第3章

对称密钥算法与 AES

3.1 简　　介

我们已经介绍了密码学、加密和解密的基本概念。计算机能够方便快捷地进行加密。几年来，对称密钥加密已经相当普及。最近已经和非对称密钥加密融合起来了。但是，稍后将会介绍，大多数实际应用都是对称与非对称密钥加密的组合。

本章介绍对称密钥加密的不同方面，介绍不同对称密钥加密算法类型与模式，还要介绍链接的作用。

本章还介绍几个对称密钥加密算法的细节，使读者了解这些算法的工作原理，包括 DES 及其变体、IDEA、RC5 与 Blowfish。我们还要介绍 Rijndael 算法，美国政府批准其为高级加密标准(AES)。

学完本章后，读者就可以完全了解对称密钥加密的工作原理。但是，如果读者不想了解这些算法的细节，则可以跳过相关部分，不会破坏连贯性。

3.2 算法类型与模式

介绍实际基于计算机的加密算法之前，我们要介绍这些算法的两个关键方面：**算法类型**与**算法模式**。算法类型定义算法每一步要加密的明文长度，算法模式定义具体类型中的加密算法细节。

3.2.1 算法类型

我们已经介绍明文消息变成密文消息，也介绍了进行这些变换的不同方法。不管用哪种方法，广义上从明文生成密文的方法有两种：**流加密法**(stream ciphers)与**块加密法**(block ciphers)，如图 3.1 所示。

流加密法

流加密法是一次加密明文中的一个位。假设原先的明文消息为 ASCII(文本格式)的 Pay 100，则将这些 ASCII 字符变成相应二进制值时，可以假设其变成 01011100(为了简单

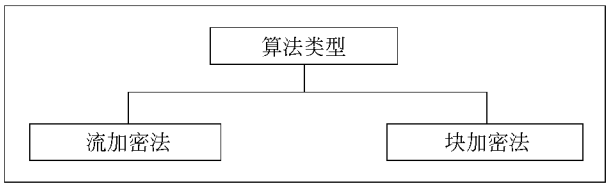

图 3.1 加密类型

起见，我们作此假设，实际上，由于每个字符占 7 个位，因此二进制文本更大）。假设采用的密钥为二进制值 10010101，再假设我们的加密算法使用异或逻辑。异或逻辑很容易理解，如图 3.2 所示。简单地说，只有一个输入为 0、一个输入为 1 时，异或运算才能得到 1，否则输出为 0。

图 3.3 显示了异或逻辑的结果。

输入 1	输入 2	输入 3
0	0	1
0	1	1
1	0	1
1	1	0

图 3.2 异或逻辑

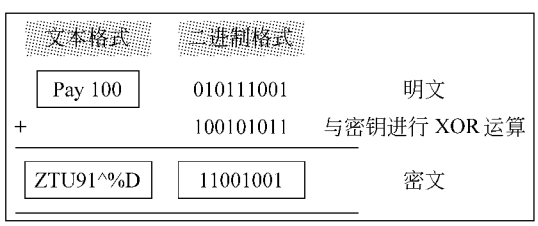

图 3.3 流加密法

对原消息中每个位采用密钥的一位后，假设密文为二进制值 11001001（文本值为 ZTU91 ^%）。注意，明文的位是逐位加密的，因此，传输的是二进制值 11001001，但其 ASCII 值为 ZTU91 ^%，攻击者搞不懂，因此保护了信息。

流加密技术一次加密明文中的一个位，解密时也是一位一位地进行。

异或逻辑的另一个有趣属性是，再用一次时，可以恢复原先的数据。例如，假设有两个二进制值 A =101 与 B=110，A 和 B 进行异或操作得到 C：

$$C = A \text{ XOR } B$$

因此：

$$C = 101 \text{ XOR } 110$$
$$= 011$$

如果 C 与 A 进行异或操作，则得到 B，即：

$$B = 011 \text{ XOR } 101$$
$$= 110$$

同样，如果 C 与 B 进行异或操作，则得到 A，即：

$$A = 011 \text{ XOR } 110$$
$$= 101$$

异或操作的可逆性在加密算法中意义重大，见稍后介绍。

异或操作的可逆性可以恢复原值，这在加密算法中意义重大。

块加密法

块加密法不是一次加密明文中的一个位，而是一次加密明文中的一个块。假设要加密的明文为 FOUR_AND_FOUR，利用块加密法，可以先加密 FOUR，再加密_AND_，然后加密 FOUR，即一次加密明文中的一个块。

解密时，每个块转换回初始的格式。实际上，通信是在位上进行的，因此 FOUR 实际上是 ASCII 字符 FOUR 的相应二进制值。用任何算法加密之后，得到的位要变成相应的 ASCII 值，因此可能得到 Vfa%之类的怪字符，对方收到二进制值后，将其译码成 ASCII 值 FOUR 的二进制形式，如图 3.4 所示。

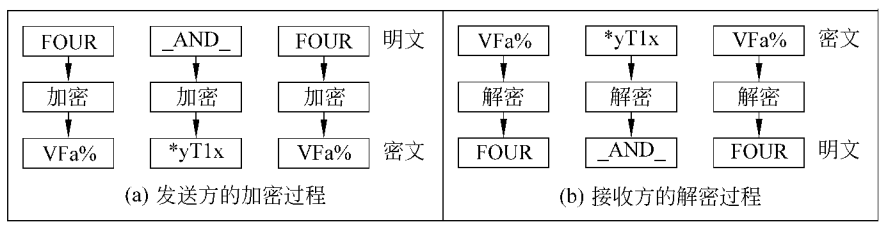

图 3.4　块加密

块加密的一个明显问题是重复文本。对重复文本模式，生成的密文是相同的，因此，密码分析员可以猜出原文的模式。密码分析员可以检查重复字符串，试图破译。如果破译成功，则可能破译明文中更大部分，从而更容易破译全部消息。即使密码分析员不能猜出其余单词，只要能在转账消息中把 debit 与 credit 变换，也会造成混乱！为了处理这类问题，块加密法是在链接模式中使用的，见稍后介绍。使用这个方法时，前面的密文块与当前块混合，从而掩护密文，避免重复内容出现重复块模式。

块加密技术一次加密明文中的一个块，解密时也是一块一块地进行。

实际上，块加密法使用的块通常包含 64 位以上，我们知道，流加密一次加密一位，是相当费时的，实际中通常是不必要的。因此，在计算机加密算法中，块加密法的用途比流加密法更加广泛。因此，我们主要介绍块加密法及其算法模式。稍后将会介绍，块加密法的两个算法模式也可以实现为流加密模式。

组结构

讨论算法时，经常会遇到是否是**组**(group)的问题。组元素是每个可能密钥构成的密文块。因此，组表示明文生成密文时的变化次数。

混淆与扩散

克劳德·香农引入了**混淆**(confusion)与**扩散**(diffusion)的概念，在计算机加密算法中非常重要。

混淆是为了保证密文中不会反映出明文线索，防止密码分析员从密文中找到模式，从而求出相应明文。我们已经知道如何混淆：就是使用前面介绍的替换技术。

扩散增加明文的冗余度，使其分布在行和列中。我们已经知道如何扩散：就是使用前面介绍的置换技术（也称变换加密技术）。

流加密法只使用混淆，而块加密法使用混淆与扩散，读者可以想想这是为什么。

3.2.2 算法模式

算法模式（algorithm mode）是块加密法中一系列基本算法步骤的组合，有些要从上一步得到某些反馈，这是计算机加密算法的基础。算法模式有4种：电子编码簿（Electronic Code Book，ECB）、加密块链接（Cipher Block Chaining，CBC）、加密反馈（Cipher Feedback，CFB）和输出反馈（Output Feedback，OFB），如图3.5所示，前两种算法模式处理块加密，而后两种模式是块加密模式，可以看成处理流加密一样。

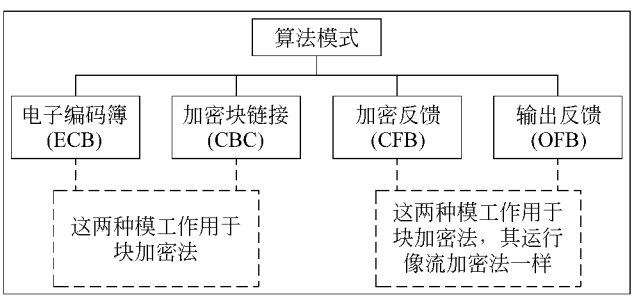

图3.5 算法模式

下面简要介绍这些算法模式。

电子编码簿（ECB）模式

电子编码簿模式是最简单的操作模式，将输入明文消息分成64位块，然后单独加密每个块。消息中的所有块用相同密钥加密，如图3.6所示。

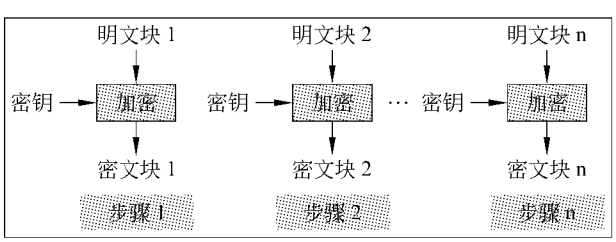

图3.6 电子编码簿模式的加密过程

接收方将收到的数据分成64位块，利用与加密时相同的密钥解密每个块，得到相应的明文块，如图3.7所示。

电子编码簿模式中用一个密钥加密消息的所有块，如果原消息中重复明文块，则加密消息中的相应密文块也会重复。因此，电子编码簿模式只适合加密小消息，因为重复相同明文

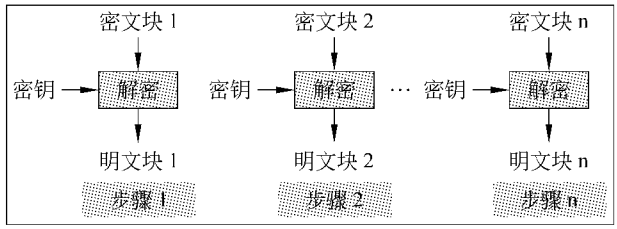

图 3.7　电子编码簿模式的解密过程

块的范围很小。

加密块链接(CBC)模式

对电子编码簿模式,在指定消息(即指定密钥)中,明文块总是产生相同的密文块。因此,如果输入中一个明文块多次出现,则输出中相应的密文块也会多次出现,从而给密码分析员提供一些线索。为了克服这个问题,加密块链接模式保证即使输入中的明文块重复,这些明文块也会在输出中得到不同的密文块。为此,要使用一个反馈机制,见下面介绍。

链接在块加密法中增加反馈机制。在加密块链接模式中,上一块的加密结果反馈到当前块的加密中,用每个块修改下一个块的加密。这样,每块密文与相应的当前输入明文块相关,与前面的所有明文块相关。

图 3.8 描述了 CBC 的加密过程。

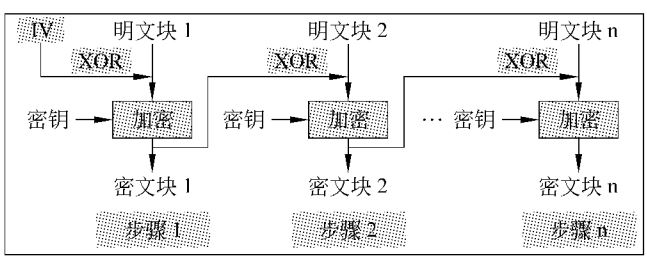

图 3.8　CBC 的加密过程

(1) 如图 3.8 所示,第 1 步接收两个输入:第一个明文块和一个随机文本块,称为**初始化向量**(Initialization Vector,IV)。

(a) 初始化向量没有什么特别意义,只是使每个消息唯一。因为初始化向量值是随机生成的,所以两个不同消息中重复初始化向量的可能性很小。因此,初始化向量可以使密文更独特,至少不同于不同消息中的其他密文。有趣的是,初始化向量不一定要保密,也可以是公开的。这好像不容易理解,但只要看看 CBC 加密过程,就可以发现,初始化向量只是第一步加密的两个输入之一。第 1 步的输出是密文块 1,它是第二步加密的两个输入之一。换句话说,密文块 1 也是第 2 步的初始化向量。同样,密文块 2 也是第 3 步的初始化向量,等等。由于所有这些密文块都要发送给接收方,因此从第 2 步起的所有初始化向量都要发送。因此,第一步的初始化向量就没有保密的必要。但是,实际上,为了提高安全性,密钥和初始化向量都保密。

(b) 第一个密文块和初始化向量用异或运算组合,然后用一个密钥加密,产生第一个密

文块。第一个密文块作为下一个明文块的反馈，见下面介绍。

（2）第2步将第二个明文块与第1步的输出（第一个密文块）用异或运算组合，然后用相同密钥加密，产生第二个密文块。第二个密文块。

（3）第3步将第三个明文块与第2步的输出（第二个密文块）用异或运算组合，然后用相同密钥加密，产生第三个密文块。

（4）这个过程对原消息的所有明文块继续。

记住，初始化向量只在第一个明文块中使用，但所有明文块的加密使用相同密钥。

解密过程如下：

（1）密文块1送入解密算法，使用所有明文块的加密使用的密钥。这一步的输出与初始化向量进行异或运算，得到第一个明文块。

（2）第2步解密密文块2，输出与第一个密文块异或操作，得到第二个明文块。

（3）这个过程对加密消息的所有密文块继续。

图3.9显示了CBC的解密过程。

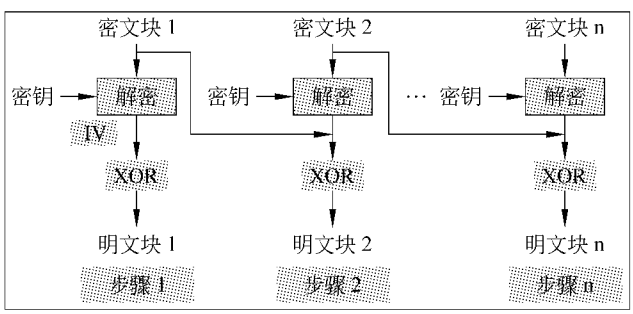

图3.9 CBC的解密过程

加密反馈（CFB）模式

不是所有应用程序都能处理数据块，面向字符的应用程序也需要安全性。例如，操作员可能在终端输入，要以安全方式在通信链路上立即传输，即使用加密方法。这时要使用流加密法，可以使用加密反馈模式。在加密反馈模式中，数据用更小的单元加密（如可以是8位，即操作员输入的一个字符的长度），这个长度小于定义的块长（通常是64位）。

下面看看加密反馈模式如何工作，假设我们一次处理j位（通常j=8）。由于加密反馈模式比前两种加密模式更复杂些，因此我们要一步一步进行介绍。

第1步：和CBC一样，加密反馈模式也使用64位的初始化向量。初始化向量放在移位寄存器中，在第1步加密，产生相应的64位初始化向量密文。如图3.10所示。

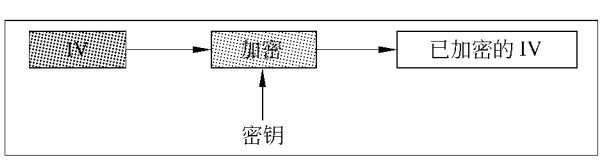

图3.10 CFB的第1步

第 2 步：加密初始化向量最左边(即最重要)的 j 位与明文前 j 位进行异或运算，产生密文第一部分(假设为 C)，如图 3.11 所示。然后将 C 传输到接收方。

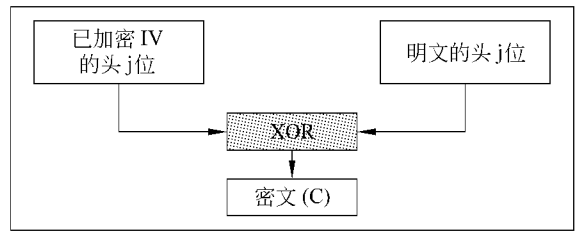

图 3.11　CFB 的第 2 步

第 3 步：初始化向量的位(即初始化向量所在的移位寄存器内容)左移 j 位，使移位寄存器最右边的 j 位为不可预测的数据，在其中填入 C 的内容，如图 3.12 所示。

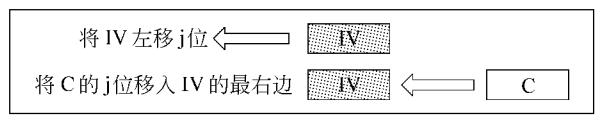

图 3.12　CFB 的第 3 步

第 4 步：重复第 1～3 步，直到加密所有明文单元，即重复下列步骤：

- 加密 IV。
- 加密得到的左边 j 位与明文的下面 j 位进行异或运算。
- 得到的明文部分(密文的下 j 位)发给接收方。
- 将 IV 的移位寄存器左移 j 位。
- 在 IV 的移位寄存器右边插入这 j 位密文。

图 3.13 显示了 CFB 的总体加密过程。

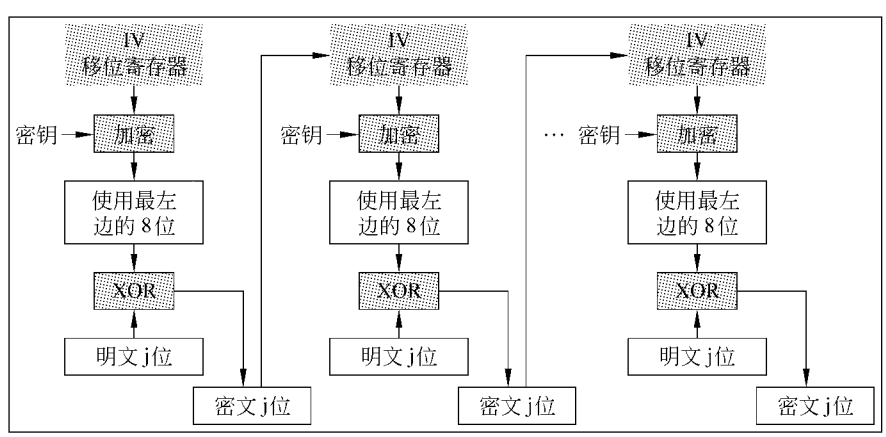

图 3.13　CFB 的总体加密过程

接收方的解密过程也很简单，只要稍作改变，这里不再重述。

输出反馈(OFB)模式

输出反馈模式与CFB很相似，唯一的差别是，CFB中密文填入加密过程下一阶段，而在OFB中，IV加密过程的输出填入加密过程下一阶段。因此，这里不准备描述OFB的细节，而只是画出OFB过程的框图，如图3.14所示，具体细节与CFB过程大致相同。

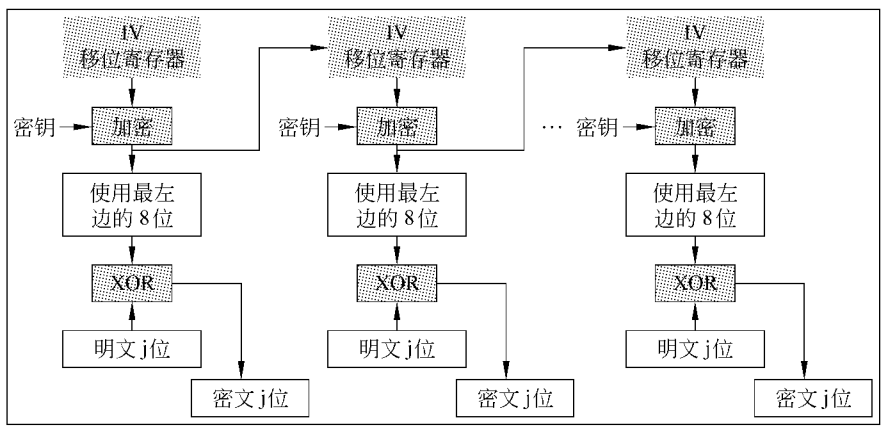

图3.14 OFB的总体加密过程

我们来归纳一下OFB模式的主要优点。简单地说，我们可以认为，在这种模式中，如果某个位有错误，那么这些错误只会保留在单个位上，不会破坏整个消息。也就是说，位错误不会扩散。如果密文位Ci出错，那么解密后，只有对应该位的值(即Pi)是错误的，其他位不受影响。在CFB模式则不同，密文位Ci作为输入反馈给移位寄存器，会破坏整个消息的其他位。

OFB模式的缺点是，攻击者可以对密文和消息的校验和以可控的方式做必要的修改。对密文的这种修改不会被检测到。换句话说，攻击者可以同时修改密文和校验和，而没有办法检测到这种修改。

计数器(CTR)模式

计数器模式与OFB模式非常类似。它使用序号(称为计数器)作为算法的输入。每个块加密后，要填充到寄存器中，使用下一个寄存器值。通常使用一个常数作为初始计数器的值，并且每次迭代后递增(通常是增加1)。计数器块的大小等于明文块的大小。

加密时，计数器加密后与明文块做XOR运算，得到密文。不需要使用链接过程。解密时，使用相同的计数器序列。其中，每个已加密的计数器与对应的密文块做XOR运算，得到初始明文。

计数器模式的整个操作如图3.15和图3.16所示。

计数器模式的加密或解密过程可以并行地作用于多个文本块，因为这里不涉及到链接的情况。这就使得计数器模式的运行速度更快。多处理系统可以利用这个特性，有助于减少整个处理时间。可以实现预处理，为加密盒准备输出(这些是XOR操作的输入)。计数器模式只负责加密过程的实现，不负责解密过程的实现。

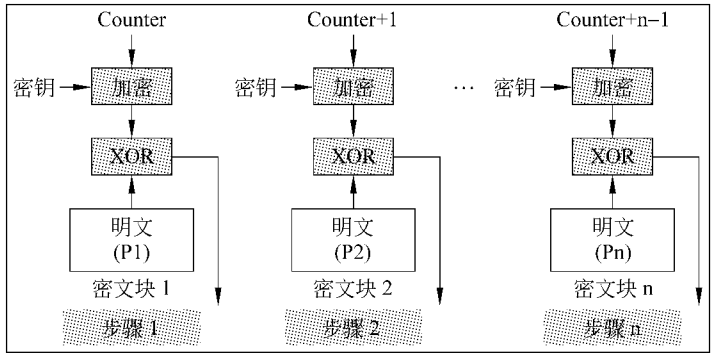

图 3.15　计数器模式：加密过程

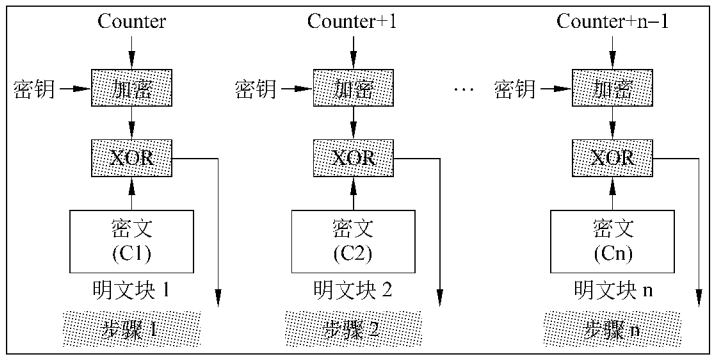

图 3.16　计数器模式：解密过程

表 3.1 归纳了各种算法模式的主要特性。

表 3.1　算法模式：细节内容与使用

算 法 模 式	细 节 内 容	使　　用
电子编码簿(ECB)	用相同的密钥分别加密文本的每个块，一次加密 64 位	以一种安全的方式传输一个值
加密块链接(CBC)	来自上一步的 64 位密文与下一步的明文进行 XOR 运算	文本块加密认证
加密反馈(CFB)	来自上一步的随机化密文的 K 位与下一步的明文的 K 位进行 XOR 运算	传输已加密的数据流认证
输出反馈(OFB)	类似于 CFB，只不过加密步骤的输入是前面的 DES 输出	传输已加密的数据流
计数器模式(CTR)	计数器与明文块一起加密，然后递增该计数器	基于块的传输需要高速的应用程序

表 3.2 归纳了各种模式的主要优点和缺点。

表 3.2 各种模式的主要优点和缺点

特 性	ECB	CBC	CFB	OFB/CTR
有关安全的问题	• 不能隐藏明文模式 • 块加密的输入与明文相同，没有随机化 • 明文容易操作，文本块可以被删除、重复或交换	• 明文块可以从消息的开头和末尾删除，可以改变第1个块的位	• 明文块可以从消息的开头和末尾删除，可以改变第1个块的位	• 明文容易操作 • 密文的改变直接改变明文
有关安全的优点	• 同一密钥可以用来加密多个消息	• 通过把明文与前一个密文块进行XOR运算，可以隐藏明文 • 同一密钥可以用来加密多个消息	• 可以隐藏明文模式 • 通过使用不同的IV，可以用同一密钥加密多个消息 • 块加密的输入是随机的	• 可以隐藏明文模式 • 通过使用不同的IV，可以用同一密钥加密多个消息 • 块加密的输入是随机的
有关效率的问题	• 通过填充块，使得密文的长度大于明文长度 • 不能预处理	• 密文比明文多一个块。 • 不能预处理 • 加密中不能引入并行性	• 密文长度与明文长度相同 • 加密中不能引入并行性	• 密文长度与明文长度相同 • 加密中不能引入并行性（只有OFB可以）

3.3 对称密钥加密法概述

下面简要复习一下对称密钥加密法。对称密钥加密法也称为**秘密密钥加密法**(Secret Key Cryptography)或**私钥加密法**(Private Key Cryptography)，只使用一个密钥，加密与解密使用相同密钥。显然，双方要先协定密钥之后才能开始传输，别人不应该知道这个密钥。图 3.17 的示例显示了对称密钥加密法的工作原理。简单地说，发送方(A)用密钥将明文消息变成密文，接收方(B)用相同密钥将密文消息变成明文，从而得到原消息。

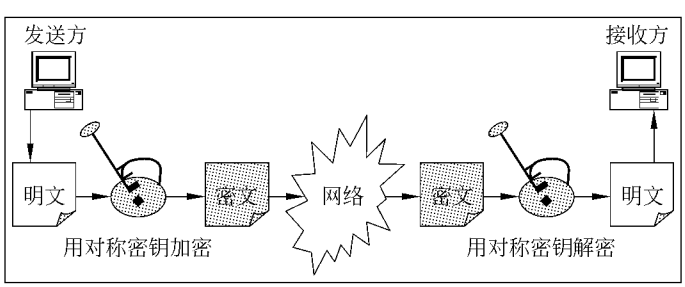

图 3.17 对称密钥加密法

前面曾介绍过，对称密钥加密法在实际应用中存在几个问题，下面简单回顾一下这些问题。

第一个问题是密钥协定或密钥发布。双方如何确定密钥？一个办法是发送方的某个人