第 3 章

信息安全的加密技术

3.1 密码学概述

密码技术是信息安全的核心技术。密码学是集数学、计算科学、电子与通信等多种学科于一身的交叉学科。它不仅能保证机密性信息的加密，而且能够实现数字签名、身份验证、系统安全等功能，是迅速发展的重要学科之一。密码学是信息安全相关议题，如数字签名、数字证书的核心。密码学的首要目的是隐藏信息的涵义，并不是隐藏信息的存在。

3.1.1 密码学的概念

密码学（Cryptology）：研究信息系统安全保密的科学。它包含两个分支：密码编码学（Cryptography），对信息进行编码实现隐蔽信息的一门学问；密码分析学（Cryptanalytics），研究在不知道密钥的情况下，分析破译密码的学问。一些密码学常见术语如下。

（1）明文（消息）（Plaintext）：被隐蔽消息。

（2）密文（Ciphertext）或密报（Cryptogram）：明文经密码变换成的一种隐蔽形式。

（3）加密（Encryption）：将明文变换为密文的过程。

（4）解密（Decryption）：加密的逆过程，即由密文恢复出原明文的过程。

（5）加密员或密码员（Cryptographer）：对明文进行加密操作的人员。

（6）加密算法（Encryption Algorithm）：密码员对明文进行加密时所采用的一组规则。

（7）接收者（Receiver）：传输消息的预定对象。

（8）解密算法：接收者对密文进行解密时所采用的一组规则。

（9）密钥（Key）：控制加密和解密算法操作的数据处理，分别称作加密密钥和解密密钥。

（10）截收者（Eavesdropper）：在信息传输和处理系统中的非授权者，通过搭线窃听、电磁窃听、声音窃听等来窃取机密信息。

（11）密码分析（Cryptanalysis）：截收者试图通过分析截获的密文推断出原来的明文或密钥。

（12）密码分析员（Cryptanalyst）：从事密码分析的人。

（13）被动攻击（Passive Attack）：对一个保密系统采取截获密文进行分析的攻击。

（14）主动攻击（Active Attack）：非法入侵者（Tamper）、攻击者（Attacker）或黑客（Hacker）主动向系统窜扰，采用删除、增添、重放、伪造等篡改手段向系统注入假消息，达到利己害人的目的。

3.1.2 密码学的产生和发展

密码学是一门既古老又新兴的学科。密码学一词源自希腊文“krypto's”及“logos”两词，直译即“隐藏”及“信息”。密码学有一个奇妙的发展历程，当然，秘而不宣总是扮演主要角色。有人把密码学的发展划分为三个阶段。

1．第一阶段（古代到 1949 年）

这一时期可以看作科学密码学的前夜时期。该阶段的密码技术更多地可以说是一种艺术，而不是一种科学。密码学专家常常凭直觉和信念来进行密码设计和分析，而不是推理和证明。人们早在古埃及就已经开始使用密码技术，但是用于军事目的，并不公开。

公元前 50 年，恺撒大帝发明了一种密码叫作恺撒密码。在恺撒密码中，每个字母都与其后第三位的字母对应，然后进行替换，如“A”对应“D”，“B”对应“E”，依次类推。如果到了字母表的末尾，就回到开始，如“Z”对应“C”，“Y”对应“B”，“X”对应“A”，如此形成一个循环。当时罗马的军队就用恺撒密码进行通信。

恺撒密码明文字母表：A　B　C　D　E　F　G　…… X　Y　Z

恺撒密码密文字母表：D　E　F　G　H　I　J　…… A　B　C

这样就可以从明文得到密文。例如：明文“VENI, VIDI, VICI”对应的密文“YHQL, YLGL, YLFL”，意思是“我来，我见，我征服”，是恺撒征服本都王法那西斯后向罗马元老院宣告的名言。

1844 年，萨米尔·莫尔斯发明了莫尔斯电码：用一系列的电子点画来进行电报通信。电报的出现第一次使远距离快速传输信息成为可能，事实上，它增强了西方各国的通信能力。

20 世纪初，意大利物理学家奎里亚摩·马可尼发明了无线电报，让无线电波成为新的通信手段，它实现了远距离通信的即时传输。马可尼的发明改变了密码世界。由于通过无线电波送出的每条信息不仅传给了己方，也传送给了敌方，这就意味着必须给每条信息加密。

随着第一次世界大战的爆发，对密码和密码分析员的需求急剧上升，一场秘密通信的全球战役打响了。

第一次世界大战前，重要的密码学进展很少出现在公开文献中。直到 1918 年，20 世纪最有影响的密码分析文章之一——威廉·F. 弗里德曼的专题论文《重合指数及其在密码学中的应用》作为私立的“河岸实验室”的一份研究报告问世了，其实，这篇论文涉及的工作是在战争期间完成的。第一次世界大战后，完全处于秘密工作状态的美国陆军和海军的机要部门开始在密码学方面取得根本性的进展，但是公开的文献几乎没有。

1918 年，加州奥克兰的爱德华·H. 赫伯特申请了第一个转轮机专利，这种装置在约 50 年里被指定为美军的主要密码设备，它依靠转轮不断改变明文和密文的字母映射关系。由于转轮的存在，每转动一格就相当于给明文加密一次，并且每次的密钥不同，而密钥的数量就是全部字母的个数——26 个。

同年，德国人亚瑟·谢尔比乌斯发明了第一台非手工编码的密码机——恩尼格玛密码机。恩尼格玛密码机是德军在第二次世界大战期间最重要的通信利器，也是密码学发展史上的一则传奇。

随着高速、大容量和自动化保密通信的要求，机械与电路相结合的转轮加密设备的出现使古典密码体制退出了历史舞台。

2．第二阶段（1949—1975 年）

1946 年，世界上第一台计算机 ENIAC 诞生。冯·诺依曼的二进制编码、计算机硬件系统的组成结构、存储程序的工作方式对 ENIAC 的产生，以及后来电子计算机的设计产生了深远影响，并延续至今。运用电子计算机对二进制形式存储的明文进行加密解密标志着现代密码学的开始。

1949 年香农的《保密系统的通信理论》，为近现代密码学建立了理论基础。从 1949 年到 1967 年，密码学文献近乎空白。密码学有许多年都是军队独家专有的领域。美国国家安全局，以及英国、法国、以色列及其他国家的安全机构已将大量财力投入加密自己的通信，同时千方百计地去破译别人的通信，面对这些政府，个人既无专门知识又无足够财力去保护自己的秘密。

1967 年，戴维·卡恩的《破译者》（*The Code Breaker*），对以往的密码学历史做了相当完整的记述。《破译者》的意义不仅在于它涉及相当广泛的领域，它还使成千上万的人了解了密码学。此后，密码学文章开始大量涌现。大约在同一时期，早期为空军研制敌我识别装置的霍斯特·菲斯特尔在位于纽约约克镇高地的国际商业机器公司沃森实验室里致力于密码学的研究，开始着手美国数据加密标准的研究。到 20 世纪 70 年代初期，国际商业机器公司发表了菲斯特尔和他的同事在这个课题方面的几篇技术报告。

3．第三阶段（1976 年至今）

1976 年迪菲和赫尔曼发表的文章《密码学的新动向》开启了密码学上的一场革命。他们首先证明了在发送端和接收端无密钥传输的保密通信是可能的，从而开创了公钥密码学的新纪元。

1978 年，罗纳德·李维斯特、阿迪·萨莫尔和伦纳德·阿德曼实现了 RSA 公钥密码体制。

1969 年，哥伦比亚大学的史蒂芬·威斯纳首次提出“共轭编码”（Conjugate Coding）的概念。1984 年，查尔斯·本尼特和吉列斯·布拉萨德在其思想启发下，提出量子理论 BB84 协议，从此量子密码理论宣告诞生。其安全性在于：可以发现窃听行为，可以抗击无限能力计算行为。

1985 年，米勒和克布里茨首次将有限域上的椭圆曲线运用到了公钥密码系统中，其安全性是基于椭圆曲线上的离散对数问题。

1989 年，马修斯、惠勒、佩科拉和卡罗尔等人首次把混沌理论应用到序列密码及保密通信理论，为序列密码研究开辟了新途径。

2000 年，欧盟启动了新欧洲数据加密、数字签名、数据完整性计划，采用了适应于 21 世纪信息安全发展全面需求的序列密码、分组密码、公开密钥密码、散列函数及随机噪声发生器等技术。

3.1.3 密码算法

密码算法是用于加密和解密的数学函数，密码算法是密码协议的基础。现行的密码算法主要包括序列密码、分组密码、公开密钥密码、散列函数等，密码算法除了保证信息的保密性，还保障信息的完整性和不可抵赖性。

假设我们想通过网络发送消息 P（P 通常是明文数据包），使用密码算法隐藏 P 的内容，可将 P 转化成密文，这个转化过程称为加密。得到与明文 P 相对应的密文 C，需要依靠一个附加的参数 K_1，称为密钥。密文 C 的接收者为恢复明文，需要另一个密钥 K_2 完成反方向的运算，这个反向计算的过程称为解密。

信息加密解密的传输模型如图 3-1 所示。

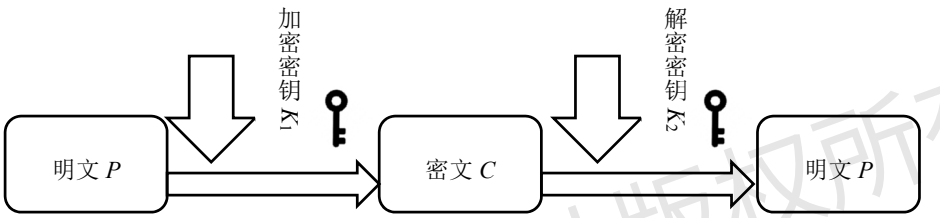

图 3-1 信息加密解密的传输模型

相关概念如下。

（1）发送者和接收者。假设发送者想安全地发送消息给接收者，他想确保窃听者不能阅读发送的消息，就需要对其加密。

（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）。涉及计算机，P 则是简单的二进制数据。明文可被传送或存储，无论在哪种情况下，M 都指待加密的消息。

密文用 C 表示，它也是二进制数据，有时和 M 一样大，有时比 M 稍大（通过压缩和加密的结合，C 有可能比 M 小些；然而，单单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，用数学公式表示为

$$E(M)=C$$

相反地，解密函数 D 作用于 C 产生 M

$$D(C)=M$$

先加密再解密消息，要想恢复原始的消息，下面的等式必须成立：

$$D[E(M)]=M$$

（3）鉴别、完整性检验和抗抵赖。除提供机密性外，密码学还有其他作用：

① 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。

② 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原消息。

③ 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：一个函数用作加密，另一个函数用作解密）。

如果算法的保密性是基于保持算法的秘密，那么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，因为每当有一个用户离开这个组织时，其他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，那么所有人都必须改变他们的算法。

但是，受限制的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，因为用户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 的可能值的范围叫作密钥空间。加密和解密运算都使用这个密钥（运算都依赖于密钥，并用 K 作为下标表示），这样，加/解密函数就变成

$$E_K(M)=C$$
$$D_K(C)=M$$

这些函数具有下面的特性

$$D_K[E_K(M)]=M$$

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 K_1 与相应的解密密钥 K_2 不同，在这种情况下有

$$E_{K_1}(M)=C$$
$$D_{K_2}(C)=M$$
$$D_{K_2}[E_{K_1}(M)]=M$$

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，对于可以大量生产使用算法的产品，即使窃听者知道你的算法也没有关系，因为只要他不知道你使用的具体密钥，他就不可能阅读你的消息。

密码系统是由算法及所有可能的明文、密文和密钥组成的。基于密钥的加密算法通常有两类：对称加密算法和非对称加密算法。

1．对称加密算法

对称加密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，泄露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

$$E_K(M)=C$$
$$D_K(C)=M$$

对称加密算法可分为两类。一类是一次只对明文中的单个位（有时对字节）运算的算

法，称为序列算法或序列密码。另一类是对明文的一组位进行运算的算法，这些位组称为分组，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 位——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进行运算，可认为是序列算法对字符序列的运算）。后来，随着破译能力的发展，分组长度又提高到 128 位或更长。

对称加密算法是应用较早的加密算法，技术成熟。在对称加密算法中，发送者将明文（原始数据）和加密密钥一起进行特殊加密算法处理，使其变成复杂的加密密文发送出去。接收者收到密文后，若想解读，只有使用加密用过的密钥及相同算法的逆算法对密文进行解密，才能使其恢复成可读明文。在对称加密算法中，使用的密钥只有一个，发送/接收双方都使用这个密钥对数据进行加密或解密，这要求接收者必须事先知道加密密钥。对称加密算法的优点是算法公开、计算量小、加密速度快、加密效率高；缺点是发送/接收双方使用同样的密钥，安全性得不到保证。此外，发送/接收双方每次使用对称加密算法时，都需要使用其他人不知道的唯一密钥，这会使发送/接收双方所拥有的密钥数量以几何级数增长，密钥管理成为负担。对称加密算法在分布式网络系统上使用较为困难，主要是因为密钥管理困难，使用成本较高。在计算机专网系统中广泛使用的对称加密算法有数据加密标准（Data Encryption Standard，DES）算法和国际数据加密算法等。

2．非对称加密算法

非对称加密算法，也称公开密钥算法，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）。之所以其叫作公开密钥算法，是因为加密密钥能够公开，即陌生人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，加密密钥叫作公开密钥（简称公钥），解密密钥叫作私人密钥（简称私钥）。私人密钥有时也叫作秘密密钥。为了避免与对称算法混淆，此处不用秘密密钥这个名字。

公开密钥 K_1 加密表示为

$$E_{K_1}(M)=C$$

虽然公开密钥和私人密钥是不同的，但用相应的私人密钥 K_2 解密可表示为

$$D_{K_2}(C)=M$$

有时消息用私人密钥加密而用公开密钥解密，如数字签名（后面将详细介绍），这些运算可分别表示为

$$E_{K_2}(M)=C$$

$$D_{K_1}(C)=M$$

当前的公开密钥算法的运算速度，比对称算法要慢得多，这使公开密钥算法在大数据量的加密中应用有限。

公开密钥算法使用完全不同但又是完全匹配的一对钥匙——公开密钥和私人密钥。在使用公开密钥算法加密消息时，只有使用匹配的一对公开密钥和私人密钥，才能完成对消息的加密和解密过程。加密明文时采用公开密钥进行，解密密文时使用私人密钥才能完成，而且发送者（加密者）知道接收者的公开密钥，但接收者（解密者）是唯一知道自己私人密钥的人。由于公开密钥算法拥有两个密钥，因而特别适用于分布式系统中的数据加密。

广泛应用的公开密钥算法有 RSA 算法。以公开密钥算法为基础的加密技术应用非常广泛，如数字证书等，后面章节将详细阐述。

3．单向散列函数

单向散列函数是不需要密钥的密码算法，在现代公钥基础设施中有着广泛的用途，是数字签名、数字证书的重要基础。MD5 和 SHA 是常用的单向散列函数。

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他特性：

（1）给定 M，很容易计算出 h；

（2）给定 h，要根据 $H(M)=h$ 计算出 M 是很难的，或者说是不可能的；

（3）给定 M，要找到另一个消息 M' 并满足 $H(M)=H(M')$ 是很难的。

在许多应用中，仅有单向性是不够的，还需要有“抗碰撞”的条件。也就是说，要找出两个随机的消息 M 和 M'，很难满足 $H(M)=H(M')$。由于单向散列函数的这些特性，以及公开密钥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，让发送者对 $H(M)$ 签名而不是对 M 签名。

单向散列函数常用于不可逆加密算法，以及数字签名算法（Digital Signature Algorithm，DSA）的设计。所谓不可逆加密算法，是指加密过程中不需要使用密钥，输入明文后，明文由系统直接经过加密算法处理形成密文，这种加密后的密文是无法被解密的，只有重新输入明文，并再次经过同样不可逆的加密算法处理，得到相同的加密密文并被系统重新识别后，才能真正解密。显然，在这类加密过程中，加密者是自己，解密者还得是自己，而所谓解密，实际上就是重新加一次密，所应用的“密码”也就是输入的明文。不可逆加密算法不存在密钥保管和分发问题，非常适合在分布式网络系统上使用，但因加密计算复杂，工作量相当繁重，通常只在数据量有限的情形下使用，如广泛应用在计算机系统中的口令加密利用的就是不可逆加密算法。近年来，随着计算机系统性能的不断提高，不可逆加密算法的应用领域逐渐增大。在计算机网络中应用较多的不可逆加密算法有 RSA 公司发明的 MD5 算法和由美国国家标准局建议的不可逆加密标准——安全杂乱信息标准（Secure Hash Standard，SHS）等。

3.2　传统密码技术

3.2.1　单表代换密码

恺撒密码因被古罗马皇帝恺撒用于和将军们进行联系而闻名。它通常作为其他更复杂的加密方法中的一个步骤，如维吉尼亚密码。但和所有的利用字母表进行替换的加密技术一样，恺撒密码非常容易被破解，而且在实际应用中也无法保证通信安全。字母频谱分析是单表代换密码常用的密码分析方法。

单表代换密码的基本思想：通过把字母移动一定的位数来实现加密和解密。例如，把

明文字母的位数向后移动三位，那么明文字母“B”就变成了密文的“E”，依次类推，“X”变成“A”，“Y”变成“B”，“Z”变成“C”，如图 3-2 所示。

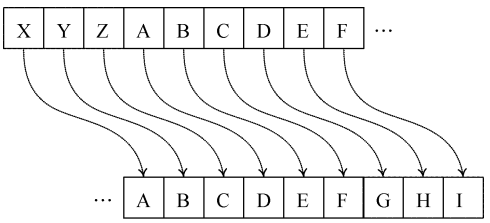

图 3-2　恺撒密码

3.2.2　多表代换密码

多表代换密码中最有名的一种密码称为维吉尼亚密码。这是一种以移位代换为基础的周期代换密码，m 个移位代换表由 m 个字母组成的密钥字确定（这里假设密钥字中的 m 个字母不相同；如果有相同的，则代换表的个数是密钥字中不同字母的个数）如果密钥字为“deceptive”，则明文“wearediscoveredredsaveyourself”被加密过程中的明文、密钥、对应数字密钥和密文分别如下。

明文：w e a r e d i s c o v e r e d s a v e y o u r s e l f

密钥：d e c e p t i v e d e c e p t i v e d e c e p t i v e

对应数字密钥：3 4 2 4 15 19 8 21 4 3 4 2 4 15 19 8 21 4 3 4 2 4 15 19 8 21 4

密文：Z I C V T W Q N G R Z G V T W A V Z H C Q Y G L M G J

其中，密钥字母 a，b，c，d，…，x，y，z 对应数字 0，1，2，3，…，23，24，25。密钥字母“d”对应数字 3，因而明文字母“w”在密钥字母“d”的作用下向后移动 3 位，得到密文字母“Z”；密钥字母“e”对应数字 4，因而明文字母“e”在密钥字母“e”的作用下向后移动 4 位，得到密文字母“I”；以此类推。解密时，密文字母在密钥字母的作用下向前移位。

在维吉尼亚密码中，如果密钥字的长度是 m，则明文中的一个字母能够映射成这 m 个可能字母的一个。容易看出，维吉尼亚密码中长度为 m 的可能密钥字的个数是 26^m，甚至对于一个较小的 m 值，如 $m=5$，密钥空间也超过了 1.1×10^7 次，这个空间已经足以阻止手工穷举密钥搜索。

为方便记忆，维吉尼亚密码的密钥字常常取于英文中的一个单词、一个句子或一段文字。事实上，单表代换密码是多表代换密码的特例。因此，维吉尼亚密码的明文和密文频率分布相同，仍然能够用统计技术进行分析。要抗击这样的密码分析，只能选择与明文长度相同且与之没有统计关系的密钥内容。1918 年美国电话电报公司的弗纳姆提出这样的密码系统：明文英文字母编成 5 位二元数字，称为五单元波多代码（Baudot Code），选择随机二元数字流作为密钥，加密通过执行明文和密钥的逐位异或操作，产生密文，可以简单表示为

$$C_i = p_i \oplus k_i$$

式中，p_i 表示明文的第 i 个二元数字；k_i 表示密钥的第 i 个二元数字；C_i 表示密文的第 i 个

二元数字；⊕ 表示异或操作。解密仅需执行相同的逐位异或操作：

$$p_i = C_i \oplus k_i$$

弗纳姆密码系统的密钥若不重复使用，就能得到“一次一密”密码。若密钥有重复，虽然使用长密钥增加了密码分析的难度，但只要有了足够的密文，使用已知的或可能的明文序列，或将二者结合也就能破译。

3.2.3　多字母代换密码

前面介绍的密码都是以单个字母作为代换对象的。对多于一个字母的对象进行代换，就是多字母代换密码。它的特点是将字母的频度隐藏，从而抗击统计分析。首先介绍希尔密码，它是数学家希尔于 1929 年研制的。这类密码虽然由于加密操作复杂而未能广泛应用，但仍在很大程度上推进了传统密码学的研究。

希尔密码将明文组分为 m 个字母一组的明文组，若最后一组明文不够 m 个字母则用字母补足，每组明文用 m 个密文字母代换。这种代换由 m 个线性方程决定，其中字母 a, b, c,…, x, y, z 对应数字 0, 1, 2,…, 23, 24, 25。若 m=3，则该系统可以描述为

$$C_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26$$
$$C_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26$$
$$C_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26$$

用列向量和矩阵可表示为

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

或

$$\boldsymbol{C} = \boldsymbol{KP}$$

式中，$\boldsymbol{C}$ 和 $\boldsymbol{P}$ 分别是密文向量和明文向量；$\boldsymbol{K}$ 是密钥矩阵。操作要执行模 26 运算。例如，用密钥

$$\boldsymbol{K} = \begin{pmatrix} 11 & 3 \\ 8 & 7 \end{pmatrix}$$

来加密明文“july”。将明文分成两组——“ju”和“ly”，分别为(9, 20)和(11, 24)，计算如下：

$$\begin{pmatrix} 11 & 3 \\ 8 & 7 \end{pmatrix} \begin{pmatrix} 9 \\ 20 \end{pmatrix} = \begin{pmatrix} 99+60 \\ 72+140 \end{pmatrix} = \begin{pmatrix} 26\times 6+3 \\ 26\times 8+4 \end{pmatrix} \rightarrow \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} 11 & 3 \\ 8 & 7 \end{pmatrix} \begin{pmatrix} 11 \\ 24 \end{pmatrix} = \begin{pmatrix} 121+72 \\ 88+168 \end{pmatrix} = \begin{pmatrix} 26\times 7+11 \\ 26\times 9+22 \end{pmatrix} \rightarrow \begin{pmatrix} 11 \\ 22 \end{pmatrix}$$

因此，“july”的加密结果为“DELW”。为了解密，必须先计算密钥矩阵的解密密钥矩阵。

$$\boldsymbol{K}^{-1} = \begin{pmatrix} 7 & 23 \\ 18 & 11 \end{pmatrix}$$

然后计算

$$\begin{pmatrix}7 & 23\\18 & 11\end{pmatrix}\begin{pmatrix}3\\4\end{pmatrix}=\begin{pmatrix}21+92\\54+44\end{pmatrix}=\begin{pmatrix}26\times4+9\\26\times3+20\end{pmatrix}\rightarrow\begin{pmatrix}9\\20\end{pmatrix}$$

$$\begin{pmatrix}7 & 23\\18 & 11\end{pmatrix}\begin{pmatrix}11\\22\end{pmatrix}=\begin{pmatrix}77+506\\198+242\end{pmatrix}=\begin{pmatrix}26\times22+11\\26\times16+24\end{pmatrix}\rightarrow\begin{pmatrix}11\\24\end{pmatrix}$$

最后，得到正确的明文“july”。

3.2.4 轮转密码

轮转密码（Rotor Cipher）机是用一组转轮或接线编码轮（wired code wheel）所组成的机器，可实现长周期的多表代换密码，是机械密码时代最杰出的成果，曾被广泛应用于军事通信中。其中，最有名的两种密码机是恩尼格玛密码机和哈格林密码机。恩尼格玛密码机由德国亚瑟·谢尔比乌斯发明，在第二次世界大战中希特勒曾用它装备德军，作为陆海空最高级密码。哈格林密码机由瑞典威廉·哈格林发明，在第二次世界大战中曾被广泛应用。哈格林 C-36 曾广泛用于装备法国军队。哈格林 C-48，即 M-209 密码机具有质量轻、体积小、结构紧凑等优点，曾装备美国师、营级，总生产量达 14 万部，美军在朝鲜战争中还在使用。此外，在第二次世界大战中，日本采用的红（red）密和紫（purple）密都是轮转密码。今天，周期更长、更复杂的密码可以用超大规模集成电路（Very Large Scale Integrated Circuit，VLSI）实现，所以这类密码机已逐步被淘汰。

总而言之，传统密码技术存在的缺点在于，由于大多数传统密码技术起源于计算机诞生之前，它们仅限于字母组合，无法解决字母的大小写和其他非字母的字符问题。事实上，它们是一种对称密码技术，属于对称密码技术的早期阶段。计算机的诞生和 0～1 字节流处理，使对称密码学得到了快速发展。

3.3 对称密码技术

常用的采用对称密码技术的加密方案有 5 个组成部分，如图 3-3 所示。

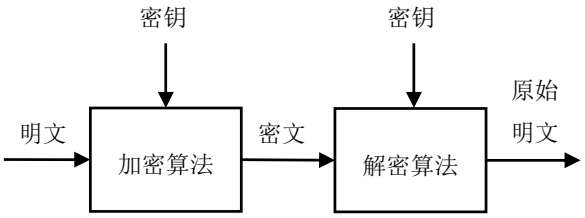

图 3-3 采用对称密码技术的加密方案的组成部分

（1）明文：原始信息。

（2）加密算法：以密钥为参数，对明文进行多种置换和转换的规则和步骤，变换结果为密文。

（3）密钥：加密与解密算法的参数，直接影响对明文进行变换的结果。

（4）密文：对明文进行变换的结果。

（5）解密算法：加密算法的逆变换，以密文为输入、密钥为参数，变换结果为原始明文。

对称密码中有几种常用的数学运算。这些运算的共同目的就是把被加密的明文数码尽可能地打乱，从而加大破译的难度。

（1）移位和循环移位：移位就是将一段数码按照规定的位数整体性地左移或右移。循环右移就是当右移时，把数码最后的位移到数码的最前方；循环左移正相反。例如，对十进制数码 12345678 循环右移 1 位（十进制位）的结果为 81234567；循环左移 1 位的结果为 23456781。

（2）置换：置换就是将数码中某一位的值根据置换表的规定，用另一位代替。它不像移位操作那样整齐有序，其看上去杂乱无章，这正是加密所需的，因而被广泛使用。

（3）扩展：扩展就是将一段数码扩展成比原来位数更长的数码。扩展方法有很多种，例如，可以用置换的方法，以扩展置换表来规定扩展后的数码每一位的替代值。

（4）压缩：压缩就是将一段数码压缩成比原来位数更短的数码。压缩方法有多种，例如，可以用置换的方法，以置换表来规定压缩后的数码每一位的替代值。

（5）异或：这是一种二进制布尔代数运算。异或的数学符号为⊕，它的运算法则如下：

$$1 \oplus 1 = 0$$
$$0 \oplus 0 = 0$$
$$1 \oplus 0 = 1$$
$$0 \oplus 1 = 1$$

我们可以将其简单地理解为，参与异或运算的两数位如相等，则结果为 0；不等则为 1。

（6）迭代：迭代就是多次重复相同的运算，这在密码算法中经常被使用，以使形成的密文更加难以破解。

3.3.1　DES 算法

DES 算法是由国际商业机器公司研制的一种对称加密算法，美国国家标准局于 1977 年公布，把它作为非机要部门使用的数据加密标准，它一直活跃在国际保密通信的舞台上，扮演着十分重要的角色。

DES 算法是一个分组加密算法，典型的 DES 算法以 64 位为分组对数据加密，加密和解密用的是同一种算法。它的密钥长度是 56 位（每个第 8 位都用作奇偶校验），密钥可以是任意的 56 位的数，而且可以在任意时候改变。其中有极少数被认为是易破解的弱密钥，但是我们可以很容易地避开不用它们，所以其保密性依赖于密钥。

1．DES 加密的算法框架

首先要生成一套加密密钥，从用户处取得一个 64 位的密码口令，然后通过等分、移位、选取和迭代形成一套 16 个加密密钥，分别供每一轮运算中使用。

DES 算法对 64 位的明文分组 M 进行操作，M 经过一个初始置换 IP，置换成 M_0。将明文 M_0 分成左半部分和右半部分 $M_0=(L_0,R_0)$，各 32 位；然后进行 16 轮完全相同的运算

（迭代），这些运算被称为函数 f，在每一轮运算过程中数据与相应的密钥结合。

在每一轮运算中，密钥位移位，然后从密钥的 56 位中选出 48 位；通过一个扩展置换将数据的右半部分扩展成 48 位；并通过一个异或操作替换成新的 48 位数据；再将其压缩置换成 32 位。这四步运算构成了函数 f。然后，通过另一个异或操作，函数 f 的输出与左半部分结合，其结果成为新的右半部分，原来的右半部分成为新的左半部分。将该操作重复 16 次。

经过 16 轮迭代后，左、右半部分合在一起经过一个末置换（数据整理），就完成了加密过程。加密流程如图 3-4 所示。

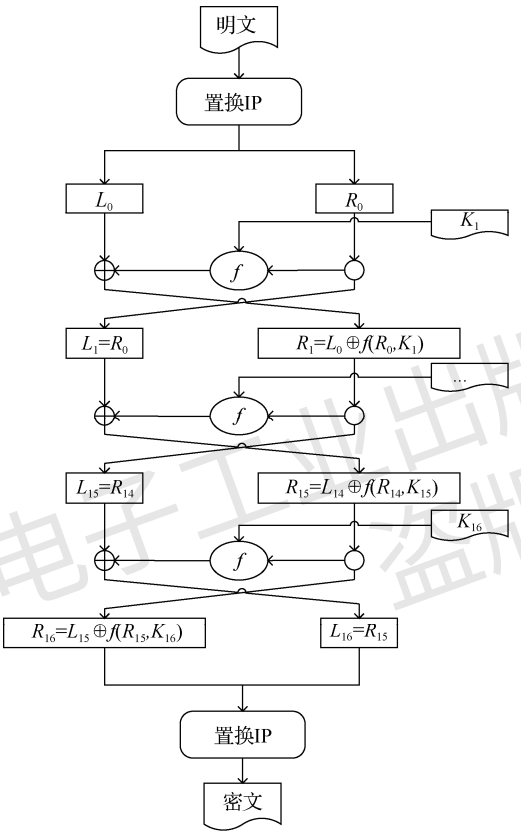

图 3-4　加密流程

2．DES 算法的解密过程

在了解了加密过程中所有的移位、置换、异或和循环迭代之后，读者也许会认为，解密算法应该是加密的逆运算，与加密算法完全不同。恰恰相反，经过密码学家精心设计选择的各种操作，DES 算法获得了一个非常有用的性质：加密和解密使用相同的算法。

DES 加密和解密唯一的不同是密钥的次序相反。如果各轮加密密钥分别是 $K_1, K_2, K_3, \cdots, K_{16}$，那么解密密钥就是 $K_{16}, K_{15}, K_{14}, \cdots, K_1$。这也就是 DES 算法被称为对称加密算法的原因。

3．DES 算法的实际操作

1）密钥生成

第 A-1 步　取得密钥。

从用户处取得一个 64 位（本书中均指二进制位）的密码口令 key，去除 64 位密码中作为奇偶校验位的第 8 位、第 16 位、第 24 位、第 32 位、第 40 位、第 48 位、第 56 位、第 64 位，剩下的 56 位作为有效输入密钥。

第 A-2 步　等分密钥。

把在第 A-1 步中生成的 56 位输入密钥分成均等的 A、B 两部分，每部分为 28 位，但不是简单地前后一分为二，而是参照表 3-1 和表 3-2 把输入密钥的位值填入相应的位置。如表 3-1 所示，A 的第一位为输入的 64 位密钥的第 57 位，A 的第 2 位为输入的 64 位密钥的第 49 位，依次类推，A 的最后一位（第 28 位）为输入的 64 位密钥的第 36 位。这样，形成了 A、B 两部分：

$$k = k_1 k_2 k_3 \cdots k_{55} k_{56}$$
$$A = k_{57} k_{49} k_{41} \cdots k_{44} k_{36}$$
$$B = k_{65} k_{55} k_{47} \cdots k_{12} k_4$$

表 3-1　输入密钥位序/ *A* 位序对照表

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	50	44	36

注：表中每个单元格的数字是输入密钥位序，每个单元格的位置排序就是 *A* 位序，从左向右排，排完一行接着排下一行。

表 3-2　输入密钥位序/ *B* 位序对照表

65	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

注：表中每个单元格的数字是输入密钥位序，每个单元格的位置排序就是 *B* 位序，从左向右排，排完一行接着排下一行。

第 A-3 步　密钥移位。

DES 算法的密钥是经过 16 次迭代（循环左移）得到一组密钥的，把在第 A-1 步中生成的 *A*、*B* 视为迭代的起始密钥，表 3-3 为每一次迭代时密钥循环左移的位数。例如，在第 1 次迭代时密钥循环左移 1 位，第 3 次迭代时密钥循环左移 2 位，第 9 次迭代时密钥循环左移 1 位，第 14 次迭代时密钥循环左移 2 位。

表 3-3　每次迭代时密钥循环左移的位数

迭 代 序 号	1	2	3	4	5	6	7	8
循环左移的位数	1	1	2	2	2	2	2	2
迭 代 序 号	9	10	11	12	13	14	15	16
循环左移的位数	1	2	2	2	2	2	2	1

第 1 次迭代：

$$A(1)=(1)\,A$$
$$B(1)=(1)\,B$$

第 i 次迭代：

$$A(i)=(i)\,A(i-1)$$
$$B(i)=(i)\,B(i-1)$$

第 A-4 步　密钥的选取。

在第 A-3 步中第 i 次迭代生成的 2 个 28 位长的密钥为

$$A^{(i)}=A^{(i)}{}_1A^{(i)}{}_2A^{(i)}{}_3\cdots A^{(i)}{}_{27}A^{(i)}{}_{28}$$
$$B^{(i)}=B^{(i)}{}_1B^{(i)}{}_2B^{(i)}{}_3\cdots B^{(i)}{}_{27}B^{(i)}{}_{28}$$

将其合并，得

$$C^{(i)}=A^{(i)}B^{(i)}=A^{(i)}{}_1A^{(i)}{}_2A^{(i)}{}_3\cdots A^{(i)}{}_{27}A^{(i)}{}_{28}B^{(i)}{}_1B^{(i)}{}_2B^{(i)}{}_3\cdots B^{(i)}{}_{27}B^{(i)}{}_{28}$$
$$=C^{(i)}{}_1C^{(i)}{}_2C^{(i)}{}_3\cdots C^{(i)}{}_{47}C^{(i)}{}_{48}$$

如表 3-4 所示，K 的第一位为 56 位密钥的第 14 位，K 的第 2 位为 56 位密钥的第 17

位，依次类推，K 的最后一位（第 48 位）为 56 位密钥的第 32 位。这样，就生成了一个 48 位使用密钥：

$$K^{(i)} = C^{(i)}_{14}C^{(i)}_{17}C^{(i)}_{11}\cdots C^{(i)}_{29}C^{(i)}_{32}$$
$$= K^{(i)}_{1}K^{(i)}_{2}K^{(i)}_{3}\cdots K^{(i)}_{47}K^{(i)}_{48}$$

这个密钥在加密运算中将与进行第 i 次迭代加密的数据进行按位异或。

表 3-4　56 位密钥 C 的位序与加密密钥 K 的位序对照表

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

第 A-5 步 迭代。

DES 算法密钥生成需要进行 16 次迭代，在完成 16 次迭代前，循环执行第 A-3 步和第 A-4 步，最终形成 16 套加密密钥：key[0], key[1], key[2],…, key[14], key[15]。

2）数据的加密操作

第 B-1 步 取得数据。

把明文数据分成 64 位的数据块，不够 64 位的数据块以适当的方式补足。

第 B-2 步 初始置换。

如表 3-5 所示，把输入的 64 位数据的原第 58 位换到第 1 位，原第 50 位换到第 2 位，依次类推，原第 7 位换到第 64 位，最后得到新的 64 位数据。

$$\text{oldData} = D_1D_2D_3\cdots D_{63}D_{64}$$
$$\text{newData} = D_{58}D_{50}D_{42}\cdots D_{15}D_7$$

表 3-5　初始置换表

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

注：表中每个单元格的位置排序是新数据的位序，下同。

第 B-3 步 数据扩展。

第 1 次迭代将第 B-2 步中生成的 newData 作为输入数据，第 i（$i > 1$）次迭代将第 i–1 次的 64 位输出数据作为输入数据，把 64 位数据按位置等分成左右两部分：

$$\text{newData} = D_1D_2D_3\cdots D_{63}D_{64}$$
$$\text{left} = I_1I_2I_3\cdots I_{31}I_{32} = D_1D_2D_3\cdots D_{31}D_{32}$$
$$\text{right} = R_1R_2R_3\cdots R_{31}R_{32} = D_{33}D_{34}D_{35}\cdots D_{63}D_{64}$$

保持 left 不变，根据表 3-6 把 right 由 32 位扩展置换成 48 位。在数据扩展操作中，有些输入数据位（如第 1、4、5、17、28、29、32 等数位）用了 2 次，因此数据得到了扩展。这样得到右半部分 $\text{right} = R_{32}R_1R_2\cdots R_{31}R_{32}R_1$，把扩展后的 48 位 right 与第 i 次迭代生成的 48 位加密密钥进行按位异或操作（$\text{right}^{(i)}\wedge = \text{key}[i]$）形成一个新的 48 位的 right，$\text{right} = R_1'R_2'R_3'\cdots R_{47}'R_{48}'$。

表 3-6　数据扩展对照表（输入数据位序/生成新数据位序）

32	1	2	3	4	5	4	5
6	7	8	9	8	9	10	11
12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21
22	23	24	25	24	25	26	27
28	29	28	29	30	31	32	1

第 B-4 步 数据压缩。

表 3-7 和表 3-8 所示为 *a*、*b* 对应的数据压缩置换表。*c*～*h* 对应的数据压缩置换表与表 3-8、表 3-9 形式完全相同，仅数值不同，为节省篇幅从略。

表 3-7　*a* 对应的数据压缩置换表

	1	2	3	4	5	6	7	8
1～8	e	0	4	f	d	7	1	4
9～16	2	e	f	2	b	d	b	e
17～24	3	a	a	6	6	c	c	b
25～32	5	9	9	5	0	3	7	8
33～40	4	f	1	c	e	8	8	2
41～48	d	4	6	9	2	1	b	7
49～56	f	5	c	b	9	3	7	e
57～64	3	a	a	0	5	6	0	d

注：阴影栏内是置换前的十进制数字，白色栏内是置换后的十六进制数字。

表 3-8　*b* 对应的数据压缩置换表

	1	2	3	4	5	6	7	8
1～8	f	3	1	d	8	4	e	7
9～16	6	f	b	2	3	8	4	f
17～24	9	c	7	0	2	1	d	a
25～32	c	6	0	9	5	b	a	5
33～40	0	d	e	8	7	a	b	1
41～48	a	3	4	f	d	4	1	2

续表

	1	2	3	4	5	6	7	8
49～56	5	b	8	6	c	7	6	c
57～64	9	0	3	5	2	e	f	9

注：阴影栏内是置换前的十进制数字，白色栏内是置换后的十六进制数字。

在第 B-3 步中形成了 48 位的 right， $\text{right} = R_1R_2R_3\cdots R_{47}R_{48}$ ，需要把 48 位的 right 转换成 32 位的 right。置换的方法如下。

第一步，先把 right 视为由 8 个 6 位二进制块组成的，即

$$a = a_1a_2\cdots a_6 = R_1R_2R_3R_4R_5R_6$$
$$b = b_1b_2\cdots b_6 = R_7R_8R_9R_{10}R_{11}R_{12}$$
$$c = c_1c_2\cdots c_6 = R_{13}R_{14}R_{15}R_{16}R_{17}R_{18}$$
$$d = d_1d_2\cdots d_6 = R_{19}R_{20}R_{21}R_{22}R_{23}R_{24}$$
$$e = e_1e_2\cdots e_6 = R_{25}R_{26}R_{27}R_{28}R_{29}R_{30}$$
$$f = f_1f_2\cdots f_6 = R_{31}R_{32}R_{33}R_{34}R_{35}R_{36}$$
$$g = g_1g_2\cdots g_6 = R_{37}R_{38}R_{39}R_{40}R_{41}R_{42}$$
$$h = h_1h_2\cdots h_6 = R_{43}R_{44}R_{45}R_{46}R_{47}R_{48}$$

$a, b,\cdots, h$ 都是 6 位二进制数，转换成十进制整数的值都应当不大于 64。

第二步，将 $a, b,\cdots, h$ 转换成十进制整数后，在对应的表中根据转换后的整数值取得对应位置的替代值。这些替代值都是一个十六进制的个位数，因此，每个替代值只占二进制数 4 位。

转换时各查各表，例如：

$a = 32$，则到表 3-7 中找到 32 的位置，把对应的替代值十六进制的 8 赋给 a；

$b = 53$，则到表 3-8 中找到 53 的位置，把对应的替代值十六进制的 c 赋给 b；

这样，将每 6 位用一个 4 位替换，就完成了从 48 位向 32 位数据的压缩置换。

有些资料中介绍 6 位转 4 位的实现方法与本书中所采用的不同，但殊途同归，最终的目的是相同的。

第 B-5 步 数据换位置换。

把第 B-4 步形成的 32 位 right：

$$\text{right} = R_1R_2R_3\cdots R_{31}R_{32}$$

根据表 3-9 进行转换：数据的原第 16 位换到第 1 位，原第 7 位换到第 2 位，依次类推，最后得到新的 32 位数据：

$$\text{right} = R_{16}R_7R_{20}\cdots R_4R_{25}$$

表 3-9 数据换位置换表

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

第 B-6 步　交换数据。

把 right 和 left 按位异或后的值赋给 right，然后将本轮输入的原始 right 值赋给 left。

第 B-7 步　迭代。

RES 算法需要进行 16 次迭代，在完成 16 次迭代前，把第 i–1 次得到的 left 和 right 的值作为第 i 次的输入数据，重复第 B-3 步到第 B-6 步。但是要记住一点：在第 B-3 步中第 i 次迭代选择第 i 次迭代生成的密钥与数据进行按位异或。

第 B-8 步　数据整理。

为保证加密和解密的对称性，每完成 RES 算法的前 15 次迭代中的 1 次迭代都要交换 left 和 right 的值，第 16 次迭代不交换两者的值。

至此，把 32 位的 left 和 right 合并成 64 位的 Data：

$$\text{Data}=D_1D_2D_3\cdots D_{63}D_{64}=I_1I_2I_3\cdots I_{32}R_1R_2R_3\cdots R_{32}$$

根据表 3-10 调整 Data 中各数据的位值，数据的原第 40 位换到第 1 位，原第 8 位换到第 2 位，依次类推，最后得到新的 64 位数据：

$$\text{Data}=D_{40}D_8D_{48}\cdots D_{57}D_{25}$$

表 3-10　数据整理表

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

经过了这么多次数学运算，我们最终得到的 Data 即密文。

3）数据的解密操作

数据解密的算法与加密算法基本相同，区别只在于第 B-3 步中和数据进行按位异或的密钥使用顺序不同，在加密中第 i 次迭代就采用第 i 次迭代生成的密钥和数据进行异或，而解密时第 i 次迭代采用第 17–i 次迭代生成的密钥和数据进行异或。

4．DES 算法的安全性和发展

DES 算法的安全性首先取决于密钥的长度。密钥越长，破译者利用穷举法破解密钥的难度就越大。根据当今计算机的处理速度和能力，56 位长度的密钥已经能够被破解，而 128 位的密钥则被认为是安全的，但随着时间的推移，这个数字也迟早会被突破。

另外，对 DES 算法进行某种变形和改进也是提高 DES 算法安全性的途径。

例如，演变出的 3-DES 算法使用了 3 个独立密钥进行三重 DES 加密，与 DES 算法相比，大大提高了安全性。如果 56 位 DES 算法密钥用穷举搜索来破译需要 2^{56} 次运算，而 3-DES 算法密钥则需要 2^{112} 次计算。

又如，独立子密钥 DES 算法由于每轮都使用不同的子密钥，意味着其密钥长度在 56

位的基础上扩大到 768 位。DES 算法还有 DESX 算法、CRYPT 算法、GDES 算法、RDES 算法等变形。这些变形和改进的目的都是加大破译难度及提高密码运算的效率。

3.3.2 国际数据加密算法

国际数据加密算法的第一版是由来学嘉和詹姆斯·梅西于 1990 年公布的，叫作推荐加密标准（Proposed Encryption Standard，PES）。在比哈姆和萨莫尔演示了差分密码分析之后，1991 年设计者为抵抗此攻击，增加了密码算法的强度，称新算法为改进型推荐加密标准（Improved Proposedtional Encryption Standard，IPES）算法，IPES 算法在 1992 年改名为国际数据加密算法（International Data Encryption Algorithm，IDEA）。

这里首先定义五种算法。

1．三种加密运算

（1）异或运算：按位做不进位加法运算，规则为

$$1 \oplus 0=0 \oplus 1=1，\ 0 \oplus 0=1 \oplus 1=0$$

（2）模 2^{16} 加运算 Ξ：16 位整数做加法运算，$X \Xi Y \equiv (X+Y) \bmod (2^{16})$。

（3）模 $2^{16}+1$ 乘运算 Θ：16 位整数做乘法运算，$X \Theta Y \equiv (X \times Y) \bmod (2^{16}+1)$。

2．两种子密钥运算

（1）16 位整数加法逆运算：X 的加法逆为 $-X$，即

$$X \Xi (-X) \equiv (X+(-X)) \bmod (2^{16}) \equiv 0$$

X 的加法逆为

$$-X = 2^{16} - X$$

（2）16 位整数乘法逆运算：X 的乘法逆为 X^{-1}，即

$$X \Theta X^{-1} \equiv (X \times X^{-1}) \bmod (2^{16}+1) \equiv 1$$

乘法逆的运算比较复杂，现举例说明。

例：求 X=43 679 的乘法逆 X^{-1}。

$$2^{16}+1 = 65\,537$$

$$65\,537=43\,679 \times 1+21\,858$$

$$43\,679=21\,858 \times 1+21\,821$$

$$21\,858=21\,821 \times 1+37$$

$$21\,821=37 \times 589+28$$

$$37=28 \times 1+9$$

$$28=9 \times 3+1$$

所以

$$\begin{aligned} 1 &= 28-9 \times 3 \\ &= 28-3 \times (37-28) \\ &= 4 \times 28-3 \times 37 \\ &= 4 \times (21\,821-37 \times 589)-3 \times 37 \\ &= 4 \times 21\,821-(4 \times 589+3) \times 37 \end{aligned}$$

=4×21 821−2359×(21 858−21 821)
=2363×21 821−2359×21 858
=2363×(43 479−21 858)−2359×21 858
=2363×43 679−(2363+2359)×21 858
=2363×43 679−(2363+2359)×(65 537−43 679)
=7085×43 679−4722×65 537

即 $7085\times43\,679\equiv1 \bmod (2^{16}+1)\equiv1$，所以，$X$=43 679 的乘法逆 $X^{-1}=7085$。

对于模 $2^{16}+1$ 乘运算 Θ，由于 0 用 $2^{16}\equiv-1$ 来表示，因此 0 的乘法逆是 0。

3．IDEA 的描述

把明文分成多个 64 位的分组，使用 128 位的密钥对每个 64 位分组加密。

把 64 位分组分成 4 个 16 位子分组，由 128 位的密钥产生 52 个 16 位的子密钥，即首先将 128 位密钥分成 8 个 16 位子密钥（前 6 个用于第 1 轮加密，后 2 个用于第 2 轮加密）；然后密钥向左环移 25 位，分成 8 个子密钥（前 4 个用于第 2 轮加密，后 4 个用于第 3 轮加密），如此 6 次，将产生 48 个子密钥；密钥再次向左环移 25 位，只在前 64 位生成 4 个 16 位子密钥，合计 52 个子密钥，完成加密全过程。IDEA 的加密运算都在 16 位子分组上运行，只使用 3 种加密运算算法，而没有位置换。

加密子密钥如表 3-11 所示。

表 3-11　加密子密钥

轮　数	加密子密钥					
1	Z(1,1)	Z(1,2)	Z(1,3)	Z(1,4)	Z(1,5)	Z(1,6)
2	Z(2,1)	Z(2,2)	Z(2,3)	Z(2,4)	Z(2,5)	Z(2,6)
3	Z(3,1)	Z(3,2)	Z(3,3)	Z(3,4)	Z(3,5)	Z(3,6)
4	Z(4,1)	Z(4,2)	Z(4,3)	Z(4,4)	Z(4,5)	Z(4,6)
5	Z(5,1)	Z(5,2)	Z(5,3)	Z(5,4)	Z(5,5)	Z(5,6)
6	Z(6,1)	Z(6,2)	Z(6,3)	Z(6,4)	Z(6,5)	Z(6,6)
7	Z(7,1)	Z(7,2)	Z(7,3)	Z(7,4)	Z(7,5)	Z(7,6)
8	Z(8,1)	Z(8,2)	Z(8,3)	Z(8,4)	Z(8,5)	Z(8,6)
输出变换	Z(9,1)	Z(9,2)	Z(9,3)	Z(9,4)		

解密子密钥仍为 52 个，其要么是加密子密钥的加法逆，要么是加密子密钥的乘法逆。解密子密钥如表 3-12 所示。

表 3-12　解密子密钥

轮　数	解密子密钥					
1	$(Z(9,1))^{-1}$	−Z(9,2)	−Z(9,3)	$(Z(9,4))^{-1}$	Z(8,5)	Z(8,6)
2	$(Z(8,1))^{-1}$	−Z(8,3)	−Z(8,2)	$(Z(8,4))^{-1}$	Z(7,5)	Z(7,6)
3	$(Z(7,1))^{-1}$	−Z(7,3)	−Z(7,2)	$(Z(7,4))^{-1}$	Z(6,5)	Z(6,6)
4	$(Z(6,1))^{-1}$	−Z(6,3)	−Z(6,2)	$(Z(6,4))^{-1}$	Z(5,5)	Z(5,6)
5	$(Z(5,1))^{-1}$	−Z(5,3)	−Z(5,2)	$(Z(5,4))^{-1}$	Z(4,5)	Z(4,6)

续表

轮　数	解密子密钥					
6	$(Z(4,1))^{-1}$	$-Z(4,3)$	$-Z(4,2)$	$(Z(4,4))^{-1}$	Z(3,5)	Z(3,6)
7	$(Z(3,1))^{-1}$	$-Z(3,3)$	$-Z(3,2)$	$(Z(3,4))^{-1}$	Z(2,5)	Z(2,6)
8	$(Z(2,1))^{-1}$	$-Z(2,3)$	$-Z(2,2)$	$(Z(2,4))^{-1}$	Z(1,5)	Z(1,6)
输出变换	$(Z(1,1))^{-1}$	$-Z(1,2)$	$-Z(1,3)$	$(Z(1,4))^{-1}$		

注：有些文献中解密密钥第 2～8 轮的 2、3 子密钥的顺序与本表不同。

4．IDEA 的算法流程

IDEA 的算法流程如图 3-5 所示。64 位分组分成 4 个 16 位子分组，这 4 个子分组为算法的第 1 轮输入，总共有 8 轮。在每一轮中，这 4 个子分组之间相异或、相加、相乘，且与 6 个 16 位子密钥相异或、相加、相乘。在轮与轮间，第 2 和第 3 个子分组交换。在最后输出变换中，4 个子分组与 4 个子密钥进行运算。在每轮运算中，执行如下顺序：

（1） X_1 和第 1 个子密钥相乘；

（2） X_2 和第 2 个子密钥相加；

（3） X_3 和第 3 个子密钥相加；

（4） X_4 和第 4 个子密钥相乘；

（5）将第（1）步和第（3）步的结果相异或；

（6）将第（2）步和第（4）步的结果相异或；

（7）将第（5）步的结果与第 5 个子密钥相乘；

（8）将第（6）步和第（7）步的结果相加；

（9）将第（8）步的结果与第 6 个子密钥相乘；

（10）将第（7）步和第（9）步的结果相加；

（11）将第（1）步和第（9）步的结果相异或；

（12）将第（3）步和第（9）步的结果相异或；

（13）将第（2）步和第（10）步的结果相异或；

（14）将第（4）步和第（10）步的结果相异或。

每一轮的输出是第（11）步、第（12）步、第（13）步和第（14）步的结果形成的 4 个子分组。将中间 2 个分组交换（最后 1 轮除外），即下一轮的输入。

经过 8 轮运算之后，有 1 个最终的输出变换：

（1） X_1 和 Z(9, 1)子密钥相乘；

（2） X_2 和 Z(9, 2)子密钥相加；

（3） X_3 和 Z(9, 3)子密钥相加；

（4） X_4 和 Z(9, 4)子密钥相乘。

最后，将这 4 个子分组重新连接到一起产生密文。

解密过程与加密过程基本一样，只是使用对应的解密子密钥。

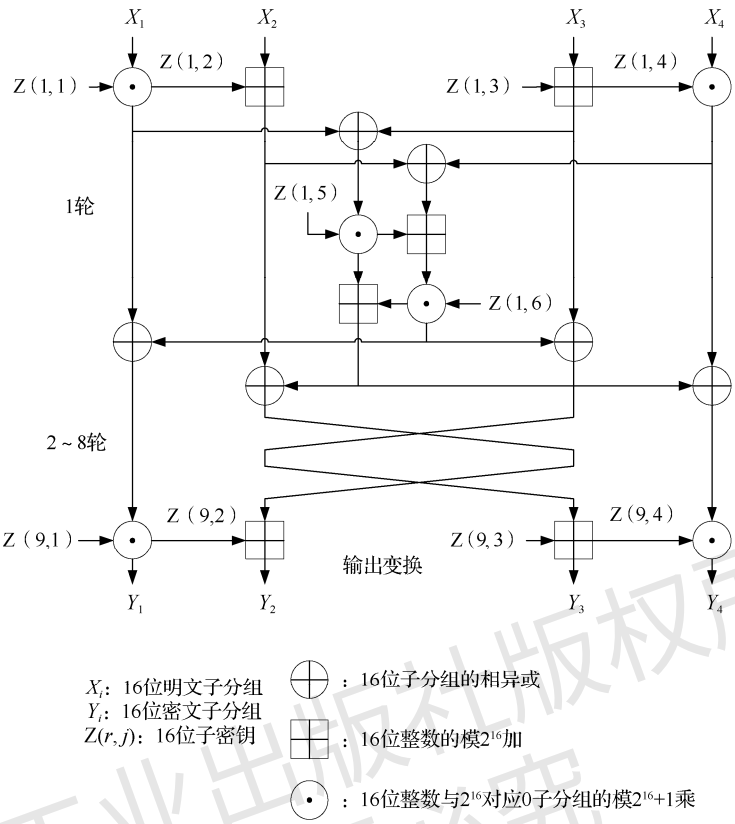

图 3-5　IDEA 的算法流程

5．IDEA 的安全性

IDEA 的密钥为 128 位（DES 算法的密钥为 56 位），抗穷举攻击能力强；IDEA 已改进了抗差分密码分析攻击能力，在 IDEA 的 8 轮运算中，第 4 轮运算后，就对差分密码分析免疫了。到目前为止，IDEA 是安全的。

3.4　非对称密码技术

3.4.1　RSA 密码体制

回顾图 3-1 展示的信息加密解密的传输模型，当加密密钥与解密密钥一致时，对明文加密的过程称为对称加密。这一过程存在的问题在于，对称密钥如何从发送者传输给接收者。如果在信道上被攻击者截获，对称加密也就失去了加密的意义。这个问题推动了非对称加密技术 RSA 算法的诞生与发展。

1．RSA 算法的历史

1976 年以前，所有的加密方法都是同一种模式：

（1）甲方选择某一种加密规则，对信息进行加密；

（2）乙方使用同一种规则，对信息进行解密。

由于加密和解密使用同样规则（密钥），因此这种加密模式被称为“对称加密算法”（Symmetric-key Algorithm）。这种加密模式有一个最大弱点：甲方必须把加密规则告诉乙方，否则无法解密。保存和传递密钥就成了最令人头疼的问题。

1976 年，两位美国计算机学家惠特菲尔德·迪菲和马丁·赫尔曼，提出了一种崭新构思，即“迪菲-赫尔曼密钥交换算法”，也称为 Diffie-Hellman 密钥交换算法，利用该算法可以在不直接传递密钥的情况下，完成解密。这个算法启发了其他科学家，他们认识到，加密和解密可以使用不同的规则，这两种规则之间存在某种对应关系即可，这样就避免了直接传递密钥。这种新的加密模式被称为“非对称加密算法”。

（1）乙方生成两把密钥（公开密钥和私人密钥，下面简称公钥和私钥）。公钥是公开的，任何人都可以获得；私钥则是保密的。

（2）甲方获取乙方的公钥，然后用它对信息加密。

（3）乙方得到加密后的信息，用私钥解密。

如果公钥加密的信息只有私钥解得开，那么只要私钥不泄露，通信就是安全的。

1977 年，三位数学家罗纳德·李维斯特、阿迪·萨莫尔和伦纳德·阿德曼设计了一种算法，可以实现非对称加密，这种算法以他们三个人的名字命名，叫作 RSA 算法。直到现在，RSA 算法一直是最广为使用的非对称加密算法。毫不夸张地说，只要有计算机网络的地方，就有 RSA 算法。

RSA 算法使用的是大素数相乘容易而大整数分解难的思想。下面是 RSA 算法的例子。

（1）选择两个大素数 p、q。为便于理解，这里令 p=5，q=11。

（2）计算 $n=pq$=55，以及欧拉函数 $\varphi(n)=(p-1)(q-1)=4\times10=40$。

（3）随机选择一个$(1, \varphi(n))$范围内的正整数 e，使最大公约数 $\gcd(e, \varphi(n))=1$，如 e=7。实际中尽可能大，如素数 65 537。

（4）计算 e 对于$\phi(n)$的模反元素 d。“模反元素”就是指有一个整数 d，可以使 ed 被$\phi(n)$除的余数为 1。

$$ed \equiv 1(\mathrm{mod}(\varphi(n))$$

等价于

$$ed-1 \equiv k\varphi(n)$$

这里 d=23。

（5）公钥(e, n)，这里是(7,55)。

（6）私钥 d，这里是 23。

数据加密和解密的过程如图 3-6 所示。

我们可以看到，如果不知道 d，就没有办法从 c 求出 m。而前面已经说过，要知道 d 就必须分解 n，这是极难做到的，所以 RSA 算法保证了通信安全。

读者可能会产生疑问：公钥(n, e)只能加密小于 n 的整数 m，如果要加密大于 n 的整数，该怎么办。有两种解决方法：一种是把长信息分割成若干段短消息，对每段分别加密；另一种是先选择一种对称加密算法（如 DES 算法），用这种算法的密钥加密信息，再用 RSA 算法的公钥加密 DES 算法的密钥。

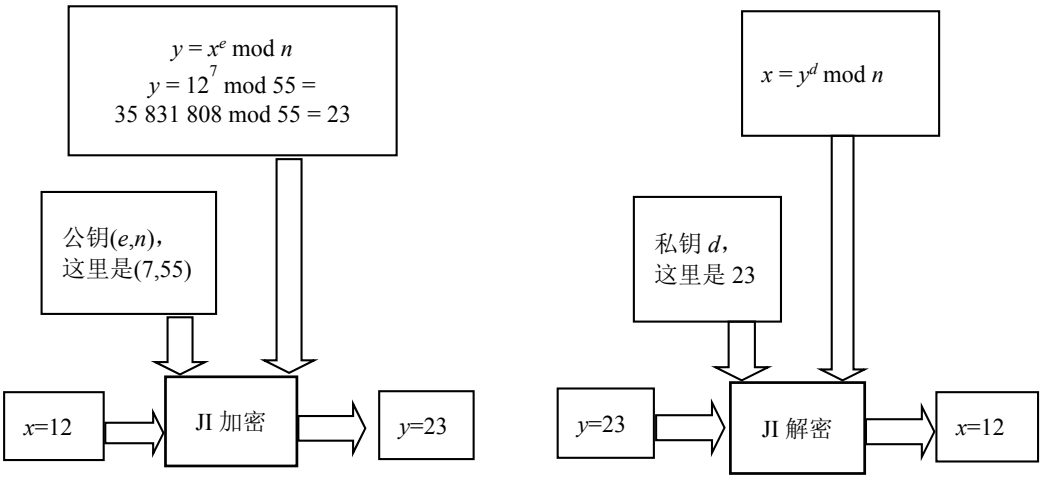

图 3-6　数据加密和解密的过程

这种算法非常可靠，密钥越长，它就越难破解。根据相关文献，目前被破解的最长 RSA 密钥是 768 个二进制位。也就是说，长度超过 768 位的密钥，尚无法被破解（至少没人公开宣布）。因此可以认为，1024 位的 RSA 密钥基本安全，2048 位的密钥极其安全。

2．RSA 算法的可靠性

回顾上面的密钥生成步骤，一共出现六个数字：p、q、n、$\varphi(n)$、e、d 。

这六个数字之中，公钥用到了两个（n 和 e），其余四个数字都是不公开的。其中最关键的是 d，因为 n 和 d 组成了私钥，一旦 d 泄露，就等于私钥泄露。

读者可以思考，有无可能在已知 n 和 e 的情况下推导出 d。

（1）$ed \equiv 1(\mathrm{mod}(\varphi(n))$，只有知道 e 和 $\phi(n)$，才能算出 d。

（2）$\varphi(n) = (p-1)(q-1)$，只有知道 p 和 q，才能算出 $\phi(n)$。

（3）$n = pq$，只有将 n 因数分解，才能算出 p 和 q。

结论：如果 n 可以被因数分解，d 就可以被算出，也就意味着私钥被破解。

可是，大整数的因数分解是一件非常困难的事情。目前除了暴力破解，还没有发现别的有效方法。维基百科这样写道：“对极大整数做因数分解的难度决定了 RSA 算法的可靠性。换言之，对一极大整数做因数分解越困难，RSA 算法越可靠。”

假如有人找到一种快速因数分解的算法，那么 RSA 算法的可靠性就会大幅下降。但找到这样的算法的可能性是非常小的。今天只有短的 RSA 算法的密钥才可能被暴力破解。只要密钥足够长，用 RSA 算法加密的信息实际上是不能被解破的。

举例来说，你可以对 3233 进行因数分解（61×53），但是你没法对下面这个整数进行因数分解。

1230186684530117755130494958384962720772853569595334
7921973224521517264005072636575187452021997864693899
564749427740638459251925573

326303453731548268507917026122142913461670429214311602222124047927473779408066535141959745985690214341333478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489×367460436667995904282446337996279526322791581643430876426760322838157396665112792333734171433968102700927987363089173

事实上，这大概是人类已经分解的最大整数（232 个十进制位，768 个二进制位）。比它更大的因数分解，还没有被报道过，因此目前能被破解的最长 RSA 密钥就是 768 位。

3．私钥解密的证明

最后，我们来证明，为什么用私钥解密，一定可以正确地得到 m，也就是证明：

$$cd \equiv m(\mathrm{mod}\, n)$$

根据加密规则

$$me \equiv c(\mathrm{mod}\, n)$$

c 可以写成下面的形式：

$$c = me - kn$$

将 c 代入我们要证明的那个解密规则：

$$(me - kn)d \equiv m(\mathrm{mod}\, n)$$

它等同于求证

$$med \equiv m(\mathrm{mod}\, n)$$

由于

$$ed \equiv 1[\mathrm{mod}\, \phi(n)]$$

所以

$$ed = h\phi(n) + 1$$

即求证

$$m(h\phi(n) + 1) \equiv m(\mathrm{mod}\, n)$$

接下来，分成两种情况证明上面这个式子。

（1）m 与 n 互质。根据欧拉定理，此时

$$m\phi(n) \equiv 1(\mathrm{mod}\, n)$$

得到

$$[m\phi(n)]h \times m = m(\mathrm{mod}\, n)$$

原式得证。

（2）m 与 n 不是互质关系。此时，由于 n 等于质数 p 和 q 的乘积，所以 m 必然等于 kp 或 kq。

以 $m = kp$ 为例，考虑到这时 k 与 q 必然互质，根据欧拉定理，下面的式子成立：

$$(kp)q - 1 = 1(\mathrm{mod}\, q)$$

进一步得到

$$[(kp)q - 1]h(p-1) \times kp \equiv kp(\mathrm{mod}\, q)$$

即

$$(kp)ed \equiv kp(\mathrm{mod}\, q)$$

将它改写成下面的等式

$$(kp)ed \equiv tq + kp$$

这时 t 必然能被 p 整除，即 $t = t'p$

$$(kp)ed = t'pq + kp$$

因为 $m = kp, n = pq$，所以

$$med \equiv m(\mathrm{mod}\, n)$$

原式得证。

3.4.2　Rabin 密码体制

Rabin 算法是一种基于模平方和模平方根的非对称加密算法。$a = x^2 \Leftrightarrow x = \sqrt{a}$，称 a 为 x 的算术平方，称 x 为 a 的算术平方根。

$$a \equiv x^2 \bmod m \Leftrightarrow x \equiv \sqrt{a} \bmod m$$

称 a 为 x 模 m 时的平方，称 x 为 a 模 m 时的平方根。

设私钥 p、q 为两素数，公钥 n=p×q。对于明文 m 和密文 c，定义以下加密过程（公钥加密过程）：

$$c_i = m_i^2 \bmod n$$

对应的解密过程相当于求解以下的同余方程：

$$m_i^2 \equiv c_i \bmod n$$

根据“中国剩余定理”，当且仅当模互素时同余方程有解，因此上述同余方程不可解（公钥无法解密公钥加密后的密文）。

但使用私钥 p、q，可以通过以下方式得到明文。

（1）令 $\begin{cases} m_{pi} = \sqrt{c_i} \bmod p \\ m_{qi} = \sqrt{c_i} \bmod q \end{cases}$，其中等式右边表示模平方根；

（2）根据扩展欧几里得算法 $y_p \cdot p + y_q \cdot q = 1$，得到 p、q 的一个线性表示；

（3）可以证明每一个密文对应四个原文，分别记为 r、$-r$、s 和$-s$，且有

$$r=(y_p\cdot p\cdot m_p+y_q\cdot q\cdot m_p)\bmod n$$
$$-r=n-r$$
$$s=(y_p\cdot p\cdot m_q-y_q\cdot q\cdot m_p)\bmod n$$
$$-s=n-s$$

当然，Rabin 算法具有其致命的缺陷：一个密文对应四个明文。但此算法仍然包含了密码学中的基本概念和技巧，如单向函数、整数的因数分解等。

Rabin 算法的安全性基于整数的因式分解问题：只有将公钥 n 正确分解为私钥 p、q 后，才可以将公钥加密后的密文还原为原文，而通常 p、q 都会取相当大的素数，因此 n 也是一个非常大的数字；数字越大，其因式分解越困难。

3.4.3 ElGamal 密码体制

ElGamal 加密方法是一种用于对采用迪菲-赫尔曼方式进行交换的公钥进行加密的非对称加密方法。

在群论中，循环群（Cyclic Group）是指能由单个特殊元素生成的群。生成循环群的单个特殊元素 g 称为生成元（Generator），群中元素的个数称为阶（Order）。

1．ElGamal 加密

设 n 为素数，定义集合 $\{g^i\mid i=0,1,\cdots,n-1\}$ 上的乘法运算：

$g_i\neq g_j\in G$，$g_i\times g_j=g^i\times g^j=g^{(i+j)\bmod n}$，则乘法对集合成群，称作由生成元 g 生成的 n 阶乘法循环群，记作 $\mathrm{MG}^{(n)}$。乘法循环群，如 $\{g^0,g^1,g^2,\cdots,g^{n-1}\}$，其中 g^0 称为单位元，记作 e。

设 n 为素数，定义集合 $\{g\times i\mid i=0,1,\cdots,n-1\}$ 的加法运算：

$g_i+g_j=g\times i+g\times j=g\times(i+j)\bmod n=(i+j)g\bmod n$，则加法对集合成群，称作由生成元 g 生成的 n 阶加法循环群。加法循环群，如 $\{0,g,2g,\cdots,(n-1)g\}$，其循环群的性质与生成元 g 无关。有时由特殊生成元可以生成特殊的循环群，例如，当 g=1 时，加法循环群 $\{0,1,2,\cdots,n-1\}$ 由 0 和前 n−1 个正整数组成，称作 n 阶基础加法循环群，记作 $\mathrm{BAG}(n)$。

可以证明，给定对应法则 f 满足：

$$\forall i\in\mathrm{BAG}(n)\xrightarrow{f}g^i\in\mathrm{MG}^{(n)}$$
$$\forall g^i\in\mathrm{MG}^{(n)}\xrightarrow{f^{-1}}i\in\mathrm{BAG}(n)$$

任意 n 阶乘法循环群同构于同阶基础加法循环群。

2．ElGamal 解密

例如，若 $G=\{e,g_1,g_2,g_3,g_4,g_5\}$，则 G 为循环的，且 G 同构于模 6 的加法群：{\overline{0}, \overline{1}, \overline{2}, \overline{3}, \overline{4}, \overline{5}}。

在乘法循环群的基础上可以定义指数方幂运算及其逆运算——对数运算，此时，指数方幂运算和对数运算是代数运算中指数和对数定义在同余下的推广：

$$k=\log_l x\Leftrightarrow l^k=x$$

称 k 为 x 以 l 为底的对数，记为 $\log_l x$。

$$\ln d_l x = k \bmod \phi(m) \Leftrightarrow l^k \equiv x \bmod m$$

称 k 为 x 以 l 为底，模 $\phi(m)$ 时的离散对数，记为 $\ln d_l x$ 。

离散对数假设目前没有很好的算法计算离散对数，即计算离散对数几乎不可能。ElGamal 方法可以在任何循环群中实现，其安全强度依赖于循环群上离散对数假设的强度。

3．密钥生成

假设爱丽丝和鲍勃为通信的双方，通信发起方爱丽丝按以下方法生成公钥：

爱丽丝通过生成元 g 和阶 q 定义一个乘法循环群 G；

爱丽丝在集合 R={0, 1, 2,…, q−1}中随机选择一个整数 x；

爱丽丝根据群 G 的生成元和阶生成群中的一个元素 h，h = g^x\,；

爱丽丝将{G, q, g, h}作为公钥发布，x 作为私钥妥善保存。

（1）加密过程。通信接收方的鲍勃在加密过程中通过公钥{G, q, g, h}对明文 m 进行加密（其中 1～3 步可以事先完成）：

鲍勃在集合 R={0, 1, 2,…, q−1}中随机选择一个整数 y；

鲍勃根据{G, q, g, h}生成群中的一个元素 c_1=g^y\,；

鲍勃根据 s=h^y\,得到对称密钥（由于 Bob 每次接收消息后都会生成 s，因此 s 也称为临时密钥）；

鲍勃将明文 m 转换为群 G 中的一个元素 m'\,（如将特定信息进行编码）；

鲍勃计算 c_2=m'\cdots ；

鲍勃将(c_1，c_2)\,作为密文发送。

（2）解密过程。爱丽丝使用私钥 x 对密文(c_1，c_2)\,进行解密，步骤为：

爱丽丝计算 s=c_1^x\,；

爱丽丝计算群中的元素 m'=c_2 \cdot s^{−1}\,，并将其还原为明文（将编码还原为信息）。

下述等式保证了爱丽丝计算出的编码与鲍勃转换的编码相同：

c_2 \cdot s^{−1} = m'\cdot h^y \cdot (g^{xy})^{−1} = m'\cdot g^{xy} \cdot g^{−xy} = m'

正如开篇介绍的那样，ElGamal 加密方法通常用于复合加密通信中，采用这种通信方式时，消息本身采用对称加密方式，对称加密所采用的密钥采用 ElGamal 方法加密，然后使用迪菲-赫尔曼方法进行传送（消息长度通常远远大于密钥长度）。

4．安全性保证

（1）如果迪菲-赫尔曼计算假设在群 G 上成立，则加密方法是单向方法；

（2）如果迪菲-赫尔曼判定假设在群 G 上成立，则 ElGamal 加密方式是语义安全（Semantic Security）的（单纯的迪菲-赫尔曼计算无法保证这一点）；

（3）ElGamal 加密可通过选择密文攻击方式篡改（或捏造）信息。例如，给定明文 m 及其密文(c1, c2)，篡改者可以在不知道明文 m 的条件下直接构造出 2m 的密文(c1, 2c2)。因此在实际使用时必须对方法进行修改以避免选择密文攻击，这种修改有时会破坏迪菲-赫尔曼判定。

（4）效率：ElGamal 方法中 1 个明文对应 2 个加密结果（g^a 和 g^b），因此密文空间的大小是明文空间大小的 2 倍，也就是说纵观整个通信过程，收发密文的大小是实际明文大小的 2 倍。

5．迪菲-赫尔曼密钥交换方法及其系列假设

迪菲-赫尔曼密钥交换方法（简称 DH 方法）是一种在不安全信道上交换密钥的算法，协议使用一个素数 p 的整数模 n 乘法群及其原根 g，方法的理论依据就是离散对数问题。假设爱丽丝和鲍勃为交换密钥的双方，则 DH 方法的步骤采用数学语言描述如下：

爱丽丝和鲍勃写上一个有限循环群 G 和它的一个生成元 g（通常在协议开始以前就已经规定好，且 g 是公开的，并可以被所有攻击者看到）。

爱丽丝选择一个随机自然数 a 并且将 $ga \bmod p$ 发送给鲍勃。

鲍勃选择一个随机自然数 b 并且将 $gb \bmod p$ 发送给爱丽丝。

爱丽丝计算 $(gb)a$。

鲍勃计算 $(ga)b$。

由于 mod p 运算下的 gab 和 gba 相等，因此爱丽丝和鲍勃得到相同的结果，并将其作为对称密钥使用。

下面是一个实际的例子：

爱丽丝和鲍勃决定使用 p=23 及 g=5。

爱丽丝选择一个秘密整数 a=6，计算 $A = ga \bmod p$ 并发送给鲍勃。

$$A = 56 \bmod 23 = 8$$

鲍勃选择一个秘密整数 b=15，计算 $B = gb \bmod p$ 并发送给爱丽丝。

$$B = 515 \bmod 23 = 19$$

爱丽丝计算 $s = B\text{a} \bmod p$，196 mod 23 = 2。

鲍勃计算<$s = Ab \bmod p$，815 mod 23 = 2。

当然，为了使这个例子变得安全，必须使用非常大的 a、b 及 p，否则可以遍历所有 gab mod 23 的可能取值（总共有最多 22 个这样的值，就算 a 和 b 很大也无济于事）来破解。如果 p 是一个至少 300 位的质数，并且 a 和 b 至少有 100 位长，那么即使使用世界上所有的计算资源和当今最好的算法也不可能从 g、p 和 $ga \bmod p$ 中计算出 a。这个问题就是著名的离散对数问题。

注意，g 不需要很大，并且在一般的实践中通常是 2 或者 5。

6．迪菲-赫尔曼计算假设

迪菲-赫尔曼计算假设（也称循环群计算性假设，简称 CDH 假设）认为循环群是难以计算的。也就是说，对于一个 q 阶循环群 G，在给定 g 、g^a 、g^b 的情况下难以计算出 g^{ab}，其中 g 为群 G 的生成元，$a,b \in \{0,\cdots,q-1\}$。CDH 假设的数学依据是目前暂时没有找到求解离散对数的有效算法，一旦找到这样的算法，不但 CDH 假设不成立，而且所有建立在 CDH 假设基础上的密码系统也将被破解。

7．迪菲-赫尔曼判定假设

与 CDH 假设密切相关的是另一个假设——迪菲-赫尔曼判定假设（简称 DDH 假设），DDH 假设认为任意给定 q 阶循环群 G 中的任意两个元素 g^a 和 g^b，其中 $a,b \in \{0,\cdots,q-1\}$，g^{ab} 看起来就像 G 中的任意一个元素，换句话说，DDH 假设认为无法区分三元组 $\{g^a, g^b, g^{ab}\}$ 中的元素：从三元组 $\{g^a, g^b, g^{ab}\}$ 中分别随机抽取的两个元素为 a 和 b 的概率与从三元组

$\{g^a, g^b, g^{ab}\}$ 中随机抽取的三个元素为 a、b 和 c 的概率一样。三元组 $\{g^a, g^b, g^{ab}\}$ 因此被称为 DDH 三元组。

8．CDH 假设、DDH 假设和离散对数假设之间的关系

CDH 假设、DDH 假设和离散对数假设是三个相互关联的假设，CDH 假设和 DDH 假设都建立在离散对数假设的基础上，一旦离散对数假设被证明不成立，则 CDH 假设和 DDH 假设都不成立；相比之下，CDH 假设几乎等价（或稍强）于离散对数假设，而 DDH 假设强于 CDH 假设（如果存在 DDH 假设不成立的群，则 CDH 假设在这个群中仍成立）。

与基于 CDH 假设的加密系统可以在循环群中实现不同，基于 DDH 假设的加密系统只能在 DDH 假设成立的群中实现。

3.5　本章实验——磁盘文件加密系统

在微软操作系统系列中，NTFS 是在 Windows NT 操作系统上诞生和发展的磁盘文件系统，在 Windows 7 及以上的版本，NTFS 提供了应用非对称加密技术的加密文件系统（Encrypting File System，EFS）。这是一件有趣的事情，本次实验将展现如何对 Windows 7 上的文件进行公钥加密和私钥解密。本章实验环境要求：Windows 7 及以上操作系统版本，其他条件不限。

首先，在桌面新建一个需要加密的文本文档，将其命名为“机密”，并在文档中输入内容并保存。如图 3-7 所示。

接着，右击文件，选择“属性”选项，如图 3-8 所示。

图 3-7　新建并保存文本文档

图 3-8　右击文件并选择“属性”选项

单击“高级”按钮，弹出“高级属性”对话框，如图 3-9 所示，在原有基础上，勾选

“加密内容以便保护数据”复选框后单击“确定”按钮，并在“机密.txt 属性”对话框中单击“确定”按钮。至此，文件内容得以加密。这个时候会弹出“加密警告”对话框，如图 3-10 所示，按照需求选择后单击“确定”按钮即可。

图 3-9 “高级属性”对话框

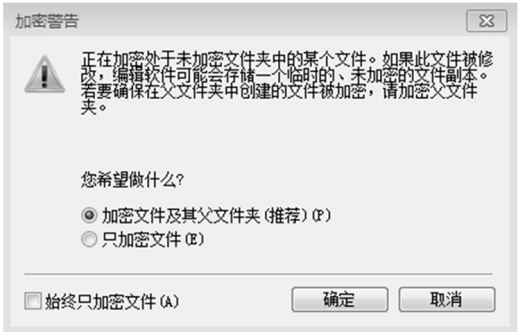

图 3-10 “加密警告”对话框

接下来，在“高级属性”对话框中单击“详细信息”按钮，选中用户，单击“备份密钥”按钮，弹出如图 3-11 所示的对话框。在进一步的证书导出向导中，将用于访问该机密记事本文档的证书文件导出，导出的文件扩展名选择.pfx，这是包括公钥和私钥的一套证书。此外，可以设置安装该证书时的口令。

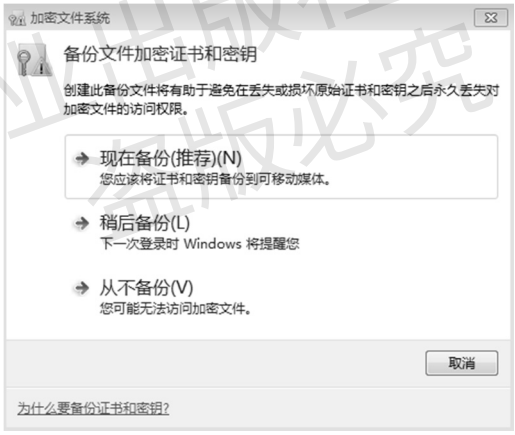

图 3-11 备份秘钥

新建用户、注销或者切换当前用户到新建的用户中，尝试在没有证书的情况下打开加密文件，弹出拒绝访问提示框，如图 3-12 所示。

图 3-12 访问加密文件被拒绝

之后，在勾选“加密内容以便保护数据”复选框并单击“确定”按钮后，会自动弹出“备份提示”提示框。单击“确定”按钮并输入口令，并将密钥保存于 U 盘，多次单击“下一步”按钮，弹出“导出成功”提示框，则表示加密成功且证书已导入计算机中。此时，尝试两种场景：一种场景是切换用户，新用户因为没有此 U 盘私钥，不能访问打开该文件；另一种场景是，若在原有用户环境中，拔去 U 盘并删除证书，则该文件同样拒绝访问。

3.6　本章习题

3.6.1　基础填空

（1）密码学是研究信息系统安全保密的科学，包含两个分支：密码编码学，对信息进行编码实现隐蔽信息；_____，研究分析破译密码。密码算法是用于加密和解密的_____，密码算法是_____的基础。

（2）传统加密技术有单表代换密码、多表代换密码、_____、轮转密码等。其中，轮转密码是用一组转轮或接线编码轮所组成的机器，用以实现长周期的_____。

（3）网络安全通信中要用到两类密码算法，一类是对称密码算法，另一类是_____。对称算法又可分为两类：一类算法是一次只对_____中的单个位（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位进行运算，这些位组称为分组，相应的算法称为_____。

3.6.2　概念简答

（1）请以重要事件为线索，简述密码学的三个发展阶段。

（2）请结合算法框架、流程步骤、安全性等比较对称加密技术中 DES 算法和 IDEA。

（3）请结合 RSA 算法的原理和过程简要论证其可靠性。

3.6.3　上机实践

（1）请自行选择一种高级编程语言，编写代码实现恺撒密码，并思考优化方案。

（2）请参考本章实验的加密解密步骤，对其他类型的文件进行公钥加密和私钥解密，完成实验报告。