摘要

本文以应用密码学在信息安全、网络数据安全等方面的应用为背景，重点分析和研究了目前最受欢迎的 RSA 公钥密码体制。RSA 算法是应用广泛的公开密钥算法，它算法简单、保密性强，而且没有密钥管理的麻烦，特别适合现代保密通信的需要，但是 RSA 算法的最大缺陷就是加密速度较慢。针对这样的问题，作者介绍了几种提高 RSA 速度的方法。重点提出一种新的组合的快速 RSA 算法，并提出如何利用 COM 组件实现该算法。

以 COM 为基础的组件技术，提供了有力、灵活、安全的开发平台，它可以提高软件产品的重用性、安全性和可靠性。作者在上述基础上，设计并实现了一个加密传输系统的 COM 组件，组件中加密和解密采用了 2048 位的 RSA 快速密码算法，可实现加密和解密的操作。加密组件对象封装了许多加/解密所需的功能，如果需要可以进行更新现有接口的功能或是开发新的接口。应用此组件可以对网络传输的数据进行加密和解密，防止信息被非法窃取。

关键词：密码学；RSA 公钥密码体制;快速算法;COM 组件

Abstract

This text regard the computer cryptography apply in the information safe and network data safety etc. for background, The author emphatically analysis and research the most popular RSA public key cryptosystem currently. The RSA algorithm is the most widely deployed public key cryptosystem. Its algorithm is simple, performance of cipher is better, and no trouble of manage key, which is very suitable for modern secret communication. But the most drawback of RSA algorithm is the speed of encryption very slowly. Aimed at this, the author introduces several methods to improve the RSA algorithm' s speed. And based on it, the author put forward a new combined fast RSA algorithm.

Component technology based on COM can provide exploit flat that is emollient, vivid and safety, and it can improve the reused, safety and reliability of software product. According to these, the author design and implement a COM component of cipher transmission system. In this component, encryption and decryption used RSA fast algorithm of 2048-bit, and can use it to encrypt or decrypt, Cipher component object envelope many functions, if it is needed that we can update the function of existent interface or exploit the new interface. This COM component can be used to encrypt or decrypt data that transmit in Internet, and avoids that the information is filched illegal.

Keywords:

Cryptography; RSA public key cryptosystem; fast algorithm; COM component

声　　明

本人所呈交的学位论文是在导师的指导下完成的。论文中所取得的研究成果除加以标注和致谢的地方外，不包含其他人已经发表或撰写过的研究成果，也不包含本人为获得其他学位而使用过的材料。对本研究工作有所贡献的同志均已在论文中作了明确的说明并表示谢意。

本人签名：何秋歌

日　　期：2004年1月18日

第一章 绪论

1.1 研究背景

二十世纪末，一场信息技术革命席卷了社会的每一个角落，其变革速度之快，影响之大，涉及面之广均是无可比拟的。信息技术的发展对世界范围内的经济、科教、信息交流及社会发展乃至各国的国家安全都产生了十分深刻的影响。世界各国都在大力发展信息产业以便跟上时代发展的步伐并抢占信息控制权。在我们惊叹网络技术日新月异、在无边无际的 Internet 上畅游时，通信网络已经成为那些具有一定经济资源和技术专长的人搞破坏的靶子。雇员、黑客、恐怖组织、操纵政治和情报活动的外国政府、涉及工业谍报活动的国内外企业均在不断提高对计算机和通信系统的攻击水平。在信息社会中，信息安全是一个永恒的话题。

2000 年 1 月上旬，美国出台了电脑空间安全计划，随即又提出追加 20 亿美元的信息安全预算，网络安全市场陡然获得全球重视。此后的骇客事件频繁爆发，震动了全世界。2000 年 2 月 7 日至 9 日，美国著名的 Yahoo, Amazon, eBay 等 8 大顶级网站遭来历不明的电子攻击导致服务中断，整个互联网的使用率两天之内下降了 20%，这三天的袭击造成的直接损失达 12 亿美元，间接损失更达数十亿美元[18]。信息安全问题就这样在正面的倡导和反面的佐证面前，成为了信息化和计算机产业的主角。另外，黑客的话题频见报道，黑客的身影仿佛无处不在，多起信用卡账户和密码被公开的事件，让人感受到了网络的便利的同时也体会到了潜在的风险。

随着全球信息化的飞速发展，我国大力建设各种信息化系统。目前，我国信息安全产品的营业额约占整个网络产品的 10%，据此估计全国网络信息安全产品 1 年的营业额将在 50 亿元人民币左右。截至 2000 年 9 月底，在我国从事计算机信息系统安全产品研究开发、生产的企业达到 230 家，已经领取销售许可证的安全专用产品达 335 种。2000 年 7 月我国第一个国家信息安全产业化基地在成都隆重奠基，不久国家第二个信息安全产业基地又在上海拉开了序幕[18]。我国的网络信息安全产品和技术已经从加密、防火墙，过渡到包括身份认证、入侵监测、防病毒等五大系列全方位的防护体系。这说明我国的计算机信息系统安全产品已形成了一定的产业规模。

但是，我国信息安全产业的发展现状与信息化建设尚不相符，网络信息安全日益突出，已成为亟待解决的影响国家全局和长远利益的关键问题之一。

网络信息安全问题是一个关键问题，它所包含的内容也十分丰富。其中加密是一种主动的防卫措施。其核心将建立在密码学理论与技术上。密码技术是保护信息安全的主要手段之一，使用密码技术不仅可以保证信息的机密性，而且可以保证信息的完整性和确定性，防止信息被篡改、伪造和假冒。

1.2 密码学的发展史

密码学(Cryptography)一词来源于古希腊的 Crypto 和 Graphing，意思是密写。它是以认识密码变换的本职、研究密码保护与破译的基本规律为对象的学科。经典密码学主要包括两个既对立又统一的分支：密码编码学和密码分析学。研究密码变化的规律并用之于编制密码以保护秘密信息的科学，称为密码编码学；研究密码变化的规律用之于密码以获取信息情报的科学，称为密码分析学，也叫密码破译学。前者是实现对信息保护的，后者是实现对信息反保护的，密码编码学与密码分析学相辅相成，共处于密码学的统一体中。

现代密码学除了包括密码编码学和密码破译学两个主要学科外，还包括近几年才形成的新分支——密码密钥学，它是以密码的核心部分——密钥作为研究对象的学科。密钥管理是一种规程，它包括密钥的产生、分配、存储、保护、销毁等环节，在保密系统中至关重要。上述三个分支学科构成现代密码学的主要学科体系。

密码学本与计算机科学无关。早在远古时代由于战争通信的需要便产生了密码学。它有过自己辉煌的经历。但成为人们独立的学科是近二十余年的事。由于计算机网络的飞速发展，它与密码学的结合，使得近代密码学理论得到革命性的变革和迅速发展。从而形成了所谓计算机密码学。它实际上是计算机科学与技术、通信科学和数学等多学科的交叉。

1949 年，信息论的创始人，美国数学家 Shannon 发表了题为“保密通信的信息理论”的论文[8]，奠定了近代密码学理论的基础。

近代计算机密码学的伟大成就之一事 1977 年美国国家标准局正式公布实施的所谓数据加密标准(DES 算法)[9]。它公开了加密算法，从此揭开了密码学神秘的面纱。计算机密码学的第二个伟大成就就是由 Diffie 和 Hellman 开创的公钥密码学。特别是 RSA 公钥密码算法的提出是密码学发展史上的又一个里程碑[6]。

1.3 密码学的基本概念

1 消息和加密

密码学就是研究加密和解密变换的科学，一般来说，最初要通过网络传输又未经任何变换的消息称为明文(plaintext)，用某种方法伪装消息以隐藏它的内容的过程称为加密(encryption)，被加密的消息称为密文(cipher text)，而把密文转换为明文的过程称为解密(decryption)。

明文(plaintext)用 M 或 P 表示，它可能位序列、文本文件、位图、数字化的语音序列或数字化的视频图像等等。对于计算机，M 指简单的二进制数据。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 M 小些。仅通过加密通常达不到这一点)。加密函数 E 作用于 M 得到密文 C，可用数学公式表示：

$$E(M)=C \tag{1.3.1}$$

相反地，解密函数 D 作用于 C 产生 M：

$$D(C)=M \tag{1.3.2}$$

先加密后再解密，原始的明文将恢复，故下面的等式必须成立：

$$D(E(M))=M \tag{1.3.3}$$

2 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：

—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。

—完整性：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。

—抗抵赖：发送者事后不可能虚假地否认他发送的消息。

这些功能是通过计算机进行社会交流至关重要的需求，就像面对面交流一样。某人是否就是他说的人；某人的身份证明文件是否有效；声称从某人那里来的文件是否就是从那个人那里来的；这些事情都是通过鉴别、完整性和抗抵赖来实现的。

3 算法和密钥

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

密钥是由密码体制的用户随机选取，并能唯一确定明文/密文之间变换的一个(对)随机字符串。密钥(Key)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：

$$E_K(M)=C \tag{1.3.4}$$

$$D_K(C)=M \tag{1.3.5}$$

这些函数具有下面的特性，如图 1.1 所示：

$$D_K(E_K\ (M))=M \tag{1.3.6}$$

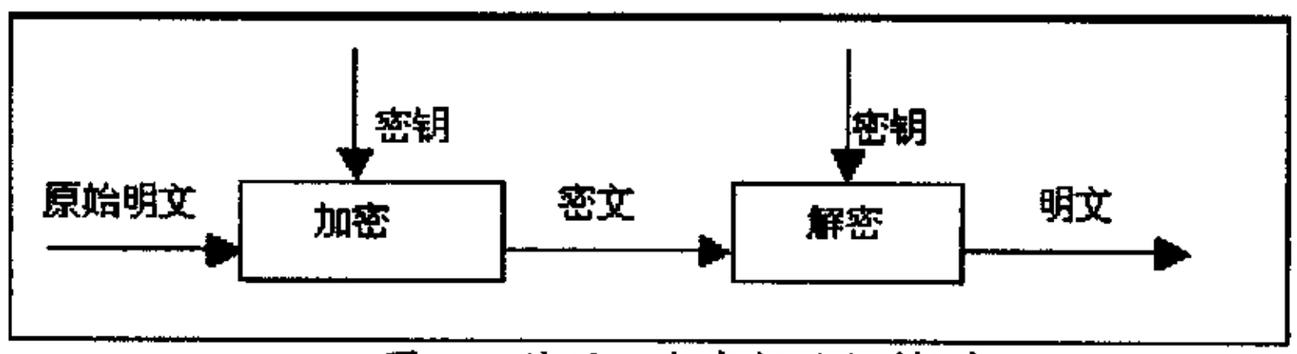

图 1.1 使用一个密钥的加/解密

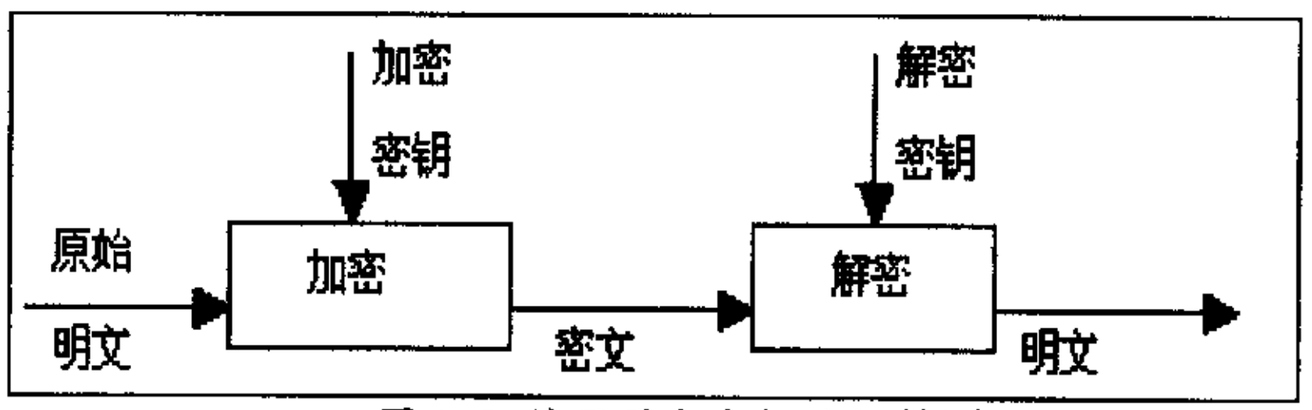

图 1.2 使用两个密钥的加/解密

有些算法使用不同的加密密钥和解密密钥，如图 1.2 所示，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：

$$E_{K1}(M)=C \tag{1.3.7}$$

$$D_{K2}(C)=M \tag{1.3.8}$$

$$D_{K2}(E_{K1}(M))=M \tag{1.3.9}$$

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。

4 对称算法

基于密钥的算法通常有两类：对称算法和公开密钥算法。

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。这些算法也叫私密密钥算法

或单密密钥算法，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：$E_K(M)=C$ 和 $D_K(C)=M$。

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进行运算，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。

5 公开密钥算法

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做解密的密钥，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)。

用公开密钥 K 加密表示为：$E_K(M)=C$，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：$D_K(C)=M$。

6 密码分析

密码编码学的主要目的是保持明文(或密钥，或明文和密钥)的秘密，以防止偷听者(也叫对手、攻击者、截取者、入侵者、敌手或干脆称为敌人)知晓。这里假设偷听者完全能够截获收发者之间的通信。

密码分析学是在不知道密钥的情况下，恢复出明文的科学。成功的密码分析能恢复出消息的明文或密钥。密码分析也可以发现密码体制的弱点，最终得到上述结果(密钥通过非密码分析方式的丢失叫做泄露)。

对密码进行分析的常识称为攻击。常用的密码分析攻击有以下几类：

(1)唯密文攻击　密码分析者有一些消息的密文，这些消息都用同一加密算法加密。密码分析者的任务是恢复尽可能多的明文，或者最好是能够推算出加密消息的密钥由来，以便可采用相同的密钥解出其他被加密的消息。

已知：$C_1=E_K(P_1)$，$C_2=E_K(P_2)$，……，$C_i=E_K(P_i)$

推导出：P_1，P_2，…，P_i；K 或者找出一个算法从 $C_{i+1}=E_K(P_{i+1})$ 推出 P_{i+1}。

(2)已知明文攻击　密码分析者不仅可能达到一些消息的密文，而且也知道这些消

息的明文。分析者的任务就是用加密信息推出用来加密的密钥或导出一个算法，此算法可以对用同一密钥加密的任何新的消息进行解密。

已知：P_1，$C_1=E_K(P_1)$，P_2 ，$C_2=E_K(P_2)$，……P_i ，$C_i=E_K(P_i)$

推导出：密钥 K，或从 $C_{i+1}=E_K(P_{i+1})$ 推出 P_{i+1} 的算法。

(3)选择明文攻击法　分析者不仅可以得到一些消息的密文和相应的明文，而且他们也可以选择被加密的明文。这比已知明文攻击更有效。因为密码分析者能选择特定的明文块去加密，哪些块可能产生更多关于密钥的信息，分析者的任务是推出用来加密消息的密钥或导出一个算法，此算法可以对用同一密钥加密的任何新的消息进行解密。

已知：P_1，$C_1=E_K(P_1)$，P_2 ，$C_2=E_K(P_2)$，……P_i ，$C_i=E_K(P_i)$

其中 P_1，P_2，…，P_i 是由密码分析者选择的。

推导出：密钥 K，或从 $C_{i+1}=E_K(P_{i+1})$ 推出 P_{i+1} 的算法。

(4)自适应选择明文攻击　这是选择明文攻击的特殊情况。密码分析者不仅能选择被加密的明文，而且也能基于以前加密的结果修正这个选择。在选择明文攻击中，密码分析者还可以选择一大块被加密了的明文。而后再基于第一块的结果选择另一明文块，以此类推，此外还有其他的密码分析攻击方法。

(5)选择密文攻击。密码分析者能选择不同的被加密的密文，并可得到对应的解密的明文，例如密码分析者存取一个防窜改的自动解密盒，密码分析者的任务是推出密钥。

已知：C_1，$P_1=D_K(C_1)$，C_2 ，$P_2=D_K(C_2)$，……C_i ，$P_i=D_K(C_i)$，

推导出：　K。

这种攻击主要用于公开密钥体制。选择密文攻击有时也可以有效地用于对称算法。

(6)选择密钥攻击。这种攻击并不表示密码分析者能够选择密钥，它只表示密码分析者具有不同密钥之间的关系的有关知识。这种方法有点奇特和晦涩，不是很实际。

最好的算法是哪些已经公开的，并经过世界上最好的密码分析家们多年的攻击，但是还是不能破译的算法。

7 算法的安全性

根据被破译的难易程度，不同的密码算法具有不同的安全等级。如果破译算法的代价大于加密数据的价值，那么你可能是安全的。如果破译算法所需的时间比加密数据保密的时间更长，那么你可能是安全的；如果用单密钥加密的数据量比破译算法需要的数据量少得多，那么你可能是安全的。

之所以说“可能”是因为在密码分析中总有新的突破。另一方面，大多数数据随着

时间的推移，其价值会越来越小，而数据的价值总是比突破保护它的安全性的代价更小，这点是很重要的。

Lars Knudsen 把破译算法分为不同的类别，安全性的递减顺序为：

1) **全部破译**。密码分析者找出密钥 K，这样 $D_K(C)=P$。

2) **全盘推导**。密码分析者找到一个代替算法 A，在不知道密钥 K 的情况下，等价于

$$D_K(C)=P。$$

3) **实例(或局部)推导**。密码分析者从截获的密文中找出明文。

4) **信息推导**。密码分析者获得一些有关密钥或明文的信息。这些信息可能是密钥的几个比特、有关明文格式的信息等等。

如果不论密码分析者有多少密文，都没有足够的信息恢复出明文，那么这个算法就是无条件保密的，事实上，只有一次一密乱码本[6]，才是不可破的(给出无限多的资源仍然不可破)。所有其它的密码系统在唯密文攻击中都是可破的，只要简单地一个接一个地去试每种可能的密钥，并且检查所得明文是否有意义。

密码学更关心在计算上不可破译的密码系统。如果一个算法用(现在或将来)可得到的资源都不能破译，这个算法则被认为在计算上是安全的(有时叫做强的)。准确地说，“可用资源”就是公开数据的分析整理。

1.4 国内外密码学理论与技术研究现状及发展趋势

密码学理论与技术主要包括两部分，即基于数学的密码理论与技术(包括公钥密码、分组密码、序列密码、认证码、数字签名、Hash 函数、身份识别、密钥管理、PKI 技术等)和非数学的密码理论与技术(包括信息隐形，量子密码，基于生物特征的识别理论与技术)。

自从 1976 年公钥密码的思想提出以来，国际上已经提出了许多种公钥密码体制，但比较流行的主要由两类：一类是基于大整数因子分解问题的，其中最典型的代表是 RSA[6]；另一类是基于离散对数问题的，比如 ElGamal 公钥密码和影响比较大的椭圆曲线公钥密码。由于分解大整数的能力日益增强，所以对 RSA 的安全带来了一定的威胁。目前 768 比特模长的 RSA 已不安全。一般建议使用 1024 比特模长，预计要保证 20 年的安全就要选择 1280 比特的模长，增大模长带来实现上的难度。而基于离散对数问题的公钥密码在目前技术下 512 比特模长就能够保证其安全性。特别是椭圆曲线上的离散对数的计算要比有限域上的离散对数的计算更困难，目前技术下只需要 160 比特模长即可，适合于智能卡的实现，因而受到国内外学者的广泛关注。国际上制定了椭圆曲线公钥密

码标准 IEEEP1363，RSA 等一些公司声称他们已经开发出了符合该标准的椭圆曲线公钥密码。我国学者也提出了一些公钥密码算法[10]，另外在公钥密码的快速实现方面也做了一定的工作，比如在 RSA 的快速实现和椭圆曲线公钥密码的快速实现方面都有所突破。公钥密码的快速实现是当前公钥密码研究中的一个热点，包括算法优化和程序化，另一个人们所关注的问题是椭圆曲线公钥密码的安全性论证问题。

公钥密码主要用于数字签名和密钥分配。当然，数字签名和密钥分配都有自己的研究体系，形成了各自的理论框架。目前数字签名的研究内容非常丰富，包括普通签名和特殊签名。特殊签名有盲签名，代理签名，群签名，不可否认签名，公平盲签名，门限签名，具有消息恢复功能的签名等，它与具体应用环境密切相关。显然，数字签名的应用涉及到法律问题，美国联邦政府基于有限域上的离散对数问题制定了自己的数字签名标准(DSS)，部分州已经制定了数字签名法。在密钥管理方面，国际上都有一些大的举动，比如 1993 年美国提出的密钥托管理论和技术、国际标准化组织制定的 X.509 标准(已经发展到第 3 版本)以及麻省理工学院开发的 Kerboros 协议(已经发展到第 5 版本)等，这些工作影响很大。

Hash 函数主要用于完整性校验和提高数字签名的有效性，目前已经提出了很多方案，各有千秋。美国已经制定了 Hash 标准和-SHA-1，与其数字签名标准匹配使用。由于技术的原因，美国目前正准备更新其 Hash 标准，另外，欧洲也正在制定 Hash 标准，这将导致 Hash 函数的研究特别是实用技术的研究将成为热点。

在身份识别的研究中，最令人瞩目的识别方案有两类：一类是 1984 年 shamir 提出的基于身份识别方案，另一类是 1986 年 Fiat 等人提出的零知识身份识别方案[15]。随后，人们在这两类方案的基础上有提出了一系列实用的身份识别方案，比如，Schnorr 识别方案、Okamoto 识别方案、Guillu-Quisquater 识别方案、Feige-Fiat-Shamir 识别方案等。目前人们所关注的是身份识别方案与具体应用环境的有机结合。

序列密码主要用于政府、军方等国家要害部门，尽管用于这些部门的理论和技术都是保密的，但由于一些数学工具(比如代数、数论、概率等)可用于研究序列密码，其理论和技术相对比较而言比较成熟。虽然，近年来序列密码不是一个研究热点，但是有很多有价值的公开问题需要进一步解决，比如自同步流密码的研究，有记忆前馈网络密码系统的研究，混沌序列密码和新研究方法的探索等。另外虽然没有制定密码序列标准，但在一些系统中广泛使用了序列密码比如 RC4，用于存储加密。事实上，欧洲的 NESSIE 计划中已经包括了序列密码标准的制定，这一举措有可能导致序列密码研究热。

美国早在 1977 年就制定了自己的数据加密标准(一种分组密码)，随着美国的数据

加密标准的出现，人们对分组密码展开了深入的研究和讨论，设计了大量的分组密码，给出了一系列的评测准则。美国国家标准技术研究所（NIST）于 1997 年初发起并组织了在全世界广泛征集新的加密标准算法的活动，同时要求每一种后选算法应当支持 128、192、和 256 比特的密钥长度。经过 3 年多时间的反复较量，对首轮入选的 15 种不同算法进行了广泛的评估与测试，筛选出 5 种算法进入决赛。最终，由比利时的密码专家 Joan Daemen(Proton world International 公司)及 Vincent Rijmen(Leuven 大学)所提出的加密算法 Rijndael 入选，成为 21 世纪新的高级加密算法 AES[17]。其他国家，如日本和苏联也纷纷提出了自己的数据加密标准。我国目前的做法是针对每个或每一类安全产品需要开发所用的算法，而且算法和源代码都不公开，这样一来，算法的需求量相对就比较大，继而带来了兼容性、互操作性等问题。

目前最为人们所关注的实用密码技术是 PKI 技术。国外的 PKI 应用已经开始，开发 PKI 的厂商也有多家。许多厂家，如 Baltimore ，Entrust 等推出了可以应用的 PKI 产品，有些公司如 VerySign 等已经开始提供 PKI 服务。网络许多应用正在使用 PKI 技术来保证网络的认证、不可否认、加解密和密钥管理等。尽管如此，总的说来 PKI 技术仍在发展中。IDC 公司的 Internet 安全分析家认为：PKI 技术将成为所有应用计算机基础结构的核心部件，包括那些越出传统网络界限的应用。BTB 电子商务活动需要的认证、不可否认等只有 PKI 产品才有能力提供这些功能。

密码技术特别是加密技术是信息安全技术中的核心技术，国家关键基础设施中不可能引进或采用别人的加密技术，只能是自主开发。目前我国在密码技术的应用水平方面与国外有一定的差距。国外的密码技术必将对我们有一定的冲击力，特别是在加入 WTO 组织后这种冲击力有增无减。有些做法必须要逐渐与国际接轨，不能再采用目前这种关门造车的做法，因此，我们必须要有自己的算法来对付未来的挑战。实用密码技术的基础是密码基础理论，没有好的密码理论不可能有好的密码技术、也不可能有先进的、自主的创新的密码技术。因此首先必须持之以恒地坚持和加强密码基础理论研究，与国际保持同步。另一方面，密码理论研究也是为了应用，没有应用的理论是没有价值的。我们应在现有理论和技术基础上充分吸收国外先进经验形成自主的、创新的密码技术。

1.5 现代密码学

1 对称密码系统——分组密码(Block cipher)

对称式密钥系统既是所谓的单密钥系统，即加密方利用一个密钥对数据进行加密，解密方接收到数据后需要用同一密钥来进行解密。1976 年以前存在的所有密码系统，

即属于此种类型。这种加密技术的特点具有很高的保密强度，可以达到经受破译力量的分析和攻击。而且数学运算量小，加密速度快，易于处理。但是它的密钥分发和管理比较困难，一旦密钥泄漏，那么以后的秘密通信也就难以保证了。因此密钥管理成为影响系统安全的关键因素，使它难以满足系统开放性要求。另外，当有许多人希望进行保密通信时，彼此间不能使用相同的密钥，所以 n 个用户要使用 C(n,2)=n(n-1)/2 个密钥，显然密钥管理难度比较大。

分组密码的工作方式是将明文分成固定长度的组(块)，如 64 比特一组，用同一密钥和算法对每一个块加密，输出也是固定长度的密文。设计分组密码算法的核心技术是在相信复杂函数可以通过简单函数迭代若干圈得到的原则下，利用简单函数及对合等运算，充分利用非线性运算。以 DES 算法为例，它采用美国国家安全局核心设计的 8 个 S 盒盒 P 置换，经过 16 圈迭代最终产生 64 位比特密文，每轮迭代值用的 48 比特子密钥是由原始的 56 比特产生的。

对称密钥算法主要是由两个动作组成的，一个是换位(Transposition)，另一个是替换(或代换 Substitution)。无论多么复杂的对称加密绝对脱离不了这两种最基本的动作。在对称密码系统中，最具有代表性的当属 DES 算法。DES 算法一般使用 64 位(实际上用到 56 位)密钥对 64 位二进制文本块进行加密，具有很高的安全强度，而且它设计精巧，容易实现，加密速度快。此外还有三重 DES 算法、FEAL 算法、IDEA 算法、SKIPJACK 算法和前苏联国家标准 NSSU 等。在这章中，就这两个动作加以说明，之后介绍几种对称加密算法，并做一些安全方面的分析。

➢ **换位加密法(Transposition)**

换位加密法早在公元 400 年前就已开始被希腊人所使用。它的基本精神简单地来说，就是依照某种特定的规则来重新排列明文，也就是搅乱明文字母间的顺序。我们以下面的例子来说明。

STRIKE WHILE THE IRON IS HOT

此明文经过一个简单的换位之后，我们可以得到密文

TOH SI NORI EHT EKIRTS

以上的例子，其换位的规则就是将明文以相反的顺序重写。这里必须强调一点，此法只是改变字母之间原有的顺序而非变动字母原本的意义。也就是说“T”经过换位之后，仍然代表原来的字母“T”。只是原本所处的位置(在明文中)遭到变动而已，此点是非常重要的概念。

➢ **替换加密法(substitution)**

前面介绍的换位加密法，其基本精神是改变明文字母之间的相对位置，对于每一个字母而言，其意义不变。而替换加密法与此精神完全相反，对于明文的每一个字母不改变它的位置，只是将它以别的字母或符号代替。如：假设明文的字母集是大写的 26 个英文字母，而我们的替换方式如下：

原字母	A	B	C	D	F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
替代字母	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

也就是说，在密文中的 Z 代表文中 A，其他则依此类推。因此。若我们的明文为

FAITH　CAN　MOVE　MOUNTAINS

则相对的密文为(空白不计)

UZRGSXZMNLEVNLFMGZRMH

当然这个例子是一个非常简单的替换加密法，还有其它如旋转替换法、代换替换法等复杂一些的替换加密法。

（1） 美国数据加密标准(DES)

DES(Data Encryption　Standard)是在 1970 年代中期由美国 IBM 公司发展出来的，且被美国国家标准局公布为数据加密标准的一种分组加密法(Block Cipher)。直到今日，尽管 DES 已经经历了 20 个年头，但是在已知的公开文献中还是无法完全地、彻底地把 DES 给破解掉。

DES 属于分组加密法，而分组加密法就是对一定大小的明文或密文捉加密或解密动作。而在 DES 这个加密系统中，其每次加密或解密的分组大小均为 62 位，所以 DES 没有明文的扩展问题。就一般数据而言，无论是明文或密文，其数据大小通常都大于 64 位。这时只需要将明文/密文中每 64 位当成一个分组加以切割，再对每一个分组左加密或解密即可。但是对明文左分组切割时，可能最后一个分组会小于 64 位。这时，要在此分组之后附加“0”位，直到此分组大小成为 64 位为止。

另一方面，DES 所用的加密或解密密钥(Key)也是 64 位大小，但因其中有 8 个位是用来左奇偶校验，所以 64 位中真正起作用的只有 56 位。而 DES 加密与解密所用的算法除了子密钥的顺序不同之处，其他的部分则是完全相同的。DES 的总体方案如图 1.3 所示：

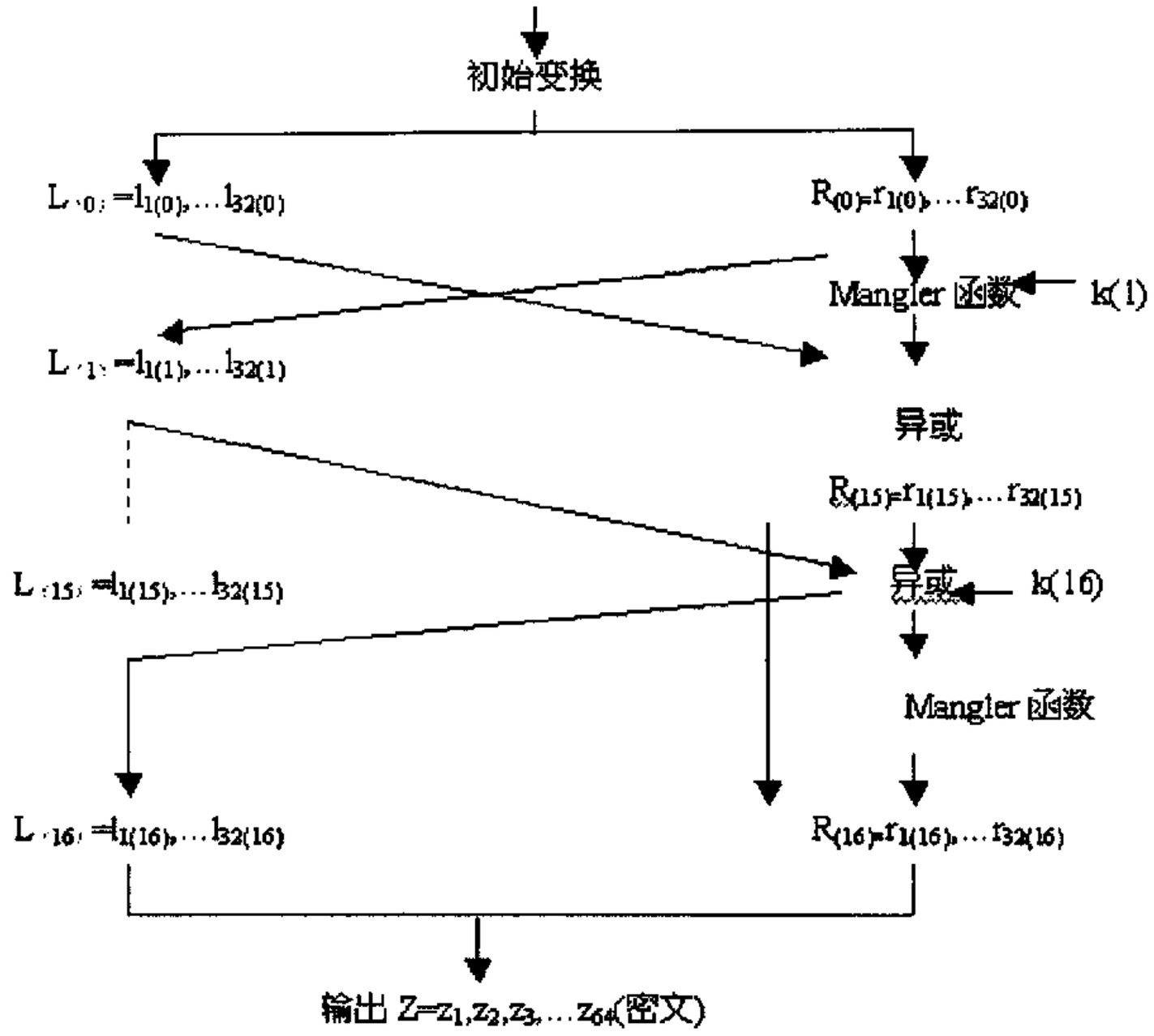

图 1.3　DES 加密算法的一般描述

DES 对 64 位的输入数据块进行 16 论编码。在每轮编码时，以 48 位的“每轮”密钥值由 56 位完整密钥得出来。在每轮的编码过程中，64 位数据和每轮密钥值被输入一个称为“S”的盒中，由一个编码函数对数据进行编码。另外，在每轮编码开始，过后以及每轮之间，64 位数码被以一种特别的方式置换(数位顺序被打乱)。在每一步处理中都要从 56 位的主密钥中的处一个唯一的轮次密钥。最后，输入的 64 位原始数据被转换成 64 位看起来被完全打乱了的输出数据，但可以用解密算法(实质上时加密过程的逆过程)将其转换回输入时的状态。当然，这个解密过程要使用加密数据使用的同样的密钥。

(2)　国际数据加密算法(IDEA)

IDEA(International Data Encryption Algorithm)是由瑞士联邦理工学院学院的 Xuejia Lai 和 James MASSEY 研制的一个对称分组密码。最初的版本发表在[LAI90]中。为了对于差分密码分析有更强的抗击能力的一个修改版本在[LA91]给出，而在[LA92]中则有对其更详细的描述。

IDEA 是最近几年提出的用来替代 DES 的许多算法中一种。许多人认为 DES 将要甚至已经达到了使用周期的最后阶段。从被采用的角度看，IDEA 是这些方案中最成功的一种。例如，IDEA 被 PGP 采用，仅此一点就可以使这个算法得到广泛使用。

IDEA 是利用 128 位的密钥对 64 位的明文分组，经过连续加密产生 64 位的密文分

组。IDEA 的加密原理如图 1.5 所示：

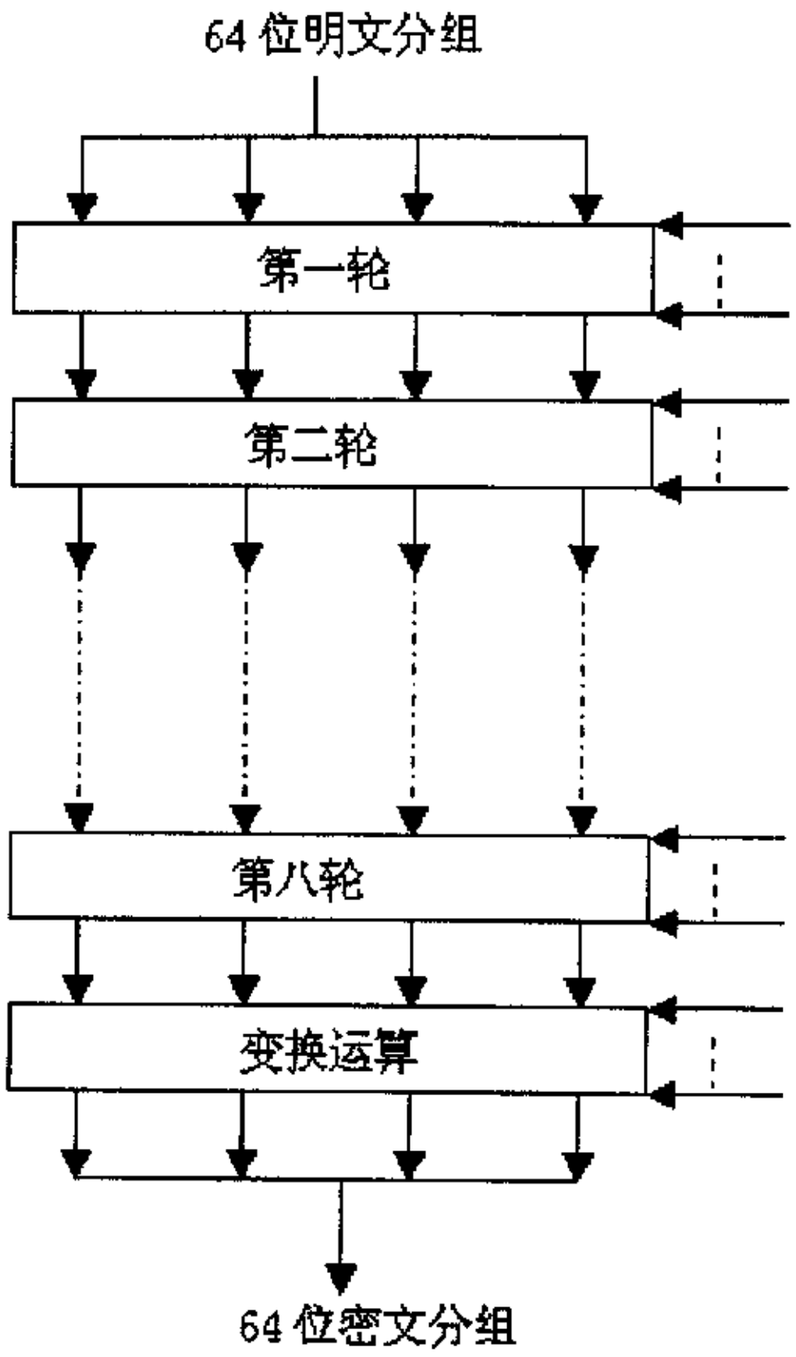

图 1.5 IDEA 加密流程

IDEA 方法包括 8 轮的重复运算，加上最后的输出变换运算(Tansformation)。64 位明文分组在每一轮中都是被分成 4 份，每份 16 位为一单元来处理。每一轮中有 6 个不同的子密钥参与作用。最后的输出运算用到了另外的 4 个子密钥。所以 IDEA 加密过程中共享到了 52 个子密钥。这些子密钥都是由一个 128 位的加密密钥产生的。

IDEA 的解密方法和加密方法有同样的结构，可以将密文分组当作输入而逐步恢复明文分组解密过程，与加密过程中唯一不同的是子密钥的生成方法。

（3） 快速数据加密法(FEAL)

FEAL(Fast Data Encipherment Algorithm)是一套类似美国 DES 的分组加密法。其最早期的版本是于 1987 年 Miyaguchi 两个人在欧洲密码学会议上所提出。FEAL 被提出的原意是着眼于当时的 DES 只用硬件去实现它，因此不适用于较小的系统，如个人计算机。而 FEAL 则强调其在每一轮的安全度都比 DES 高，所以使用较少的轮数就可以达到与 DES 采用 16 轮相同的安全度，如此一来，就比较适合用软件实现了。

从输入输出的观点看，FEAL 分组加密法与 DES 是相同的。即 FEAL 的加密或解密分组，FEAL 的密钥中并没有校验位，所以它的有效加密或解密密钥均是 64 个位，FEAL 的

加密或解密所用的算法也是同一个，且同 DES 一般，FEAL 解密是所用到的子密钥的顺序也是加密时子密钥被使用顺序的反转。另一个相同点是，FEAL 也只有用到“异或”的位运算。

但是 FEAL 加密算法的真正加密结构，则与 DES 有极大的差异。FEAL 完全没有使用置换函数来搅乱加密或解密过程中的数据，更没有如 DES 般具有神秘的 S-盒。FEAL 使用了异或(XOR)、旋转(Rotation)\加法与模(Modulus)运算。

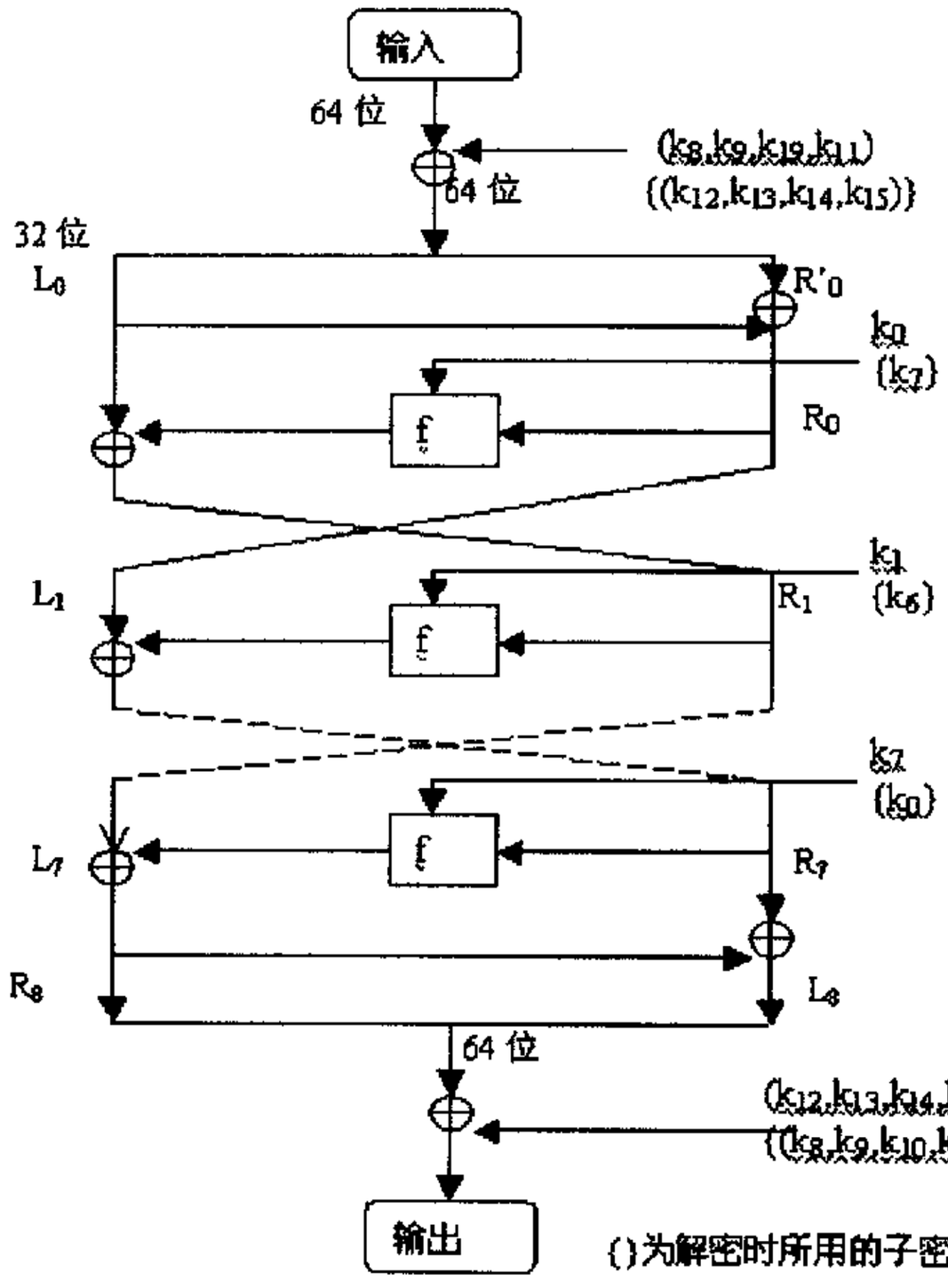

图 1.6 是 FEAL-8 的加密与解密结构

图 1.6 是 FEAL-8 的加密与解密过程的结构图，输入的 64 位分组，可能是明文或密文，视使用者加密或解密而定。首先，输入的 64 位分组数据与 4 个各为 16 位长度的子密钥 k_8，k_9，k_{10}，k_{11}所连接而成的 64 位子密钥组成做异或运算，在二分为各是 32 位的两部分，最后一轮所产生的 R8 则与 R7 做异或运算产生 L8，然后再将 R8 摆在高位。L8 放在低位，二者连接成 64 位的分组，再与子密钥组(k_{12}，k_{13}，k_{14}，k_{15})做异或运算，结果即为此次加密的输出。至于解密，其过程完全一样，不同之处只在于子密钥的使用顺序相反。

2 对称式密码系统——序列密码(Stream Cipher)

序列密码也称序列加密算法，即明文的位串与伪随机数产生器经过适当的运算耳得到密文。我们也可视序列加密算法为 1 个位的分组加密算法。序列加密算法的主要缺点在于若一个伪随机序列发生错误便会使整个密文发生错误，致使再解密过程中无法还原为明文。虽说是缺点，但也可视为优点，即可以做相同的明文位串可以有不同密文位串。图 1.7 是序列加密系统的简单结构图。

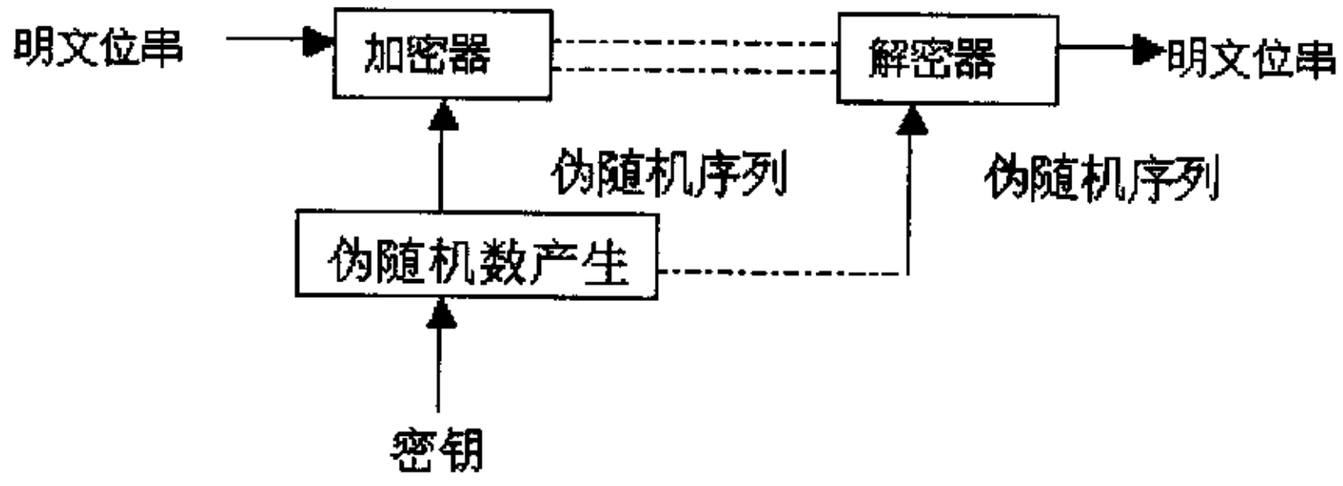

图 1.7 序列加密系统方框图

序列密码的主要原理是通过有限状态机产生性能优良的伪随机序列，使用该序列加密信息流(逐比特加密)得到密文序列。所以序列密码算法的安全性完全取决于它所产生的伪随机序列的好坏。衡量一个伪随机序列的好坏标准又有多种，比较通用的有著名的 Golamb 的三个条件：Rueppel 的线性复杂度随机走动条件、线性逼近以及产生该序列的布尔函数满足的相关免疫条件等。产生好的序列密码的主要途径之一是利用移位寄存器产生伪随机序列。

3 公钥密码体制

公钥密码体制的最大特点是加密和解密能力分开：一个密钥公开作为加密密钥；一个为用户专用，作为解密密钥，通信双方无须事先交换密钥就可进行保密通信。而要从公开的公钥或密文分析出明文或私钥，则是计算不可行的。若以公钥作为加密密钥，以用户私钥作为解密密钥，则可实现多个用户加密的消息只能由一个用户解读；反之，以用户私钥作为加密密钥而以公钥作为解密密钥，则可实现由一个用户加密的消息而使多个用户解读。前者用于保密通信，后者用于数字签名。这无疑为解决计算机通信网络中的安全提供了新的理论基础。

一个公钥密码体制可用图 1.8 表示，形式上，一个公钥密码体制是一个八元组(P,C,R,SK,PK,G,E,D)，其中明文空间 P 是可能明文的有限集，密文空间 C 是可能密文的有限集，私钥空间 SK 是可能私钥的有限集，公钥空间 PK 是可能公钥的有限集，E 和 D

分别是加密规则集和解密规则集，R 是随机参数空间，G 是一个密钥生成算法，满足对任意 r∈R, x∈SK, y∈PK, m∈Pc(7C，若(x,y)=G(r),c=Encry(y,m),则以下两个条件成立：由 x 容易计算出 y，但由 y 难以计算出 x; m=Decry(x,c),且由 c 难以计算出 m 和 x。

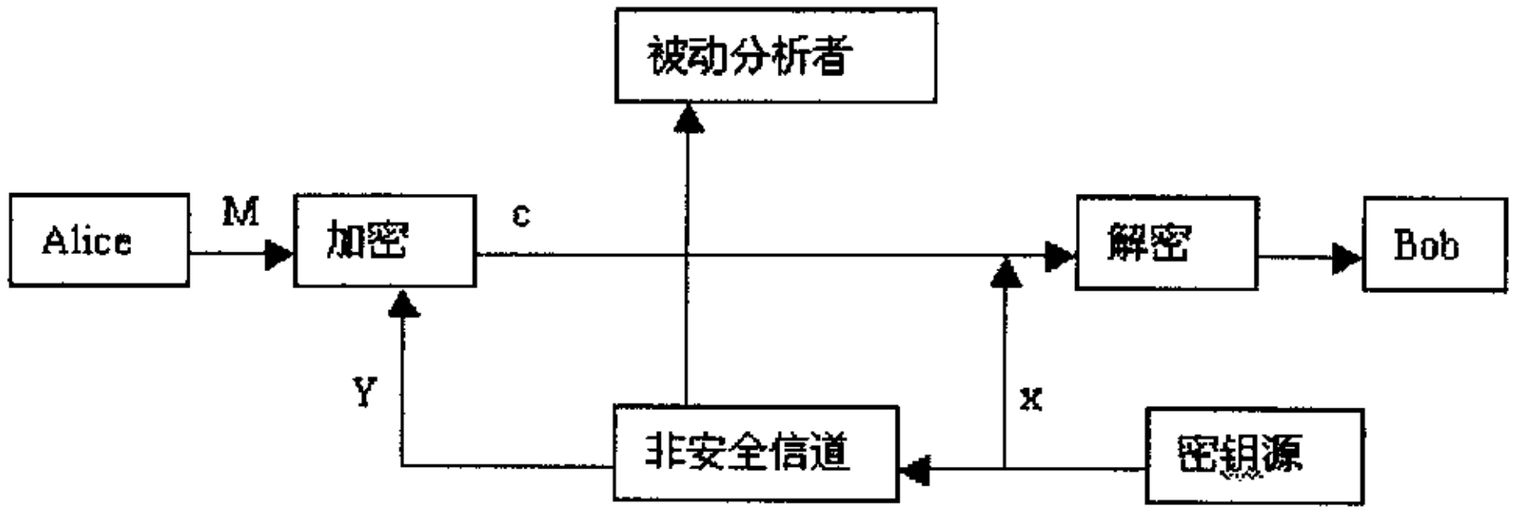

图 1.8　公钥密码体制模型

公钥密码体制的安全性主要取决于构造公钥算法所依赖的数学问题，一般要求加密函数具有单向性，即求逆困难性。因此，设计公钥体制的关键是先要寻求一个合适的单向函数。

定义 1.5.1(单向函数)：一个可逆函数 f:A→B，若它满足:(1)对所有 x ∈A,易于计算.t(x); (2)对几乎所有 x∈A 由 f(X)求 x 极为困难，以至于实际上无法做到，则称 f 为一单向(One-way)函数。

定义中的“极为困难”是指对现有的计算资源和算法而言。Massey 称此为视在困难性(Apparent Difficulty)，相应的函数称之为视在单向函数。以此来和本质困难性((Essentially Difficul Difficulty)相区分。单向函数是求逆困难的函数，而陷门单向函数(Trapdoor One-way Function)则是在不知道陷门信息下求逆困难的函数，而当知道陷门信息后求逆就易于实现。这是 Diffie 与 Hellman 引入的重要概念。

定义 1.5.2(陷门单向函数)：陷门单向函数是一类满足如下条件的单向函数:f:A→B,, z∈Z,Z 是陷门信息集。

(1)对所有:z∈Z，在给定 z 下容易找到一对算法 E_z 和 D_z 使对所有 x∈A，易于计算 f_z 及其逆，即:$f_z(x)= E_z(x)$，$D_z(f_z(x)) = x$;而且当给定 z 后容易找到一种算法 F_z，称 F_z 为可用消息集鉴别函数，对所有 x∈A,易于检查是否 x∈Az (Az 是 A 的子集)，Az 是可用明文集；

(2)对几乎所有 z∈Z，当只给定 E_z 和 F_z 时，对几乎所有 x∈A_z，很难以即实际上不可能从 $y=F_z(x)$计算出 x；

(3)对任一 z，集 A_z 必须是保密系统中明文集中的一个方便集，即易于实现明文到

它的映射。

第(1)条让我们注意识别 x 是否在允许的定义范围内;第(2)条半精确地定义了陷门单向函数为一单向函数:第(3)条则是公钥密码体制的默认条件。陷门单向函数的定义并没有说明这类函数是否存在，但 Diffie 与 Hellman 指出：一个单钥密码体制，如果其能抵抗选择明文攻击，就可规定一个陷门单向函数。以该单钥密码体制的密钥作为陷门信息，则相应的加密函数就是这类函数，这是构造公钥密码体制的途径。

- RSA 公钥密码体制

1976 年 Diffie 与 Hellman 发表的文章虽未提出一个具体的公钥密码体制，但大大推动了这方面的研究工作，1978 年，美国麻省理工学院的三位教授 R. L. Rivest, A. Shamir 和 L. Adleman 发现了一种用数论结果构造公钥体制的方法，这就是后来被广泛称为 RSA 体制的公钥密码体制。它既可用于加密，又可用于数字签名，易懂且易实现，是目前仍然安全并且最被广泛应用的一种公钥体制。国际上一些标准化组织 ISO, ITU 及 SWIFT 等均已接受 RSA 作为标准。在 Internet 中所采用的 PGP(Pretty Good Privacy)也将 RSA 作为传送会话密钥和数字签名的标准算法。RSA 算法的安全性基于大整数分解的困难性。

RSA 是一种分组密码，其中的明文和密文都是对于某个 n 的从 0 到 n-1 之间的整数。RSA 算法基于一个十分简单的数论事实：将两个大素数相乘十分容易，但是想分解他们的乘积却十分困难，因此可以将乘积公开作为加密密钥。RSA 的数学原理是：产生一对公开密钥和私密密钥的方法如下：假设 m 为要传送的报文，任取两个足够大的素数 p、q，使得 p 和 q 的乘积为 n，且大于 m；由 p 和 q 算出另一个数 z=(p-1)(q-1)，再取一个与 z 互素的大奇数 e(称为公开指数)。对于这个 e 值，可以找出另一个值 d 满足 e*d=1 mod z ，(n, e)和(n, d)这两组数分别为公开密钥和私密密钥，简称公钥和私钥。对于明文 m, 用公钥(n, e)加密得到密文 c；对于密文 c，用私钥(n, d)解密可得到明文 m。具体关系是：

公开密钥：n=pq,(p, q 分别为两个互异的大素数)，e 与(p-1)(q-1)互素；

私密密钥：de=1(mod(p-1)(q-1)), d 为私密密钥；

加密：$c=m^e(\mathrm{mod}\ n)$，其中 m 为明文，c 为密文；

解密：$m=c^d(\mathrm{mod}\ n)$

一般要求 p, q 为安全素数，计算出 e, n, d 后，两个素数 p 和 q 不再需要保留，并且不让任何人得到。因为 RSA 算法的安全性依赖于大数分解，所以 n 的长度至少大于 512 比特。目前无论软件还是硬件实现 RSA，速度都无法与对称密钥密码算法相比，因此因子分解问题是研究的关键。

从上述 RSA 公钥体制的算法可以看出，实现 RSA 关键问题之一是生成私密密钥对(e,d)的有效方法。目前比较成熟的方法有 Euclidean 算法，近几年一些研究人员又提出了一种避免使用 Euclidean 算法的 Derome 算法以及该方法的扩展算法。

- **Rabin 公钥密码系统**

Rabin 系统是 RSA 体制的变种算法。

密钥的产生：与 RSA 系统相同，接收者 B 任选两个大素数 p 及 q 并求出 n=pq；

公开密钥：n；

私密密钥：p 及 q；

加密：若 A 欲传送明文 m 给 B，0≤m<n，则 A 首先取得 B 的公开密钥 n，并求出密文 $C=m^e \bmod n$。A 将 C 传送给 B。

解密：B 收到密文后，C 在模 n 的四个平方根 m1、m2、m3 及 m4，则 m 必为此 4 个根之一。若明文 m 满足某种格式(如后面加时间或许多 0)，则由格式可判定此 4 个根何者为真正明文。Rabin 系统的安全性等价于因式分解，但它并不是比 RSA 更安全，例如我们以选择密文法攻击来说明。

破译者任选一随机数 x 并求出 $x^e=c \bmod n$，破译者将 C 送给 B 并要求 B 解密。若 B 解密获得一明文 m 送给破译者且 $m \neq \pm x \bmod n$(其概率为 1/2)，则破译者可获得 C 的 4 个平方根得两个独立解，破译者可解因子 n，故此系统可被破解。

为了防止上述攻击法，在使用 Rabin 系统时必须特别注意。利用时戳加在明文上以辨别明文，或明文具有某种格式。如最后 100 个位 0 等，均为不错的方法。

- **背包公钥体制**

背包公钥体制，1978 年，Merkle-Hellman 提出了一个基于组合论中背包问题的公钥密码体制。自此以后，背包体制因其加解密迅速而隐忍注目。所谓背包密码是指：

已知一个长度为 b 的背包，其厚度分别为 $a_1, a_2, a_3 \cdots a_n$ 的 n 个物品，假定这些物品的半径和背包相同，要求从这 n 个物品中选定若干个，正好装满这个背包，这个问题导致求 $x_i=0$ 或 1，$i=1,2,3\cdots,n$，使其满足

$$\sum_{i=1}^{n} a_i x_i = b$$

这里 $a_1, a_2, \cdots, a_n$ 和 b 都是整数。

取 $a=(a_1, a_2, \cdots, a_n)$ 作为加密密钥加以公开，其中 $a_1, a_2, \cdots, a_n$ 为整数，$m=m_1 m_2 \cdots m_n$ 是 n 位 0，1 明文字符串。利用公钥加密如下：

$$c= a_1 m_1 + a_2 m_2 + \cdots + a_n m_n$$

从密文求明文等价于解背包问题。

背包问题是著名的难题，至今还没有好的求解方法。若对 2n 种所有的可能性进行穷举搜索在实际上是不可能的。在计算复杂性理论上已经证明，背包问题属于 NP 完全类。基于这个事实，才有了背包公钥体制。

此外还有 Dilffie—Hellman 加密算法、椭圆曲线加密算法、概率加密算法、混沌加密算法等各种公钥密码，这里不再叙述。

第二章　RSA 算法快速实现研究

2.1　RSA 算法

RSA 是一个既能用于数据加密也能用于数字签名的算法。它易于理解和操作，也很流行。算法的名字以发明者的名字命名：Ron Rivest, Adi Shamir 和 Leonard Adleman。但 RSA 的安全性一直未能得到理论上的证明。它经历了各种攻击，至今未被完全攻破。

1　基本理论

RSA 的安全基于大数分解。其公开密钥和私人密钥是一对大素数（100 到 200 个进制数或更大）的函数。从一个公开密钥和密文中恢复出明文的难度等价于分解两个大素数之积。

RSA 算法的理论描述如下：

- 为了产生两个密钥，选取大素数 p 和 q。（保密）
- 计算乘积　　n=pq（公开）
- 计算　　$\Phi(n)=(p-1)(q-1)$　　（保密）
- 随机选取加密密钥 e，使得 e 和 $(p-1)(q-1)$ 互素
- 用欧几里德扩展算法计算解密密钥 d, 满足

$$ed \equiv 1 \bmod (p-1)(q-1) \tag{2.1.1}$$

则

$$d \equiv e^{-1} \bmod (p-1)(q-1) \tag{2.1.2}$$

e 和 n 是公开密钥，d 是私人密钥。而剩余的参数 p, q, $\Phi(n)$ 和 d 形成陷门。陷门信息显然不是由 4 个独立的参数组成的。例如，知道 p 时可以立即揭示出其他 3 个参数，因此不可以泄露其中任何一个参数。

2　安全性

RSA 公开密钥密码体制的安全性取决于从公开密钥 (n, e) 计算出私密密钥 (n, d) 的困难程度，而后者则等同于从 n 找出它的两个素因子 p 和 q。

若 n=pq 被因子分解，则 RSA 便被攻破。因为若 p, q 已知，则 $\Phi(n)=(p-1)(q-1)$ 便

可算出。解密密钥 d 对于 e 满足：de=1 mod z,故 d 便也很容易求得。因此 RSA 的安全性依赖于分解的困难性。

若 n 被分解成功，则 RSA 便被攻破，但还不能证明对 RSA 的攻击，其难度和分解 n 相当，故对 RSA 的攻击的困难不比大数分解更困难。若从求Φ(n)入手对 RSA 进行攻击，它的难度和分解 n 相当，而且没有分解 n 容易。但还是不能证明对 RSA 的攻击，其难度和分解 n 相当，也没有比因子分解 n 更好的方法。因此，寻求有效的因子分解算法是关键问题。至今为止，大数的分解问题仍然是极其困难的任务，因此 RSA 算法仍然被认为是最安全的公钥加密算法。

3　RSA 的缺点

(1) 产生密钥很困难。受到大素数产生技术的限制，因而难以做到一次一密，而且对于普通用户，如果没有专业知识，很难自己生成所需的密钥。而且随着大数分解技术的发展，密钥的长度还在增加，增加了密钥生成的困难性。

(2) 运算速度慢。RSA 运算代价很高，速度较慢，与对称密码算法相比，要慢几个数量级。硬件实现 RSA 时，RSA 比 DES 慢大约 1000 倍。最快的具有 512 位模数的 VLSI 硬件实现吞吐量为 64K/秒。也有一些实现 1024 位 RSA 的加密芯片，但这些实现都较慢。软件实现时，RSA 大约比 DES 慢 100 倍。这些数字会随着技术的发展而发生相应的变化，但 RSA 的速度将永远不会达到对称算法的速度[2]。

(3) 密钥的标准化。关于 RSA 密钥的格式和数据的编码方式还没有形成统一的标准，不利于密钥的传输、存储和使用。

2.2　对 RSA 的攻击

1　公共模数攻击

若系统中共有一个模数，只是不同的人拥有不同的 e 和 d，系统将是危险的。最普遍的情况是同一信息用不同的公钥加密，这些公钥共模而且互质，那么该信息无需私钥就可得到恢复[23]。

设 p 为信息明文，两个加密密钥为 e_1和 e_2，公共模数是 n，两个密文为：

$C1=p^{e_1} \bmod n$

$C2=p^{e_2} \bmod n$

假定用户 B 有一个 RSA 算法，模为 n，加密指数为 e_1。用户 C 也是一个具有同样算法且模为 n，加密指数为 e_2，e_1与 e_2互素。如果用户 A 想加密同一个明文 x 送给 B 和 C，

那么 A 先计算 $y_1=x^{e1} \bmod n$ 和 $y_2=x^{e2} \bmod n$，然后将 y_1 发送给 B，将 y_2 发送给 C。假定 D 截获了 y_1 和 y_2，那么 D 就可以按下述步骤计算出 x：

1）　计算 $c_1=e^{-1} \bmod e_2$

2）　计算 $c_2=(c_1e_1-1)/e_2$

3）　计算 $x=y_1^{c1}(y_1^{c2})^{-1} \bmod n$

这表明，无论密码系统多么“安全”，D 解密 A 发送的明文是可能的。在有些时候，需要一个可信中心选择一个 RSA 模 n，并对网上的每个用户分配不同的加、解密指数对 (e_i, d_i)。如果加密同一个明文送给 2 个或更多用户，由上可知，一个窃听者能够利用公开可获得的信息以很高的概率恢复出明文。解决的办法只有一个，那就是不要在不同的用户之间共享模数 n。

2　选择密文攻击

实际上，攻击利用的都是同一个弱点，即存在这样一个事实：乘幂保留了输入的乘法结构：

$(xm)^d \bmod n=x^d*m^d \bmod n$

这是公钥密码系统的最有用的特征——每个人都能使用公钥。解决这一问题，主要措施有两条：一条是采用好的公钥协议，保证工作过程中实体不对其他实体任意产生的信息解密，不对自己一无所知的信息签名；另一条是决不对陌生人送来的随机文档签名，签名时首先使用 Hash 函数对文档作 Hash 处理，或同时使用不同的签名算法。

3　RSA 的低加密指数攻击

在 RSA 加密和数字签名中，如果选取了较低的 e 值可以加快速度，但这样作是不安全的。如采用不同的 RSA 公开密钥及相同的 e 值，对 e(e+1)/2 个线性相关的消息加密，存在能攻击该系统的方法。如果消息比较短，或者消息不相关，就不存在这个问题。如果消息相同，那么 e 个消息就够了。对付办法就是 e 和 d 都取较大的值。

2.3　RSA 算法快速实现研究

RSA 算法是第一个能同时用于加密和数字签名的算法，也易于理解和操作。RSA 是被研究得最广泛的公钥算法，从提出到现在已近二十年，经历了各种攻击的考验，逐渐为人们接受，普遍认为是目前最优秀的公钥方案之一。但在实际中，RSA 并没有得到很好的应用，主要用来加密其他算法的密钥，而没有用来真正传输数据进行加密。主要问题是 RSA 的加密解密速度太慢，而影响其速度的主要因素在于大数的模幂乘运算效率较

低，所以提高大数模幂乘的效率一直是提高 RSA 算法速度的一个非常重要的课题。本节概要介绍几种可以提高 RSA 速度的快速算法，然后提出一种组合的快速实现方法，对 RSA 的速度问题的解决提出一点建议，最后给出一种用 DSP 快速实现 RSA 算法的方法。

1　大数乘幂算法分析

（1）BR 算法

在 RSA 算法中，主要的运算是大素数的乘幂运算。Rivest 等人在建立 RSA 公钥密码体制时，对于求 $X^e \bmod M$ 的算法，提出了用重复平方求模和相乘后求模的迭代方法来实现，即 BR(Binary　Represenations)方法。几乎所有的快速 RSA 算法的实现都是建立在 BR 算法基础上，先对 BR 算法进行简要的介绍。

首先将十进制指数 e 表示成二进制形式即为：

$$e=\sum_{i=0}^{n-1} e_i 2^i \qquad e_i \in \{0,1\},\ i=0,1,2\cdots\cdots k-1 \tag{2.3.1}$$

BR 算法是将幂乘余变成一系列平方剩余和乘同余的迭代，即加密和解密表达式 $Y \equiv x^e \bmod M$ 变成：

$$Y=(((((((1.x^{e_{n-1}})^2_M.x^{e_{n-2}})_M)^2_M\cdots x^{e_1})_M)^2_M.X^{e_0})_M \tag{2.3.2}$$

式中 $(.)_M$ 表示括弧中的数为 M 求模运算；$(.)^2_M$ 表示先对括号中的数求平方再对 M 求模。这样，在计算 $x^e \bmod M$ 时，先做依次平方运算，然后根据 e_i 的值再做一次乘法运算，以此来简化乘幂运算的复杂性。其算法如下：

(1) 设 $\bar{e}=e_{k-1}\,e_{k-2}..e_1e_0$；

(2) 置变量 c:=1；

(3) 对于 i=k-1, k-2, …, 1, 0 重复计算：

I　$c:=(c\times x^{e_i})^2 \bmod M$

II　若 $e_i=1$，则 $c:=c\times x \bmod M$

(4) c 即为所求。

上述算法中，影响运算速度的主要因素是平方和乘法运算的速度。设 e 的长度为比特，则此算法要做 k 次平方运算和 k 次乘法运算。若考虑 $e_i=0$ 的概率(一般为 1/2)则该算法要做的平均运算量为：

平方次数：k

乘法次数：k/2

BR 算法的迭代步数为：$2\log_2^e$。

2　改进的乘幂算法

BR 算法将指数 e 二进制化后，e 的比特序列长度变的很大(实际应用中往往超过 400 位)，不得不在计算过程中进行多次迭代，导致了其计算机效率低。因此，寻找最短的指数可能序列来使迭代次数减少是一个困难问题。基于这个思想，BR 算法的改进算法是将指数 e 进行 2^k 进制化(k=0, 1, 2, …)，使指数序列的长度减少，从而减少迭代次数，提高运算的效率。

因为任何一个整数 n 可唯一的表示为 $n=C_k^{m_k}+C_{k-1}^{m_{k-1}}+\ldots+C_1^{m}+C_0$

其中，m 是大于 1 的正整数，c_i 是整数，满足 $0\leqslant c_i<M<m$

所以当 $m=2^k$ 时，指数可表示为：$e=e_{n-1}(2^k)^{n-1}+e_{n-2}(2^k)^{n-2})+\cdots+e_1(2^k)+e_0$

其中 $e_i\in\{0, 1, 2, \cdots 2^{k-1}\}$

因此，幂乘余运算式 $x^e \bmod M$ 即可表示为下面的迭代式：

$$((((((1.x^{e_{n-1}})^{2^k}{}_{M}.x^{e_{n-2}})_M)^{2^k}{}_{M1}\cdots x^{e_1})_{M1})^{2^k}{}_{M1}.x^{e_0})_{M1} \qquad (2.3.3)$$

设指数 e 的二进制序列长度为 N；2^k 进制化后序列长度为 n，改进的 BR 算法一般分两步进行，如取 k=3 时算法如下：

(1)　设 $e=\sum_{i=0}^{n-1}8^i e_i$，$\bar{e}=(e_{n-1}e_{n-2}\cdots e_1e_0)_8$ 其中：$e_i\in\{0, 1, 2, \cdots, 7\}$

(2)　求出 x^2，x^3，$\cdots x^7 \pmod M$

(3)　置变量 c:=1;

(4)　对于 i= n-1, n-2, …1, 0 重复计算：

I. $c:=(c\times x^{e_i})^2 \bmod M$(平方)

II.　$c:=(c\times x^{e_i})^2 \bmod M$ (四次方)

III.　$c:=(c\times x^{e_i})^2 \bmod M$(八次方)

IV.　若 $e_i\neq 0$ 则 $c:=c\times x^{e_i} \bmod M$

(5)　c 即为所求。

分析上述算法，平均运算量为：平方次数 k

乘法次数 7*n/3*8+14=7n/24+14

其总的迭代步数为 2^k+n, 即 $2^k+\frac{n}{k}$。

3　基于乘同余对称特性的快速算法——SMM 算法

SMM(Symmdery　of　Modulo Mutiplicaction)算法是利用乘同余对称性来减少 RSA

加密计算中乘法和求模运算的一种快速算法。

令 i, j∈{0, 1, 2, …, m-1/2, m+1/2, ……m-1} 由于乘同余和平方剩余的对称性有：

$$(M-i)^2 \equiv i^2 (\mathrm{mod}\ \ M) \tag{2.3.4}$$

$$(M-i)(M-j) \equiv ij (\mathrm{mod}\ \ M) \tag{2.3.5}$$

$$i(M-j) \equiv (M-i)j \equiv ij (\mathrm{mod}\ \ M) \tag{2.3.6}$$

式 (2.3.1)的迭代运算实际只含两种基本运算，令 r_i 为第 i 步迭代后的中间结果，x 为待加密明文，则 r_i，$x \in \{0, 1, \cdots\cdots, m-1/2, m+1/2, \cdots\cdots, m-1\}$

两种基本运算分别式 $x_i^2 (\mathrm{mod}\ \ M)$ 和 $r_i \times x \bmod M$, $i=1, 2, \cdots\cdots, m$

在式(2.3.1)中的每步迭代计算种进行有条件代换即可使上述计算速度更快，代换原则是：如果 $r_i > (m-1/2)$ 或 $x > (m+1/2)$，则利用 $m-r_i$ 或 m-x 代替 r_i 或 x 进行平方剩余或乘同余计算，根据公式(2.3.4-2.3.6)其计算结果不变，但由于减少了乘法时间和求模运算量，使算法速度提高。

4　伪余数

在公式 $r=x^e \bmod M$ 中，r 是 M 的真余数，为了减少运行时间，对模进行放大。如放大一倍，即 $M_1=M*2$。这时对 x^e 进行求余有：$r_1 = x^e \bmod M_1$，其中 $0 < r_1 < M_1-1$ 这时 r_1 是 M_1 的真余数，但对 M 来说不一定是真余数，称 r_1 伪 M 的伪余数。因为此时有：

$r_1 - r = m \times j$(j 伪小于 0 的正整数)

因此 BR 算法中的迭代式子(2.3.3)变为：

$$((((((1.X^{e_{n-1}})^2_{M1}.X^{e_{n-2}})_{M1})^2_{M1}\cdots X^{e_1})_{M1})^2_{M1}.X^{e_0})_{M1} \tag{2.3.7}$$

这样使每一步的求余和平方剩余的时间减少。

2.4　组合快速 RSA 算法设计

RSA 公钥密码体制的快速算法研究一直受到密码学界的高度重视，RSA 的速度问题不解决就难以在实际应用中得到广泛应用。本文前面介绍了几种快速实现 RSA 的算法。他们分别从不同的角度对 RSA 算法中的乘幂运算速度进行提高。BR 算法的改进算法中，由于减少了指数序列的速度而减少了迭代次数。SMM 算法是在每步迭代计算中通过有条件代换，使部分乘数和被乘数的绝对值减小。而伪余数则是通过减少运行时间提高乘幂速度。

1　算法设计

如果将 BR 算法的改进算法的每步迭代时间减少，在此基础上，再利用伪余数的概

念，放大 M，使每步的求余时间减少。这样，可以进一步提高 RSA 算法的运算速度，即可以形成一种组合的快速算法。

算法设计思路如下：

1）利用伪余数的概念，将式子 $x^e \bmod M$ 中的 M 放大为 M_1，即 $M_1=M^2$；

2）计算出从 $x^e \bmod M_1$，到 $x^{2^k-1} \bmod M_1$ 的值，先放在一个表中保存；

3）由于放大了模，则 BR 算法的改进算法中的幂剩余迭代即可表示为下面的式子：

$$((((((1.x^{e_{n-1}})^{2^k}_{M1}.x^{e_{n-2}})_{M1})^{2^k}_{M1}\cdots x^{e_1})_{M1})^{2^k}_{M1}.x^{e_0})_{M1} \qquad (2.4.1)$$

其中 n 为指数 e 被 2k 进制化后的序列长度。

4）在式子(2.4.1)的每步迭代中，使用 SMM 方法进行代换。迭代运算所包含的两种各种基本运算分别为：$x_i^{2^k} \pmod{M_1}$ 和 $r_i \times x \bmod M_1$，其中 $i=1,2,3,\cdots,m_1$。如果第 i 步迭代后的中间结果 $r_i>(m_1-1)/2$ 或明文 $x>(m_1-1)/2$ 则利用 M_1-r_i 或 x 进行平方乘余或乘同余计算，根据乘同余对称的特性，其结果不变。

5）最后用上面的结果对 M 求模，即：

$$((((((((1.x^{e_{n-1}})^{2^k}_{M}.x^{e_{n-2}})_{M})^{2^k}_{M}\cdots x^{e_1})_{M})^{2^k}_{M}.x^{e_0})_{M})_{M} \qquad (2.4.2)$$

结果即为所求。

算法的 C 程序描述如下，分为两步：

step1:计算从 $x^0 \bmod M_1$ 到 $x_i^{2^k-1} \bmod M_1$ 的值

```
k=3; /*给定一个 k 值，则对指数进行 2k 进制化*/
r[0]=1;
j=1;
M1=M*2;
while(j<2k)
  { r[j]=r[j-1]*X mod M1;
    j++;
  }
```

Step1 需要作 2K 次迭代运算

step2:幂乘余和乘同余定理的计算

```
r1=1;
for(i=n-1;j>=0;i--)
  {for(j=1;j<=k;j++)
```

```
{if(r1>(m1-1)/2)    r1=M1-r1;
        r1=r12 mod M1;}
    if(r[ei]>(m1-1)/2)  R[e1]=M1-r[ei];
   r1=r1*r[ei] mod  M1;
  }
 r=r1 mod M;
```

r 即为所求；

step2 的迭代步骤为 n。

2　组合算法的速度分析

在组合算法中，使用了几种提高 RSA 算法的高效算法。一方面该算法不仅减小了部分乘余数和被乘余数的绝对值，使乘法速度加快，并且使迭代次数明显减少，所以组合算法比上面的任意算法速度更快。

随机的选取若干大整数进行实例实验，将比较结果列于表 2.1，由表 2.1 可以看出，新的组合算法使用结果与理论分析是一致的。

由于新算法是在 SMM 算法的基础上进一步减少迭代运算步数得到的，而 SMM 算法已经使每步迭代的求模运算量平均减少了 30%以上，而且采用伪余数可进一步减少运行时间，考虑到这一因素，实际获得的速度改善要大于 40%。

表 2.1 新算法与其他快速算法随机比较结果(K=3)

序号	E 的十进制序列长度	E 的二进制序列长度	E 的 2^k 进制序列长度	BR 迭代次数	改进的 BR 迭代次数	新算法所貉时间减少量
1	50	164	55	164	63	39%
2	60	196	66	196	74	40%
3	70	229	77	229	85	40%
4	80	261	87	261	95	40%
5	90	295	99	295	107	41%
6	100	328	110	328	118	45%
7	110	262	121	362	129	39%
8	120	394	132	394	140	40%

3　结论

本文提出的快速算法是在 BR 的改进算法和 SMM 算法的基础上加以组合得到的，由于 RSA 算法的模数大到十进制的 200 倍以上，指数大到 100 位以上，其运算速度非常慢，因而获得 40%以上的速度改善是相当可观的，关于新算法与幂乘余各参数之间的定量关

系，还有待进一步研究。

2.5　高效取余算法研究

在前面(2.3.1)的迭代算法中，每一步迭代都对 M 求模，以控制中间结果数的大小，每一步迭代最多需要 2 次乘法和 2 次求模，总共需要 $\log_2 e$ 次迭代。显然，在 $2\log_2 e$ 次求模运算中，后者是影响 RSA 速度的关键。下面给出两种高效求余的方法。

1　算法的基本思想如下：

(1)生成一个表 TAB[i](0≤i≤511)，每个表元素 TAB[i]的长度为 n+2 个字节，其内容为：

TAB[0]=0

TAB[i]=gN　(0≤i≤511，g 为一正整数)

满足 int(gN/T)=i-1，int((g+1)N/T)=i 这里 int 表示存在一个整数 s，x 满足 s≤x<s+1，则 int(x)=s。

(2)在 W 的高位加一个空字节 0，使的 W 的长度为 2n+1 字节。

(3)每次从 W 的高位开始取 n+2 个字节作为 W′，根据 W′ 的前两个字节的值 j 确定表中的元素 TAB[j]，将 W′ 减去 TAB[j]，由 TAB 表的生成规则可知，减的结果使得 W′ 的第一个字节为 0，第二个字节为 0 或 1，将 W′ 的内容写回 W，然后从第二个字节开始，重复上述过程，知道 W 剩余字节数为 n+1。此时，若 W 的最高字节为 1，再将 W 减去 N 或 2N，使得 W 最高字节为 0，若此时 W 的值还未大于 N，余下的值就是余数 R。

算法如下：

设：$W=W_{2n-1}W_{2n-2}\cdots W_0=\sum_{i=0}^{2n-1} 2^{8i}W_i$

$$N=N_{n-1}N_{n-2}\cdots N_0=\sum_{i=0}^{n-1} 2^{8i}N_i$$

且 $N\geqslant T/2$

则 $R=W \bmod N$

$$=R_{n-1}R_{n-2}\cdots R_0=\sum_{i=0}^{n-1} 2^{8i}R_i$$

生成表 TAB[i](0≤i≤511)

TAB[0]=0

g=1

对于 i=1, 2, …511 重复执行

如果 int(gN/T)=i-1 且 int((g+1)N/T)=i 则 TAB[i]=gN 否则 g=g+1

计算余数 R

① 再 W 的高位加一个空字节，使得 W 的长度为 2n+1，即

$W=W_{2n}W_{2n-1}\cdots W_1W_0$(其中 $W_{2n}=0$)

② 对于 i=n-1, n-2, …1, 0 重复执行

I　令 j= W_{n+i+1} W_{n+1}，则 0≤i≤511

II　将 W_{n+i+1} $W_{n+1}\cdots W_i$ 减去 TAB[j]，即 W_{n+i+1} $W_{n+1}\cdots W_i=W_{n+i+1}$ $W_{n+1}\cdots W_i$-TAB[j]

由 TAB[j]的生成规则可知，上述运算后：$W_{n+i+1}=0$，$W_{n+i}=\{0, 1\}$

③若 $W_n=1$，再将 $W_nW_{n-1}\cdots W_0$ 减去 N 或 2N，即：

$W_nW_{n-1}\cdots W_0= W_nW_{n-1}\cdots W_0-N$

$W_nW_{n-1}\cdots W_0= W_nW_{n-1}\cdots W_0-2N$

以保证 $W_n=0$

④若 $W_{n-1}W_{n-2}\cdots W_0\geqslant N$，则 $W_{n-1}W_{n-2}\cdots W_0= W_{n-1}W_{n-2}\cdots W_{0-N}$

以保证 $W_{n-1}W_{n-2}\cdots W_0<N$

⑤R= $W_{n-1}W_{n-2}\cdots W_0$

在该算法中由于要满足 N≥T/2，因此在实际应用中，若 N<T/2，必须同时将 N 与 W 在左移 d 位，以保证 N≥T/2，这样对于求的余数 R 必须右移 d 位，以得到真正的余数。

上述算法的正确性证明参见文献[15]，在此对称算法的速度作一分析。从算法上看，该算法只做加法和减法运算，没有乘法和除法运算，生成 TAB 表需要做 g 次 n+2 个字节的加法(g<1022, 因为 gN<511，而 n≥T/2)，每次取余运算最多要做 n+3 次 n+2 个字节的减法。由于 RSA 算法中模数 N 是确定的，对于 TAB 表只要生成一次即可，而在加解密运算中要用到多次取余运算，因此建表时间可以忽略。由此可见，该算法的运算速度要比除法求余的运算速度快。

2　递归余数和算法

该算法上是将 RSA 算法中的取余运算变成一系列的乘幂的和，即

$$(x)_N\equiv(\sum_{i=0}^{k-1}x_i2^i)_N\equiv((\sum_{i=0}^{k-1}x_i2^i)_N)_N,\quad x_i\in\{0,\ 1\}\tag{2.5.1}$$

令 $r_i\equiv(2^i)_N$　(2.5.2)

则 $(x)_N\equiv(\sum_{i=0}^{k-1}x_ir_i)_N$　(2.5.3)

由于 $x_i \in \{0, 1\}$，所以式(2.5.1)表示 X 求模等于 X 的非 0 位所对应的余数之和。由式(2.5.2)得：

$$r_i \equiv (2.2^{i+1})_M \equiv (2.(2^i)_M)_M \equiv (2^{i-1})_M \qquad (2.5.4)$$

说明第 i 个余数可由第(i-1)个余数迭代而来，如果用 m 表示 M 的位数，j 表示 M 的各素因子欧拉数的最小公倍数，即 p、q 为素数，M=pq，Φ(p) Φ(q)分别表示 p 和 q 的欧拉数，则 j=[Φ(p), Φ(q)]。

由费马小定理有，$2^{j+1} \equiv 2 \pmod{M}$说明 2 的各次幂对 M 求模以 j 为周期。因为在(2.5.1)的每步迭代中，求积的最大值不超过(2m-1)位，所以在求 2 (mod M)时最多只需计算(2m-1)个余数。

又因为 i<m-1 时，$2^j \pmod{M} = 2^j$，不需求余数。因此只需计算 $2^m - 2^{2m-2}$对 M 的(m-1)个余数。当 m≤j≤2m-1 时，则秩序计算 $2^m - 2^j$对 M 的(j-m+1)个余数。那么算法就是实现迭代生成(m-1)或者(j-m+1)个余数，建成迭代余数表。在进行求模运算时，只需查出与被模数非 0 位相对应的余数并求和，如果和超过了 M，重复次方法，知道结果 M 的完全剩余系里为止。

算法如下：

令 $M' = M(M_{k-1}M_{k-2}\cdots M_1M_0)_2$, $x' = (x_{n-1}x_{n-2}\cdots x_1x_0)$

(1)　n:=m-3

(2)　建立余数表:

对于 i=0, 2, …n-1 重复计算 $A[i] \equiv 2^{m+1} \pmod{M}$

(3)　当 x′ >M′ 对于 i=0, 1, 2…n-1

用 X_{m+i} 去查余数表，然后与 $X_mX_{m-1}\cdots X_1X_0$ 相加

(4)　X 即为所求

需要说明的是，用本算法实现 RSA 算法虽然速度较快，但使用于分组较多的情况。由于事先建立余数表需要耗费一定的时间，当信息分组较多时，分摊在每个分组的时间可以忽略不计，仅当模数位费马数时，由于不需要建立余数表，且每步迭代只需一次加法，使用于任何情况。

3　Montgomery 算法

式子(2.3.1)得每步迭代实质都是模乘,包括乘法和求模两种基本运算。

$$\begin{cases} X = x_{n-1}x_{n-2}\cdots x_1x_0 \\ Y = y_{n-1}y_{n-2}\cdots y_1y_0 \qquad x_i、m_i\{0,1\},\ i=0,1,\cdots n-1,\ m_0=1 \end{cases} \qquad (2.5.5)$$

$$Z=z_{n-1}z_{n-2}\cdots z_i z_0$$

式子 X、Y、M 分别为 n 位二进制被乘数、乘数和奇数模，且 X<M, Y<M。 Montgomery 模乘算法采用模右移的方法避免了求模运算中费时的除法。基-2 的 Montgomery 算法 $MM(X、Y、M)=XY2^{-(n+2)} \bmod M$ 步骤如下：

（1） $y_0=0$

（2） 对 i=0, 1, 2, . . . , n，计算

$$q_i=y_i \bmod 2 \tag{2.5.6}$$

$$s_{i+1}=(s_i+q_iM)/2+y_ix \tag{2.5.7}$$

（3） 输出 $S=s_{i+1}$。

用 Montgomery 算法计算式(2.5.2)的每步迭代，则可以构造从左到右的二进制幂剩余算法(基于 Montgomery 算法的 BR 算法)，其步骤如下：

（1） 预计算 $\overline{X} \equiv X2^{n+2} \bmod M$;

（2） Y= $\overline{X}$, i=n-1。若 $e_i=0$, i=i-1(找到 E 中第一个 1)；

（3） 计算 Y=MM(Y, Y, M)(自乘得到 $Y \equiv Y^2 2^{-(n+2)} \bmod M$)；

（4） 若 $e_i=1$, 计算 Y=MM(Y, $\overline{X}$, M)(乘 X 得到 $Y \equiv Y\overline{X}\,2^{-(n+2)} \bmod M$)；

（5） i=i-1, 若 i≥0 返回到步骤(3)(得到 $Y \equiv X^E 2^{n+2} \bmod M$)；

（6） 计算 Y=MM(1, Y, M)，得到 Y≡Y-M;

快速算法分为以下三部分：1)预计算 $\overline{X} \equiv X2^{n+2} \bmod M$；2)幂剩余计算，由 BR 迭代算法完成，每步迭代运用 Montgomery 算法；3)后处理，使最终结果 Y<M。用 Montgomery 算法映射、并由脉动阵列实现的 512 比特 RSA 算法，在 125MHz 时钟下速率可达 164kbps。

2.6 用 DSP 快速实现 RSA 算法研究

自从 RSA 算法提出以来，人们已经研究出多种方法用硬件和软件来实现 RSA 算法的快速乘幂取余算法。但其中用软件实现的比较多，硬件实现的比较少。而就 RSA 算法的特性来看，用硬件实现要比用软件实现速度快。但是专门的 VLSI 芯片一般花费较大，而且通常它的微处理器不适合用来处理这种精确的算法。数字信号处理器(DSP)则是一个理想的选择，它具有强大的运算能力，速度快，而且具有很好的灵活性，特别适合完成高复杂度的算法，还有一个优点是几乎每台 PC 机都有一个来支持 DSP 应用，这就大大简少了一些必要的或额外的开销。

现在有关用 DSP 硬件实现 RSA 的报道还是不多的。这里给出一种用 DSP 实现 RSA 算法的方法，使用的是最新的 TM320C620EVM 定点 DSP 芯片和 TI 公司的集成 DSP 开发工具

套件。理论研究结果表明，用 DSP 硬件加密的效果比在软件环境中实现 RSA 算法要快。

1 算法设计

（1） 硬件

主机是一台普通的奔腾 PC 机，主频 400MHZ，主板商有可用于即插即用 DSP 板的标准 PCI 扩展槽。RSA 的核心操作是指数和取模运算，即 $X^e \bmod M$, 其中 e 和 M 都是大整数，实现 RSA 的运算要进行多次乘法运算和减法运算。德州仪器厂的 TMS320C6x 顶点 DSP，CPU 内核中八个功能单元可以完全并行运行，功能单元执行逻辑、位移、乘法、加法和数据寻址等操作。TMS320C6x 系列芯片的体系结构采用长指令字方式，单指令字长为 32 位，取指令、指令分配和指令译码单元每周期可以从程序存储器到功能单元传递 8 条指令，这 8 条指令组成一个指令包，总字长为 8×32=256 位，芯片内部设置了专门的指令分配模块，可以将 256 位指令包分配到 8 个功能单元中，并由 8 个功能单元并行运行，TMS320C6x 芯片的最高时钟频率可以达到 200MHz，8 个功能单元同时运行时，该芯片处理能力高达 1600MHz。TMS320C6x 芯片的内存储器总容量为 1M 位，其中 2K×256 位用于数据内存和数据 cache，用户可以访问 8 位、16 位和 32 位的数据。它还提供了一个满足 PCI Local BUS 协议的兼容接口，可以使主机直接和 DSP 板上的 JTAG 控制器、DSP 主机接口(PHI)和板上的控制/状态寄存器进行通信。

（2）软件

C6X EVM 软件包括株距支持软件和 DSP 支持软件。C6X EVM 提供的株距支持软件包括 WIN32 主机应用程序和库函数。这些软件运行在操作系统 windoes95 或 windowsNT4.0 的 Intel 兼容 PC 机上。支持 TM320C6X EVM 板的开发软件 CCS 使一个集编辑、编译、连接、实时调试和分析应用程序于一体的开发软件包。CCS 能够加快用户的开发进度，增强用户的应用程序的性能。

（3）RSA 算法中素数的生成

从安全角度，素数 p、q 应该足够大的，一般来说大于 10^{100}。因此，选择和处理这两个大素数构成了 DSP 实现的主要困难，对于素数的生成本文提出了一个很有吸引力的算法。

根据 Wilson 定理，p 位素数的充要条件为：$(p-1)! \equiv -1 \bmod p$

而事实上，p 为大素数的情况下，由于计算量过大，在图灵机结构的计算机上使用 Wilson 来检测素数是没有实际价值的[10]。一个简化的方法是采用 p 为大素数的必要条件：

若 p 为素数，则对于 X 的所有整数值，满足：$X^{(p-1)}!\equiv 1 \mod p$

取 X 为连续的三到五个整数，若满足条件，p 不是素数的概率极小。在实际应用中，若在后续过程中判断 p 正好不是素数的情况下，可重新搜寻另一个后选素数，而不是影响速度性能。

（4）素数的选择

A. 敌手有试图发给 A 的公钥 e、模数 n，若想从密文 c 中得到明文 m，敌手首先因式分解 n，然后计算 e 和 d，象 A 一样生成密钥对，一旦 d 被推算出来，敌手就能解密所有的想发给 A 的密文。

B. 如果选取了较低的加/解密指数 e 和 d 可以加快速度，但这是不安全的，建议加密解密密钥 e 和 d 和 n 的长度大体相同。

C. 给明文加上垃圾信息

D. 为了避免椭圆曲线因式分解算法攻击的方法是 p 和 q 应当有相同的位数，并且足够大。另外，p-q 不能太小。如果 p-q 很小，那么 $p\approx q$，因此 $p=\sqrt{n}$。

基于以上原因，专家建议 p、q 应该是强素数。强素数就是某些满足某些特性的素数，使得用某些特殊的因子分解方式对它们的乘积 n 进行分解很困难；

A. p-1 和 q-1 的最大公因子应该较小。

B. p-1 和 q-1 都应有大的素因子，分别记作 p′ 和 q′。

C. p′ -1 和 q′ -1 都应有大的素因子。

D. p+1 和 q+1 都应有大的素因子。

E. (p-1)/2 和(q-1)/2 都应该是素数。

（5）生成私密密钥于公开密钥

RSA 运算是多精度整数运算，操作数一般是 512 比特或更长，因此几乎所有的操作数，包括 n、p、q、e 和 d，都需要保存在以 16 比特为单位的数组中。RSA 算法所需要的所有运算都要专门开发。如前所述，RSA 的核心操作，也是最耗时的操作时模乘运算，因此，开发快速指数运算和取余运算是问题的关键。

我们可以根据前面的快速算法乘算法和快速取余算法进行组合，找出一种速度快又适合 DSP 中开发的算法来进行模幂运算。

为了提高算法的运算速度，在计算 $m=c^d \mod n$ 时，不是直接计算，而是先计算 $m1=c^{dp} \mod n$ 和 $m2=c^{dq} \mod n$ ($d_p=d^m \mod (p-1)$, $d_q=d^m \mod (q-1)$)，然后再使用 Garner 算法。这个过程看起来似乎比直接计算更复杂，但实际上变的简单了，因为幂变小了，所以时间减小了，这样可以缩短解密的时间。

2 结论

RSA 公钥秘密体制的快速算法研究一直受到密码学界的高度重视，RSA 的效率问题不解决就难以实际应用。近年来，有关用 DSP 来实现 RSA 算法的论文也不断涌现，研究工作也取得了一定的进展，而且现在用 DSP 来实现 RSA 算法问题也吸引了国外众多学者的关注。本节借鉴了若干用 DSP 快速实现 RSA 问题的最新研究成果，并在此基础上进行改进和提出新的实现方案。这种方式力图通过强素数的概念来增强 RSA 算法的安全性，并用软件硬化的手段来增强算法似的速度性能。这对于 RSA 公钥密码体制的应用很有实际意义，同时也希望给提高 RSA 速度问题的解决提供一点建议。

第三章　数据加密传输系统的 COM 组件实现

随着计算机技术的迅速发展，特别是网络技术的发展，信息系统计算体系也从单机体系发展为 C/S 模式的分布式系统。但 C/S 模式不能适应不同厂家的网络产品、硬件平台、网络协议。于是 C/S 模式引入中间件发展为 C/S/S 模式(Client/Server/Server)。中间件主要作用是用来评比网络硬件平台的差异性和操作系统与网络协议的异构性，使应用软件比较平滑地运行于不同平台上。中间件这种分布式对象被称为组件(Component)，它是一些独立的代码的封装体，提供一定的服务。微软的分布式对象技术方案是 windowsDNA 体系(WINDOWS distributed Internet Applications Architecture)。其中间件实现采用了基于 COM 规范的组件，即 COM 组件。

3.1　COM 组件的含义及其分类

开发 COM 的目的是为了使应用程序更易于定制、更为灵活。最初的目标是提供对对象链接及嵌入的支持，其基本的想法是以文档为中心的观点。在这种观点下,用户可以从一个字处理程序中编辑电子表格。对象链接及嵌入的 Microsoft 版本被称作是 OLE。

COM 比较小、速度快、灵活。因此，OLE 的第二版使用了 COM 作为它的基础技术。但 OLE 作为开发出来的第一个 COM 系统，它并不能很好地说明 COM 的功能与作用。并且由于实现上的原因，OLE 显然比较庞大、缓慢，并且难于编码，但这并不是由于使用 COM 造成的。相反，正是在 OLE2 的发展过程中，建立了一个精确定义的模型 COM，该模型具有可扩充的特点，这个特点使得可以在不改变基础部分的前提下，按结构来引入新的组件服务。

总的说来，微软的第一代组件是 DLL，它提供了 C 语言的函数调用接口:第二代组件是 Windows 开放系统体系结构,提供了一些重要的服务如 ODBC 和 Windows Sockets,但其仍然提供的是 C 的函数调用接口。第三代就是我们所说的 COM，它允许不同程序语言实现的程序之间交互操作.在 Windows NT 中又发展了分布 COM (DCOM)技术，DCOM 把 COM 技术扩展到了整个网络。现在微软又在发展它的下一代组件结构 COM+，COM+简单的说就是把 MTS(Microsoft Transaction Server)集成到 COM 中，从而为 COM 调用提供了一种新的基于 MSMQ 的通信方法。COM+提供了无缝连接系统，在该系统下，创建服务器应用就像实现客户应用一样简单,最重要的是 COM+还提供了可以用来建立高伸缩性应用

的多种服务，这些服务包括事物处理、安全、并行处理、消息队列等。但要注意到 COM+虽然在 COM 的基础上增加了属性配置和强有力的运行服务，但是要想加入 COM+服务，就必须首先实现 COM 组件。所以用一句话来概括，可以把 COM+理解为建立在 COM 基础之上的更丰富的服务提供者。

3.2　COM 编程中的几个关键概念

1　组件和接口

COM 组件是以 Win32 动态链接库(DLL)或可执行文件(EXE)的形式发布的可执行代码组成的。遵循 COM 规范编写的组件将能够满足组件架构的所有要求:动态链接性、封装性。COM 使用 DLL 将组件动态链接起来，所以它满足动态链接性。

COM 的封装性体现在:

（1）　语言封装性，即 COM 组件是完全与语言无关的，任何过程性语言都可以用来开发组件，并且可以修改任何一种语言使它能够使用组件;

（2）　COM 组件可以以二进制形式发布，从而做到了二进制代码级复用;

（3）　版本级封装性，COM 组件提供了一种实现同一组件的不同版本的标准方法;

（4）　网络封装性，COM 组件可以透明地处理在网络上被从新分配位置，对远程机器的组件同对本地机器上的组件处理方式没有什么差别。

接口是组件客户和组件之间的一种通信协议，这种协议具有强烈的类型约束，组件客户通过这种协议可以向组件请求服务。从这个意义上来说，接口只不过是一个函数集合声明，客户使用这个函数来访问组件提供的服务。

2　COM 对象

对象是类的实例，COM 对象是 COM 组件提供给客户以对象形式封装起来的实体，COM 对象是客户程序与组件程序通信的主体。类似于 C++语言中类(Cl.Gss)的概念，COM 对象也包括属性也称为数据成员)和方法(也称为成员函数)，COM 对象的属性反映了对象的存在和状态，即区别于其他对象的要素，COM 对象的方法就是对象提供给外界服务的接口，客户必须通过接口才能获得对象的服务。对于 COM 对象来说，接口是它与外界进行交互的唯一途径。COM 对象的区分是依靠 128 位的全局唯一标识符随机数来标识的，称为类型标识符，用类型标识符标识对象可以保证在全球范围内的唯一性。

3　COM 接口

COM 接口有两个含义。首先，它是一组可以调用的函数。由此客户可以让该对象做

某些事情；其次，也是更重要的，接口是组件及其客户程序之间的协议，也就是说，接口不但定义了可用什么函数，也定义了当调用这些函数时对象要做什么。接口提供了不同对象之间的一种连接，每个接口被分配一个 IID，每个 IID 也是一个长度为 128 位的 GUID 结构。COM 规范使用 1DL(interface description language，接口描述语言)来定义接口。接口内存模型如图 1 所示。

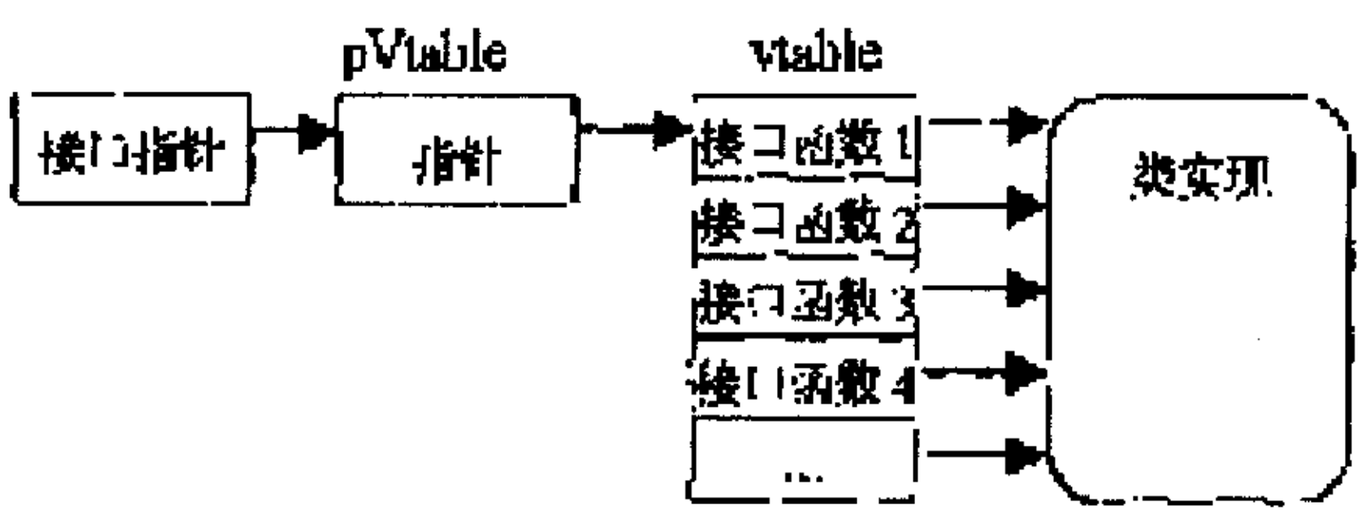

图 5.1　接口内存模型

由图 5.1 可知客户程序用一个指向接口数据结构的指针球调用接口成员函数。接口指针实际上又指向另一个指针，这第二个指针指向一组函数，称为接口函数表，接口函数表中每一项为 4 个字节长的函数指针，每个函数指针与对象的具体实现相连接。通过这种方式客户只要获得了接口指针就可以调用 COM 对象的所有功能。通常把接口函数表称为。对具体接口来说，它的虚函数表是确定的，因此接口成员函数和成员函数在虚函数表中的顺序是不变的；对于每个成员函数来说，其参数和返回值也是确定的。

4　IUnknown 接口

根据 COM 规范，COM 定义的每个接口都是 IUnknown 接口直接或间接继承过来。IUnknown 接口提供两个非常重要的特性:生存期控制和接口查询。IUnknown 引入了“引用计数”方法可以有效地控制对象的生存周期。

IUnknown 接口定义如下:

```
Interface IUidaiown
{
HRESULT QueryIiiterface([in] REFIID aid,[out] Void **ppv);
  ULONG AddRef(void);
  ULONG Release(void);
}
```

IUnknown 接口包括 3 个成员函数:QueryInterface，AddRef，Release。函数 QueryInterface 用于查询 COM 对象的其他接口指针。函数 AddRe 接口 Release 用于对引

用计数进行操作。当客户得到一个指向该对象的接口指针时，引用计数增加 1;当客户用完了该接口指针后，引用计数减 1。当引用计数减少到 0 时，COM 对象就把它自己从内存中清除。当客户程序对一个接口指针进行复制，则引用计数也应该增加。IUnknown 接口成员函数 AddRef-Welease 分别完成引用计数的增 1 和减 1 的操作。

3.3　COM 组件实现原理与特性

1　类厂

类厂是 COM 类的工厂，确切些，应该是“对象厂”n1。因为类厂是 COM 对象的生产基地。COM 库通过类厂创建 COM 对象，对应每一个 COM 类，有一个类厂专门用于 COM 类的对象创建操作。类厂本身也是一个 COM 对象，它支持一个特殊的接口 IcclassFactory 源型如下:

```
Class ICalssFactory:public IUnlatown
    {
    virtual HRESULTto stdcall Createlttstance(Iunlutown
      *pUnlatowttouter,const IID&lld,void "* ppv)=0;
    virtual HRESULT stdcall LockServer(Bool block)--0;
    }
```

接口 IclassFactory 成员函数 CreateInstance 用于创建 COM 对象，因为每个类厂只针对特定的 COM 类对象，所以 CreateInstance 成员函数知道创建什么样的 COM 对象。CreateInstance 的参数中，pUnknownOuter 用于对象类被聚合的情形，iid 为对象创建完成后客户得到的接口 IID, ppv 用于存放返回的接口指针。IcclassFactor 的另一个成员函数 LockServer 用于组件的生存周期。COM 规定，每一个 COM 对象类应该有一个相应的类厂对象，如果一个组件实现一个 COM，则有一个类厂;如果一个组件程序实现多个 COM 对象类，则相应有多个类厂。

2　COM 库

COM 库是组件对象实现所必须的一些建立在二进制级的系统级代码。COM 库提供以下服务:①提供少量的 API 函数实现客户服务器端应用的创建过程;② COM 库通过注册表查找本地服务器即 EXE 程序及程序名与 CLSID 的转换;③提供一种标准的内存控制方法，使应用控制进程中的内存分配。在 COM 库中，有 3 个 API 函数可用于组件 COM 对象的创建，COM 对象的创建是客户程序、COM 库和组件程序 3 者交互的结果。具体创建过

程如图 5.2 所示：

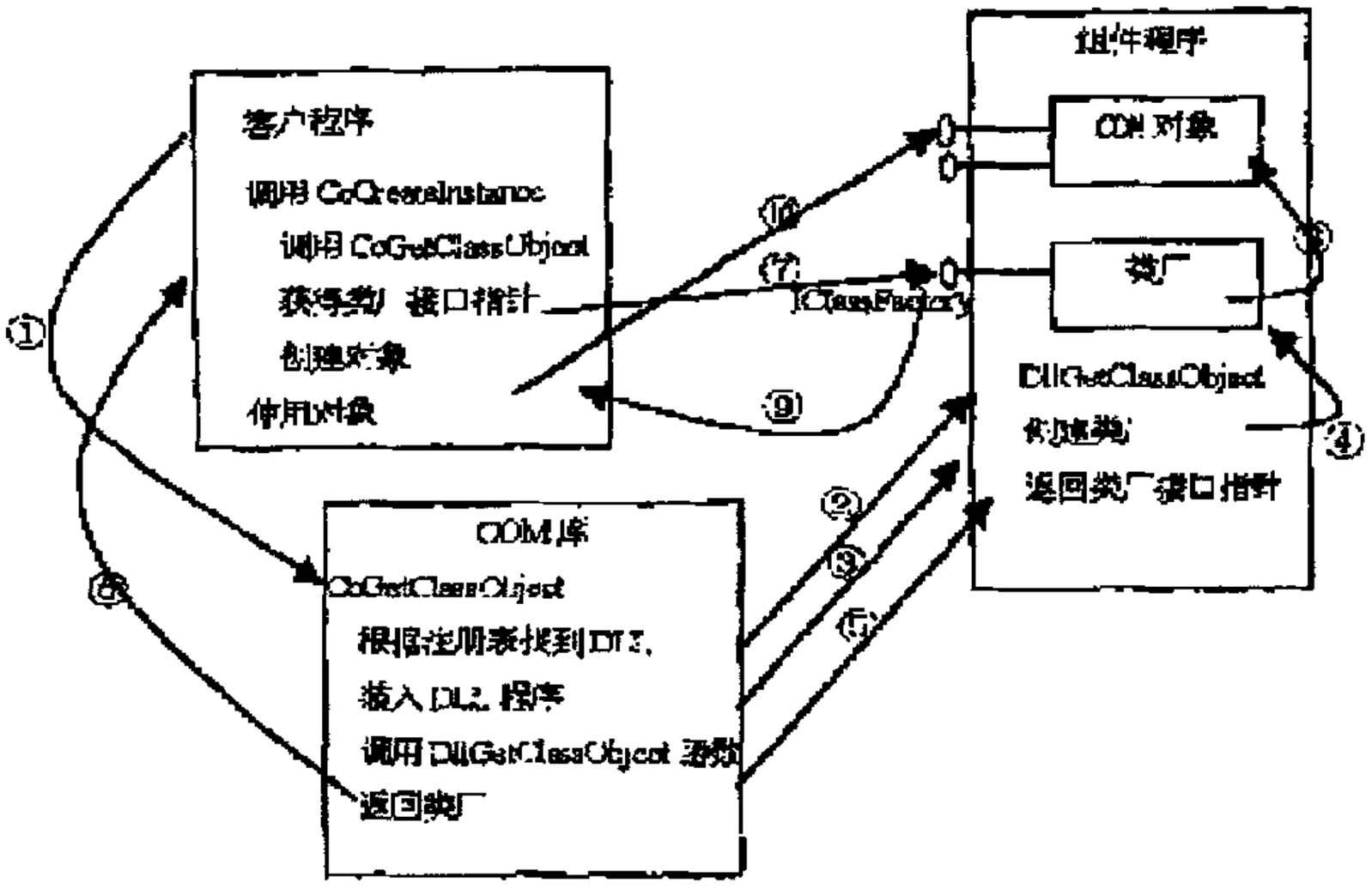

图 5.2 组件对象创建过程

① CoCreateInstance 调用 CoGetClass 函数；

② COM 库找到 DLL 程序并进入进程；

③ 调用 DllGetClassObject 函数；

④ DllGetClassObject 函数创建类厂；

⑤ DllGetClassObject 函数把类厂接口指针返回给 CoGetClassOjbectiA 数；

⑥ CoGetClassObject 函数把类厂接口指针返回给 CoCreateInstance 函数；

⑦ CoCreateInstance 函数得到类厂后，调用类厂的对象创建函数；

⑧ 类厂创建 COM 对象；

⑨ 类厂把 COM 对象返回给 CoCreateInstance 函数，得到 COW 对象接口指针；

⑩ 客户直接调用 COM 对象。

3　COM 的特性

COM 除了具有基本的面向对象特性如封装性、多态性等外还具有如下关键特性：

（1）　语言无关性：COM 标准不是面向源代码级的标准，而是采用一种基于二进制代码级的标准。COM 规范的定义不依赖于特定的语言，编写组件所使用的语言与编写客户使用的语言可以不同，只要他们能够生成符合 COM 规范的可执行代码即可。C 一语言定义的 COM 对象可以在 VB，Java，Delphi 等语言环境中调用，反之亦然。

（2）　进程透明性：进程透明性是指无论是进程内组件还是进程外组件，其创建过程和接口调用对用户是透明的。COM 通过列集解决进程透明性问题。列集过程分为两种

方式：自定义列集和标准列集。

（3） 可重用性：COM 的重用包括两种方式:包容方式和聚合方式。包容是指外部的对象作为内部对象的客户，外部的对象调用内部对象的方法来实现自己的功能，而这些方法对客户来说是不可见的。外部对象利用内部对象的界面来实现它自己的部分界面。聚合是指外部对象将内部对象的界面展露出来，就像它自己实现的界面一样。当客户请求该界面时，外部对象直接返回内部对象的界面指针，客户仅知道外部对象而不必关心内部对象。

（4） 安全特性:COM 提供的安全机制类似于 Windows NT 的鉴定服务机制。包括两种类型的安全性：第一种激活安全性，它包括 COM 对象如何被安全地启动、客户如何与对象建立连接，以及如何保护公共的资源；第二种类型被称为调用安全性，这是指已经建立连接的基础上，客户调用组件程序的安全保护功能。

3.4 基于 COM 的加密组件的实现

1 ATL 简介

在严格意义上讲，COM 组件的接口是一个包含一个函数指针数组的内存结构，因此用 Microsoft Visual C++6.0 进行开发是理想的选择方案。

ATL(Activae Template Library)是 Visual C++提供的一套基于模板的 C++类库，利用这些模板类，我们可以建立小巧、快速的 COM 组件程序。所以说，ATL 主要是针对COM应用开发的,它内部的模板类COM的一些基本特征,比如一些基本COM接口 Iunknown、iclassFactory、Idispatch 等，也支持 COM 的一些高级特性，如双接口、连接点、Activex 控制等，并且 ATL 建立的 COM 对象支持套件线程和自由线程两种模型。

ATL 提供了两个基本模板类:CcomObjectRootEx 和 CcomCoClass,每一个标准的 COM 对象都必须继承于这个模板。CcomObjectRootEx 模板实现了 Iunknown 成员方法包括对引用计数的处理以及对接口请求的处理，它也支持对象被聚合的情形；CcomCoClass 模板类为对象定义了缺省的类厂，并且支持聚合模型。为了支持 COM 对象的创建操作，ATL 还定义了对象映射表。对象映射表是一张全局表，它被放 ATL 主源文件(与工程同名)中，对象映射表中没没一项指定了对象的 CLSID 及它的一些创建函数，包括注册表操作函数指针以及类厂的接口指针等信息，全局的 DllGetClassObject 函数利用对象映射表获取类厂对象。ATL 定义的 COM 对象类，还不是最终 COM 对象的 C++对象的创建过程很复杂，也不利于理解，但模板类与对象映射表的结合是对象创建更为灵活了。

本加密 COM 组件就是 Windows2000 平台上，在 Visual C++环境中用 ATL 进行开发

的。

2　加密组件的创建

（1）组件的规划原则

创建自己的 COM 组件之前，首先必须从功能方面弄清楚组件的需求。一般来说，COM 组件的规划应遵循以下原则：

➢ 用户对于数据库的访问请求应通过 COM 组件来实现。将访问权限授予 COM 组件，可有效保证数据库的安全性；把每个用户的连接变成和 COM 组件的连接，以避免数据库资源的浪费和系统崩溃的危险。

➢ 组件粒度不宜过大，争取每一个 COM 组件实现一个或一类相似的应用内容，而不必追究起功能的过分庞大。保证每个组件对象完成的逻辑功能相对单一，有助于重用机制的发挥和调节 COM 的适应性。

➢ COM 组件与用户的接口应尽量简单、友好。COM 组件如果是可视的，则只能有一个可视化界面。

（2）组件接口设计的考虑

组件的接口在整个应用系统中起决定性的作用。一般接口应具有较高的通用性，以提高整个应用系统的复用能力，同时还要兼顾简单和实用性。另外，组件内部实现细节不能反映到接口中，接口同内部实现细节的隔离程度越高，组件或应用发生变化对接口的影响越小。

3　RSA 加密组件的实现

RSA 运算基于大整数，因此如何表示大整数是实现 RSA 算法首先遇到的问题。对于 RSA 算法中的各个素数 p 和 q 只有 512 位(二进制)以上算法才是安全的，一般用字符串或位组表示大整数，从而实现大整数的算术运算和指数运算等。本系统中 RSA 算法的大整数采用 2048 位的，大约 10 进制 $O(10^{620})$ 量级，如此巨大的整数远远超出了计算机的表示能力，因此我们用长度为 256 的字符串表示大整数，下面给出的类 vlong 提供了实现 RSA 算法所需要的大整数的加、减、乘、除、模幂、位操作等基本运算。

```
Class vlong    //非常大的数
{
public;    //标准算术运算
friend vlong operator+(const vlong&x,const vlong&y);
friend vlong operator-(const vlong&x,const vlong&y);
```

```
friend vlong operator*(const vlong&x,const vlong&y);
friend vlong operator/(const vlong&x,const vlong&y);
friend vlong operator%(const vlong&x,const vlong&y);
        friend vlong operator^(const vlong&x,const vlong&y);
friend vlong pow2(DWORD n);
friend vlong operator&(const vlong&x,const vlong&y);
friend vlong operator<<(const vlong&x,dword n);
vlong& operator+=(const vlong& x);
vlong& operator-=(const vlong& x);
vlong& operator>>=(DWORD n);
//标准比较运算
friend long operator! =(const vlong&x,const vlong&y);
friend long operator==(const vlong&x,const vlong&y);
friend  long operator>=(const vlong&x,const vlong&y);
friend  long operator<=(const vlong&x,const vlong&y);
friend  long operator>(const vlong&x,const vlong&y);
        friend  long operator<(const vlong&x,const vlong&y);
friend  vlong form_str(char*s);
friend  vlong form_vlong(const vlong &x);
friend  vlong abs(const vlong &x);//Absolute value
//构造函数和析构函数
vlong(DWORD x=0)
vlong(const vlong& x);
~vlong();
friend DWORD to_unsigned(const vlong& x);
//位运算
DWORD bits() const;
DWORD bit)DWORD i)const;
Void setbit(DWORD i);
Void clearbit(DWORD i);
Vlong& operator^=(const vlong& x);
```

```
Vlong& operator&=(const vlong& x);
Vlong& ror(DWORD n);//single  bit  rotate
Vlong& rol(DWORD n);//single  bit  rotate
Friend long product(const vlong & x,const vlong & y);
Void load(DWORD*a,DWORD n);
Void store(DWORD*a,DWORD n)const;//low level save
Void load(DWORD*a,DWORD n);//自己添加
Void store(DWORD*a,DWORD n)const;//自己添加
Class vlong_value*value;
Long negative;
Long cf(const vlong & x)const;
Void  docopy();
Friend class monty;
};
```

(1) 随机大数的产生

随机性时密码学安全性的基础，RSA 算法也不例外。不论一个密码体制在数学上有多么安全，如果它的体制参数产生时随机性差，就很容易被重复，因此也就是不安全的。实现 RSA 算法首先要保证能够产生生成公钥和私钥的两个随机大素数。一般情况下计算机产生的随机数都是伪随机数，但是用一些好的算法产生的伪随机数非常接近真正的随机数，可以满足密码学的要求。现有的伪随机数产生算法产生的伪随机序列的周期高达 O（10^{50}）量级，因而可以满足本是算法的要求。此外随机种子影响着随机数的产生，它必须有足够的随机性，以免攻击着发现随机种子的规律后，重现密码的生成过程。本方法中通过捕获当前的随时间不同这个随机事件产生随机种子。

```
srand(unsigned)time(NULL);
for(i=0;i<2;i++)
{
  for(j=0;j<(LEVEL*2);j++)
   {
      tc=(char)(0x41+rand()%0xAF);
      prand[i][j]=tc;
   }
```

```
            prand[i][j]=0;
        }
        p=pf.find_prime(from_str(prand[0]));//计算生成两个达奇数 p,q
        q=pf.find_prime(from_str(prand[1]));
```

(2) 素数的产生

工程实现 RSA 算法的另一个关键是素性检测。对于 RSA 公钥密码体制，产生体制参数还需要两个不相同的大素数，这两个大素数必须是随机产生的，否则就容易被复制，因而也就不安全。通常产生素数的方法可以分为两种：确定性产生法和概率产生法。利用确定性产生法所生成的素数带有一定限制。如果算法设计不当。构造的素数容易带有规律性，使密码分析人员可以容易地追踪到素数的变化，直接猜测到所使用的素数，但由于生成的随机序列大整数不一定都是素数，因而就需要测试由随机序列构成的整数是否是素数，即素性检测。不可能用因式分解的方法检测一个大整数是否为素数，事实上，这正是 RSA 算法的安全性保证。只能用一些判决的算法测试一个大整数是否为素数。每执行一次测试，可以判断该整数不为合数的概率是多少。通过多次测试，即可判定一个大整数为素数的概率接近 1。比如，有一个素数检测法 f，用它测试大整数 p，如果 p 每通过一次测试，其为合数的概率是 1/t(t>1),则如 p 通过 n 次测试，根据乘法原理，p 为素数的概率为$(1-\frac{1}{t\hat{}n})$。显然。当 n→∞，$(1-\frac{1}{t\hat{}n})$→1,即可以判定 p 为素数。但在实际应用中,不可能取 n→∞,只能取一个有限的值。常见的随机测试法有威尔逊(Milosn)算法[2]，米勒-勒宾(Miller-Rabin)算法[2]、勒曼(D.J.Lehmann)算法等。我们所采用的是米勒-勒宾算法。它的思路如下：

已知 $n-1=2^c m$。

S1:在{1,2,…,N-1}中随机均匀地产生一数 a;j←0，计算 z←am mod n，若 z=1 或 n-1,则 n 通过测试，结束；

S2:若 j=c 则 n 非素数，结束，否则转 S3；

S3:j←j+1,z←z^2 mod n;若 z←-1 则通过测试，否则转 S2。

Prime_factor 类的定义描述如下，它主要用来生成 RSA 算法中所需要的两个大素数。

```
Class prime_factor
{
  public:
```

```
        DOWORD np;
        DOWORD *pl
        Prime_factory(DWORD MP=200);//sive  size
        ~prime_factory();
        vlong  find_prime(vlong & start);
        long make_prime(vlong & r,vlong &k,const vlong & rmin);
};
```

(3) 密钥的生成

根据文档 PKCS#，两个大整数 p,q 产生之后，计算 n=pq，然后给出公钥 e 和私钥 d,可以采用 X.509 中的建议,选取欧几里德扩展算法计算解密密钥 d,使之满足 ed=1 mod (p-1)(q-1)。将公钥 n 和 e，私钥 d 和 e 等封装成 RSA 公钥类 public_key 和 RSA 私钥类 private_key,其中这两个类的定义如下：

```
class public_key
{
public;
        public_key();//构造函数，派生类 private_key 也会调用这个构造函数;
        //要求 plain 必须小于 m;
        vlong encrypt(const vlong & plan);
        void  PK_to_vlong(PK pk);
        void  vlong_to _PK(PK &pk);
public;
       void  set_requires(int red);
       int  get_requires();
       int   requires;//判断加密的数是否小于 m
       vlong  m,e;
};
class private_key:public  public_key
{私钥类 private_key 是从公钥 public_key 中派生出来的
public:
    void   create();//生成 m,e,p,q 安全级别是 2048 位
    vlong  decrypt(const  vlong & cipher);//要求 0<=cipher<n
```

```
        void  SK_to_vlong(SK sk);
public:
        void  vlong_to_SK(SK &sk);
        vlong  p,q;
};
```

其中成员函数 create()为产生 RSA 密钥对函数，其实现过程如下：

```
viod private_key::create()
    {
        char prand[2][LEVEL*4/*必须不小于 LEVEL*2*/],tc;
        DWORD  i,j;
        DWORD validate[LEVEL].// 验证 m 的最高位是否为 0
        Prime_factory  pf;
        vlong tmp;
start:  //条件不成立，重新再生成 p 和 q
 srand(unsigned)time(NULL);
 for(i=0;i<2;i++)
{  for(j=0;j<(level*2);j++)
    {
          tc=(char)(0x41+rand()%0xaf);
          prand[i][j]=tc;
     }
     prand[i][j]=0;
}
//choose  primes
p=pf.find_prime(from_str(prand[0]));//计算生成两个大奇数 p,q
q= pf.find_prime(from_str(prand[1]));
if(p>q)
{
  tmp=p;
  p=q;
  q=tmp;
```

```
}
//calculate  public  key
m=p*q;    //m 最高位是非 0 时，要加密的大整数最高位是 0，就满足加密的条件了
m.store(validate,LEVEL);
//如果这个 m 不能满足条件，就重新生成 p 和 q，直到条件成立，
//其实这也不能完全解决问题还是要有 requires
if(validate[LEVEL-1]==0x0000000 goto start;
e=65537;  //如果这个固定的 e 不能满足条件，
//就重新生成 p 和 q，//直到条件成立
if(gcd(p-1,e)!=1||gcd(q-1,e)!=1) goto start;
}
```

(4) 加密和解密

用 publi_key 类中 encrypt 函数用来实现数据的加密操作，而用 private_key 类中的 decrypt 函数来实现数据的解密操作。解密过程应用了中国剩余定理，虽然步骤看起来变多了，但是解密速度却必直接计算要快得多，因为迭代的步数减少了。其中 encryp 函数和 decryp 函数的实现如下所示：

```
vlong public_key::encrypt(const  vlong& plain)
{
if(plain>=m)
{
   requires=false;
}
if(!requires)
{   //出错一次就不能进行加密，只有再把 requires=true
    return  m;
    }
    return modexp(plain,e,m);
}
vlong  private_key::decrypt(const  vlong cipher)
{
    if(cipher>=m)
```

```
        {
          requires=false;
          }
  if(!requires)
    {  //只要出错一次就不能进行加密，只有再把 requires=true
        return  m;
    }
    //应用中国剩余定理计算解密过程
    vlong d=modinv(e,(p-1)*(q-1));
    vlong u=modinv(p,q);
    vlong dp=d%(p-1);
    vlong dq=d%(q-1);
    vlong a=modexp(cipher%p,dp,p);
    vlong b=modexp(cipher%q,dq,q);
    if(b<a)b+=q;
    return a+p*(((b-a)*u)%q);
    }
```

(5) 接口的设计与实现

组件的接口在整个系统中起决定性的作用，一般接口应具有较高的通用性，以提高整个应用系统的复用能力，同时还要具有简单和实用性。因此将所有功能全部封装在组件的内部，并没有在接口上反映出来，而接口设计的十分简单，只有一个接口，在这个接口上有两个方法可以使用，一个是加密方法，一个是解密方法。这两个方法十分简单，这样可能组件的重用性会小一些，但不影响这个加密系统的实现。下面是加密组件对象 enctyption 的 IDL 定义：

```
import" oaidl.idl" ;
import" oaidl.idl" ;
[
object,
uuid(0f962554-f9e4-4478-b60A-36621FB57969),
dual,
helpstring(" icrypt Interface" ),
```

```
    pointer_default(unique)
  ]
  interface Icrypt:Idispatch
  {
    [id(1),helpstring( " method encrypt " )]HRESULT encrypt[in]BSTR
ctext,[out,retval] BSTR *mtext);
    [id(2), helpstring( " method encrypt " )]HRESULT encrypt[in]BSTR
ctext,[out,retval] BSTR *mt);
      };
  [
  uuid(15D7A647-BDD4-4C4D-9E22-58A89B495CF6),
  version(1.0),
  helpstring(" icrypt Interface" ),
  ]
  library CRYPTIONLib
  {
  imoprtlib(" stdole32.tlb" );
  imoprtlib(" stdole32.tlb" );
  [
    uuid(58B291FD-F52B-401A-A795-F90856AF851D),
    helpstring(" crypt Class" )
  ]
  coclass crypt
  {
    [default] interface Icrypt;
  };
  };
```

在 encrypt.cpp 文件中将对象的每个方法具体实现，编译成为 dll 二进制代码，即可注册生成进程内组件程序。其中加密方法的实现如下，解密方法与之相似。

```
STDMETHODIMP CCRYPT::ENCRYPT(BSTR ctext,BSTR *mtext)
{
```

```
*mtexp=ctext;
pv_key.create();//计算生成两个大奇数公钥和私钥：p,q,e,m
char *p1=_com_util::ConvertBSTRToString(ctext);
m1=from_key.encrypt(m1);
char *s=from_vlong(c1);
BSTR b=_com_util::ConvertStringToBSTR(S);
*mtext=b;
return S_OK;
}
```

3.5　结束语

信息技术的飞速发展，使得信息安全变得尤为重要，信息安全问题的解决主要依赖于现代密码学核心。理论研究、安全体系结构、网络安全通讯协议等关键技术以及借助于此产品的安全产品。本文以公钥密码体制中 RSA 算法为核心，深入系统地研究了 RSA 算法的基本概念和基本理论以及它的软硬件实现方法，并依此给出了一个基于 RSA 算法的加密组件。由于时间和精力有限，所以作者的研究还没有达到一定的深度，在以后的工作和学习中会继续努力作进一步的研究。

参考文献

[1] 张周.我国企业开始重视网络安全[N].计算机世界,2000-03-20(A9).

[2]吴世忠 祝世雄等 应用密码学——协议、算法与C语言程序 [M]北京：机械工业出版社，2001.1

[3]卢开澄. 计算机密码学一计算机网络中的数据保密与安全[M] 第二版 清华大学出版社，1998.12

[4]赖溪松. 计算机密码学及其应用[M] 国防工业出版社 2001.7

[5] 杨义先等. 网络信息安全与保密[M] 第一版 北京邮电大学出版社 1999.11

[6] R.L.Rivest, A.Shamir and L.Adelman, on digital digital signatures and public key cryptosystems, Commun.ACM, vol, 21,pp, 120-126,1978.

[7] T.ElGamal, A public-key Cryptosystem and a Signature Scheme Base on Discrete Logaruthms, Advances in Cryptology: Proceedings of CRYPT 84,Springer-Verlag, 1985,pp.10-18.

[8] C.E.Shannon. Communiation theory of secrecy systems [J]. Bell system Technical Journal,1949,28(4):656-715.

[9] D.K.BRANSTAD,J.Gait,and S.katzke,Report on Workshop on Cryptography in support of Computer Security,NBSIR 77-1291,Nztional Bureau of Standards,sep 21-22,1976,September 1977.

[10] 刘尊全.刘氏高强度公开加密算法原理与装置（第二板）[M].北京：清华大学出版社，1998.

[11] Dale R. Inside COM. Redmond, WA: Microsoft Press, 1996:116-138

[12] 潘爱民.COM原理与应用.北京:清华大学出版社，1999:26-30,65- 69

[13] HTTP://www.microsoft.com/com Microsoft的COM技术WWW站点

[14] 曹晓阳，刘锦德 COM及其应用一面向对象的组件集成技术.计算机应用，1999, 19(1):1-4

[15] A Selby C,Mitchell.Algorithm for Software implementation of RSA.IEEEproceeding 1989,136,pt E.(3):166-170

[16] A.Fiat and A.Shamir,How to Prove Yourself: Practical Solutions to Identification and Signature problems, Advances in Cryptology—CRYPTO' 86 Proceedings, Springr-Verlag, 1987,pp.186-194.

[17]谷大武 徐胜波 高级加密标准(AES)算法——Rijndael 的设计[M] 北京：清华大学出版社 2003.3

[18]崔光耀，一夜好风吹，新花近万枝一中国信息安全市场扫描，信息安全与通信保密[J]，2000年第2期，7-9

[19] RSA Laboratories. RSA Laboratories Frequently Asked Questions About Today's Cryptograp, RSA Data Securitv, Inc, 1998

[20] RSA Laboratories. High-Speed RSA Implementation. RSA Data Sect--ritzy, Inc., 1991

[21] M. J. B. Robshaw. Security Estimates for 512-bit RSA. RSA Data Security, Inc., 1995

[22] Tanenbaum A S. Computer networks. 3rd Ed. Pren-tice-Hall, 1996. 587-601

[23] G. J. Simmons, "A 'Wea' Privacy Protocol Using the RSA cryptosystem," Cryptologia, v. 7, n. 2, Apr 1983, pp. 51-67

致 谢

通过一年多的研究生论文的撰写和课题的研究工作，作者从理论素质和实践能力两方面都得到了极大的提高和锻炼，这对作者以后的工作和学习都打下了坚实的基础，对以后继续从事科研或教学方面的工作是大有裨益的。

在一年多的课题研究过程中，作者的导师——张祥德教授一直给予作者无尽的关怀和细心的指导。张老师以一丝不苟的工作作风和扎实的专业理论基础给作者指出研究工作中的各种实际困难和解决方法，他的循循善诱和谆谆教诲使我不但学到了严谨的科研作风和解决实际问题的动手能力，而且学到了许多做人的基本道理，使自己知道作为一名从事科学研究的工作者所应该具备的基本素质。在此，作者想借这一宝贵机会，谨向张老师表示最真诚的感谢，并向他的家人致以衷心的祝福。同时还要感谢在做论文期间给予我帮助过的同事。