一个数据段（总是一个字节）对应着“ 置换表 ”中的一个偏移量，偏移量所对应的值就输出成为加密后的文件。加密程序和解密程序都需要一个这样的“ 置换表 ”。事实上，80x86 cpu 系列就有一个指令‘ xlat ’在硬件级来完成这样的工作。这种加密算法比较简单，加密解密速度都很快，但是一旦这个“ 置换表 ”被对方获得，那这个加密方案就完全被识破了。更进一步讲，这种加密算法对于黑客破译来讲是相当直接的，只要找到一个“ 置换表 ”就可以了。这种方法在计算机出现之前就已经被广泛地使用。

对这种“ 置换表 ”方式的一个改进就是使用 2 个或者更多的“ 置换表 ”，这些表都是基于数据流中字节的位置的，或者基于数据流本身。这时，破译变得更加困难，因为黑客必须正确地做几次变换。通过使用更多的“ 置换表 ”，并且按伪随机的方式使用每个表，这种改进的加密方法已经变的很难破译。比如，我们可以对所有的偶数位置的数据使用 a 表，对所有的奇数位置使用 b 表，即使黑客获得了明文和密文，他想破译这个加密方案也是非常困难的，除非黑客确切的知道用了两张表。

与使用“ 置换表 ”相类似，“ 变换数据位置 ”也在计算机加密中使用。但是，这需要

更多的执行时间。从输入中读入明文放到一个 buffer 中，再在 buffer 中对他们重排序，然后按这个顺序再输出。解密程序按相反的顺序还原数据。这种方法总是和一些别的加密算法混合使用，这就使得破译变得特别的困难，几乎有些不可能了。例如，有这样一个词，变换其字母的顺序，slient 可以变为 listen，但所有的字母都没有变化，没有增加也没有减少，但是字母之间的顺序已经变化了。

但是，还有一种更好的加密算法，只有计算机可以做，就是字/字节循环移位和 xor 操作。如果我们把一个字或字节在一个数据流内做循环移位，使用多个或变化的方向（左移或右移），就可以迅速的产生一个加密的数据流。这种方法是很好的，破译它就更加困难。而且，更进一步的是，如果再使用 xor 操作，按位做异或操作，就使破译密码更加困难了。如果再使用伪随机的方法，这涉及到要产生一系列的数字，我们可以使用 fibbonaci 数列。对数列所产生的数做模运算（例如模 3），得到一个结果，然后循环移位这个结果的次数，将使破译次密码变得几乎不可能！但是，使用 fibbonaci 数列这种伪随机的方式所产生的密码对我们的解密程序来讲是非常容易的。

在一些情况下，我们想能够知道数据是否已经被篡改了或被破坏了，这时就需要产生一些校验码，并且把这些校验码插入到数据流中。这样做对数据的防伪与程序本身都是有好处的。但是感染计算机程序的病毒才不会在意这些数据或程序是否加过密，是否有数字签名。所以，加密程序在每次 load 到内存要开始执行时，都要检查一下本身是否被病毒感染，对与需要加、解密的文件都要做这种检查！很自然，这样一种方法体制应该保密的，因为病毒程序的编写者将会利用这些来破坏别人的程序或数据。因此，在一些反病毒或杀病毒软件中一定要使用加密技术。

循环冗余校验是一种典型的校验数据的方法。对于每一个数据块，它使用位循环移位和 xor 操作来产生一个 16 位或 32 位的校验和 ，这使得丢失一位或两个位的错误一定会导致校验和出错。这种方式很久以来就应用于文件的传输，例如 xmodem-crc。 这是方法已经成为标准，而且有详细的文档。但是，基于标准 crc 算法的一种修改算法对于发现加密数据块中的错误和文件是否被病毒感染是很有效的。

2．基于公钥的加密算法

一个好的加密算法的重要特点之一是具有这种能力：可以指定一个密码或密钥，并用它来加密明文，不同的密码或密钥产生不同的密文。这又分为两种方式：对称密钥算法和非对称密钥算法。所谓对称密钥算法就是加密解密都使用相同的密钥，非对称密钥算法就是加密解密使用不同的密钥。非常著名的 pgp 公钥加密以及 rsa 加密方法都是非对称加密算法。加密密钥，即公钥，与解密密钥，即私钥，是非常的不同的。从数学理论上讲，几乎没有真正不可逆的算法存在。例如，对于一个输入‘ a ’执行一个操作得到结果‘ b ’，那么我们可以基于‘ b ’，做一个相对应的操作，导出输入‘ a ’。在一些情况下，对于每一种操作，我们可以得到一个确定的值，或者该操作没有定义（比如，除数为 0）。对于一个没有定义的操作来讲，基于加密算法，可以成功地防止把一个公钥变换成为私钥。因此，

要想破译非对称加密算法，找到那个唯一的密钥，唯一的方法只能是反复的试验，而这需要大量的处理时间。

rsa 加密算法使用了两个非常大的素数来产生公钥和私钥。即使从一个公钥中通过因数分解可以得到私钥，但这个运算所包含的计算量是非常巨大的，以至于在现实上是不可行的。加密算法本身也是很慢的，这使得使用 rsa 算法加密大量的数据变的有些不可行。这就使得一些现实中加密算法都基于 rsa 加密算法。pgp 算法(以及大多数基于 rsa 算法的加密方法)使用公钥来加密一个对称加密算法的密钥，然后再利用一个快速的对称加密算法来加密数据。这个对称算法的密钥是随机产生的，是保密的，因此，得到这个密钥的唯一方法就是使用私钥来解密。

3．一个崭新的多步加密算法

现在又出现了一种新的加密算法，几乎不可能被破译的。这个算法在 1998 年 6 月 1 日才正式公布。下面详细介绍这个算法:

使用一系列的数字（比如说 128 位密钥），来产生一个可重复的但高度随机化的伪随机的数字的序列。一次使用 256 个表项，使用随机数序列来产生密码转表，如下所示：

把 256 个随机数放在一个矩阵中，然后对他们进行排序，使用这样一种方式（我们要记住最初的位置）使用最初的位置来产生一个表，随意排序的表，表中的数字在 0 到 255 之间。如果不是很明白如何来做，就可以不管它。下面提供了一些原码使我们明白是如何来做的。现在，产生了一个具体的 256 字节的表。让这个随机数产生器接着来产生这个表中的其余的数，以至于每个表是不同的。下一步，使用"shotgun technique"技术来产生解码表。基本上说，如果 a 映射到 b，那么 b 一定可以映射到 a，所以 b[a[n]] = n.（n 是一个在 0 到 255 之间的数)。在一个循环中赋值，使用一个 256 字节的解码表它对应于我们刚才在上一步产生的 256 字节的加密表。

使用这个方法，已经可以产生这样的一个表，表的顺序是随机，所以产生这 256 个字节的随机数使用的是二次伪随机,使用了两个额外的 16 位的密码。现在，已经有了两张转换表，基本的加密解密是如下这样工作的。前一个字节密文是这个 256 字节的表的索引。或者，为了提高加密效果，可以使用多余 8 位的值，甚至使用校验和或者 crc 算法来产生索引字节。假定这个表是 256*256 的数组,将会是下面的样子:

```
crypto1 = a[crypto0][value]
```

变量'crypto1'是加密后的数据，'crypto0'是前一个加密数据（或着是前面几个加密数据的一个函数值)。很自然的，第一个数据需要一个“ 种子 ”，这个“ 种子 ” 是我们必须记住的。如果使用 256*256 的表，这样做将会增加密文的长度。或者，可以使用你产生出随机数序列所用的密码，也可能是它的 crc 校验和。顺便提及的是曾作过这样一个测试