现代密码学的基础理论

1、加密技术概述

一个密码系统的安全性只在于密钥的保密性，而不在算法的保密性。

对纯数据的加密的确是这样。对于你不愿意让他看到这些数据(数据的明文) 的人，用可靠的加密算法，只要破解者不知道被加密数据的密码，他就不可解读这些数据。

但是，软件的加密不同于数据的加密，它只能是“隐藏”。不管你愿意不愿意让他（合法用户，或 Cracker）看见这些数据（软件的明文），软件最终总要在机器上运行，对机器，它就必须是明文。既然机器可以“看见”这些明文，那么 Cracker，通过一些技术，也可以看到这些明文。

于是 ,从理论上 ,任何软件加密技术都可以破解。只是破解的难度不同而已。有的要让最高明的 Cracker 忙上几个月，有的可能不费吹灰之力，就被破解了。所以 ,反盗版的任务(技术上的反盗版 ,而非行政上的反盗版)就是增加 Cracker 的破解难度。让他们花费在破解软件上的成本 ,比他破解这个软件的获利还要高。这样 Cracker 的破解变得毫无意义——谁会花比正版软件更多的钱去买盗版软件 ？

2 密码学简介

2.1 概念

(1) 发送者和接收者

假设发送者想发送消息给接收者 , 且想安全地发送信息 : 她想确信偷听者不

能阅读发送的消息。

(2) 消息和加密

消息被称为明文。用某种方法伪装消息以隐藏它的内容的过程称为加密，加了密的消息称为密文，而把密文转变为明文的过程称为解密。

明文用 M（消息）或 P（明文）表示，它可能是比特流（文本文件、位图、数字化的语音流或数字化的视频图像）。至于涉及到计算机，P 是简单的二进制数据。明文可被传送或存储，无论在哪种情况，M 指待加密的消息。

密文用 C 表示，它也是二进制数据，有时和 M 一样大，有时稍大（通过压缩和加密的结合，C 有可能比 P 小些。然而，单单加密通常达不到这一点）。加密函数 E 作用于 M 得到密文 C，用数学表示为：

E（M）=C.

相反地，解密函数 D 作用于 C 产生 M

D（C）=M.

先加密后再解密消息，原始的明文将恢复出来，下面的等式必须成立：

D（E（M））=M

(3) 鉴别、完整性和抗抵赖

除了提供机密性外，密码学通常有其它的作用：.

(a) 鉴别

消息的接收者应该能够确认消息的来源；入侵者不可能伪装成他人。

(b) 完整性检验

消息的接收者应该能够验证在传送过程中消息没有被修改;入侵者不可能用假消息代替合法消息。

(c) 抗抵赖

发送者事后不可能虚假地否认他发送的消息。

(4) 算法和密钥

密码算法也叫密码，是用于加密和解密的数学函数。（通常情况下，有两个相关的函数：一个用作加密，另一个用作解密）

如果算法的保密性是基于保持算法的秘密，这种算法称为受限制的算法。受限制的算法具有历史意义，但按现在的标准，它们的保密性已远远不够。大的或经常变换的用户组织不能使用它们，因为每有一个用户离开这个组织，其它的用户就必须改换另外不同的算法。如果有人无意暴露了这个秘密，所有人都必须改变他们的算法。

更糟的是，受限制的密码算法不可能进行质量控制或标准化。每个用户组织必须有他们自己的唯一算法。这样的组织不可能采用流行的硬件或软件产品。但窃听者却可以买到这些流行产品并学习算法，于是用户不得不自己编写算法并予以实现，如果这个组织中没有好的密码学家，那么他们就无法知道他们是否拥有安全的算法。

尽管有这些主要缺陷，受限制的算法对低密级的应用来说还是很流行的，用户或者没有认识到或者不在乎他们系统中内在的问题。

现代密码学用密钥解决了这个问题，密钥用 K 表示。K 可以是很多数值里的任意值。密钥 K 的可能值的范围叫做密钥空间。加密和解密运算都使用这个密钥（即运算都依赖于密钥，并用 K 作为下标表示），这样，加/解密函数现在变成：

EK(M)=C

DK(C)=M.

这些函数具有下面的特性：

DK（EK（M））=M.

有些算法使用不同的加密密钥和解密密钥，也就是说加密密钥 K1 与相应的解密密钥 K2 不同，在这种情况下：

EK1(M)=C

DK2(C)=M

DK2 (EK1(M))=M

所有这些算法的安全性都基于密钥的安全性 ;而不是基于算法的细节的安全性。这就意味着算法可以公开，也可以被分析，可以大量生产使用算法的产品，即使偷听者知道你的算法也没有关系 ;如果他不知道你使用的具体密钥 ,他就不可能阅读你的消息。

密码系统由算法、以及所有可能的明文、密文和密钥组成的。

基于密钥的算法通常有两类：对称算法和公开密钥算法。下面将分别介绍：

2.2 对称密码算法

对称算法有时又叫传统密码算法 ,就是加密密钥能够从解密密钥中推算出来，反过来也成立。在大多数对称算法中，加/解密密钥是相同的。这些算法也叫秘密密钥算法或单密钥算法，它要求发送者和接收者在安全通信之前，商定一个密钥。对称算法的安全性依赖于密钥，泄漏密钥就意味着任何人都能对消息进行加/解密。只要通信需要保密，密钥就必须保密。

对称算法的加密和解密表示为：

EK（M）=C

DK（C）=M

对称算法可分为两类。一次只对明文中的单个比特（有时对字节）运算的算法称为序列算法或序列密码。另一类算法是对明文的一组比特亚行运算，这些比

特组称为分组，相应的算法称为分组算法或分组密码。现代计算机密码算法的典型分组长度为 64 比特——这个长度大到足以防止分析破译，但又小到足以方便使用（在计算机出现前，算法普遍地每次只对明文的一个字符运算，可认为是序列密码对字符序列的运算）。

2.3 公开密码算法

公开密钥算法（也叫非对称算法）是这样设计的：用作加密的密钥不同于用作解密的密钥，而且解密密钥不能根据加密密钥计算出来（至少在合理假定的长时间内）。之所以叫做公开密钥算法，是因为加密密钥能够公开，即陌生者能用加密密钥加密信息，但只有用相应的解密密钥才能解密信息。在这些系统中，加密密钥叫做公开密钥（简称公钥），解密密钥叫做私人密钥（简称私钥）。私人密钥有时也叫秘密密钥。为了避免与对称算法混淆，此处不用秘密密钥这个名字。

用公开密钥 K 加密表示为

EK(M)=C.

虽然公开密钥和私人密钥是不同的，但用相应的私人密钥解密可表示为：

DK(C)=M

有时消息用私人密钥加密而用公开密钥解密，这用于数字签名（后面将详细介绍），尽管可能产生混淆，但这些运算可分别表示为：

EK(M)=C

DK(C)=M

当前的公开密码算法的速度，比起对称密码算法，要慢的多，这使得公开密码算法在大数据量的加密中应用有限。

2.4 单向散列函数

单向散列函数 H(M) 作用于一个任意长度的消息 M，它返回一个固定长度

的散列值 h，其中 h 的长度为 m 。

输入为任意长度且输出为固定长度的函数有很多种，但单向散列函数还有使其单向的其它特性：

(1) 给定 M ，很容易计算 h ；

(2) 给定 h ，根据 H(M) = h 计算 M 很难 ；

(3) 给定 M ，要找到另一个消息 M ‘ 并满足 H(M) = H(M’) 很难。

在许多应用中，仅有单向性是不够的，还需要称之为“抗碰撞”的条件：

要找出两个随机的消息 M 和 M ‘，使 H(M) = H(M’) 满足很难。

由于散列函数的这些特性，由于公开密码算法的计算速度往往很慢，所以，在一些密码协议中，它可以作为一个消息 M 的摘要，代替原始消息 M，让发送者为 H(M) 签名而不是对 M 签名 。如 SHA 散列算法用于数字签名协议 DSA 中。

2.5 数字签名

提到数字签名就离不开公开密码系统和散列技术。

有几种公钥算法能用作数字签名。在一些算法中，例如 RSA，公钥或者私钥都可用作加密。用你的私钥加密文件，你就拥有安全的数字签名。在其它情况下，如 DSA，算法便区分开来了??数字签名算法不能用于加密。这种思想首先由 Diffie 和 Hellman 提出 。

基本协议是简单的 ：

(1) A 用她的私钥对文件加密，从而对文件签名。

(2) A 将签名的文件传给 B。

(3) B 用 A 的公钥解密文件，从而验证签名。

这个协议中，只需要证明 A 的公钥的确是她的。如果 B 不能完成第（3）步，那么他知道签名是无效的。

这个协议也满足以下特征：

(1) 签名是可信的。当 B 用 A 的公钥验证信息时，他知道是由 A 签名的。

(2) 签名是不可伪造的。只有 A 知道她的私钥。

(3) 签名是不可重用的。签名是文件的函数，并且不可能转换成另外的文件。

(4) 被签名的文件是不可改变的。如果文件有任何改变，文件就不可能用 A 的公钥验证。

(5) 签名是不可抵赖的。B 不用 A 的帮助就能验证 A 的签名。

在实际应用中，因为公共密码算法的速度太慢，签名者往往是对消息的散列签名而不是对消息本身签

名。这样做并不会降低签名的可信性。