# 第1单元

# 信息保密技术实验

信息的加密变换是目前实现信息安全的基本手段之一，采用信息保密技术对信息进行保密处理是最常用、最有效的安全保护手段。信息保密技术是信息安全的基础内容，也被认为是信息安全的核心技术。在信息系统中采用信息保密技术的目的是保护信息的保密性、完整性和可用性。

研究信息加密和解密变换的学科称为密码学，密码学是信息保密技术的核心，它是一门古老而深奥的学科。从密码学的发展进程来看，它经历了古典密码、对称密钥密码和非对称密钥密码3个阶段。本单元中，第1章分别从密码算法和加解密工具两方面对信息保密技术进行实践，介绍了对称加密算法中的DES算法以及非对称加密算法中的RSA算法。通过实验使读者进一步理解信息保密技术的功能、作用、基本原理和实现机制。

除了使用传统加密技术实现信息的存储与传输安全外，随着Internet应用的增长，如何利用Internet这样的公共网络实现快捷安全的信息传输，成为各类用户，尤其是企业用户普遍关心的问题。虚拟专用网络(Virtual Private Network，VPN)就是解决这一问题的有效途径。VPN通过公共网络(如Internet)建立一个临时的安全连接，构建一条穿过公共网络的安全、稳定的隧道，为远程用户、企业分支机构、商业伙伴等建立可信的安全连接，确保数据的安全传输。本单元中，第2章分别对Windows和Linux下VPN的简单构建方法进行了介绍。

# 第 1 章

# 密 码 算 法

## 实验 1-1　对称密码算法 DES

### 一、实验目的

通过对 DES 算法进行分析，并使用 DES 算法对数据进行加密和解密，进一步理解 DES 的实现和加解密原理。

### 二、实验环境

运行 Windows 或 Linux 操作系统的计算机，具有 VC(Windows)、gcc(Linux)等 C 语言编译环境。

### 三、预备知识

#### 1. 对称密码算法

根据采用的密钥类型，可将现有的加密算法分为两种：对称(单钥)密码算法和非对称(双钥或公钥)密码算法，前者如 DES，后者如 RSA。

对称密码算法的加解密使用的加密密钥和解密密钥相同，或者虽然不同，但可以从其中任意一个推导出另一个。对称密码的关键问题是将密钥安全地传送给参与保密通信的双方。对称加密算法要求通信双方在建立安全信道之前，约定好所使用的密钥。对于好的对称加密算法，其安全性完全决定于密钥的安全，算法本身是可以公开的，因此一旦密钥泄漏就等于泄漏了被加密的信息。对称算法是传统常用的算法，它最广泛使用的是 DES 算法。

#### 2. DES 算法

DES(Data Encryption Standard，数据加密标准)算法原是 IBM 公司为保护产品的机密于 1971—1972 年研制成功的，后被美国国家标准局和国家安全局选为联邦信息加密标准，并于 1977 年颁布使用。ISO 也已将 DES 作为数据加密标准。DES 是世界上最早被公认的实用密码算法标准，多年来它一直活跃在国际保密通信的舞台上，扮演了十分重要的角色。

DES 是一个分组加密算法，数据分组长度为 64 位(8 字节)，密文分组长度也是 64 位，没有数据扩展；密钥长度为64位，其中有8位为奇偶校验位，有效密钥长度为

56 位。DES 的整个体制是公开的，系统的安全性全靠密钥的保密。

1）DES 加密

（1）DES 加密流程

DES 算法处理的数据对象是一组 64 位的明文分组。设该明文分组为 $M=m_1m_2\cdots m_{64}$（$m_i=0$ 或 1）。明文分组经过 64 位的密钥 $K$ 来加密，最后生成长度为 64 位的密文 $E$。其加密过程如图 1-1 所示。

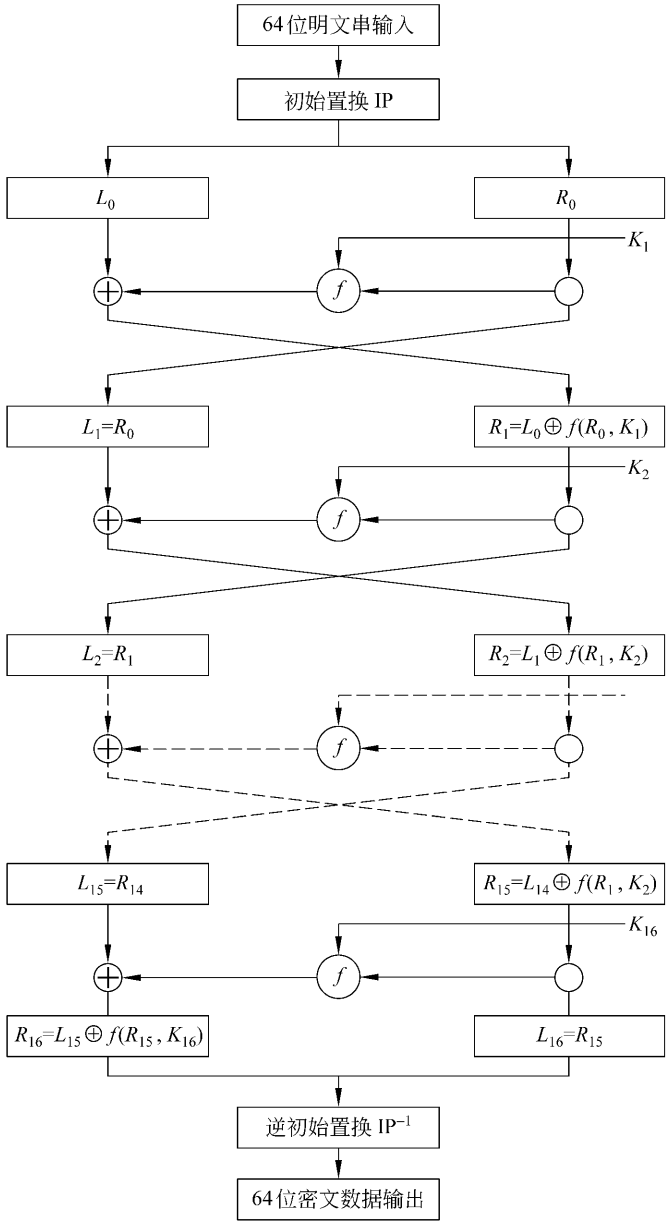


图 1-1　DES 加密流程

待加密的 64 位明文串 $M$，经过 IP 置换后，得到的比特串的下标列表如表 1-1 所示。

**表 1-1　初始置换 IP**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| IP | 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 |

该分组被分为 32 位的 $L_0$ 和 32 位的 $R_0$ 两部分。$R_0$ 与子密钥 $K_1$ 经过变换 $f(R_0, K_1)$ 输出 32 位的位串 $f_1$，$f_1$ 与 $L_0$ 做不进位的二进制加法运算。运算规则为：

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

$f_1$ 与 $L_0$ 做不进位的二进制加法运算后的结果赋给 $R_1$，$R_0$ 则原封不动地赋给 $L_1$。$L_1$ 与 $R_0$ 又做与以上完全相同的运算，生成 $L_2$，$R_2$。以此类推，一共经过 16 次运算，最后生成 $R_{16}$ 和 $L_{16}$。其中 $R_{16}$ 为 $L_{15}$ 与 $f(R_{15}, K_{16})$ 做不进位二进制加法运算的结果，$L_{16}$ 是 $R_{15}$ 的直接赋值。

$R_{16}$ 与 $L_{16}$ 合并成 64 位的比特串。值得注意的是，$R_{16}$ 一定要排在 $L_{16}$ 前面。$R_{16}$ 与 $L_{16}$ 合并后生成的比特串，经过置换 $IP^{-1}$ 后所得比特串的下标列表如表 1-2 所示。

**表 1-2　逆初始置换 $IP^{-1}$**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $IP^{-1}$ | 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 |

经过置换 $IP^{-1}$ 后生成的比特串即密文 $E$。

(2) $f$ 变换

$f$ 变换，即 $f(R_{i-1}, K_i)$ 函数，其功能是将 32 位的输入再转化为 32 位的输出。其过程如图 1-2 所示。

输入 $R_{i-1}$(32 位)，经过变换 $E$ 后，扩展为 48 位。扩展后的位串的下标列表如表 1-3

所示。

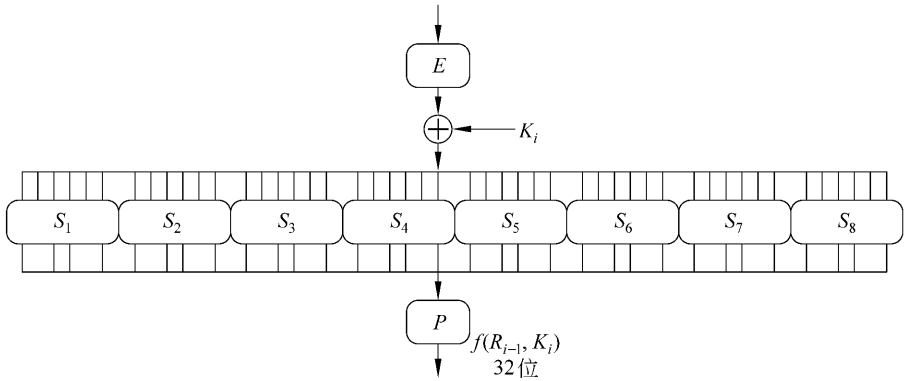


图 1-2　DES 算法中 $f$ 变换流程

**表 1-3　扩展后的位串下标**

| | | | | | | |
|---|---|---|---|---|---|---|
| $E$ | 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 | 31 |

扩展后的位串分为 8 组，每组 6 位。各组经过各自的 $S$ 盒后，又变为 4 位，合并后又成为 32 位。该 32 位经过 $P$ 变换后，其下标列表如表 1-4 所示。

**表 1-4　经过 $P$ 变换后的位串下标**

| | | | | |
|---|---|---|---|---|
| $P$ | 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 |

经过 $P$ 变换后输出的位串即是 32 位的 $f(R_{i-1}, K_i)$。

(3) S 盒

下面讲解 S 盒的变换过程。任取一 S 盒，如图 1-3 所示。

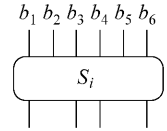

图 1-3　DES 中的 S 盒示意

在其输入 $b_1$、$b_2$、$b_3$、$b_4$、$b_5$ 和 $b_6$ 中，计算出 $x=b_1\times2+b_6$，$y=b_5+b_4\times2+b_3\times4+b_2\times8$，再从 $S_i$ 表中查出 $x$ 行、$y$ 列的值 $S_{xy}$。将 $S_{xy}$ 转换为二进制，即得 $S_i$ 盒的输出。DES 中 8 个 S 盒如图 1-4 所示。

| | \ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | 0 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| | 1 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| | 2 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| | 3 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |
| $S_2$ | 0 | 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| | 1 | 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| | 2 | 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| | 3 | 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |
| $S_3$ | 0 | 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| | 1 | 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| | 2 | 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| | 3 | 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |
| $S_4$ | 0 | 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 |
| | 1 | 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 |
| | 2 | 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 |
| | 3 | 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |
| $S_5$ | 0 | 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| | 1 | 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| | 2 | 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| | 3 | 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |
| $S_6$ | 0 | 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| | 1 | 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| | 2 | 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| | 3 | 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |
| $S_7$ | 0 | 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| | 1 | 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| | 2 | 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| | 3 | 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |
| $S_8$ | 0 | 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| | 1 | 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| | 2 | 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| | 3 | 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

图 1-4　DES 中的 8 个 S 盒

2）DES 子密钥的生成

DES 算法中，由 64 位的密钥生成 16 个 48 位的子密钥。其生成过程如图 1-5 所示。

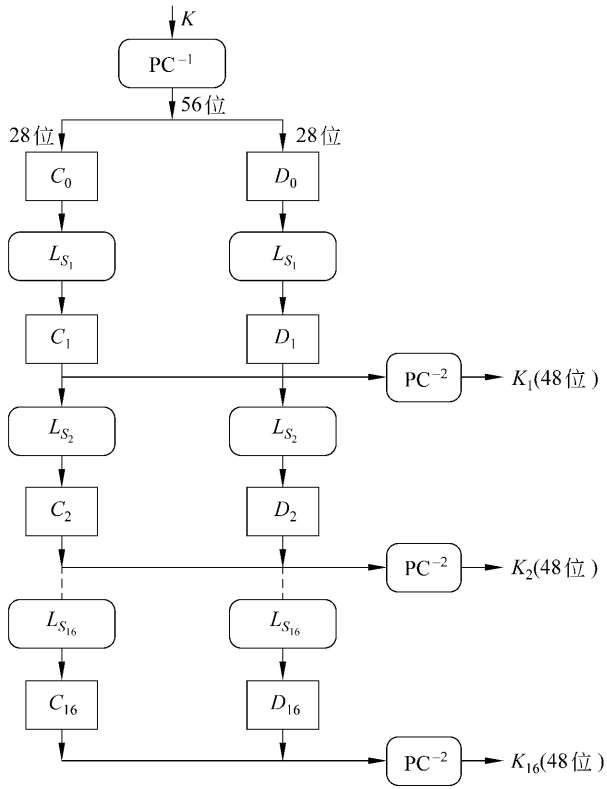


图 1-5　DES 算法子密钥的生成过程

子密钥生成过程具体解释如下：64 位的密钥 $K$，经过 $PC^{-1}$ 后，生成 56 位的串。其下标如表 1-5 所示。

**表 1-5　密钥 $K$ 经 $PC^{-1}$ 后的位串下标**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $PC^{-1}$ | 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 | 52 | 44 | 36 |
| | 63 | 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 |

该位串分为长度相等的位串 $C_0$ 和 $D_0$。然后将 $C_0$ 和 $D_0$ 分别循环左移 1 位，得到 $C_1$ 和

$D_1$。$C_1$和 $D_1$合并起来生成 $C_1D_1$，$C_1D_1$经过 $PC^{-2}$变换后即生成 48 位的 $K_1$。$K_1$的下标列表如表 1-6 所示。

**表 1-6　密钥 $K_1$的下标列表**

| | | | | | | |
|---|---|---|---|---|---|---|
| $PC^{-2}$ | 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 |

$C_1$、$D_1$分别循环左移 $L_{S_2}$位，再合并，经过 $PC^{-2}$，生成子密钥 $K_2$。以此类推，直至生成子密钥 $K_{16}$。

**注意**：$L_{S_i}(i=1,2,\cdots,16)$的值是不同的，具体如表 1-7 所示。

**表 1-7　$L_{S_i}(i=1,2,\cdots,16)$的值**

| 迭代顺序 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 左移位数 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

3) DES 解密流程

DES 的解密过程和 DES 的加密过程完全类似，只不过将 16 圈的子密钥序列 $K_1$，$K_2$，…，$K_{16}$的顺序倒过来。即第一圈用第 16 个子密钥 $K_{16}$，第二圈用 $K_{15}$，其余类推。DES 解密流程的第一圈运算如图 1-6 所示。

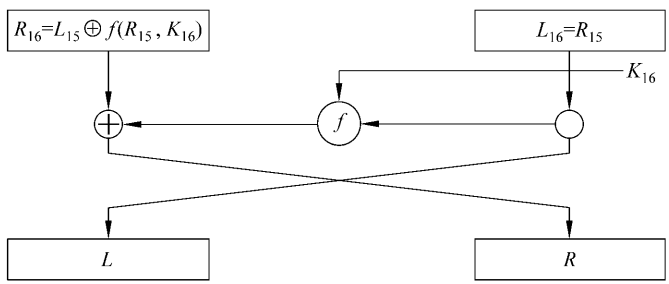


图 1-6　DES 解密流程的第一圈运算

DES 的实际密钥长度为 56 位。对 DES 最尖锐的批评之一是密钥太短。就目前计算机的计算机能力而言，DES 不能抵抗对密钥的穷举搜索攻击。

目前人们仍然不知道 DES 中是否存在陷门。所谓陷门，通俗地讲，就是设计者在算法的设计中留了一个后门，知道这一秘密的人可以进入这一后门获得使用该算法的用户

的秘密密钥。

1997 年 1 月 28 日，RSA 数据安全公司在 RSA 安全年会上悬赏 10 000 美金破解 DES，科罗拉多州的程序员 Verser 在 Internet 上数万名志愿者的协作下用 96 天的时间找到了密钥长度为 40 位和 48 位的 DES 密钥。

1998 年 7 月，电子前沿基金会(EFF)使用一台价值 25 万美元的计算机在 56 小时之内破译了 56 位的 DES。

1999 年 1 月，电子前沿基金会(EFF)通过因特网上的 10 万台计算机合作，仅用 22 小时 15 分就破解了 56 位的 DES。不过破译的前提是，破译者能识别出破译的结果确实是明文，亦即破译的结果必须容易辨认。如果明文加密之前经过压缩等处理，辨认工作就比较困难。

为了增加密钥的长度，人们建议将一种分组密码进行级联，在不同的密钥作用下，连续多次对一组明文进行加密，通常把这种技术称为多重加密技术。使用多重 DES 这一点目前基本上已达成共识。

## 四、实验任务

**1. 算法分析**

1) 任务描述

分析 DES 算法源代码，理解 DES 的加解密原理和过程。

2) 操作过程与步骤

(1) 程序内容

DES 程序包括一个头文件和一个实现 DES 算法的 C 语言文件。头文件中主要是一些宏定义和函数声明，其中还包括保证程序可移植性的一些定义。DES 程序通过宏定义可选择小代码模式(＃define small_code)或者大代码模式。在大代码模式下，程序定义了多个表，从而使 DES 算法中的很多运算都可以通过查表实现，速度较快，但要求有较多的存储空间；在小代码模式运行时，可以不查表，从而节省了存储空间，但速度较慢。

(2) 主要函数

加解密时主要用到以下 5 个函数。

① int des_setup(const unsigned char * key,int keylen,
int num_rounds, des_key * skey)

**函数名称**：密钥生成函数。

**参数说明**：key 是一个指针，指向用户输入的初始密钥；keylen 是输入密钥的长度，以字节为单位；num_rounds 是加密圈数，当输入 0 时，使用算法默认的圈数；skey 是一个指向结构体变量的指针，结构体变量用于存储加密和解密时每圈使用的子密钥。

当密钥生成时，返回值为 CPYPT_OK(0)，结果保留在 skey 指向的结构体 des_key。

des_key 的定义如下：

```
typedef struct des_key {
                    ulong32 ek[32],dk[32];
                  }des_key;
```

结构体中的 ek 存储加密时使用的子密钥，dk 存储解密时使用的子密钥。

结构体中用 2 个 32 位的整数来存储一圈的 48 位密钥，每一个 32 位整数被分成 4 个 8 位，每个 8 位的第 6 位存储密钥。如果把 48 位密钥分成 8 组，则这 8 组按存储顺序从高到低分别为 1、3、5、7、2、4、6、8。这样做是为了加密时可以把扩展和查表运算结合进行。

② void des_ecb_encrypt(const unsigned char * pt,unsigned char * ct,
des_key * key)

**函数名称**：加密函数。

**参数说明**：pt 是指向待加密的明文数组的指针，ct 是指向存储加密结果（密文数组）的指针，key 是调用密钥生成函数后存储每一圈子密钥的结构体变量。

加密成功时返回 CPYPT_OK。

③ void des_ecb_decrypt(const unsigned char * ct,unsigned char * pt,
des_key * key)

**函数名称**：解密函数。

**参数说明**：ct 是指向待解密的密文数组的指针，pt 是指向存储解密结果（明文数组）的指针，key 是调用密钥生成函数后存储每一圈子密钥的结构体变量。

解密成功时返回 CPYPT_OK。

加密和解密时，pt 和 ct 可以指向同一块内存。

④ int des_test(void)

**函数名称**：测试函数。

该函数用于对加密算法进行测试。函数体内部定义了对应的明文和密文数组，并且进行了多圈加密和解密。该函数还可以用来测试函数的运行时间。

⑤ int des_keysize(int * desired_keysize)

**函数名称**：密钥长度检验函数。

**参数说明**：desired_keysize 是使用者所希望的密钥长度。当密钥长度小于所希望的密钥长度时，返回值为 CRYPT_INVALID_KEYSIZE，否则，desired_keysize 指向的变量被置为 8。

**2. 算法实例**

1）任务描述

分析实例代码，并对程序进行编译运行，实际演示和体验 DES 算法的实现过程。

2）操作过程与步骤

（1）实例代码

```
#include "des.h"
int main (int arc,char * arv[])
{
 unsigned char pt[9]="abcdefgh",ct[9],
               key[8]={'a','b','c','d','a','b','c','d'};
 des_key skey;
 pt[9]=ct[9]= '\0';
 des_setup(key,8,0,&skey);
```

```
 des_ecb_encrypt(pt,ct,&skey);
 des_ecb_decrypt(ct,pt,&skey);
 printf("%s\n",pt);
 printf("%s\n",ct);
 system("PAUSE");
 return 0;
}
```

(2) 实例说明

该程序演示了对一组 8 字节的数据进行加密和解密的过程。pt 指向明文数组,ct 指向密文数组,skey 是密钥数组。pt 和 ct 的数组长度设为 9,是为了方便控制台字符输出。对文件进行加密时,可以指定读取和写入的字节数,这两个数组长度则应该定义为 8。

## 五、实验要求

1. 按步骤完成实验任务,撰写实验报告。
2. 对本书素材库中的 DES 源程序进行编译运行,对一个文件进行加密和解密。

# 实验 1-2 非对称密码算法 RSA

## 一、实验目的

通过分析 RSA 算法,并使用 RSA 算法及其实用工具对数据进行加密和解密来进一步理解 RSA 的实现和加解密原理。

## 二、实验环境

运行 Windows 或 Linux 操作系统的计算机,具有 VC(Windows)、gcc(Linux)等 C 语言编译环境。

## 三、实验工具

RSATool。

## 四、预备知识

### 1. 非对称密码算法

非对称密码的观点是 Diffie 和 Hellman 于 1976 年在他们的论文"密码学的新方向"一文中首次提出的,指明了 Shannon 在"保密通信的信息理论"中提出的将密码建立在求解某些已知的数学难题上的具体实现途径。

非对称加密算法也称公钥密码算法,是指用于加密的密钥与用于解密的密钥是不同的,而且从加密的密钥无法推导出解密的密钥。这类算法之所以被称为公钥算法是因为用于加密的密钥是可以广泛公开的,任何人都可以得到加密密钥并用来加密信息,但是只有拥有对应解密密钥的人才能将信息解密。在公钥算法中,加密密钥不同于解密密钥,加密密钥可以公之于众,谁都能使用,而解密密钥只有解密人自己知道。它们分别称为公开密钥(Public Key)和秘密密钥或私有密钥(Secret Key 或 Private Key)。

Rives、Shamir 和 Adleman 于 1977 年提出了第一个比较完善的公钥密码体制,它既

可用于加密，又可用于签名，这就是著名的 RSA 公钥密码体制。

非对称密码实质上是计算机复杂性理论发展的结果。对称密码的缺陷之一是通信双方在进行通信之前必须通过一个安全信道事先交换密钥，这在实际应用中通常是非常困难的，而公钥密码可使通信双方无须事先交换密钥就可建立起保密通信。

目前国际上已经提出了许多公钥密码，但比较安全并被人们认可的公钥密码主要有两类：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 公钥密码；另一类是基于离散对数问题，比如 Elgamal 公钥密码和影响比较大的椭圆曲线公钥密码。

**2. RSA 简介**

RSA 的名字来自于它的创建者们——麻省理工大学的 Ronald Rivest、以色列魏茨曼科学中心的 Adi Shamir 以及南加州大学的 Leonard Adleman。它是众所周知的不对称公共密码系统的典型。自 1977 年提出后广为流行，并易于理解和操作，目前它已被 ISO 推荐为公钥数据加密标准。后来，他们在 1982 年创办了以 RSA 命名的公司 RSA Data Security Inc. 和 RSA 实验室，该公司和实验室在公开密钥密码系统的研究和商业应用推广方面具有举足轻重的地位。

1) RSA 算法描述

(1) 公钥

选择两个互异的大素数 $p$ 和 $q$，$n$ 是二者的乘积，即 $n=pq$。$n$ 的欧拉函数 $\Phi(n)=(p-1)(q-1)$，随机选取正整数 $e$，使其满足 $gcd(e,\Phi(n))=1$，即 $e$ 和 $\Phi(n)$ 互质，将 $(n,e)$ 作为公钥。

(2) 私钥

求出正整数 $d$，使其满足 $ed=1 \bmod \Phi(n)$ 且 $0 \leqslant d \leqslant n$，将 $(n,d)$ 作为私钥。

(3) 加密

对于明文 $M$，由 $C=M^e \bmod n$ 得到密文。

(4) 解密

对于密文 $C$，由 $M=C^d \bmod n$ 得到明文。

如果破译者获得了 $n$、$e$ 和密文 $C$，为了破解密文必须计算出私钥 $d$，为此需要先分解 $n$，而 RSA 的安全性正是基于大数 $n$ 的因子分解困难的基础上的。为了提高 RSA 的安全性，提高破解难度，达到更高的安全性，一般商业应用要求 $n$ 的长度不小于 1024 位，更重要的场合不小于 2048 位。

2) RSA 的安全性

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，从提出到现在已二十多年，经历了各种攻击的考验，逐渐为人们接受，被普遍认为是目前最优秀的公钥方案之一。RSA 的安全性依赖于大数的因子分解，但并没有从理论上证明破译 RSA 的难度与大数分解难度等价。即 RSA 的重大缺陷是无法从理论上把握它的保密性能如何，而且密码学界多数人士倾向于因子分解不是 NPC 问题。RSA 的缺点主要有：①产生密钥很麻烦，受到素数产生技术的限制，因而难以做到一次一密；②分组长度太大，为保证安全性，$n$ 至少也要 600 位以上，使运算代价很高，尤其是速度较慢，较对称密码算法慢几个数量级，且随着大数分解技术的发展，这个

长度还在增加，不利于数据格式的标准化。

## 五、实验任务

### 1. 算法分析

1）任务描述

分析 RSA 算法源代码，理解 RSA 的加解密原理和过程。

2）操作过程与步骤

（1）RSA 源代码分析与程序编写

本书素材库中给出了一个密码算法库，其中包括典型对称加密算法和非对称加密算法。找出其中关于 RSA 算法的部分，并且基于标准输入/输出编写一段用 RSA 加密文件的程序。

（2）计算公私钥

为了加深对 RSA 算法的理解，根据已知参数：$p=3$，$q=11$，$M=2$，手工计算 RSA 的公私钥。

（3）加解密

使用第(2)步所计算的公私钥对明文进行加密，然后对生成的密文进行解密。

### 2. 应用实例

1）任务描述

使用 RSA 加解密工具 RSATool 生成公私钥对，并对一段文字进行加解密，以进一步了解 RSA 算法原理及应用。

2）操作过程与步骤

（1）双击 RSATool 程序图标，启动 RSATool 程序，RSATool 主窗口如图 1-7 所示。

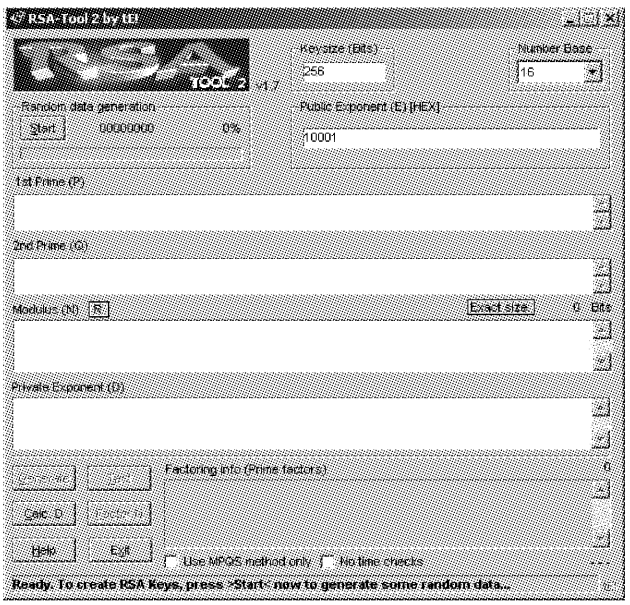


图 1-7　RSATool 主窗口

(2) 确定要生成的密钥的长度 Keysize，如 64 位，如图 1-8 所示。

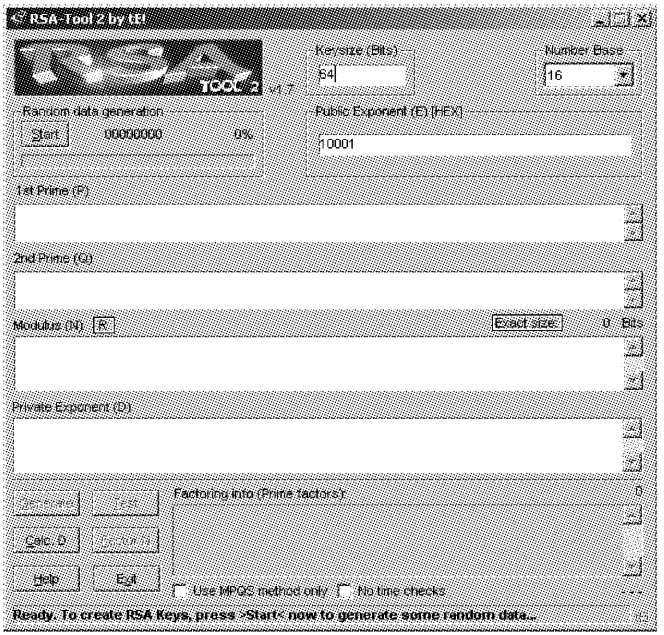


图 1-8　确定要生成的密钥的长度 Keysize

(3) 单击 Start 按钮，生成随机数，如图 1-9 所示。

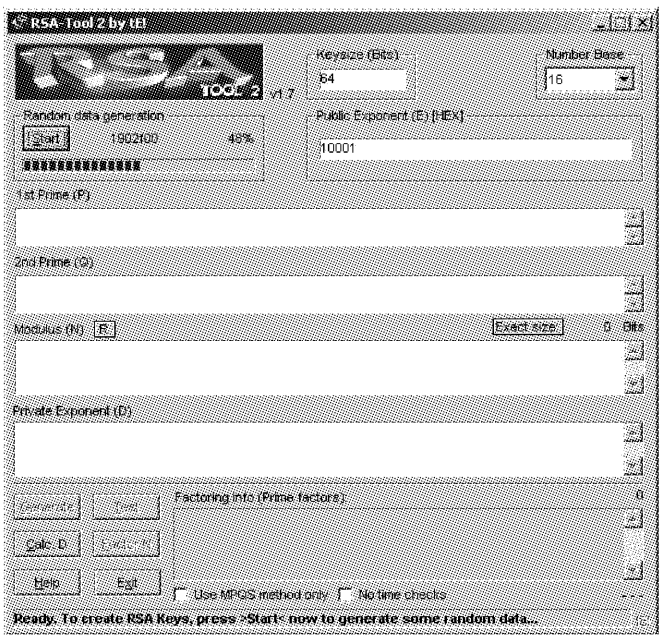


图 1-9　生成随机数

(4) 生成随机数后，单击 Generate 按钮，生成密钥，如图 1-10 所示。

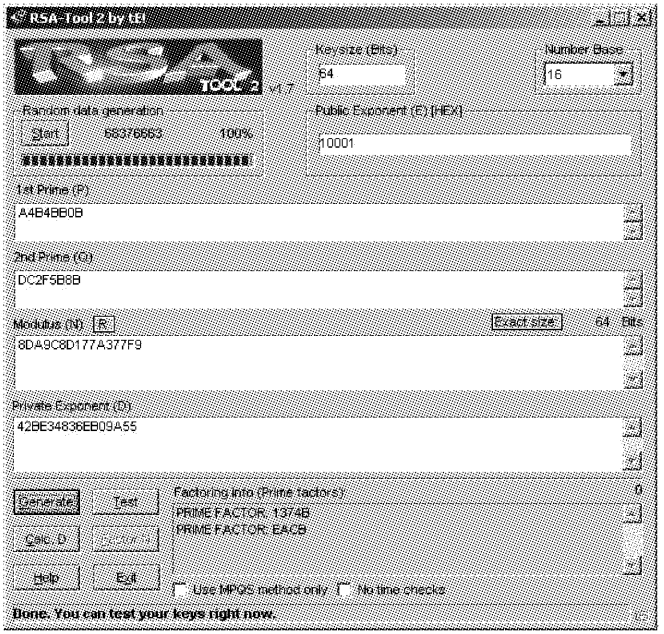


图 1-10 生成密钥

(5) 单击 Test 按钮,打开 RSA 加解密测试对话框,如图 1-11 所示。

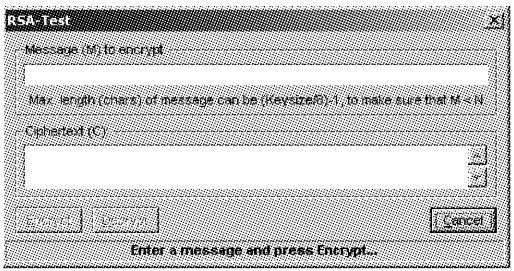


图 1-11 RSA 加解密测试对话框

(6) 在 Message to encrypt 文本框中输入要加密的文字,如图 1-12 所示。

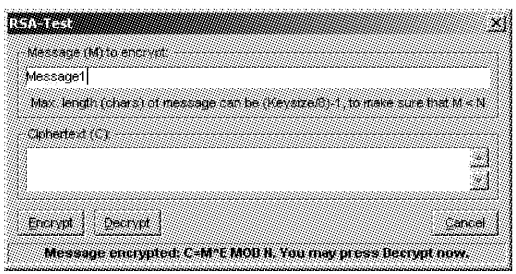


图 1-12 输入要加密的文字

(7) 单击 Encrypt 按钮,对所输入的文字进行加密,如图 1-13 所示。

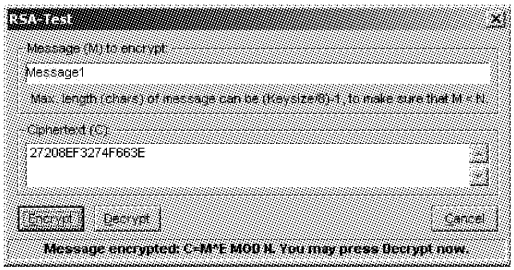


图 1-13　对所输入的文字进行加密

(8) 单击 Decrypt 按钮，对第(7)步的加密结果进行解密，如图 1-14 所示。

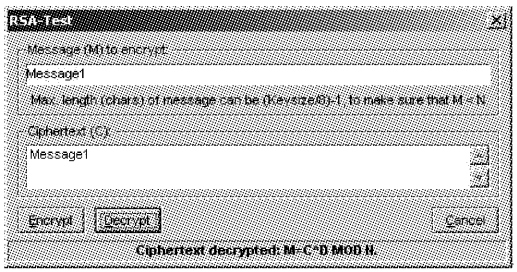


图 1-14　对加密结果进行解密

## 六、实验要求

1. 按步骤完成实验任务，撰写实验报告。

2. 对本书素材库中的 RSA 源程序进行编译运行，对一个文件进行加密和解密。

3. 计算机在生成一个随机数时，并不一定就是素数，因此要进行素性检测。思考是否有确定的方法判定一个大数是素数。查阅资料找出目前实际可行的素数判定方法，并且比较不同方法的优缺点。

# 第 2 章

# 信息保密传输

## 实验 2-1　Windows 下的 VPN

### 一、实验目的

通过实验进一步掌握虚拟专用网 VPN 的实现原理，理解并掌握 Windows 操作系统中利用 PPTP 和 IPSec 配置 VPN 连接的方法。

### 二、实验环境

多台运行 Windows 2000 Professional 或 Windows XP Professional 的计算机，一台运行 Windows 2000 Server 的计算机。

### 三、预备知识

**1. VPN**

VPN(Virtual Private Network，虚拟专用网)是利用开放的公众网络建立专用数据传输通道，将企业的分支机构、商业伙伴、移动办公等连接起来，并且提供安全的端到端的数据通信的一种广域网技术。目前，VPN 的应用越来越广泛，而且大多数的 VPN 都是用路由器或防火墙等硬件加软件实现的。实现 VPN 的协议虽然很多，但其具体实现方式都是采用隧道技术，将企业网的数据封装在隧道中进行传输。隧道协议可分为第二层和第三层隧道协议。第二层隧道协议包括点对点隧道协议 PPTP(Point-to-Point Tunneling Protocol)、第二层转发协议 L2F(Layer 2 Forwarding)和第二层隧道协议 L2TP(Layer 2 Tunneling Protocol)；第三层隧道协议包括通用路由封装 GRE(Generic Routing Encapsulation)和 IPSec(IP Security)。

Microsoft 公司在 Windows NT 4 中只实现了 PPTP 协议，而在 Windows 2000 中全面实现了 PPTP、L2TP 和 IPSec 协议。

**2. PPTP**

PPTP 最初是由 Microsoft 公司设计的，PPTP 论坛(其中包括 Ascend Communications、Microsoft、3Com、ECI Telematics 及 U. S. Robotics 等各大公司)开发的点到点安全隧道协议，用于将 PPP 分组通过 IP 网络封装传输。PPTP 协议可以建立 PC 到 LAN 的 VPN 连接，是目前较为流行的第二层隧道协议。

PPTP 协议定义了一种 PPP 分组的运载工具。它通过使用扩展的 GRE 封装，将 PPP 分组在 IP 网上进行传输。PPTP 在逻辑上延伸了 PPP 会话，从而形成了虚拟的远程拨号。

**3. IPSec**

IPSec 实际上是一套协议包而不是单个的协议，IPSec 是在 IP 网络上保证安全通信的开放标准框架，它在 IP 层提供数据源验证、数据完整性和数据保密性。其中比较重要的有 RFC2409 IKE(Internet Key Exchange，因特网密钥交换)、RFC2401 IPSec 协议、RFC2402 AH(Authentication Header，认证头)、RFC2406 ESP(Encapsulating Security Payload，封装安全载荷)等协议。IPSec 独立于密码学算法，这使得不同的用户群可以选择不同一套安全算法。

IPSec 主要由 AH 协议、ESP 协议以及负责密钥管理的 IKE 协议组成。AH 为 IP 数据包提供无连接的数据完整性和数据源身份认证。数据完整性通过消息认证码(如 MD5、SHA1)产生的校验值来保证，数据源身份认证通过在待认证的数据中加入一个共享密钥来实现。ESP 为 IP 数据包提供数据的保密性(通过加密机制)、无连接的数据完整性、数据源身份认证以及防重放攻击保护。AH 和 ESP 可以单独使用，也可以配合使用，通过组合可以配置多种灵活的安全机制。密钥管理包括 IKE 协议和安全联盟 SA (Security Association)等部分。IKE 在通信双方之间建立安全联盟，提供密钥确定、密钥管理机制，是一个产生和交换密钥材料并协商 IPSec 参数的框架。IKE 将密钥协商的结果保留在 SA 中，供 AH 和 ESP 通信时使用。

## 四、实验任务

**1. 在 Windows 下利用 PPTP 配置 VPN 连接**

1) 任务描述

在 Windows 操作系统中利用 PPTP 配置 VPN 连接，实现客户端(Windows 2000 Professional 或 Windows XP Professional)到服务器端(Windows 2000 Server)的保密传输。

2) 操作过程与步骤

(1) 在服务器端安装和启动 VPN

① 在 Windows 2000 Server 计算机上，确保到 Internet 的连接和到局域网(LAN)的连接都已正确配置。

② 选择“开始”|“程序”|“指向管理工具”|“路由和远程访问”命令，如图 2-1 所示。

③ 右击树中的服务器名称，在弹出的快捷菜单中选择“配置并启用路由和远程访问”命令，如图 2-2 所示。

④ 打开“路由和远程访问服务器安装向导”对话框，单击“下一步”按钮，在“通用配置”对话框中，选择“虚拟专用网络(VPN 服务器)”单选按钮，然后单击“下一步”按钮。

⑤ 在“远程客户协议”对话框中，确认列表中包括 TCP/IP，然后选择“是，所有要求的协议都在此列表中”单选按钮，如图 2-3 所示。

⑥ 单击“下一步”按钮，在“Internet 连接”对话框中，选择到 Internet 的连接，然后单

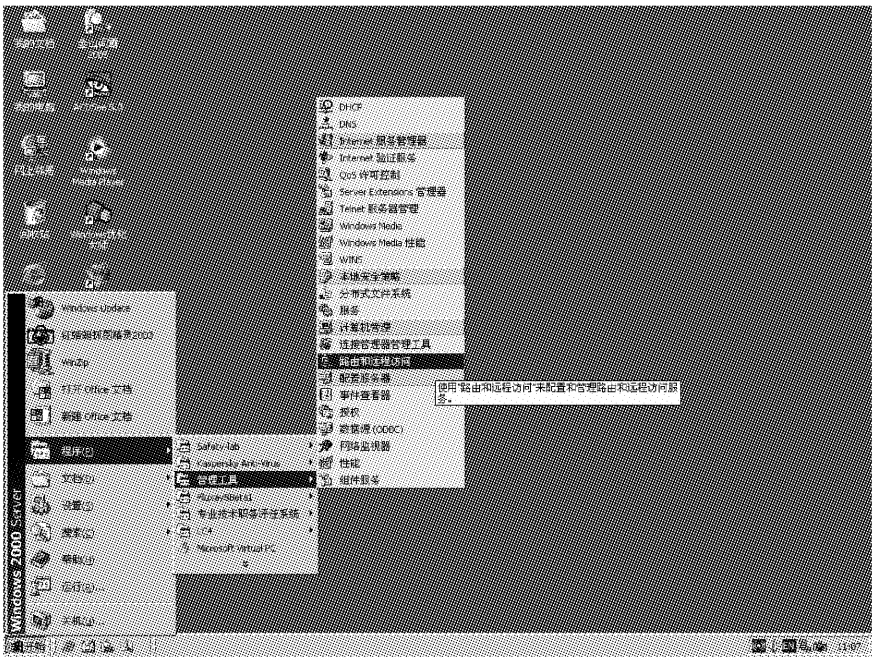


图 2-1 选择“路由和远程访问”命令

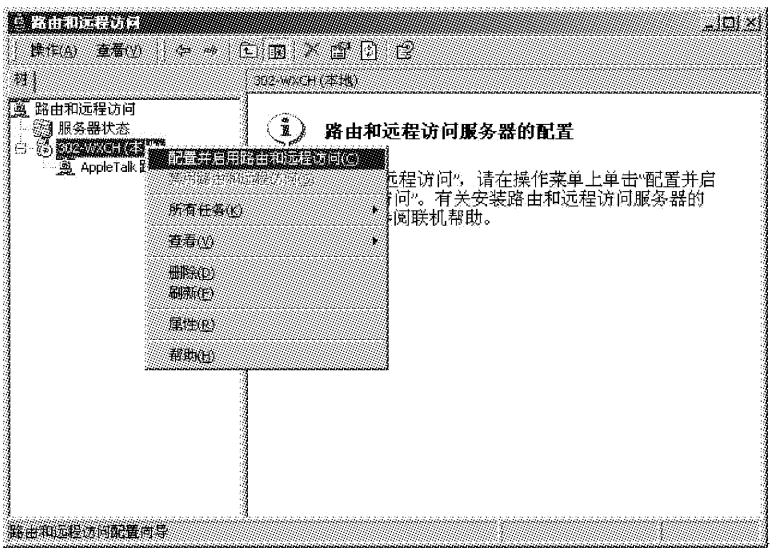


图 2-2 选择“配置并启用路由和远程访问”命令

击“下一步”按钮。

⑦ 在“IP 地址指定”对话框中，选择“来自一个指定的地址范围”单选按钮，如图 2-4 所示。

⑧ 在接下来的对话框中选择“新建”，建立新的地址范围，如图 2-5 所示。

⑨ 在“管理多个远程访问服务器”对话框中，选中“不，我现在不想设置此服务器使用

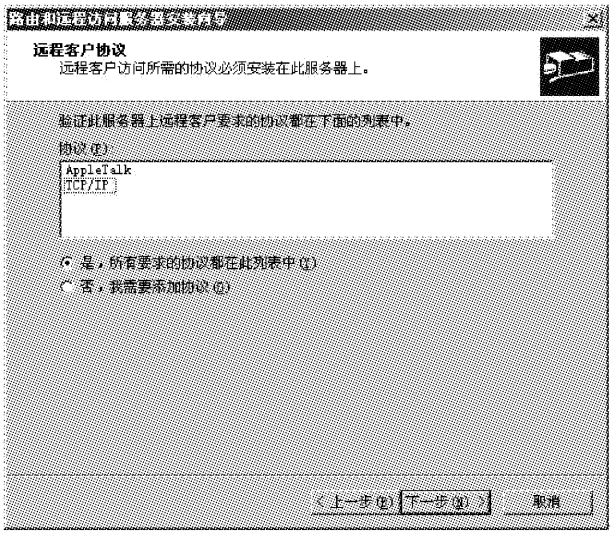


图 2-3　验证远程客户协议

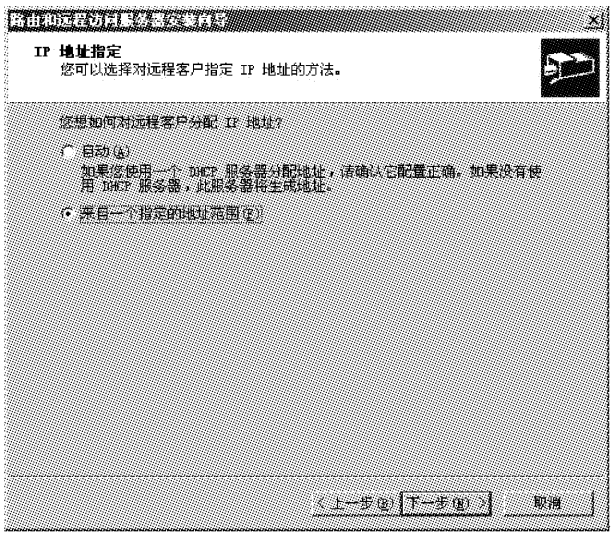


图 2-4　选择“来自一个指定的地址范围”单选按钮

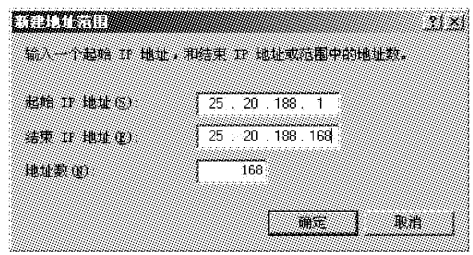


图 2-5　建立新的地址范围