本章重点

在前面的章节中，涉及的都是一些稍显死板的话题。那么在本章，就喝杯咖啡稍微休息一下吧，敬请诸位放松心情往下阅读。本章的主题是数据加密。对于公司内部的网络而言，由于只是将员工的电脑彼此相连，可能就不太需要对其间传输的数据进行加密。但是在互联网中，由于它联结的是全世界范围的企业和个人，所以会面临很多需要对数据进行加密处理的情况[①]。举例来说，在网店购物时用户输入的信用卡卡号，就是应该被加密传输的代表性数据。假设卡号未经加密就被发送出去，那么就会面临卡号被同样接入互联网的某人盗取，信用卡被其用来肆意购物的危险。因此像这种网店页面的URL，通常都是以 https:// 开头，表示数据正在使用加密的方式进行传输。其实，大家在不知不觉中就已经都是加密技术的受益者了。

然而，如何对数据进行加密呢？这的确是个有意思的话题。在本章中，我们将使用 VBScript（Visual Basic Scripting Edition）语言实际编写几个加密程序来展开这个话题[②]。请诸位不要只是阅读文字内容，还应该实际确认程序的运作。加密技术真的有趣得令人兴奋的一项技术！

10.1 先来明确一下什么是加密

在作为加密对象的数据中，蕴含着文本、图像等各种形式的信息。但是，由于计算机会把所有的数据都用数字表示，所以即便数据有各

① 当然在很多情况下，即便是企业内的局域网也需要应用加密技术，例如针对无线局域网这种传输的数据很容易被监听的环境，或者人事资料这种就算是被员工盗取也会产生恶劣影响的敏感数据。

② 从 Windows 98 以后，操作系统本身集成了 WSH（Windows Script Host）功能，通过使用记事本（notepad.exe）之类的文本编辑器，用 VBScript 语言编写的程序可以直接执行。

种展现形式，对其加密的技术却是基本相同的。因此在本章中，我们就假设加密的对象仅限于文本数据。

文本数据可以由各种各样的字符构成。其中每个字符都被分配了一个数字，我们称之为“字符编码”。定义了应该把哪个编码分配给哪个字符的字符编码体系叫作字符集。字符集分为 ASCII 字符集、JIS 字符集、Shift-JIS 字符集，EUC 字符集、Unicode 字符集等若干种。

在表 10.1 中，以十进制数字列出了大写拉丁字母（A 至 Z）的 ASCII 编码。计算机会把文本数据处理成数字序列，例如在使用了 ASCII 编码的计算机中，就会把 NIKKEI 处理成“78 73 75 75 69 73”。可是只要把这一串数字转换为对应的字符显示在屏幕上，就又变成了人们所认识的 NIKKEI 了。通常把这种未经加密的文本数据称为“明文”。

表 10.1　用于表示 A 至 Z 的 ASCII 编码（10 进制）

字符	编码	字符	编码
A	65	N	78
B	66	O	79
C	67	P	80
D	68	Q	81
E	69	R	82
F	70	S	83
G	71	T	84
H	72	U	85
I	73	V	86
J	74	W	87
K	75	X	88
L	76	Y	89
M	77	Z	90

数据一旦以明文的方式在网络中传输，就会有被盗取滥用的危险，因此要对明文进行加密，将它转换成为“密文”。当然密文也仅仅是一

串数字，但是如果是把密文显示在屏幕上，那么在人类看来显示的也只不过是读不懂、没有意义的字符序列罢了。

虽然存在各种各样的加密技术，但是其中的基本手段无外乎还是字符编码的变换，即将构成明文的每个字符的编码分别变换成其他的数值。通过反转这种变换过程，加密后的文本数据就可以还原。通常把密文还原成明文的过程（即解读密码的过程）称为“解密”。

10.2 错开字符编码的加密方式

有关加密的概念和术语先解释到这里，下面就通过运行程序来实际体验加密的过程吧。代码清单 10.1 中，列出了一段用于加密的示例程序。在该程序中，使用了如下加密方法：将文本数据中每个字符所对应的字符编码一律向后错三个，即给原字符编码的值加上 3。请把这段程序保存到以 .vbs 为扩展名的文本文件中，例如 cipher1.vbs，然后把该文件保存到合适的文件夹中。接下来只需双击 cipher1.vbs 的图标即可运行这段程序。请试着在最初弹出的窗口中输入要加密的文本数据（明文），例如就输入 NIKKEI 吧，然后单击 OK 按钮。在接下来弹出的窗口中会显示出加密后的文本数据（密文）。因为每个字符的编码都向后错了三个，所以得到的是 QLNNHL。这样的话，即便是被人偷偷地看到了，那个人也无法理解这个字符串的意义（如图 10.1 所示）。

代码清单 10.1　用给字符编码加上 3 的方法加密

```
plaintext = InputBox(" 请输入明文。")
cipher = ""
For i = 1 To Len(plaintext)
    letter = Mid(plaintext, i, 1)
    cipher = cipher & Chr(Asc(letter) + 3)
Next
MsgBox cipher
```

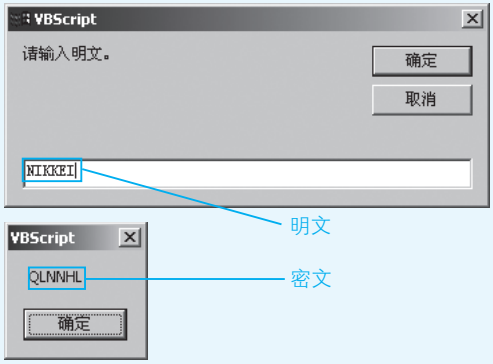

图 10.1　代码清单 10.1 的执行结果

因为加密时使用的是将字符编码向后错三个的方法，所以只要再将字符编码向前挪三个就可以实现解密。代码清单 10.2 中就是解密程序。与进行加密的程序相反，解密使用的是从字符编码中减去 3 的方法。在最初弹出的窗口中输入密文，我们就输入刚刚得到的 QLNNHL，然后单击 OK 按钮。在接下来弹出的窗口中就会显示出解密后的明文 NIKKEI（如图 10.2 所示）。怎么样？这看起来还是挺酷的吧。

代码清单 10.2　用把字符编码减去 3 的方法解密

```
cipher = InputBox(" 请输入密文。")
plaintext = ""
For i = 1 To Len(cipher)
    letter = Mid(cipher, i, 1)
    plaintext = plaintext & Chr(Asc(letter) - 3)
Next
MsgBox plaintext
```

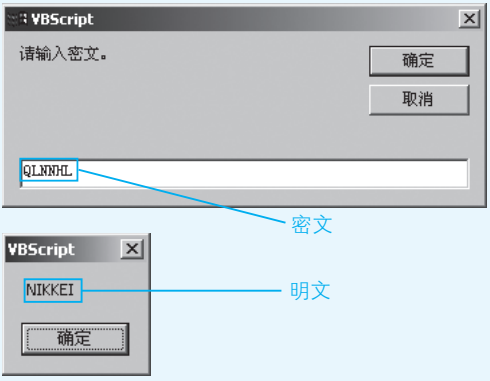

图 10.2 代码清单 10.2 的执行结果

也就是说，加上 3 就是加密，减去 3 就是解密。因此通常把像 3 这样用于加密和解密的数字称为“密钥”。如果事先就把 3 这个密钥作为只有数据的发送者和接受者才知道的秘密，那么不知道这个密钥的人，就无法对加密过的数据进行解密。

下面再试着编写一个加密程序吧。这次让密钥的值也可以由用户指定吧。该程序通过把每一个字符的编码与密钥做 XOR 运算（eXclusive OR，逻辑异或运算），将明文转换成密文（如代码清单 10.3 所示）。XOR 运算的有趣之处在于，用 XOR 运算加密后的密文，可以通过相同的 XOR 运算解密。也就是说，一个程序既可用于加密又可用于解密，很方便（如图 10.3 所示）。

代码清单 10.3 通过 XOR 运算进行加密和解密

```
k = InputBox("请输入密钥。")
key = CInt(k)
text1 = InputBox("请输入明文或密文。")
text2 = ""
```

```
For i = 1 To Len(text1)
    letter = Mid(text1, i, 1)
    text2 = text2 & Chr(Asc(letter) Xor key)
Next
MsgBox text2
```

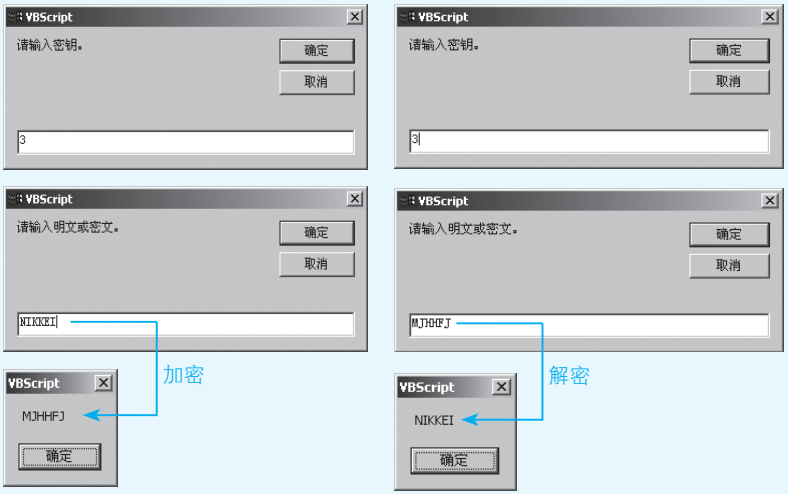

图 10.3　代码清单 10.3 的执行结果

XOR 运算的法则是把两个数据先分别用二进制表示，然后当一个数据中的某一位与另一个数据中的 1 相对时，就将这一位反转（若这一位是 0 就变成 1，是 1 就变成 0）[①]。因为是靠翻转数字实现的加密，所以只要再翻转一次就可以解密。图 10.4 中展示了密钥 3（用二进制表示是 00000011）和字母 N（其字符编码用二进制表示是 01001110）做 XOR 运算的结果，请诸位确认通过翻转和再翻转还原出字母 N 的过程：N 的字符编码先和 3 做 XOR 运算，结果是字母 M 的字符编码。

① 异或运算的法则也可以描述成如果对应位置上的两个二进制数 a、b 的值相同，则结果为 0。如果 a、b 的值不相同，则结果为 1。

M 的字符编码再和 3 做 XOR 运算，结果就又回到了 N 的字符编码。

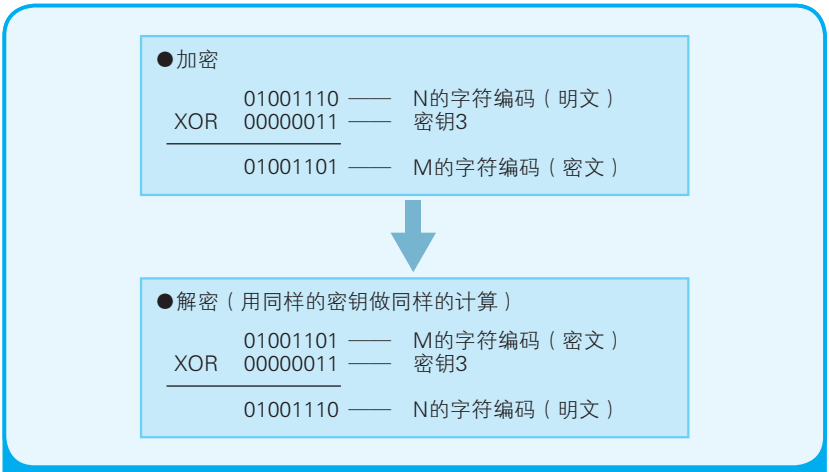

图 10.4　通过翻转和再翻转即可还原

10.3　密钥越长，解密越困难

在互联网等环境中，会有很多不固定的人群相互收发经过加密处理的数据。一般情况下，会将所使用的加密方式公开，而只对密钥的值保密。但是令人感到遗憾的是，这个世界上还是有坏人的。有些人会盗取那些并不是发送给他们的加密数据，企图破解后用于不可告人的目的。尽管这些人并不知道密钥的值，但是他们会利用计算机强大的计算能力，用密钥所有可能的取值去试着破解。例如，要想破解用 XOR 运算加密得到的密文 MJHHFJ，程序只要把 0 到 9 这几个值分别作为密钥都尝试一遍就能做到（如代码清单 10.4、图 10.5 所示）。

代码清单 10.4　通过 XOR 运算破解密文的程序

```
cipher = InputBox("请输入密文。")
plaintext = ""
```

```
For key = 0 To 9
    plaintext = plaintext & "密钥" & CStr(key) & ":"
    For i = 1 To Len(cipher)
        letter = Mid(cipher, i, 1)
        plaintext = plaintext & Chr(Asc(letter) Xor key)
    Next
    plaintext = plaintext & Chr(&HD)
Next
MsgBox plaintext
```

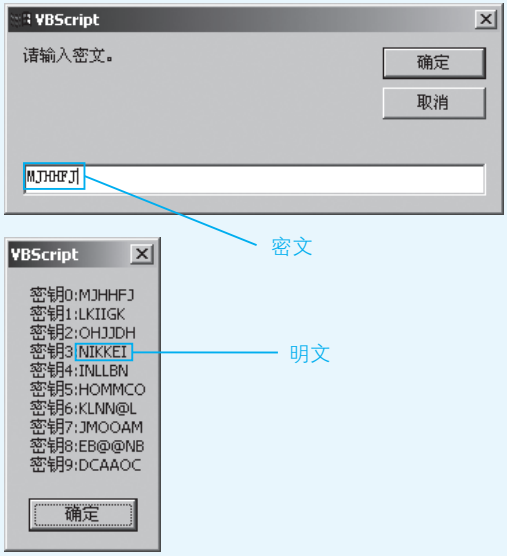

图 10.5　代码清单 10.4 的执行结果

在互联网上经过加密的数据也难免被盗，因此就要先设法做到即使数据被盗了，其内容也难以被破解。为此可以把密钥设成多位数而不仅仅是一位数。下面，我们就丢弃一位数的 3，试着以三位数的 345 为密钥，通过 XOR 运算来试着进行加密（如代码清单 10.5 所示）。将明文中的第一个字母与 3 做 XOR 运算、第二个字母与 4 做 XOR 运算、

第三个字母与 5 做 XOR 运算。从第四个字母开始，还是以三个字母为一组依次与 3、4、5 做 XOR 运算，依此类推（如图 10.6 所示）。

代码清单 10.5　通过与三位数的密钥进行 XOR 运算实现加密和解密

```
Dim key(2)
key(0) = 3
key(1) = 4
key(2) = 5
text1 = InputBox("请输入明文或密文。")
text2 = ""
For i = 1 To Len(text1)
    letter = Mid(text1, i, 1)
    text2 = text2 & Chr(Asc(letter) Xor key((i - 1) Mod 3))
Next
MsgBox text2
```

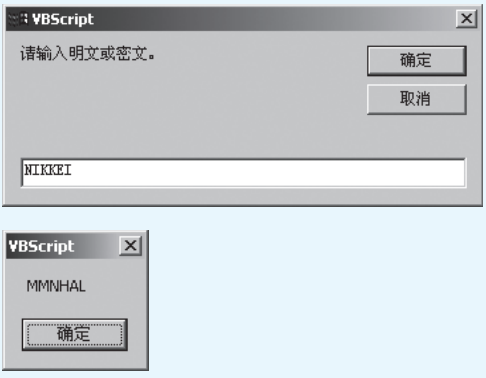

图 10.6　代码清单 10.5 的执行结果

如果仅用一位数作为密钥，那么只需要从 0 到 9 尝试十次就能破解密文。但是如果是用三位数的密钥，那么就有从 000 到 999 的 1000 种可能。如果更进一步把密钥的位数增长到十位，结果会怎样呢？那样的话，破解者就需要尝试 10 的 10 次方 = 100 亿次。就算使用了一秒钟可以进行 100 万次尝试的计算机，破解密文也还是需要花费 100 亿

÷100 万次 / 秒 = 10000 秒≈ 2.78 小时，坏人说不定就会因此放弃破解。密钥每增长一位，破解所花费的时间就会翻 10 倍。密钥再进一步增长到 16 位的话，破解时间就是 2.78 小时 ×1000000 ≈ 317 年，从所需的时间上来看，可以说破解是不可能的。

10.4　适用于互联网的公开密钥加密技术

前面几节所讲解的加密技术都属于“对称密钥加密技术”，也称作“秘密密钥加密技术”（如图 10.7 所示）。这种加密技术的特征是在加密和解密的过程中使用数值相同的密钥。因此，要使用这种技术，就必须事先把密钥的值作为只有发送者和接收者才知道的秘密保护好（如图 10.7-(1) 所示）。虽然随着密钥位数的增加，破解难度也会增大，但是事先仍不得不考虑一个问题：发送者如何才能把密钥悄悄地告诉接收者呢？用挂号信吗？要是那样的话，假设有 100 名接收者，那么发送者就要寄出 100 封挂号信，非常麻烦，而且这样也无法防止通信双方以外的其他人知道密钥。再说寄送密钥也要花费时间。互联网的存在应该意味着用户可以实时地与世界各地的人们交换信息。因此对称密钥加密技术不适合在互联网中使用。

但是世界上不乏善于解决问题的能人。他们想到只要让解密时的密钥不同于加密时的密钥，就可以克服对称密钥加密技术的缺点。（“会有这样的技术吗?”也许诸位不禁会发出这样的疑问，稍后笔者将展示具体的例子）。而这种加密技术就被称为“公开密钥加密技术”。

在公开密钥加密技术中，用于加密的密钥可以公开给全世界，因此称为“公钥”，而用于解密的密钥是只有自己才知道的秘密，因此称为“私钥”。举例来说，假设笔者的公钥是 3，私钥是 5（实际中会把位数更多的两个数作为一对儿密钥使用）。笔者会通过互联网向全世界宣