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	i
15	12783403948858939111232757568359375	15	o
22	851643319086537701956194499721106030592	22	v
10	100000000000000000000000000000	5	e

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 将要用于她的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，我们仍将表示 $n=pq$，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，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，关于这个量我们能够得到什么。因为前面提到，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（加密）再做 d 次幂（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（加密）再做 e 次幂，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，我们也能得到初始值 m。这个奇妙的结果完全遵循下列模算术：