现在假设 Alice 要向 Bob 发送一个报文。Alice 报文的最初形式（例如，“Bob，I love you. Alice”）被称为**明文**（plaintext，cleartext）。Alice 使用**加密算法**（encryption algorithm）加密其明文报文，生成的加密报文被称为**密文**（ciphertext），该密文对任何入侵者看起来是不可懂的。有趣的是在许多现代密码系统中，包括因特网上所使用的那些，加密技术本身是已知的，即公开发行的、标准化的和任何人都可使用的（例如［RFC 1321；RFC 3447；RFC 2420；NIST 2001］），即使对潜在的入侵者也是如此！显然，如果任何人都知道数据编码的方法，则一定有一些秘密信息可以阻止入侵者解密被传输的数据。这些秘密信息就是密钥。

在图 8-2 中，Alice 提供了一个**密钥**（key）K_A，它是一串数字或字符，作为加密算法的输入。加密算法以密钥和明文报文 m 为输入，生成的密文作为输出。用符号 $K_A(m)$ 表示（使用密钥 K_A 加密的）明文报文 m 的密文形式。使用密钥 K_A 的实际加密算法显然与上下文有关。类似地，Bob 将为**解密算法**（decryption algorithm）提供密钥 K_B，将密文和 Bob 的密钥作为输入，输出初始明文。也就是说，如果 Bob 接收到一个加密的报文 $K_A(m)$，他可通过计算 $K_B(K_A(m)) = m$ 进行解密。在**对称密钥系统**（symmetric key system）中，Alice 和 Bob 的密钥是相同的并且是秘密的。在**公开密钥系统**（public key system，也称为公钥系统）中，使用一对密钥：一个密钥为 Bob 和 Alice 俩人所知（实际上为全世界所知），另一个密钥只有 Bob 或 Alice 知道（而不是双方都知道）。在下面两小节中，我们将更为详细地考虑对称密钥系统和公钥系统。

8.2.1　对称密钥密码体制

所有密码算法都涉及用一种东西替换另一种东西的思想，例如，取明文的一部分进行计算，替换适当的密文以生成加密的报文。在分析现代基于密钥的密码系统之前，我们首先学习一下**凯撒密码**（Caesar cipher）找找感觉，这是一种加密数据的方法。这种非常古老而简单的对称密钥算法由 Julius Caesar 发明。

凯撒密码用于英语文本时，将明文报文中的每个字母用字母表中该字母后第 k 个字母进行替换（允许回绕，即把字母“a”排在字母“z”之后）。例如，如果 $k=3$，则明文中的字母“a”变成密文中的字母“d”；明文中的字母“b”变成密文中的字母“e”，依此类推。因此，k 的值就作为密钥。举一个例子，明文报文“bob，i love you. alice”在密文中变成“ere，l oryh brx. dolfh”。尽管密文看起来像乱码，但如果你知道使用了凯撒密码加密，因为密钥值只有 25 个，所以用不了多久就可以破解它。

凯撒密码的一种改进方法是**单码代替密码**（monoalphabetic cipher），也是使用字母表中的一个字母替换该字母表中的另一个字母。然而，并非按照规则的模式进行替换（例如，明文中的所有字母都用偏移量为 k 的字母进行替换），只要每个字母都有一个唯一的替换字母，任一字母都可用另一字母替换，反之亦然。图 8-3 为加密明文的一种可行替换规则。

```
明文字母: a b c d e f g h i j k l m n o p q r s t u v w x y z
密文字母: m n b v c x z a s d f g h j k l p o i u y t r e w q
```

图 8-3　单码代替密码

明文报文“bob，i love you. alice”变成“nkn，s gktc wky. mgsbc”。因此，与用凯撒密码情况一样，这看起来像乱码。单码代替密码的性能看来要比凯撒密码的性能好得多，可能的字母配对为 26！（10^{26} 数量级）种，而不是 25 种。尝试所有 10^{26} 种可能配对的蛮力法

要求的工作量太大，不是一种破解加密算法和解密报文的可行方式。但是，通过对明文语言进行统计分析，例如，在典型的英语文本中，由于已知字母“e”和字母“t”出现的频率较高（这些字母出现的频率分别为13%和9%），并且常见的两三个字母的组合通常一起出现（例如，“in”“it”“the”“ion”“ing”等等），这就使得破解该密文变得相对容易。如果入侵者具有某些该报文的可能内容的知识，则破解该密码就会更为容易。例如，如果入侵者Trudy是Bob的妻子，怀疑Bob和Alice有暧昧关系，则她猜想“bob”和“alice”这些名字可能会出现在密文中。如果Trudy确信这两个名字出现在密文中，并有了上述报文的密文副本，则她能够立即决定这26个字母配对中的7个，比蛮力法少检查10^9种可能性。如果Trudy的确怀疑Bob有不正当的男女关系，她可能也非常期待从该报文中找到某些其他选择的词汇。

当考虑Trudy破解Bob和Alice之间加密方案的难易程度时，可以根据入侵者所拥有的信息分为三种不同的情况。

- 唯密文攻击。有些情况下，入侵者只能得到截取的密文，也不了解明文报文的内容。我们已经看到，统计分析有助于对加密方案的**唯密文攻击**（ciphertext-only attack）。
- 已知明文攻击。前面已经看到，如果Trudy以某种方式确信在密文报文中会出现“bob”和“alice”，她就可以确定字母a、l、i、c、e、b和o的（明文，密文）匹配关系。Trudy也可能会幸运地记录到传输的所有密文，然后在一张纸上找到Bob写下的已解密的明文。当入侵者知道（明文，密文）的一些匹配时，我们将其称为对加密方案的**已知明文攻击**（known-plaintext attack）。
- 选择明文攻击。在**选择明文攻击**（chosen-plaintext attack）中，入侵者能够选择某一明文报文并得到该明文报文对应的密文形式。对于我们前面所说的简单加密算法来说，如果Trudy能让Alice发送报文“The quick brown fox jumps over the lazy dog”，则Trudy就能够完全破解Alice和Bob所使用的加密方案。但是随后我们将看到，对于更为复杂的加密技术来说，使用选择明文攻击不一定意味着能够攻破该加密机制。

500年前，发明了**多码代替密码**（polyalphabetic encryption），这种技术是对单码代替密码的改进。多码代替密码的基本思想是使用多个单码代替密码，一个单码代替密码用于加密某明文报文中一个特定位置的字母。因此，在某明文报文中不同位置出现的相同字母可能以不同的方式编码。图8-4中显示了多码代替密码机制的一个例子。它使用两个凯撒密码（其中$k=5$和$k=19$），如图中不同的行所示。我们可以选择使用这两个凯撒密码C_1和C_2，加密时采用以C_1，C_2，C_2，C_1，C_2的次序循环的模式，即明文的第一个字母用C_1编码，第二和第三个字母用C_2编码，第四个字母用C_1编码，第五个字母用C_2编码，然后循环重复该模式，即第六个字母用C_1编码，第七个字母用C_2编码，依此类推。这样一来，明文报文“bob，i love you.”加密后成为“ghu，n etox dhz.”。注意到明文报文中的第一个“b”用C_1加密为“g”，第二个“b”用C_2加密为“u”。在这个例子中，加密和解密“密钥”是两个凯撒密码密钥（$k=5$和$k=19$）和C_1，C_2，C_2，C_1，C_2的次序模式的知识。

```
明文字母 : a b c d e f g h i j k l m n o p q r s t u v w x y z
C1(k=5)  : f g h i j k l m n o p q r s t u v w x y z a b c d e
C2(k=19) : t u v w x y z a b c d e f g h i j k l m n o p q r s
```

图8-4 使用两个凯撒密码的多码代替密码

1. 块密码

我们现在回到现代社会中，考察对称密钥加密今天的工作方式。对称加密技术有两种宽泛的类型：**流密码**（stream cipher）和**块密码**（block cipher）。当我们研究无线 LAN 的安全性时，将在 8.7 节中简要地研究流密码。在本节中，我们关注块密码，该密码用在多种因特网协议的加密中，包括 PGP（用于安全电子邮件）、SSL（用于使 TCP 连接更安全）和 IPsec（用于使网络层传输更安全）。

在块密码中，要加密的报文被处理为 k 比特的块。例如，如果 $k=64$，则报文被划分为 64 比特的块，每块被独立加密。为了加密一个块，该密码采用了一对一映射，将 k 比特块的明文映射为 k 比特块的密文。我们来看一个例子。假设 $k=3$，因此块密码将 3 比特输入（明文）映射为 3 比特输出（密文）。表 8-1 给出了一种可能的映射。注意到这是一个一对一的映射，即对每种输入有不同的输出。这种块密码将报文划分成 3 比特的块并根据映射关系进行加密。可以验证，报文 010110001111 被加密成了 101000111001。

表 8-1　一种特定的 3 比特块密码

输入	输出	输入	输出
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

继续这个 3 比特块的例子，注意到上述映射只是许多可能映射中的一种。有多少种可能的映射呢？要回答这个问题，观察到一个映射只不过是所有可能输入的排列。共有 $2^3(=8)$ 种可能的输入（排列在“输入”栏中）。这 8 种输入能够排列为 8!=40 320 种不同方式。因为这些排列的每种都定义了一种映射，共有 40 320 种可能的映射。我们能够将这些映射的每种视为一个密钥，即如果 Alice 和 Bob 都知道该映射（密钥），他们能够加密和解密在他们之间发送的报文。

对这种密码的蛮力攻击即通过使用所有映射来尝试解密密文。仅使用 40 320 种映射（当 $k=3$），这能够在一台桌面 PC 上迅速完成。为了挫败蛮力攻击，块密码通常使用大得多的块，由 64 比特甚至更多比特组成。注意到对于通常的 k 比特块密码，可能映射数量是 $2^k!$，对于即使不大的 k 值（如 $k=64$），这也是一个天文数字。

如刚才所述，尽管全表块密码对于不大的 k 值能够产生健壮的对称密钥加密方案，但不幸的是它们难以实现。对于 $k=64$ 和给定的映射，将要求 Alice 和 Bob 维护一张具有 2^{64} 个输入值的表，这是一个难以实现的任务。此外，如果 Alice 和 Bob 要改变密钥，他们将不得不每人重新生成该表。因此，全表块密码在所有输入和输出之间提供了预先决定的映射（如在上述例子中那样），这简直是不可能实现的事。

取而代之的是，块密码通常使用函数模拟随机排列表。在图 8-5 中显示了当 $k=64$ 时这种函数的一个例子（引自［Kaufman 1995］）。该函数首先将 64 比特块划分为 8 个块，每个块由 8 比特组成。每个 8 比特块由一个“8 比特到 8 比特”表处理，这是个可管理的长度。例如，第一个块由标志为 T_1 的表来处理。接下来，这 8 个输出块被重新装配成一个 64 比特的块。该输出被回馈到 64 比特的输入，开始了第二次循环。经 n 次这样的循环后，该函数提供了一个 64 比特的密文块。这种循环的目的是使得每个输入比特影响最后输出比特的大部分（即使不是全部）。（如果仅使用一次循环，一个给定的输入比特将仅

影响64输出比特中的8比特。）这种块密码算法的密钥将是8张排列表（假定置乱函数是公共已知的）。

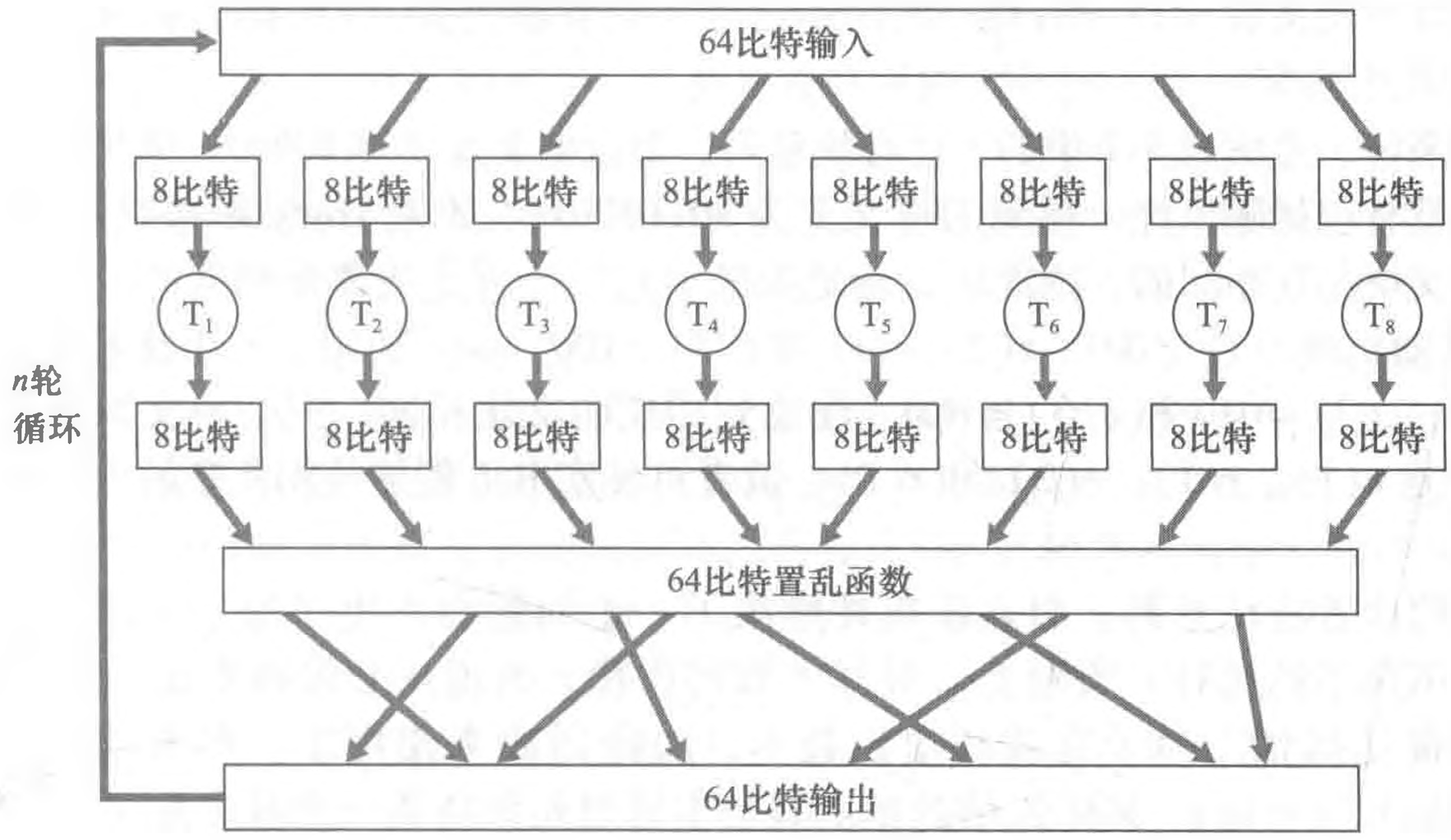

图8-5 一个块密码的例子

目前有一些流行的块密码，包括DES（Data Encryption Standard，数据加密标准）、3DES和AES（Advanced Encryption Standard，高级加密标准）。这些标准都使用了函数（而不是预先决定的表），连同图8-5的线（虽然对每种密码来说更为复杂和具体）。这些算法也都使用了比特串作为密钥。例如，DES使用了具有56比特密钥的64比特块。AES使用128比特块，能够使用128、192和256比特长的密钥进行操作。一个算法的密钥决定了特定“小型表”映射和该算法内部的排列。对这些密码进行蛮力攻击要循环通过所有密钥，用每个密钥应用解密算法。观察到采用长度为 n 的密钥，有 2^n 种可能的密钥。NIST［NIST 2001］估计，如果用1秒破解56比特DES的计算机（就是说，每秒尝试所有 2^{56} 个密钥）来破解一个128比特的AES密钥，要用大约149万亿年的时间才有可能成功。

2. 密码块链接

在计算机网络应用中，通常需要加密长报文（或长数据流）。如果使用前面描述的块密码，通过直接将报文切割成 k 比特块并独立地加密每块，将出现一个微妙而重要的问题。为了理解这个问题，注意到两个或更多个明文块可能是相同的。例如，两个或更多块中的明文可能是“HTTP/1.1”。对于这些相同的块，块密码当然将产生相同的密文。当攻击者看到相同的密文块时，它可能潜在地猜出其明文，并且通过识别相同的密文块和利用支撑协议结构的知识，甚至能够解密整个报文［Kaufman 1995］。

为了解决这个问题，可以在密文中混合某些随机性，使得相同的明文块产生不同的密文块。为了解释这个想法，令 $m(i)$ 表示第 i 个明文块，$c(i)$ 表示第 i 个密文块，并且 $a \oplus b$ 表示两个比特串 a 和 b 的异或（XOR）。（前面讲过 $0 \oplus 0 = 1 \oplus 1 = 0$ 和 $0 \oplus 1 = 1 \oplus 0 = 1$，并且两个比特串的异或是逐位进行的。例如 $10101010 \oplus 11110000 = 01011010$。）另外，将具有密钥 S 的块密码加密算法表示为 K_S。其基本思想如下：发送方为第 i 块生成一个随机的 k 比特数 $r(i)$，并且计算 $c(i) = K_S(m(i) \oplus r(i))$。注意到每块选择一个新的 k 比特随机数。则发送方发送 $c(1)$、$r(1)$、$c(2)$、$r(2)$、$c(3)$ 和 $r(3)$ 等等。因为接收方接收到 $c(i)$ 和 $r(i)$，它能够通过计算 $m(i) = K_S(c(i) \oplus r(i))$ 而恢复每个明文块。重要

的是注意到下列事实：尽管 $r(i)$ 是以明文发送的，并且因此能被 Trudy 嗅探到，但她无法获得明文 $m(i)$，因为她不知道密钥 K_S。同时注意到如果两个明文块$m(i)$和 $m(j)$ 是相同的，对应的密文块 $c(i)$ 和 $c(j)$ 将是不同的（只要随机数 $r(i)$ 和$r(i)$不同，这种情况出现的概率将很高）。

举例来说，考虑表 8-1 中的 3 比特块密码。假设明文是 010010010。如果 Alice 直接对此加密，没有包括随机性，得到的密文变为 101101101。如果 Trudy 嗅探到该密文，因为这三个密文块都是相同的，她能够正确地推断出这三个明文块都是相同的。现在假设 Alice 产生了随机块 $r(1)=001$、$r(2)=111$ 和 $r(3)=100$，并且使用了上述技术来生成密文 $c(1)=100$、$c(2)=010$ 和 $c(3)=000$。注意到即使明文块相同，三个密文块也是不同的。Alice 则发送 $c(1)$、$r(1)$、$c(2)$ 和 $r(2)$。读者可证实 Bob 能够使用共享的密钥 K_S 获得初始的明文。

精明的读者将注意到，引入随机性解决了一个问题而产生了另一个问题：Alice 必须传输以前两倍的比特。实际上，对每个加密比特，她现在必须再发送一个随机比特，使需要的带宽加倍。为了有效利用该技术，块密码通常使用了一种称为**密码块链接**（Cipher Block Chaining，CBC）的技术。其基本思想是仅随第一个报文发送一个随机值，然后让发送方和接收方使用计算的编码块代替后继的随机数。具体而言，CBC 运行过程如下：

1）在加密报文（或数据流）之前，发送方生成一个随机的 k 比特串，称为**初始向量**（Initialization Vector，IV）。将该初始向量表示为 $c(0)$。发送方以明文方式将 IV 发送给接收方。

2）对第一个块，发送方计算 $m(1)\oplus c(0)$，即计算第一块明文与 IV 的异或。然后通过块密码算法运行得到的结果以得到对应的密文块，即 $c(1)=K_S(m(1)\oplus c(0))$。发送方向接收方发送加密块 $c(1)$。

3）对于第 i 个块，发送方根据 $c(i)=K_S(m(i)\oplus c(i-1))$ 生成第 i 个密文块。

我们现在来考察这种方法的某些后果。首先，接收方将仍能够恢复初始报文。毫无疑问，当接收方接收到 $c(i)$ 时，它用 K_S 解密之以获得 $s(i)=m(i)\oplus c(i-1)$；因为接收方已经知道 $c(i-1)$，则从 $m(i)=s(i)\oplus c(i-1)$ 获得明文块。第二，即使两个明文块是相同的，相应的密文块也（几乎）总是不同的。第三，虽然发送方以明文发送 IV，入侵者将仍不能解密密文块，因为该入侵者不知道秘密密钥 S。最后，发送方仅发送一个最前面的块（即 IV），因此对（由数百块组成的）长报文而言增加的带宽用量微不足道。

举例来说，对表 8-1 中的 3 比特块密码，明文为 010010010 和 $IV=c(0)=001$，我们现在来确定其密文。发送方首先使用 IV 来计算 $c(1)=K_S(m(1)\oplus c(0))=100$。发送方然后计算 $c(2)=K_S(m(2)\oplus c(1))=K_S(010\oplus 100)=000$，并且 $c(3)=K_S(m(3)\oplus c(2))=K_S(010\oplus 000)=101$。读者可证实接收方若知道了 IV 和 K_S，将能够恢复初始的明文。

当设计安全网络协议时，CBC 有一种重要的后果：需要在协议中提供一种机制，以从发送方向接收方分发 IV。在本章稍后我们将看到几个协议是如何这样做的。

8.2.2　公开密钥加密

从凯撒密码时代直到 20 世纪 70 年代的两千多年以来，加密通信都需要通信双方共享一个共同秘密，即用于加密和解密的对称密钥。这种方法的一个困难是两方必须就共享密钥达成一致，但是这样做的前提是需要通信（可假定是安全的）！可能是双方首先会面，

人为协商确定密钥（例如，凯撒的两个百夫长在罗马浴室碰头），此后才能进行加密通信。但是，在网络世界中，通信各方之间可能从未见过面，也不会在网络以外的任何地方交谈。此时通信双方能够在没有预先商定的共享密钥的条件下进行加密通信吗？1976 年，Diffie 和 Hellman［Diffie 1976］论证了一个解决这个问题的算法（现在称为 Diffie-Hellman 密钥交换），这是个完全不同、极为优雅的安全通信算法，开创了如今的公开密钥密码系统的发展之路。我们很快就会看到公开密钥密码系统也有许多很好的特性，使得它不仅可以用于加密，还可以用于鉴别和数字签名。有趣的是，最近发现 20 世纪 70 年代早期由英国通信电子安全团体的研究人员独立研究的一系列秘密报告中的思想，与［Diffie 1976］和［RSA 1978］中的思想类似［Ellis 1987］。事实常常如此，伟大的想法通常会在许多地方独立地闪现；幸运的是，公钥的进展不仅秘密地发生，而且也在公众视野中发生。

公开密钥密码的使用在概念上相当简单。假设 Alice 要和 Bob 通信。如图 8-6 所示，这时 Alice 和 Bob 并未共享一个密钥（如同在对称密钥系统情况下），而 Bob（Alice 报文的接收方）则有两个密钥，一个是世界上任何人（包括入侵者 Trudy）都可得到的**公钥**（public key），另一个是只有 Bob 知道的**私钥**（private key）。我们使用符号 K_B^+ 和 K_B^- 来分别表示 Bob 的公钥和私钥。为了与 Bob 通信，Alice 首先取得 Bob 的公钥，然后用这个公钥和一个众所周知的（例如，已标准化的）加密算法，加密她要传递给 Bob 的报文 m；即 Alice 计算 $K_B^+(m)$。Bob 接收到 Alice 的加密报文后，用其私钥和一个众所周知的（例如，已标准化的）解密算法解密 Alice 的加密报文，即 Bob 计算 $K_B^-(K_B^+(m))$。后面我们将看到，存在着可以选择公钥和私钥的加密/解密算法和技术，使得 $K_B^-(K_B^+(m))=m$；也就是说，用 Bob 的公钥 K_B^+ 加密报文 m（得到 $K_B^+(m)$），然后再用 Bob 的私钥 K_B^- 解密报文的密文形式（就是计算 $K_B^-(K_B^+(m))$）就能得到最初的明文 m。这是个不寻常的结果！用这种办法，Alice 可以使用 Bob 公开可用的密钥给 Bob 发送机密信息，而他们任一方都无须分发任何密钥！我们很快能够看到，公钥和私钥加密相互交换同样能够得到不寻常的结果，即 $K_B^-(K_B^+(m))=K_B^+(K_B^-(m))=m$。

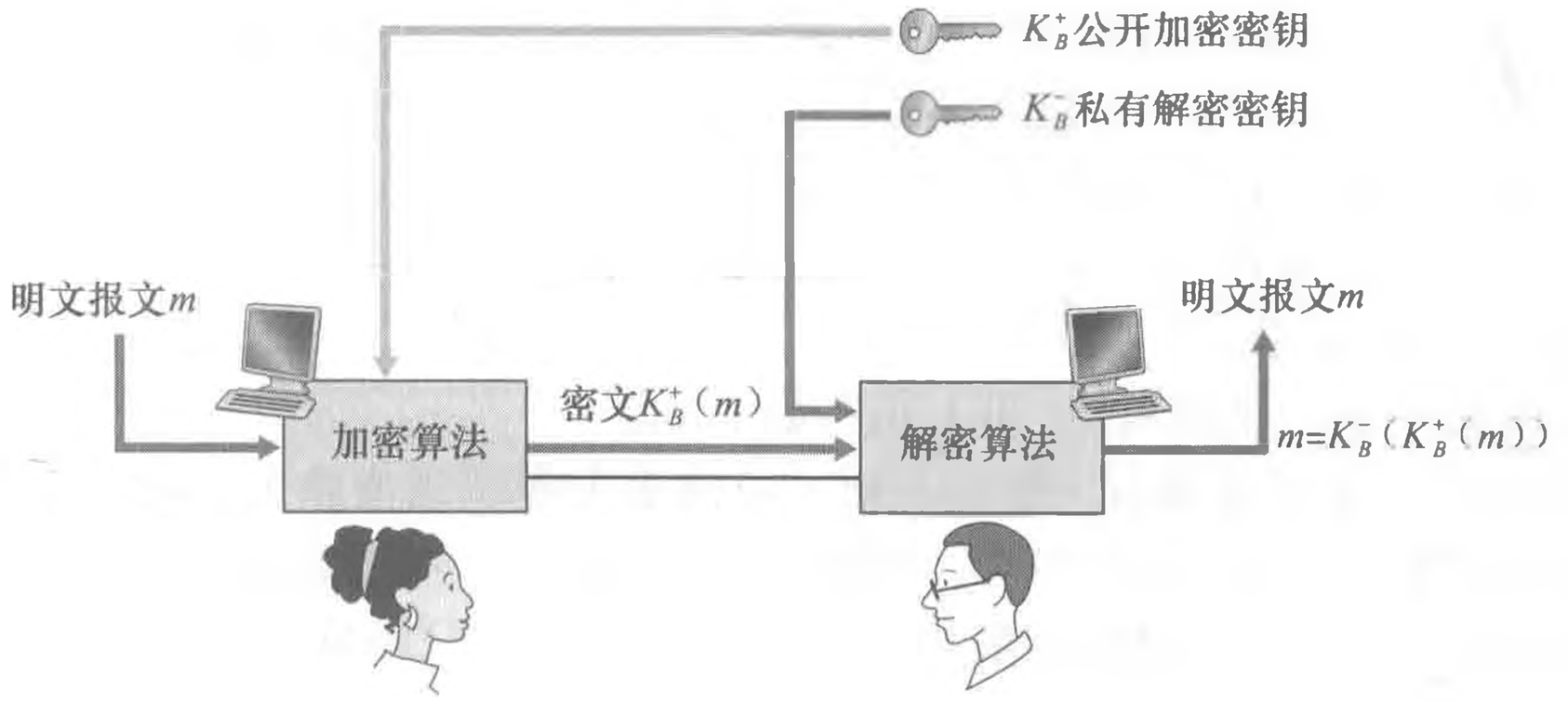

图 8-6　公开密钥密码

因此公开密钥密码体制的使用在概念上是简单的。但是有两点必须要注意。首先应关注的是，尽管入侵者截取到 Alice 的加密报文时看到的只是乱码，但是入侵者知道公钥（显然 Bob 的公钥是全世界都可以使用的）和 Alice 加密所用的算法。Trudy 可以据此发起选择明文攻击，使用已知的标准加密算法和 Bob 的公开可用的加密密钥对她所选择的任意报文编码！例如，Trudy 可以尝试对她怀疑 Alice 可能发送的全部报文或部分报文编码。很

明显，要使公开密钥密码能工作，密钥选择和加密/解密算法必须保证任意入侵者都不能（至少要困难得几乎不可能）确定出 Bob 的私钥或者以某种方式解密或猜出 Alice 发给 Bob 的报文。第二个值得关注的问题是，既然 Bob 的加密密钥是公开的，任何人（包括 Alice 和其他声称自己是 Alice 的人）都可能向 Bob 发送一个已加密的报文。在单一共享密钥情况下，发送方知道共享秘密密钥的事实就已经向接收方隐含地证实了发送方的身份。然而在公钥体制中，这点就行不通了，因为任何一个人都可向 Bob 发送使用 Bob 的公开可用密钥加密的报文。这就需要用数字签名把发送方和报文绑定起来，数字签名是我们在 8.3 节中讨论的主题。

1. RSA

尽管可能有许多算法处理这些关注的问题，但 **RSA 算法**（RSA algorithm，取算法创立人 Ron Rivest、Adi Shamir 和 Leonard Adleman 的首字母命名）几乎已经成了公开密钥密码的代名词。我们首先来理解 RSA 是如何工作的，然后再考察 RSA 的工作原理。

RSA 广泛地使用了模 n 算术的算术运算。故我们简要地回顾一下模算术。前面讲过 $x \bmod n$ 只是表示被 n 除时 x 的余数；因此如 $19 \bmod 5 = 4$。在模算术中，人们执行通常的加法、乘法和指数运算。然而，每个运算的结果由整数余数代替，该余数是被 n 除后留下的数。对于模算术的加法和乘法可由下列便于施用的事实所简化：

$$[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$$
$$[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$$
$$[(a \bmod n) \cdot (b \bmod n)] \bmod n = (a \cdot b) \bmod n$$

从第三个事实推出 $(a \bmod n)^d \bmod n = a^d \bmod n$，我们很快将会发现这个恒等式是非常有用的。

现在假设 Alice 要向 Bob 发送一个 RSA 加密的报文，如图 8-6 所示。在我们的讨论中，心中永远要记住一个报文只不过是一种比特模式，并且所有比特模式能唯一地被一个整数（连同该比特模式的长度）表示。例如，假设一个报文是比特模式 1001；这个报文能由十进制整数 9 来表示。所以，当用 RSA 加密一个报文时，等价于加密表示该报文的这个唯一的整数。

RSA 有两个互相关联的部分：

- 公钥和私钥的选择。
- 加密和解密算法。

为了生成 RSA 的公钥和私钥，Bob 执行如下步骤：

1）选择两个大素数 p 和 q。那么 p 和 q 应该多大呢？该值越大，破解 RSA 越困难，而执行加密和解密所用的时间也越长。RSA 实验室推荐，公司使用时，p 和 q 的乘积为 1024 比特的数量级。对于选择大素数的方法的讨论，参见［Caldwell 2012］。

2）计算 $n = pq$ 和 $z = (p-1)(q-1)$。

3）选择小于 n 的一个数 e，且使 e 和 z 没有（非 1 的）公因数。（这时称 e 与 z 互素。）使用字母 e 表示是因为这个值将被用于加密。

4）求一个数 d，使得 $ed - 1$ 可以被 z 整除（就是说，没有余数）。使用字母 d 表示是因为这个值将用于解密。换句话说，给定 e，我们选择 d，使得

$$ed \bmod z = 1$$

5）Bob 使外界可用的公钥 K_B^+ 是一对数（n，e），其私钥 K_B^- 是一对数（n，d）。

Alice 执行的加密和 Bob 进行的解密过程如下：

- 假设 Alice 要给 Bob 发送一个由整数 m 表示的比特组合，且 $m < n$。为了进行编码，Alice 执行指数运算 m^e，然后计算 m^e 被 n 除的整数余数。换言之，Alice 的明文报文 m 的加密值 c 就是：

$$c = m^e \bmod n$$

  对应于这个密文 c 的比特模式发送给 Bob。
- 为了对收到的密文报文 c 解密，Bob 计算：

$$m = c^d \bmod n$$

  这要求使用他的私钥 (n, d)。

举一个简单的 RSA 例子，假设 Bob 选择 $p=5$ 和 $q=7$。（坦率地讲，这样小的值无法保证安全。）则 $n=35$ 和 $z=24$。因为 5 和 24 没有公因数，所以 Bob 选择 $e=5$。最后，因为 $5\times29-1$（即 $ed-1$）可以被 24 整除，所以 Bob 选择 $d=29$。Bob 公开了 $n=35$ 和 $e=5$ 这两个值，并秘密保存了 $d=29$。观察公开的这两个值，假定 Alice 要发送字母“l”“o”“v”和“e”给 Bob。用 1～26 之间的每个数表示一个字母，其中 1 表示“a”，…，26 表示“z”，Alice 和 Bob 分别执行如表 8-2 和表 8-3 所示的加密和解密运算。注意到在这个例子中，我们认为每四个字母作为一个不同报文。一个更为真实的例子是把这四个字母转换成它们的 8 比特 ASCII 表示形式，然后加密与得到的 32 比特的比特模式对应的整数。（这样一个真实的例子产生了一些长得难以在教科书中打印出来的数！）

表 8-2　Alice 的 RSA 加密，$e=5$，$n=35$

明文字母	m：数字表示	m^e	密文 $c=m^e \bmod n$
l	12	248832	17
o	15	759375	15
v	22	5153632	22
e	5	3125	10

表 8-3　Bob 的 RSA 解密，$d=29$，$n=35$

密文 c	c^d	$m=c^d \bmod n$	明文字母
17	481968572106750915091411825223071697	12	l
15	12783403948858939111232757568359375	15	o
22	851643319086537701956194499721106030592	22	v
10	10000000000000000000000000000000	5	e

假定表 8-2 和表 8-3 中的简单示例已经产生了某些极大的数，并且假定我们前面看到 p 和 q 都是数百比特长的数，这些都是实际使用 RSA 时必须要牢记的。如何选择大素数？如何选择 e 和 d？如何对大数进行指数运算？对这些重要问题的详细讨论超出了本书的范围，详情请参见［Kaufman 1995］以及其中的参考文献。

2. 会话密钥

这里我们注意到，RSA 所要求的指数运算是相当耗费时间的过程。形成对比的是，DES 用软件实现要比 RSA 快 100 倍，用硬件实现则要快 1000～10 000 倍［RSA Fast 2012］。所以，在实际应用中，RSA 通常与对称密钥密码结合起来使用。例如，如果 Alice 要向 Bob 发送大量的加密数据，她可以用下述方式来做。首先，Alice 选择一个用于加密数据本身的密钥，这个密钥有时称为**会话密钥**（session key），该会话密钥表示为 K_S。

Alice 必须把这个会话密钥告知 Bob，因为这是他们在对称密钥密码（如 DES 或 AES）中所使用的共享对称密钥。Alice 可以使用 Bob 的 RSA 公钥来加密该会话密钥，即计算 $c = (K_S)^e \bmod n$。Bob 收到了该 RSA 加密的会话密钥 c 后，解密得到会话密钥 K_S。Bob 此时已经知道 Alice 将要用于加密数据传输的会话密钥了。

3. RSA 的工作原理

RSA 加密/解密看起来相当神奇。为什么那样应用加密算法，然后再运行解密算法，就能恢复出初始报文呢？要理解 RSA 的工作原理，我们仍将记 $n = pq$，其中 p 和 q 是 RSA 算法中的大素数。

前面讲过，在 RSA 加密过程中，一个报文 m（唯一地表示为整数）使用模 n 算术做 e 次幂运算，即

$$c = m^e \bmod n$$

解密则先对该值执行 d 次幂运算，再做模 n 运算。因此先加密再解密的结果是 $(m^e \bmod n)^d \bmod n$。下面我们来看关于这个量能够得到什么。正如前面提到的，模算术的一个重要性质是对于任意值 a、n 和 d 都有 $(a \bmod n)^d \bmod n = a^d \bmod n$。因此，在这个性质中使用 $a = m^e$，则有

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

因此剩下证明 $m^{ed} \bmod n = m$。尽管我们正试图揭开 RSA 工作原理的神秘面纱，但为了做到这一点，我们还需要用到数论中一个相当神奇的结果。具体而言，就是要用到数论中这样的结论：如果 p 和 q 是素数，且有 $n = pq$ 和 $z = (p-1)(q-1)$，则 $x^y \bmod n$ 与 $x^{(y \bmod z)} \bmod n$ 是等同的［Kaufman 1995］。应用这个结论，对于 $x = m$ 和 $y = ed$，可得

$$m^{ed} \bmod n = m^{(ed \bmod z)} \bmod n$$

但是要记住，我们是这样选择 e 和 d 的，即 $ed \bmod z = 1$。这告诉我们

$$m^{ed} \bmod n = m^1 \bmod n = m$$

这正是我们希望得到的结果！先对 m 做 e 次幂运算（加密）再做 d 次幂运算（解密），然后做模 n 的算术运算（原文中没有这句，译者认为有必要补上。——译者注），就可得到初始的 m。甚至更为奇妙之处是这样一个事实，如果我们先对 m 做 d 次幂运算（加密）再做 e 次幂运算，即颠倒加密和解密的次序，先执行解密操作再执行加密操作，也能得到初始值 m。这个奇妙的结果完全遵循下列模算术：

$$(m^d \bmod n)^e \bmod n = m^{de} \bmod n = m^{ed} \bmod n = (m^e \bmod n)^d \bmod n$$

RSA 的安全性依赖于这样的事实：目前没有已知的算法可以快速进行一个数的因数分解，这种情况下公开值 n 无法快速分解成素数 p 和 q。如果已知 p 和 q，则给定公开值 e，就很容易计算出秘密密钥 d。另一方面，不确定是否存在因数分解一个数的快速算法，从这种意义上来说，RSA 的安全性也不是确保的。

另一种流行的公钥加密算法是 Diffie- Hellman，我们将在课后习题中简要探讨它。Diffie-Hellman 并不像 RSA 那样多功能，即它不能用于加密任意长度的报文；然而，它能够用来创建一个对称的会话密钥，该密钥再被用于加密长报文。

8.3　报文完整性和数字签名

在前面一节中我们看到了能够使用加密为两个通信实体提供机密性。在本节中我们转向提供**报文完整性**（message integrity）这个同等重要的主题。报文完整性也称为报文鉴

别。连同报文完整性，在本节中我们将讨论两个相关的主题：数字签名和端点鉴别。

我们再次使用 Alice 和 Bob 来定义报文完整性问题。假定 Bob 接收到一个报文（这可能已经加密或可能是明文），并且他认为这个报文是由 Alice 发送的。为了鉴别这个报文，Bob 需要证实：

1）该报文的确源自 Alice。

2）该报文在到 Bob 的途中没有被篡改。

我们将在 8.4 ~ 8.7 节中看到，报文完整性这个问题在所有安全网络协议中都是至关重要的。

举一个特定的例子，考虑一个使用链路状态路由选择算法（例如 OSPF）的计算机网络，在该网络中决定每对路由器之间的路由（参见第 5 章）。在一个链路状态算法中，每台路由器需要向该网络中的所有其他路由器广播一个链路状态报文。路由器的链路状态报文包括直接相连邻居的列表以及到这些邻居的直接费用。一旦某台路由器从其他所有路由器收到了链路状态报文，它能够生成该网络的全图，运行它的最小费用路由选择算法并配置它的转发表。对路由选择算法的一个相对容易的攻击是，Trudy 分发具有不正确状态信息的虚假链路状态报文。因此产生了报文完整性的需求：当路由器 B 收到来自路由器 A 的链路状态报文时，路由器 B 应当证实路由器 A 实际生成了该报文，并且进一步证实在传输过程中该报文没有被篡改。

在本节中，我们描述一种由许多安全网络协议所使用的流行报文完整性技术。但在做此事之前，我们需要涉及密码学中的另一个重要主题，即密码散列函数。

8.3.1　密码散列函数

如图 8-7 所示，散列函数以 m 为输入，并计算得到一个称为散列的固定长度的字符串 $H(m)$。因特网检验和（第 3 章）和 CRC（第 6 章）都满足这个定义。**密码散列函数**（cryptographic hash function）要求具有下列附加的性质：

- 找到任意两个不同的报文 x 和 y 使得 $H(x) = H(y)$，在计算上是不可能的。

不严格地说，这种性质就意味着入侵者在计算上不可能用其他报文替换由散列函数保护的报文。这就是说，如果（m，$H(m)$）是报文和由发送方生成的报文散列的话，则入侵者不可能伪造另一个报文 y 的内容，使得该报文具有与原报文相同的散列值。

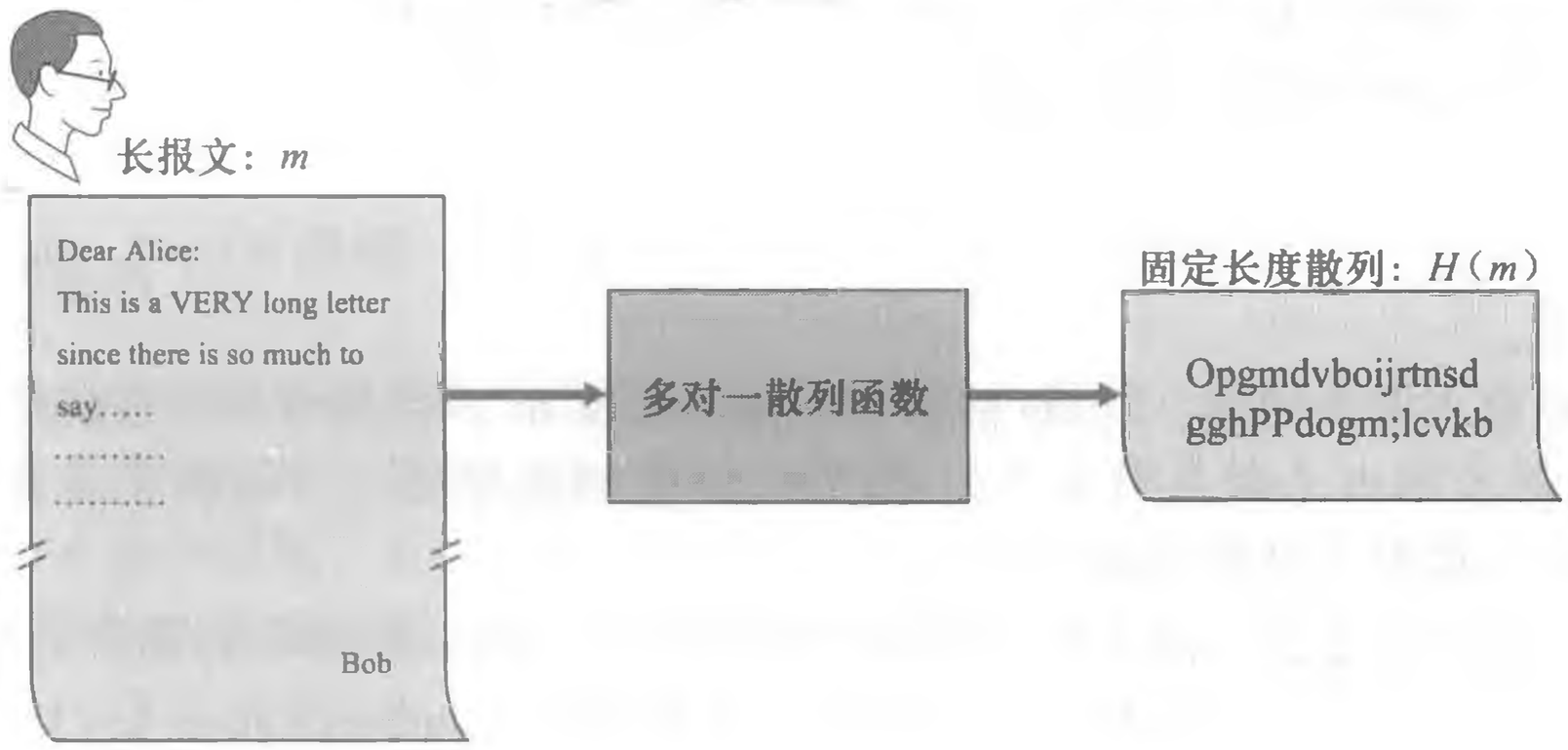

图 8-7　散列函数

我们来证实一个简单的检验和（如因特网检验和）只能算作劣质的密码散列函数。不

像在因特网检验和中执行反码运算那样，我们把每个字符看作一个字节，并把这些字节加到一起，一次用4字节的块来进行计算。假定Bob欠Alice 100.99美元并且向Alice发送一张借据，这个借据包含文本字符串“IOU100.99BOB”。这些字符的ASCII表示（以十六进制形式）为49，4F，55，31，30，30，2E，39，39，42，4F，42。

图8-8上半部分显示了这个报文的4字节检验和是B2 C1 D2 AC。图8-8下半部分显示了一条稍微不同的报文（但是Bob要付的钱却多了许多）。报文“IOU100.99BOB”和“IOU900.19BOB”有相同的检验和。因此，这种简单的检验和算法违反了上述要求。给定初始数据，很容易找到有相同检验和的另一组数据。很明显，为了安全起见，我们需要比检验和更为强有力的散列函数。

报文	ASCII表示				
I O U 1	49	4F	55	31	
0 0 . 9	30	30	2E	39	
9 B O B	39	42	4F	42	
	B2	C1	D2	AC	检验和

报文	ASCII表示				
I O U 9	49	4F	55	39	
0 0 . 1	30	30	2E	31	
9 B O B	39	42	4F	42	
	B2	C1	D2	AC	检验和

图8-8　初始报文和欺诈报文具有相同的检验和

Ron Rivest［RFC 1321］的MD5散列算法如今正在广泛使用。这个算法通过4步过程计算得到128比特的散列。这4步过程由下列步骤组成：①填充——先填1，然后填足够多的0，直到报文长度满足一定的条件；②添加——在填充前添加一个用64比特表示的报文长度；③初始化累加器；④循环——在最后的循环步骤中，对报文的16字块进行4轮处理。MD5的描述（包括一个C源代码实现）可参见［RFC 1321］。

目前正使用的第二个主要散列算法是安全散列算法SHA-1（Security Hash Algorithm）［FIPS 1995］。这个算法的原理类似于MD4［RFC 1320］设计中所使用的原理，而MD4是MD5的前身。SHA-1是美国联邦政府的标准，任何联邦政府的应用程序如果需要使用密码散列算法的话，都要求使用SHA-1。SHA-1生成一个160比特的报文摘要。较长的输出长度可使SHA-1更安全。

8.3.2　报文鉴别码

我们现在再回到报文完整性的问题。既然我们理解了散列函数，就先来看一下将如何执行报文完整性：

1）Alice生成报文m并计算散列$H(m)$（例如使用SHA-1）。

2）然后Alice将$H(m)$附加到报文m上，生成一个扩展报文$(m, H(m))$，并将该扩展报文发给Bob。

3）Bob接收到一个扩展报文(m, h)并计算$H(m)$。如果$H(m)=h$，Bob得到结论：一切正常。

这种方法存在明显缺陷。Trudy能够生成虚假报文m'，在其中声称她就是Alice，计算$H(m')$并发送给Bob$(m', H(m'))$。当Bob接收到该报文，一切将在步骤3中核对通过，并且Bob无法猜出这种不轨的行为。

为了执行报文完整性，除了使用密码散列函数外，Alice和Bob将需要共享秘密s。这个共享的秘密只不过是一个比特串，它被称为**鉴别密钥**（authentication key）。使用这个共享秘密，报文完整性能够执行如下：

1）Alice生成报文m，用s级联m以生成$m+s$，并计算散列$H(m+s)$（例如使用SHA-1）。$H(m+s)$被称为**报文鉴别码**（Message Authentication Code，MAC）。

2）然后 Alice 将 MAC 附加到报文 m 上，生成扩展报文（m，$H(m+s)$），并将该扩展报文发送给 Bob。

3）Bob 接收到一个扩展报文（m，h），由于知道 s，计算出报文鉴别码 $H(m+s)$。如果 $H(m+s)=h$，Bob 得到结论：一切正常。

图 8-9 中总结了上述过程。读者应当注意到这里的 MAC（表示“报文鉴别码”）与用于数据链路层中的 MAC（表示“媒体访问控制”）是不一样的！

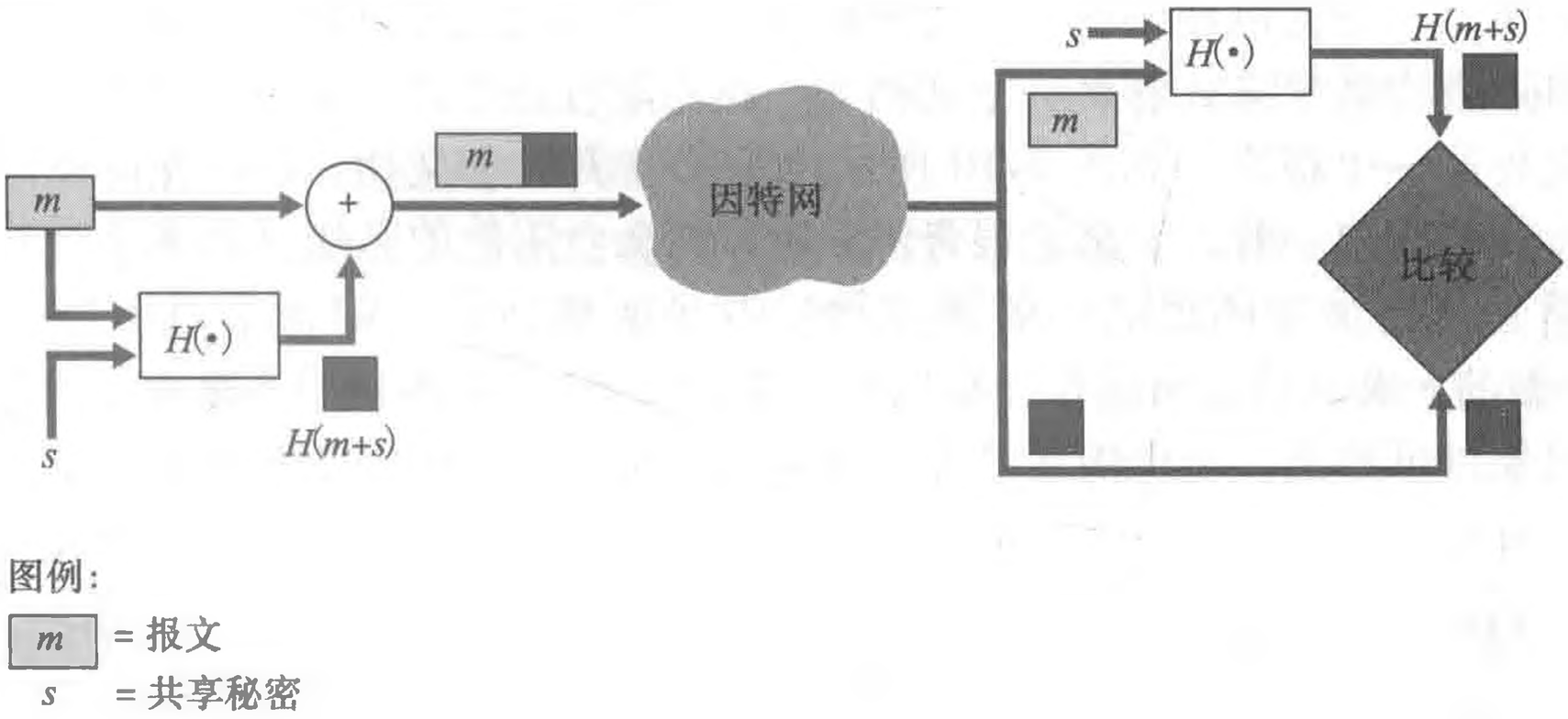

图 8-9　报文鉴别码

MAC 的一个优良特点是它不要求一种加密算法。的确，在许多应用中，包括前面讨论的链路状态路由选择算法，通信实体仅关心报文完整性，并不关心报文机密性。使用 MAC，实体能够鉴别它们相互发送的报文，而不必在完整性过程中综合进复杂的加密过程。

如你所猜测，多年来已经提出了若干种对 MAC 的不同标准。目前最为流行的标准是 HMAC，它能够与 MD5 或 SHA-1 一道使用。HMAC 实际上通过散列函数运行数据和鉴别密钥两次［Kaufman 1995；RFC 2104］。

这里还遗留下一个重要问题。怎样向通信实体分发这个共享的鉴别密钥呢？例如，在链路状态路由选择算法中，在某种程度上需要向自治系统中的每台路由器分发该秘密鉴别密钥。（注意到所有路由器都能够使用相同的鉴别密钥。）一名网络管理员能够通过物理上访问每台路由器来实际完成这项工作。或者，如果这名网络管理员不够勤快，并且每台路由器都有它自己的公钥，那么该网络管理员能够用路由器的公钥加密鉴别密钥并分发给任何一台路由器，从而通过网络向路由器发送加密的密钥。

8.3.3　数字签名

回想在过去的一周中你在纸上已经签过多少次你的名字。你可能经常会在支票、信用卡收据、法律文件和信件上签名。你的签名证明你（而不是其他人）承认和/或同意这些文件的内容。在数字领域，人们通常需要指出一个文件的所有者或创作者，或者表明某人认可一个文件内容。**数字签名**（digital signature）就是一种在数字领域实现这些目标的密码技术。

正如手工签字一样，数字签名也应当以可鉴别的、不可伪造的方式进行。这就是说，必须能够证明由某个人在一个文件上的签名确实是由该人签署的（该签名必须是可证实的），且只有那个人能够签署那个文件（该签名无法伪造）。

我们现在来考虑怎样设计一个数字签名方案。当 Bob 签署一个报文时，可以观察到 Bob 必须将某些对他独特的东西放置在该报文上。Bob 可以考虑附加一个 MAC 用作签名，其中 MAC 是由他的密钥（对他是独特的）作用到该报文上而生成的，然后得到该散列值。而 Alice 为了验证该签名，她必须也具有该密钥的副本，在这种情况下该密钥对 Bob 将不是唯一的。因此，此时 MAC 是无法胜任这项工作的。

前面讲过使用公钥密码，Bob 具有公钥和私钥，这两种密钥对 Bob 均为独特的。因此，公钥密钥是一种提供数字签名的优秀候选者。我们现在来研究一下这是怎样完成的。

假设 Bob 要以数字方式签署一个文档 m。我们能够想象这个文档是 Bob 打算签署并发送的一个文件或一个报文。如图 8-10 所示，为了签署这个文档，Bob 直接使用他的私钥 K_B^- 计算 $K_B^-(m)$。乍一看，会感觉很奇怪，Bob 怎么会用他的私钥（在 8.2 节中，我们用私钥解密用其公钥加密的报文）签署文档！但是回想加密和解密都只不过是数学运算（RSA 中所做的 e 或 d 指数幂运算；参见 8.2 节），并且 Bob 的目的不是弄乱或掩盖文档的内容，而只是以可鉴别、不可伪造的方式签署这个文档。Bob 对文档 m 签名之后所得的文档就是 $K_B^-(m)$。

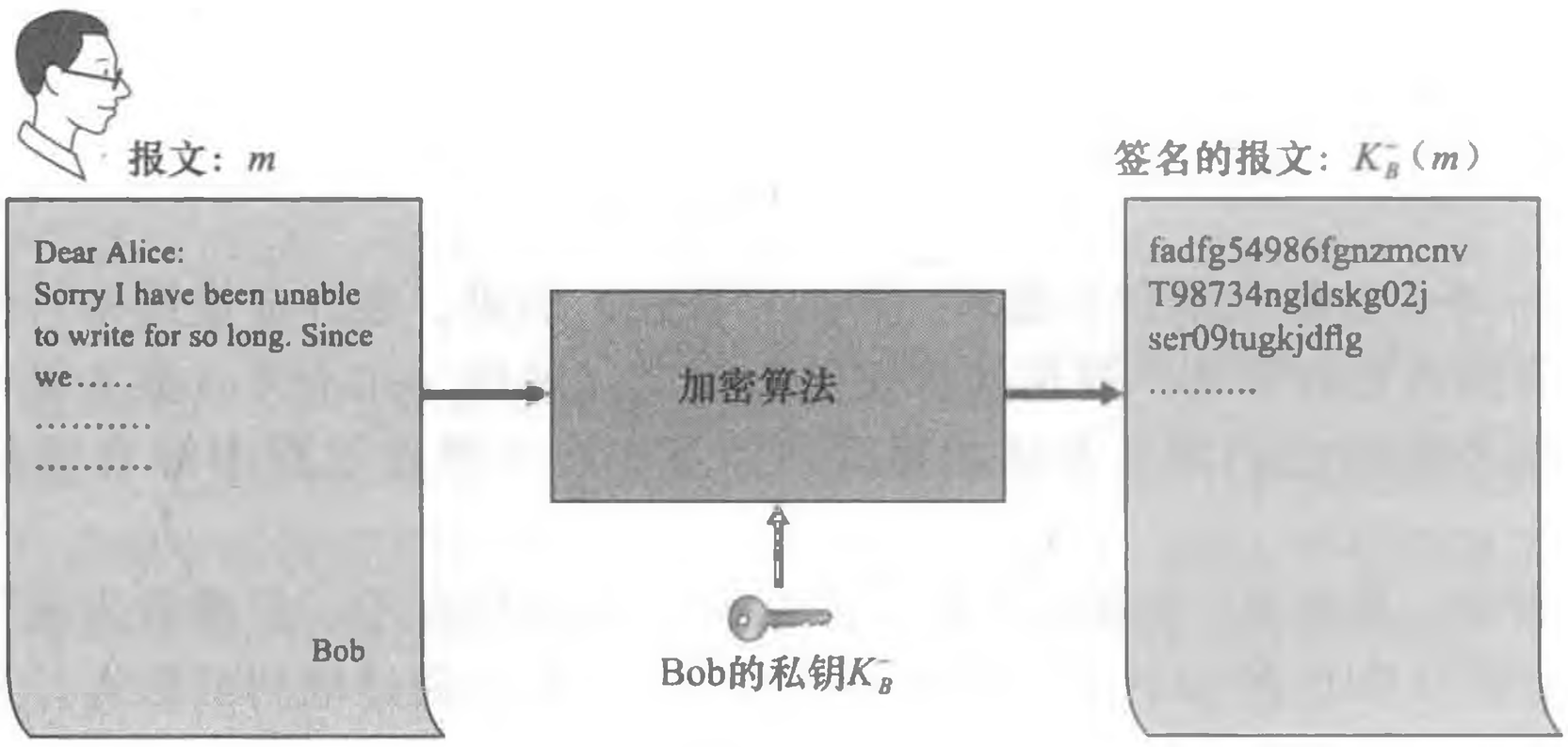

图 8-10　为一个文档生成一个数据签名

数字签名 $K_B^-(m)$ 是否满足了可鉴别、不可伪造的需求？假设 Alice 有 m 和 $K_B^-(m)$。她要在法庭上证明（进行诉讼）Bob 确实签署过这个文档，他就是唯一能够签署该文档的人。Alice 持有 Bob 的公钥 K_B^+，并把它用于 Bob 的数字签名 $K_B^-(m)$，从而得到了文档 m。也就是说，Alice 计算 $K_B^+(K_B^-(m))$。瞧！在 Alice 经历了令人注目的慌乱后得到了 m，它与初始文档完全一致。然后，Alice 就可以论证仅有 Bob 能够签署这个文档，基于如下理由：

- 无论是谁签署这个报文，都必定在计算签名 $K_B^-(m)$ 过程中使用了 K_B^- 这个私钥，使 $K_B^+(K_B^-(m)) = m$。
- 知道 K_B^- 这个私钥的唯一人只有 Bob。从 8.2 节我们对 RSA 的讨论中可知，知道公钥 K_B^+ 无助于得知私钥 K_B^- 的信息。因此，知道私钥 K_B^- 的人才是生成密钥对（K_B^+，K_B^-）的人，而这个人首当其冲就是 Bob。（注意到此处假设 Bob 没有把 K_B^- 泄露给任何人，也没有人从 Bob 处窃取到 K_B^-。）

注意到下列问题是重要的：如果源文档 m 被修改过，比如改成了另一个文档 m'，则 Bob 对 m 生成的签名对 m'无效，因为 $K_B^+(K_B^-(m))$ 不等于 m'。因此我们看到数字签名也

提供完整性，使得接收方验证该报文未被篡改，同时也验证了该报文的源。

对用加密进行数据签名的担心是，加密和解密的计算代价昂贵。给定加解密的开销，通过完全加密/解密对数据签名是杀鸡用牛刀。更有效的方法是将散列函数引入数字签名。8.3.2 节中讲过，一种散列算法取一个任意长的报文 m，计算生成该报文的一个固定长度的数据“指纹”，表示为 $H(m)$。使用散列函数，Bob 对报文的散列签名而不是对报文本身签名，即 Bob 计算 $K_B^-(H(m))$。因为 $H(m)$ 通常比报文 m 小得多，所以生成数字签名所需要的计算量大为降低。

在 Bob 向 Alice 发送一个报文的情况下，图 8-11 提供了生成数字签名的操作过程的概览。Bob 让他的初始长报文通过一个散列函数。然后他用自己的私钥对得到的散列进行数字签名。明文形式的初始报文连同已经数字签名的报文摘要（从此以后可称为数字签名）一道被发送给 Alice。图 8-12 提供了鉴别报文完整性的操作过程的概览。Alice 先把发送方的公钥应用于报文获得一个散列结果。然后她再把该散列函数应用于明文报文以得到第二个散列结果。如果这两个散列匹配，则 Alice 可以确信报文的完整性及其发送方。

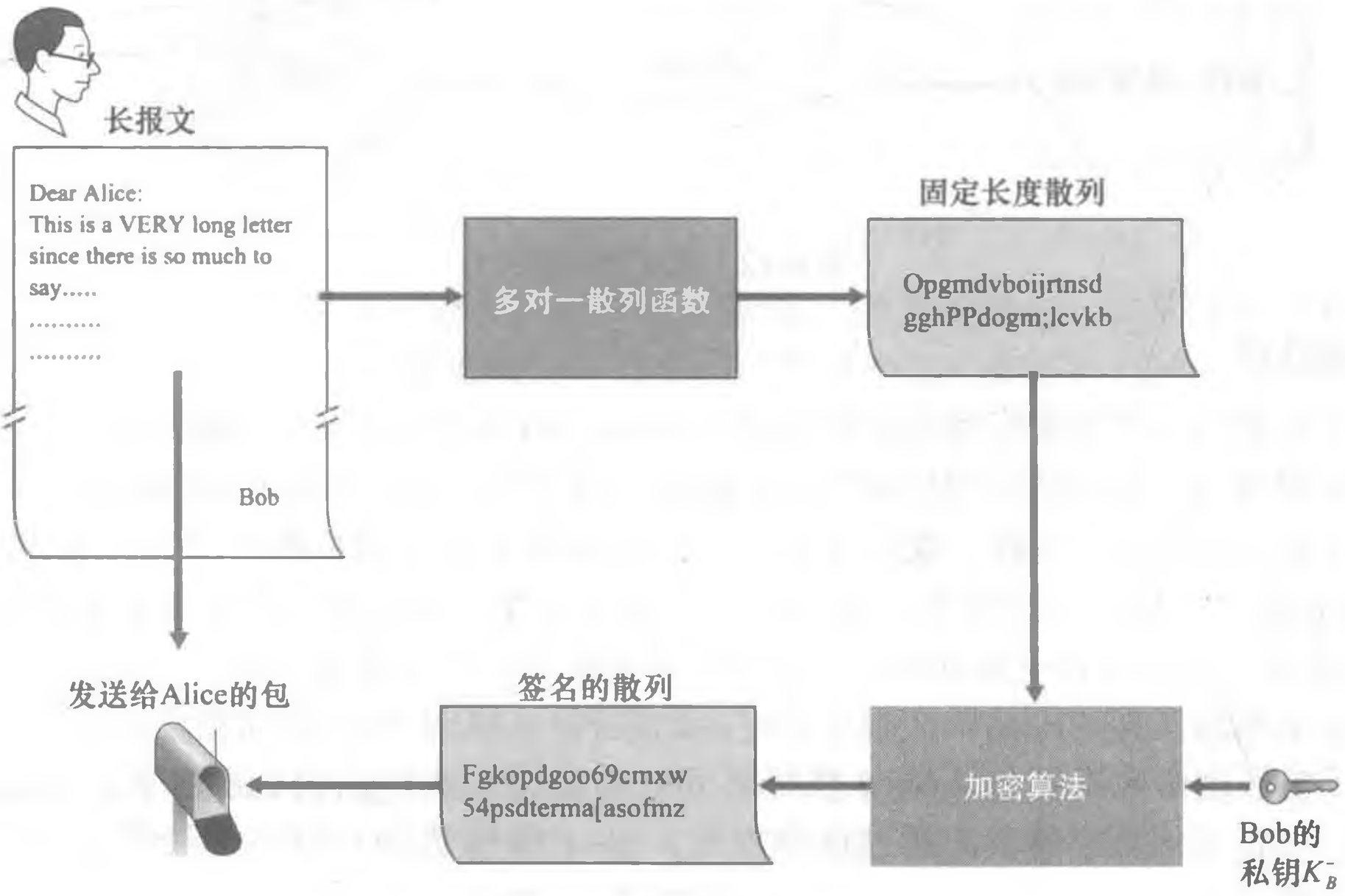

图 8-11 发送数字签名的报文

在继续学习之前，我们简要地将数字签名与 MAC 进行比较，尽管它们有类似之处，但也有重要的微妙差异。数字签名和 MAC 都以一个报文（或一个文档）开始。为了从该报文中生成一个 MAC，我们为该报文附加一个鉴别密钥，然后取得该结果的散列。注意到在生成 MAC 过程中既不涉及公开密钥加密，也不涉及对称密钥加密。为了生成一个数字签名，我们首先取得该报文的散列，然后用我们的私钥加密该报文（使用公钥密码）。因此，数字签名是一种“技术含量更高”的技术，因为它需要一个如后面描述的、具有认证中心支撑的公钥基础设施（PKI）。我们将在 8.4 节中看到，PGP 是一种流行的安全电子邮件系统，为了报文完整性而使用数字签名。我们已经看到了 OSPF 为了报文完整性而使用 MAC。我们将在 8.5 节和 8.6 节中看到 MAC 也能用于流行的运输层和网络层安全协议。

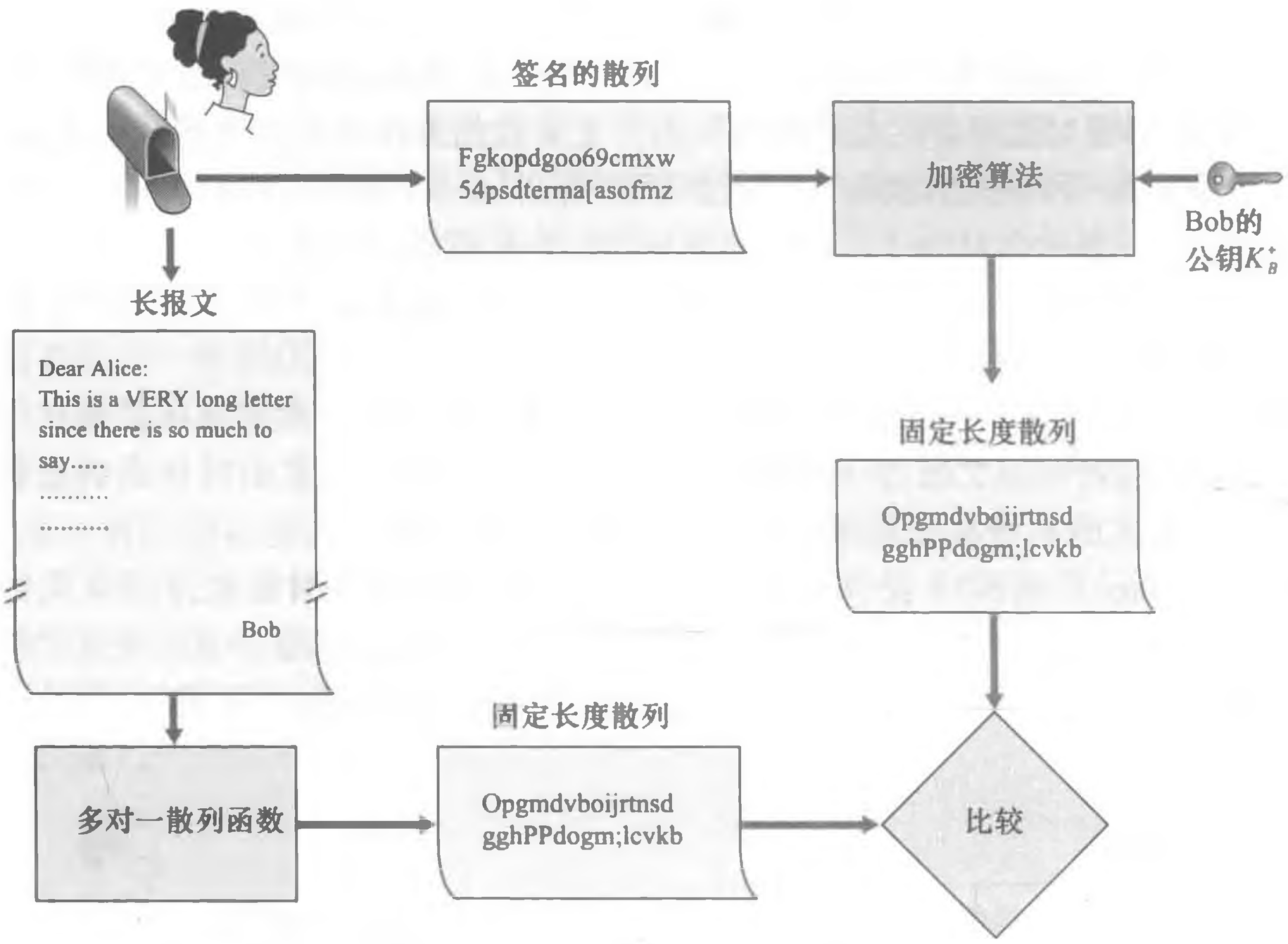

图 8-12　验证签名报文

公钥认证

数字签名的一个重要应用是**公钥认证**（public key certification），即证实一个公钥属于某个特定的实体。公钥认证用在许多流行的安全网络协议中，包括 IPsec 和 SSL。

为了深入理解这个问题，我们考虑一个因特网商务版本的经典的“比萨恶作剧”。假定 Alice 正在从事比萨派送业务，从因特网上接受订单。Bob 是一个爱吃比萨的人，他向 Alice 发送了一份包含其家庭地址和他希望的比萨类型的明文报文。Bob 在这个报文中也包含一个数字签名（即对原始明文报文的签名的散列），以向 Alice 证实他是该报文的真正来源。为了验证这个数字签名，Alice 获得了 Bob 的公钥（也许从公钥服务器或通过电子邮件报文）并核对该数字签名。通过这种方式，Alice 确信是 Bob 而不是某些青少年恶作剧者下的比萨订单。

在聪明的 Trudy 出现之前，这一切看起来进行得相当好。如图 8-13 中所示，Trudy 沉溺于一场恶作剧中。Trudy 向 Alice 发送一个报文，在这个报文中她说她是 Bob，给出了 Bob 家的地址并订购了一个比萨。在这个报文中，她也包括了她（Trudy）的公钥，虽然 Alice 自然地假定它就是 Bob 的公钥。Trudy 也附加了一个签名，但这是用她自己（Trudy）的私钥生成的。在收到该报文后，Alice 就会用 Trudy 的公钥（Alice 认为它是 Bob 的公钥）来解密该数字签名，并得到结论：这个明文报文确实是由 Bob 生成的。而当外送人员带着具有意大利辣香肠和凤尾鱼的比萨到达 Bob 家时，他会感到非常惊讶！

从这个例子我们看到，要使公钥密码有用，需要能够证实你具有的公钥实际上就是与你要进行通信的实体（人员、路由器、浏览器等）的公钥。例如，当 Alice 与 Bob 使用公钥密码通信时，她需要证实她假定是 Bob 的那个公钥确实就是 Bob 的公钥。

将公钥与特定实体绑定通常是由**认证中心**（Certification Authority，CA）完成的，CA 的职责就是使识别和发行证书合法化。CA 具有下列作用：