毒扫描。因为扫描速度很慢，所以要保持效率就应该仅对上次扫描后被改动的文件进行检查。但是，聪明的病毒会把感染过的文件日期重置为初始日期以逃避检验。于是，反病毒程序修改校验文件所在目录的日期。但是病毒接着又把目录的日期也改掉。这就像我们上面所提到的猫捉老鼠游戏一样。

反病毒软件的另一种方法是检测文件，记录和存放所有文件的长度。如果一个文件自上周以来突然增加了许多，就有可能被感染，如图9-32a所示。但是，聪明的病毒可通过程序压缩原有文件并将其填充到原有长度来逃避检查。要使这种方法奏效，病毒必须还要包含压缩和解压缩过程，如图9-32c所示。

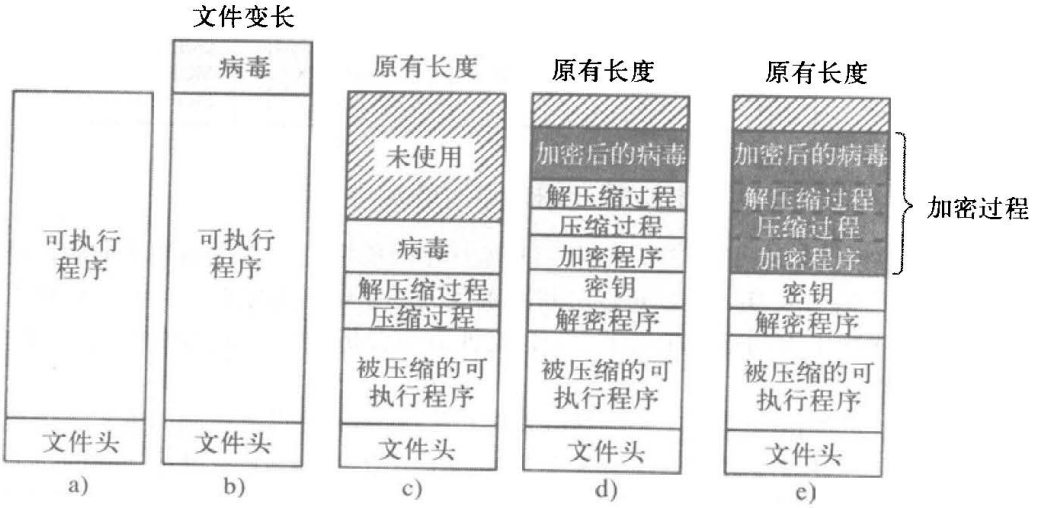

图9-32　a) 一段程序；b) 已感染的程序；c) 被压缩的已感染程序；
d) 加密的病毒；e) 带有加密压缩代码的压缩病毒

病毒还有一种逃避检测办法，那就是让自己在磁盘里呈现出的特征与病毒数据库里的特性不尽相同。要达到这一目标，方法之一是每感染一个文件就用不同的密钥将自身加密。在复制新的病毒体之前，病毒先随机产生一个32位的加密密钥，如将当前时间与内存里诸如72 008和319 992等数字进行异或。然后将病毒代码与这一密钥逐字节地异或，加密后的结果值储存在被感染文件中，如图9-32d所示。密钥也同时存放在文件中。从保密性角度来说，把密钥放进文件是不明智的。这样做的目的无非是为了对付病毒扫描，但却不能防止专家在反病毒实验室里逆向破解出病毒代码。当然，病毒在运行时必须首先对自己解密，所以在文件里也同时需要解密过程。

上述策略实际上并不完善，因为压缩、解压缩、加密和解密等过程在复制每个病毒体时都是一样的，反病毒软件可以利用这一特征来查杀病毒。把压缩、解压缩和加密过程隐藏起来较为容易：只要对它们加密并存放在病毒体里，如图9-32e所示。但是，解密过程不能被加密，它必须运行在硬件上以便将病毒体的其余部分解密，所以必须用明文格式存放。反病毒软件当然知道这些，所以它们专门搜索解密过程。

然而，Virgil喜欢笑到最后，所以他采用了下面的步骤。假设解密过程需要进行如下运算：

X=(A+B+C-4)

在普通的双地址计算机上可以运用汇编语言编写该运算，如图9-33a所示。第一个地址是源地址；第二个地址是目标地址，所以MOV A，R1是把变量*A*放入寄存器R1中。图9-33b的代码也是同样的意思，不同之处仅仅在于代码中插入了NOP（无操作）指令而降低了效率。

现在整个编码工作还未完成。为了伪装解密代码，可以用许多方法来替代NOP。例如，把0加入寄存器、自身异或、左移0位、跳转到下一个指令等，所有的都不做任何操作。所以，图9-33c在功能上与图9-33a是相同的。当病毒复制自身时，往往采用图9-33c的代码而不是图9-33a，这样在日后运行时还能工作。这种每次复制时都发生变异的病毒叫做*多形态病毒*（polymorphic virus）。

现在假设在这段代码里不再需要R5寄存器。也就是说，图9-33d与图9-33a的功能一致。最后，在许多情况下，可以交换指令而不会改变程序功能，我们用图9-33e作为另一种与图9-33a在逻辑上保持一致的代码段。这种能够交换机器码指令而不影响程序功能的代码叫做*变异引擎*（mutation engine）。较复杂的病毒在复制病毒体时，可以通过变异引擎产生不同的解密代码。变异的手段包括插入一些没用而且没