常用的加密算法名称及介绍

一、对称性加密算法

对称式加密就是加密和解密使用同一个密钥。信息接收双方都需事先知道密匙和加解密算法且其密匙是相同的，之后便是对数据进行加解密了。对称加密算法用来对敏感数据等信息进行加密。

对称性加密算法有：AES、DES、3DES，介绍如下

* DES（Data Encryption Standard）：数据加密标准，速度较快，适用于加密大量数据的场合。
* 3DES（Triple DES）：是基于DES，对一块数据用三个不同的密钥进行三次加密，强度更高。
* AES（Advanced Encryption Standard）：高级加密标准，是下一代的加密算法标准，速度快，安全级别高；AES是一个使用128为分组块的分组加密算法，分组块和128、192或256位的密钥一起作为输入，对4×4的字节数组上进行操作。众所周之AES是种十分高效的算法，尤其在8位架构中，这源于它面向字节的设计。AES 适用于8位的小型单片机或者普通的32位微处理器,并且适合用专门的硬件实现，硬件实现能够使其吞吐量（每秒可以到达的加密/解密bit数）达到十亿量级。同样，其也适用于RFID系统。

二、非对称加密算法

非对称式加密就是加密和解密所使用的不是同一个密钥，通常有两个密钥，称为"公钥"和"私钥"，它们两个必需配对使用，否则不能打开加密文件。发送双方A,B事先均生成一堆密匙，然后A将自己的公有密匙发送给B，B将自己的公有密匙发送给A，如果A要给B发送消 息，则先需要用B的公有密匙进行消息加密，然后发送给B端，此时B端再用自己的私有密匙进行消息解密，B向A发送消息时为同样的道理。

非对称性算法有：RSA、DSA、ECC，介绍如下：

* RSA：由 RSA 公司发明，是一个支持变长密钥的公共密钥算法，需要加密的文件块的长度也是可变的。RSA在国外早已进入实用阶段，已研制出多种高速的RSA的专用芯片。
* DSA（Digital Signature Algorithm）：数字签名算法，是一种标准的 DSS（数字签名标准），严格来说不算加密算法。
* ECC（Elliptic Curves Cryptography）：椭圆曲线密码编码学。ECC和RSA相比，具有多方面的绝对优势，主要有：抗攻击性强。相同的密钥长度，其抗攻击性要强很多倍。计算量小，处理速度快。ECC总的速度比RSA、DSA要快得多。存储空间占用小。ECC的密钥尺寸和系统参数与RSA、DSA相比要小得多，意味着它所占的存贮空间要小得多。这对于加密算法在IC卡上的应用具有特别重要的意义。带宽要求低。当对长消息进行加解密时，三类密码系统有相同的带宽要求，但应用于短消息时ECC带宽要求却低得多。带宽要求低使ECC在无线网络领域具有广泛的应用前景。

三、散列算法

散列算法，又称哈希函数，是一种单向加密算法。在信息安全技术中，经常需要验证消息的完整性，散列(Hash)函数提供了这一服务，它对不同长度的输入消息，产生固定长度的输出。这个固定长度的输出称为原输入消息的"散列"或"消息摘要"(Message digest)。散列算法不算加密算法，因为其结果是不可逆的，既然是不可逆的，那么当然不是用来加密的，而是签名。

散列算法（签名算法）有：MD5、SHA1、HMAC，介绍如下：

* MD5：MD5是一种不可逆的加密算法，目前是最牢靠的加密算法之一，尚没有能够逆运算的程序被开发出来，它对应任何字符串都可以加密成一段唯一的固定长度的代码。
* SHA1：是由NISTNSA设计为同DSA一起使用的，它对长度小于264的输入，产生长度为160bit的散列值，因此抗穷举(brute-force)性更好。SHA-1设计时基于和MD4相同原理,并且模仿了该算法。SHA-1是由美国标准技术局（NIST）颁布的国家标准，是一种应用最为广泛的Hash函数算法，也是目前最先进的加密技术，被政府部门和私营业主用来处理敏感的信息。而SHA-1基于MD5，MD5又基于MD4。
* HMAC：是密钥相关的哈希运算消息认证码（Hash-based Message Authentication Code）,HMAC运算利用哈希算法，以一个密钥和一个消息为输入，生成一个消息摘要作为输出。也就是说HMAC是需要一个密钥的。所以，HMAC\_SHA1也是需要一个密钥的，而SHA1不需要。

四、Base64加密算法

通常用于把二进制数据编码为可写的字符形式的数据，对数据内容进行编码来适合传输(可以对img图像编码用于传输)。这是一种可逆的编码方式。编码后的数据是一个字符串，其中包含的字符为：A-Z、a-z、0-9、+、/，共64个字符(26 + 26 + 10 + 1 + 1 = 64，其实是65个字符，“=”是填充字符。Base64要求把每三个8Bit的字节转换为四个6Bit的字节(3\*8 = 4\*6 = 24)，然后把6Bit再添两位高位0，组成四个8Bit的字节，也就是说，转换后的字符串理论上将要比原来的长1/3。原文的字节最后不够3个的地方用0来补足，转换时Base64编码用=号来代替。这就是为什么有些Base64编码会以一个或两个等号结束的原因，中间是不可能出现等号的，但等号最多只有两个。其实不用"="也不耽误解码，之所以用"="，可能是考虑到多段编码后的Base64字符串拼起来也不会引起混淆。)

说明：Base64编码是从二进制到字符的过程，像一些中文字符用不同的编码转为二进制时，产生的二进制是不一样的，所以最终产生的Base64字符也不一样。