第3章 私钥加密

3.1 计算安全性

在上一章中，我们看到完美保密性有一些根本的限制。 在本章中，我们将开始研究现代密码学，引入较弱（但足够）的计算保密性概念。 然后，我们将展示如何使用这一定义来绕过先前显示的不可实现性结果，特别是，如何使用一个短密钥（例如128位长）来加密许多长消息（例如总共千兆字节）。 在此过程中，我们将研究伪随机性的基本概念，它抓住了某个事物即使不是完全随机的，也能够“看起来”完全随机的想法。 这一强大的概念是许多现代密码学的基础，并且在这一领域之外也有应用和影响。

在第2章中，我们介绍了完美保密性的概念。 虽然完美保密性是一个值得追求的目标，但它也是不必要的强大。 完美保密性要求加密消息的绝对零信息泄露，即使是面对计算能力无限的窃听者。 然而，就所有实际目的而言，如果一个加密方案只向计算能力有限的窃听者泄露了极少量信息，它仍将被认为是安全的。 例如，对于任何现实世界的应用来说，如果一个方案对投入最快超级计算机高达200年计算努力的窃听者，信息泄露的概率最多为 2^{-60}，那么它是足够用的。 考虑到攻击者计算限制并允许小概率失败的安全定义被称为**计算安全的**，以区别于本质上是信息理论的概念（如完美保密性）。 计算安全性现在是定义所有密码学目的安全的**实际**方式。 我们强调，尽管我们放弃了获得完美保密性，但这并不意味着我们放弃了严谨的数学方法。 定义和证明仍然是必不可少的，唯一的区别是我们现在考虑的是较弱（但仍然有意义）的安全定义。 相对于信息理论安全概念（在加密的情况下，这两个放松都是必要的，以便超越上一章讨论的完美保密性的限制），计算安全性纳入了两个放松：

1. **安全性仅保证对抗在可行时间内运行的有效攻击者。** 这 意味着，如果给予足够的时间（或足够的计算资源）攻击者可能能够破坏安全性。 如果我们能让破解该方案所需的资源大于任何现实攻击者可用的资源，那么就所有实际目的而言，该方案是不可破解的。
2. **攻击者可能以一个非常小的概率成功（即，安全可能失败）。** 如果我们能使这个概率足够小，我们就无需担心它。 为了获得一个有意义的理论，我们需要精确地定义上述放松。 有两种通用的方法来实现这一点：**具体方法**和**渐近方法**。 接下来将对此进行描述。

3.1.1 具体方法

计算安全性的具体方法通过显式地限定任何（随机化）攻击者在指定时间量内运行的最大成功概率，或更精确地，投入特定计算量时，来量化一个密码方案的安全性。 因此，安全性的具体定义大致采用以下形式：

一个方案是 **(t, ε)-安全的**，如果运行时间**至多为t**的任何攻击者成功破解该方案的概率**至多为ε**。

（当然，上述只是一个通用模板，并且为了使上述陈述有意义，我们需要精确定义“破解”该方案的含义。） 作为一个例子，一个方案可能保证运行时间最多为200年，使用最快可用超级计算机的任何攻击者，成功破解该方案的概率不会超过 2^{-60}。 或者，用CPU周期来衡量运行时间可能更方便，并且构建一个方案，使得使用**至多** 2^{80} 周期 的任何攻击者，成功破解该方案的概率不会超过 2^{-60}。 了解现代密码学方案中 t 的大值和 ε 的小值是很富有启发性的。

例 3.1 现代私钥加密方案通常被假定为给出**几乎最优的安全性**，其含义是：当密钥长度为 n 时（因此密钥空间大小为 2^n），运行时间为 t （例如以计算机周期衡量）的攻击者成功破解该方案的概率**至多为** $ct/2^n$，其中 c 是某个固定常数。 （这简单对应于密钥空间的暴力搜索，并且假定没有进行预处理。） 假设 $c = 1$ 以简化计算，一个长度 $n = 60$ 的密钥对使用桌面计算机的攻击者提供了足够的安全性。 实际上，在4 GHz处理器（每秒执行 4×10^9 周期）上， 2^{60} CPU周期需要 $2^{60}/(4 \times 10^9)$ 秒，或大约9年。 然而，在撰写本文时最快的超级计算机每秒大致可以执行 2×10^{16} 浮点运算，而 2^{60} 次这样的运算只需要大约1分钟。 采用 $n = 80$ 将是一个更谨慎的选择；即使是刚刚提到的那台计算机，也需要大约2年的时间来完成 2^{80} 次运算。（上述数字仅用于说明目的；在实践中 $c > 1$，并且其他几个因素——例如访问内存所需的时间以及在计算机网络上进行并行计算的可能性——会显著影响暴力破解攻击的性能。） 然而，如今推荐的密钥长度可能是 $n = 128$。 2^{80} 和 2^{128} 之间的差异是一个 2^{48} 的乘法因子。 为了感受这个数字有多大，请注意根据物理学家的估计，自大爆炸以来的秒数约为 2^{58}。 如果攻击者在一年内成功恢复加密消息的概率至多为 2^{-60}，那么发送方和接收方在同一时间段内都被闪电击中的可能性更大。 一个每百年发生一次的事件可以大致估计为在任何给定秒内以 2^{-30} 的概率发生。 以 2^{-60} 的概率在任何给定秒内发生的事件比这要小 2^{30} 倍，预计大约每1000亿年发生一次。

具体方法在实践中很重要，因为具体保证是密码方案用户最终感兴趣的。 然而，提供精确的具体保证是很困难的。 此外，在解释具体安全声明时必须小心。 例如，声称运行时间为5年的任何攻击者都不能以大于 ε 的概率破解给定方案的说法引发了以下问题：这假定了哪种类型的计算能力（例如，桌面PC、超级计算机、数百台计算机网络）？ 这个估计是否考虑了计算能力（根据摩尔定律，计算能力大约每18个月翻一番）的未来进步？ 这个估计是假定使用“现成”算法，还是针对攻击进行了优化的专用软件实现？ 此外，这样的保证很少说明运行时间为2年的攻击者的成功概率（除了它至多为 ε 的事实），也没有说明运行时间为10年的攻击者的成功概率。

3.1.2 渐近方法

正如上面部分提到的，使用具体安全方法存在一些技术和理论上的困难。 在实践中必须处理这些问题，但当具体安全不是首要关注点时，使用**渐近方法**来处理安全性会更方便；本书采用的就是这种方法。 这种植根于复杂性理论的方法引入了一个**整数值安全参数**（用 n 表示），该参数化了密码方案以及所有涉及的各方（即诚实方和攻击者）。 当诚实方初始化一个方案时（即他们生成密钥时），他们会为安全参数选择一个值 n；出于讨论的目的，可以将安全参数视为对应于密钥的长度。 假定任何攻击该方案的攻击者都知道安全参数，我们现在将攻击者的运行时间及其成功概率视为安全参数的函数，而不是具体的数字。 然后：

1. 我们将**“有效攻击者”**等同于在**多项式时间**内运行的随机化（即概率性）算法 A。 这意味着存在某个多项式 p 使得当安全参数为 n 时，攻击者运行的时间至多为 $p(n)$。 出于实际效率的考虑，我们也要求诚实方在多项式时间内运行，尽管我们强调攻击者可能比诚实方强大得多（并且运行时间长得多）。
2. 我们将**“成功的微小概率”**概念等同于小于任何 n 的逆多项式的成功概率（参见定义 3.4）。 这样的概率被称为**可忽略的（negligible）**。 令 **ppt** 代表“概率多项式时间”（probabilistic polynomial-time）。 渐近安全性的定义大致采用以下通用形式：

一个方案是**安全的**，如果任何ppt攻击者成功破解该方案的概率**至多为可忽略的**。

这种安全性的概念是**渐近的**，因为安全性依赖于方案在**足够大**的 n 值时的行为。 下面的例子清楚地说明了这一点。

例 3.2 假设我们有一个渐近安全的方案。 那么，可能会出现这样的情况：运行 n^3 分钟的攻击者可以以 $2^{40} \cdot 2^{-n}$ 的概率“破解该方案”（这是一个可忽略的 n 的函数）。 当 $n \leq 40$ 时，这意味着运行40^3 分钟（大约6周）的攻击者可以以1的概率破解该方案，因此这样的 n 值不是很有用。 即使对于 $n = 50$，运行 50^3 分钟（大约3个月）的攻击者也可以以大约 $1/1000$ 的概率破解该方案，这可能是不可接受的。 另一方面，当 $n = 500$ 时，运行200年的攻击者只能以大约 2^{-500} 的概率破解该方案。

正如上例所示，我们可以将安全参数视为一种机制，允许诚实方**“调整”**方案的安全性到某个期望的水平。 （增加安全参数也会增加运行该方案所需的时间，以及密钥的长度，因此诚实方希望将安全参数设置得尽可能小，同时能防御他们关注的攻击类别。） 将安全参数视为密钥长度，这大致对应于暴力搜索攻击所需时间随密钥长度呈指数增长的事实。 通过增加安全参数来“提高安全性”**的能力具有重要的实际意义，因为它使诚实方能够防御计算能力（随摩尔定律）的增加。 下面的例子说明了这在实践中可能如何发挥作用。

例 3.3 让我们看看更快的计算机的可用性可能对实践中的安全性产生什么影响。 假设我们有一

个密码方案，其中诚实方运行 $10^6 \cdot n^2$ 周期，而攻击者运行 $10^8 \cdot n^4$ 周期可以以至多 $2^{-n/2}$ 的概率“破解”该方案。 （这些数字旨在使计算更容易，并且不代表任何现有密码方案。） 假设所有各方都使用2 GHz计算机，并且诚实方设置 $n = 80$。 那么诚实方运行 $10^6 \cdot 6400$ 周期，即3.2秒，而攻击者运行 $10^8 \cdot (80)^4$ 周期，即大约3周，可以以仅 2^{-40} 的概率破解该方案。 假设8 GHz计算机变得可用，并且所有各方都升级。 诚实方可以将 n 增加到160（这需要生成一个新的密钥）并保持3.2秒的运行时间（即，以 $8 \cdot 10^9$ 周期/秒的速度运行 $10^6 \cdot 160^2$ 周期）。 相比之下，攻击者现在必须运行超过800万秒，即超过13周，才能实现 2^{-80} 的成功概率。 更快的计算机的影响是使攻击者的工作更难。

即使在使用渐近方法时，也必须记住，最终，当密码系统在实践中部署时，将需要一个具体的安全保证。 （毕竟，必须决定 n 的某个值。） 然而，正如上面的例子所示，通常情况下，渐近安全声明可以转化为任何期望 n 值的具体安全界限。

渐近方法详解

我们现在更正式地讨论“多项式时间算法”和“可忽略的成功概率”的概念。

高效算法。 我们将算法定义为高效的，如果它在多项式时间内运行。 一个算法 A 在多项式时间内运行，如果存在一个多项式 p 使得，对于每一个输入 $x \in \{0,1\}^*$， $A(x)$ 的计算在至多 $p(|x|)$ 步内终止。 （这里 $|x|$ 表示字符串 x 的长度。） 正如前面提到的，我们只对运行时间在安全参数 n 中是多项式的攻击者感兴趣。 由于我们根据算法输入的长度来衡量算法的运行时间，我们有时会提供以一元形式写入的安全参数（即，作为 1^n，或由 n 个一组成的一个字符串）作为输入。 各方（或更精确地说，他们运行的算法）可能会接受除安全参数以外的其他输入——例如，一个要加密的消息——我们允许他们的运行时间在他们的（总）输入长度中是多项式的。 默认情况下，我们允许所有算法都是概率性的（或随机化的）。 任何这样的算法可以在其执行的每一步“抛硬币”；这是一种隐喻的说法，即算法可以访问一个无偏的随机比特。 等价地，我们可以将一个随机化算法视为一个除了输入之外，还被赋予一个**足够长**的均匀分布的随机带的算法，它可以在其整个执行过程中根据需要使用这些比特。 我们默认考虑随机化算法有两个原因。 首先，**随机性对密码学至关重要**（例如，为了选择随机密钥等等），因此诚实方必须是概率性的；鉴于此，允许攻击者也是概率性的就很自然。 其次，随机化是实用的，并且——据我们所知——赋予了攻击者额外的权力。 由于我们的目标是模拟所有现实攻击，我们倾向于更自由的有效计算定义。

可忽略的成功概率。 可忽略函数是一个**渐近地小于任何逆多项式函数**的函数。 形式上：

定义 3.4 一个从自然数到非负实数的函数 f 是 **可忽略的 (negligible)**，如果对于每一个正多项式 p 来说，存在一个 N 使得对于所有整数 $n > N$，都有 $f(n) < 1/p(n)$ 成立。

简而言之，上述内容也可以表述为：对于每一个多项式 p 和所有足够大值的 n，都有 $f(n) < 1/p(n)$ 成立。 上述内容的一个等价表述是：要求对于所有常数 c 来说，存在一个 N 使得对于所有 $n > N$，都有 $f(n) < n^{-c}$ 成立。 我们通常用 $negl$ 来表示一个任意的可忽略函数。

例 3.5 函数 2^{-n}, $2^{-\sqrt{n}}$ 和 $n^{-\log n}$ 都是可忽略的。 然而，它们以非常不同的速率趋近于零。例如，我们可以看看每个函数小于 $1/n^5$ 的最小 n 值：

1. 求解 $2^{-n} < n^{-5}$ 得到 $n > 5\log n$。 满足此条件的最小整数值为 $n = 23$。
2. 求解 $2^{-\sqrt{n}} < n^{-5}$ 得到 $\sqrt{n} > 25\log_2 n$。 满足此条件的最小整数值约为 $n \approx 3500$。
3. 求解 $n^{-\log n} < n^{-5}$ 得到 $\log n > 5$。 满足此条件的最小整数值为 $n = 33$。 从上面您可能会产生 $n^{-\log n}$ 比 $2^{-\sqrt{n}}$ 更快趋近于零的印象。 然而，这是不正确的；对于所有 $n > 65536$ 来说，有 $2^{-\sqrt{n}} < n^{-\log n}$ 成立。 尽管如此，这确实表明对于 n 为数百或数千的值，攻击成功的概率为 $n^{-\log n}$ 优于攻击成功的概率为 $2^{-\sqrt{n}}$。

使用可忽略的成功概率的一个技术优势是它们服从某些**闭包属性**。 下面的命题是一个简单的练习。

命题 3.6 令 $negl_1$ 和 $negl_2$ 是可忽略函数。 则：

1. 由 $negl_3(n) = negl_1(n) + negl_2(n)$ 定义的函数 $negl_3$ 是可忽略的。
2. 对于任何正多项式 p，由 $negl_4(n) = p(n) \cdot negl_1(n)$ 定义的函数 $negl_4$ 是可忽略的。

上述命题的第二部分意味着，如果某个事件在某个实验中以可忽略的概率发生，那么即使该实验重复**多项式**次，该事件发生的概率仍然是可忽略的。 （这依赖于并集界；参见命题A.7。） 例如， n 次公平抛硬币全部正面朝上的概率是可忽略的。 这意味着，即使我们将抛 n 次硬币的实验重复**多项式**次，任何这些实验导致 n 次正面朝上的概率仍然是可忽略的。

上述命题的第二部分的一个推论是，如果一个函数 g 不是可忽略的，那么对于任何正多项式 p 来说，由 $f(n) \stackrel{\text{def}}{=} g(n)/p(n)$ 定义的函数 f 也不是可忽略的。

渐近安全性：总结

任何安全定义都包含两个部分：对什么被认为是方案的**“破解”**的定义，以及对攻击者能力的规格说明**。 攻击者的能力可以涉及许多问题（例如，在加密的情况下，我们是假定密文-only攻击还是选择-明文攻击）。 然而，当涉及到攻击者的计算能力时，我们从现在起将攻击者建模为**高效的**，因此只考虑可以在**概率多项式时间**内实现的攻击策略。 定义也总是以这样的方式制定，即以**可忽略的概率**发生的破解不被认为是显著的。 因此，任何安全定义的通用框架如下：

一个方案是**安全的**，如果对于每一个**概率多项式时间**攻击者 A 来说，它执行某种正式指定类型

的攻击时， A 成功的概率（成功也正式指定）是**可忽略的**。

这样的定义是**渐近的**，因为它可能对于小值 n 来说，攻击者可以以高概率成功。 为了更详细地了解这一点，我们在上述陈述中扩展了**“可忽略的”**一词：

一个方案是**安全的**，如果对于每一个 **ppt** 攻击者 A 来说，它执行某种正式指定类型的攻击时，对于每一个**正多项式** p 来说，存在一个整数 N 使得当 $n > N$ 时， A 成功的概率小于 $1/p(n)$。

请注意，对于 $n \leq N$ 的值，不作任何保证。

关于定义渐近安全性所做的选择

在定义渐近安全性的通用概念时，我们做出了两个选择：我们将**高效攻击策略**等同于**概率多项式时间算法**的类别，并将**小的成功机会**等同于**可忽略的概率**。 这两个选择——在某种程度上——是任意的，一个人可以通过定义，例如，将**高效策略**定义为在二次时间内运行的那些，或将**小的成功概率**定义为被 2^{-n} 限定的那些，来建立一个完全合理的理论。 尽管如此，我们还是简要地证明我们所做的选择（它们是标准的）。

熟悉复杂性理论或算法的人将认识到，将**高效计算**等同于**（概率）多项式时间算法**的想法并非密码学所独有。 使用（概率）多项式时间作为我们效率度量的一个优点是，这使我们不必精确地指定我们的计算模型，因为扩展的丘奇-图灵论题指出，所有“合理的”计算模型在多项式上是等价的。 因此，我们不需要指定我们是使用图灵机、布尔电路还是随机存取机；我们可以用高级伪代码呈现算法，并且相信如果我们的分析表明这些算法在多项式时间内运行，那么任何合理的实现也会如此。

（概率）多项式时间算法的另一个优点是它们满足合意的**闭包属性**：特别是，一个对多项式时间子程序进行多项式次调用（并且只进行额外的多项式计算）的算法本身也将在多项式时间内运行。

可忽略概率的最重要的特性是我们在命题3.6(2)中已经看到的闭包属性：**任何多项式乘以一个可忽略函数仍然是可忽略的**。 这特别意味着，如果一个算法对某个“失败”概率可忽略的子程序进行了多项式次调用，那么任何一次调用失败的概率仍然是可忽略的。

放松的必要性

计算安全性引入了对完美保密性的两种放松：首先，安全性仅保证对抗**有效攻击者**；其次，允许有**小的成功概率**。 这两种放松对于实现实用的加密方案都是至关重要的，特别是为了绕过完美保密性的负面结果。 我们非正式地讨论一下为什么会是这种情况。 假设我们有一个加密方案，

其中密钥空间 K 的大小远小于消息空间 M 的大小。（如前一章所示，这意味着该方案不可能是完美保密的。） 无论加密方案如何构建，都会出现两种攻击： • **给定一个密文 c，攻击者可以使用所有密钥 $k \in K$ 来解密 c。** 这会给出一个密文 c 可能对应的所有消息的列表。 由于这个列表不能包含 M 的所有元素（因为 $|K| < |M|$），这次攻击会泄露关于被加密消息的一些信息。 此外，假设攻击者执行了**已知明文攻击**，并得知密文 $c_1, \ldots, c_l$ 分别对应于消息 $m_1, \ldots, m_l$。 攻击者可以再次尝试使用所有可能的密钥解密这些密文中的每一个，直到找到一个密钥 k 使得对所有 i 都有 $Dec_k(c_i) = m_i$。 随后，给定一个未知消息 m 的密文 c，几乎可以肯定 $Dec_k(c) = m$。 像上面那样的**穷举搜索攻击**允许攻击者在与 $|K|$ 成线性关系的时间内以基本上为1的概率成功。 • **再次考虑攻击者得知密文 $c_1, \ldots, c_l$ 对应于消息 $m_1, \ldots, m_l$ 的情况。** 攻击者可以猜测一个均匀密钥 $k \in K$，并检查是否对所有 i 都有 $Dec_k(c_i) = m_i$。 如果是这样，那么如上所述，攻击者可以使用 k 来解密随后由诚实方加密的任何内容。 在这种情况下，攻击者在基本上**恒定时间**内运行，并以非零（尽管非常小）的概率 $1/|K|$ 成功。

由此可见，如果我们要使用单个短密钥加密许多消息，只有当我们限制攻击者的运行时间（以便攻击者没有足够的时间执行暴力搜索）并且愿意允许**非常小的**成功概率（以便排除第二种“攻击”）时，才能实现安全性。

3.2 定义计算安全加密

有了上一节的背景，我们准备提出私钥加密的计算安全定义。 首先，我们重新定义私钥加密的语法；这与第2章中引入的语法基本相同，只是我们现在显式地考虑安全参数 n。 我们还允许解密算法在遇到无效密文时输出错误消息。 最后，默认情况下，我们让消息空间为 $\{0,1\}^*$（所有（有限长度）二进制字符串的集合）。

定义 3.7 一个私钥加密方案是一个概率多项式时间算法的元组 (Gen, Enc, Dec)，满足以下条件：

1. **密钥生成算法** Gen 以 1^n（即以一元形式写入的安全参数）为输入，并输出一个密钥 k；我们记为 $k \leftarrow Gen(1^n)$（强调 Gen 是一个随机化算法）。 我们假设由 $Gen(1^n)$ 输出的任何密钥 k 都满足 $|k| \geq n$，这不会失去一般性。
2. **加密算法** Enc 以密钥 k 和明文 $m \in \{0,1\}^*$ 为输入，并输出一个密文 c。 由于 Enc 可能是随机化的，我们将其记为 $c \leftarrow Enc_k(m)$。
3. **解密算法** Dec 以密钥 k 和密文 c 为输入，并输出一个消息 m 或一个特殊符号 $\perp$ 表示失败。 我们假设 Dec 是确定性的，因此我们将其记为 $m := Dec_k(c)$（这里假设 Dec 不返回错误）。 我们用符号 $\perp$ 表示一个通用错误。 **要求** 对于每一个 n，由 $Gen(1^n)$ 输出的每一个密钥 k，以及每一个 $m \in \{0,1\}^*$，都满足 $Dec_k(Enc_k(m)) = m$。

如果 (Gen, Enc, Dec) 满足对于由 $Gen(1^n)$ 输出的 k，算法 Enc_k 仅对长度为 $l(n)$ 的消息 $m \in \{0,1\}^{l(n)}$ 有定义，则我们称 (Gen, Enc, Dec) 是一个**定长私钥加密方案**，用于加密长度为 $l(n)$ 的消息。

几乎总是， $Gen(1^n)$ 只是输出一个均匀的 n 比特字符串作为密钥。 在这种情况下，我们将省略 Gen，而只用一对算法 (Enc, Dec) 来定义私钥加密方案。

上述定义考虑了**无状态方案**，其中 Enc（和 Dec）的每次调用都与所有先前的调用无关。 在本章后面，我们将偶尔讨论**有状态方案**，其中发送方（和可能还有接收方）需要在调用之间维护状态。 除非明确另有说明，我们所有的结果都假定无状态加密/解密。

3.2.1 基本安全定义

我们首先介绍私钥加密最基本的安全概念：在**密文-only攻击**下（即攻击者仅观察到**单个密文**），或者等价地，当给定密钥仅用于加密**单个消息**时的安全性。 我们将在本章后面考虑更强的安全定义。

定义动机。 正如我们已经讨论过的，任何安全定义都包含两个不同的组成部分：**威胁模型**（即，攻击者假定能力的规格说明）和**安全目标**（通常通过描述什么构成了方案的“破解”来指定）。 我们从考虑最简单的威胁模型开始我们的定义处理，即我们有一个**窃听攻击者**，它观察到一个**单个消息**的加密。 这与上一章中考虑的威胁模型完全相同，除了一个例外，即我们现在只对**计算有界**的攻击者感兴趣，因此他们的运行时间限制在多项式时间内。

尽管我们对攻击者的能力做了两个假设（即，它只是窃听，并且它在多项式时间内运行），我们对攻击者尝试解密它观察到的密文的**策略**不作任何假设。 这对于获得有意义的安全概念至关重要；该定义确保了对**任何**计算有界攻击者的保护，无论它使用什么算法。

正确定义加密的安全目标并非易事，但我们已经在第1.4.1节和上一章中详细讨论了这个问题。因此，我们只回顾一下定义背后的思想是：攻击者应该无法从密文中了解关于明文的**任何部分信息**。 **语义安全性**（参见第3.2.2节）精确地形式化了这一概念，并且是第一个被提出的计算安全加密的定义。 语义安全性很复杂且难以处理。 幸运的是，存在一个等价的定义，称为**不可区分性**，它简单得多。

不可区分性的定义是根据作为定义2.5给出的**完美保密性**的替代定义来模式化的。 （这进一步激励了不可区分性定义是一个好的定义。） 回想一下，定义2.5考虑了一个实验 $PrivK^{\text{eav}}_{A,\Pi}$，其中攻击者 A 输出两个消息 m_0 和 m_1，然后被赋予其中一个消息使用均匀密钥的加密。 该定义表明，如果没有任何攻击者 A 能够以任何不同于 $1/2$ 的概率确定哪个消息被加密，那么方案 Π 是安全的，而 $1/2$ 是 A 只是进行随机猜测时正确的概率。

在这里，我们保持实验 $PrivK^{\text{eav}}_{A,\Pi}$ 几乎完全相同（除了下面讨论的一些技术差异），但在定义本身中引入了两个关键修改：

1. **我们现在只考虑在多项式时间内运行的攻击者**，而定义2.5甚至考虑了计算时间无限的攻击者。
2. **我们现在承认攻击者可能以略微大于 $1/2$ 的概率确定被加密的消息。**

正如上一节广泛讨论的那样，上述放松构成了**计算安全性**的核心要素。

至于其他差异，最突出的是我们现在用**安全参数** n 来参数化实验。 然后，我们测量攻击者 A 的运行时间及其成功概率作为 n 的函数。 我们写 $PrivK^{\text{eav}}_{A,\Pi}(n)$ 来表示以安全参数 n 运行的实验，并写

$$\Pr\left[PrivK^{\text{eav}}_{A,\Pi}(n) = 1\right] \tag{3.1}$$

来表示实验 $PrivK^{\text{eav}}_{A,\Pi}(n)$ 输出为1的概率。 请注意，给定 A, Π，公式(3.1)是 n 的一个函数。

实验 $PrivK^{\text{eav}}_{A,\Pi}$ 中的第二个区别是，我们现在明确要求攻击者输出两个**等长**的消息 m_0, m_1。（在定义2.5中，如果消息空间 M 仅包含固定长度的消息，则此要求是隐含的，正如一次性密码方案的情况一样。） 这意味着，默认情况下，我们不要求安全的加密方案隐藏明文的长度。我们将在本节末尾重新讨论这一点；另请参见练习3.2和3.3。

存在窃听者的不可区分性。 我们现在给出形式化的定义，从上述实验开始。 该实验是为任何私钥加密方案 $\Pi = (Gen, Enc, Dec)$、任何攻击者 A 和安全参数的任何值 n 定义的：

窃听不可区分性实验 $PrivK^{\text{eav}}_{A,\Pi}(n)$:

1. 攻击者 A 获得输入 1^n，并输出一对**等长**的消息 $m_0, m_1 \in M$。
2. 使用 $Gen(1^n)$ 生成一个密钥 k，并选择一个均匀比特 $b \in \{0,1\}$。 计算密文 $c \leftarrow Enc_k(m_b)$ 并将其提供给 A。 我们称 c 为**挑战密文**。
3. A 输出一个比特 b'。
4. 实验的输出定义为如果 $b' = b$，则为1，否则为0。 如果 $PrivK^{\text{eav}}_{A,\Pi}(n) = 1$，我们说 A **成功**。

m_0 和 m_1 的长度没有限制，只要它们相等即可。 （当然，如果 A 在多项式时间内运行，那么 m_0 和 m_1 的长度在 n 中是多项式的。） 如果 Π 是一个用于加密长度为 $l(n)$ 的消息的定长方案，则通过要求 $m_0, m_1 \in \{0,1\}^{l(n)}$ 来修改上述实验。

攻击者只能窃听的事实隐含在她的输入仅限于单个密文这一事实中，并且攻击者没有与发送方或接收方进行任何进一步的交互。 （正如我们稍后将看到的，允许额外的交互会使攻击者明显更强大。）

不可区分性的定义表明，如果没有任何 **ppt** 攻击者 A 能够以明显优于随机猜测（正确概率为 $1/2$）的概率猜出哪个消息被加密，那么一个加密方案是安全的：

定义 3.8 一个私钥加密方案 $\Pi = (Gen, Enc, Dec)$ 具有**在窃听者存在下的不可区分加密性**，或简称 **EAV-安全的**，如果对于所有概率多项式时间攻击者 A 来说，存在一个**可忽略函数** $negl$ 使得，对于所有 n 来说，

$$\Pr\left[PrivK^{\text{eav}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + negl(n),$$

其中概率是根据 A 使用的随机性以及实验中使用的随机性（用于选择密钥和比特 b，以及 Enc 使用的任何随机性）来计算的。

注意： 除非另有说明，当我们写 "$f(n) \leq g(n)$" 时，我们的意思是该不等式对所有 n 都成立。

显然，定义3.8**弱于**等价于完美保密性的定义2.5。 因此，任何完美保密加密方案都具有在窃听者存在下的不可区分加密性。 我们的目标是展示存在满足上述条件的加密方案，其中密钥比消息短。 也就是说，我们将展示满足定义3.8但不能满足定义2.5的方案。

一个等价的表述。 定义3.8要求没有 **ppt** 攻击者可以以明显优于 $1/2$ 的概率确定哪个消息被加密。 一个等价的表述是，当 A 看到 m_0 的加密还是 m_1 的加密时，每个 **ppt** 攻击者 A 的行为是**相同的**。 由于 A 输出一个比特，**"行为相同"**意味着它在每种情况下都以几乎相同的概率输出1。 为了形式化这一点，定义 $PrivK^{\text{eav}}_{A,\Pi}(n, b)$ 如上所述，但使用固定的比特 b（而不是随机选择）。 令 $out_A(PrivK^{\text{eav}}_{A,\Pi}(n, b))$ 表示实验中 A 的输出比特 b'。 下面的命题本质上表明，没有 A 能够判断它是在实验 $PrivK^{\text{eav}}_{A,\Pi}(n, 0)$ 中运行还是在实验 $PrivK^{\text{eav}}_{A,\Pi}(n, 1)$ 中运行。

定义 3.9 一个私钥加密方案 $\Pi = (Gen, Enc, Dec)$ 具有**在窃听者存在下的不可区分加密性**，如果对于所有 **ppt** 攻击者 A 来说，存在一个**可忽略函数** $negl$ 使得：

$$\left|\Pr[out_A(PrivK^{\text{eav}}_{A,\Pi}(n, 0)) = 1] - \Pr[out_A(PrivK^{\text{eav}}_{A,\Pi}(n, 1)) = 1]\right| \leq negl(n).$$

这个事实等价于定义3.8，留作练习。

加密与明文长度

安全加密的默认概念不要求加密方案隐藏明文长度，事实上，所有常用的加密方案都会泄露明文长度（或其近似值）。 主要原因是**不可能**在隐藏所有关于明文长度的信息的同时支持任意长度的消息（参见练习3.2）。 在许多情况下，明文长度已经是公开的或不敏感的，这无关紧要。然而，情况并非总是如此，有时泄露明文长度是有问题的。 例如： • **简单数字/文本数据：** 假设使用的加密方案精确地泄露了明文长度。 那么加密的薪水信息将泄露某人是5位数还是6位数的薪水。 类似地，加密“是”/“否”回答将精确地泄露答案。 • **自动建议：** 网站通常包含“自动完成”或“自动建议”功能，通过该功能，网络服务器根据用户已输入的部分信息建议潜在的单词或短语列表。 这个列表的**大小**可以泄露关于用户目前已输入字母的信息。 （例如，为“th”返回的自动完成数量远大于为“zo”返回的数量。） • **数据库搜索：** 考虑一个用户查询数据库以查找与某个搜索词匹配的所有记录。 返回的记录**数量**可以泄露关于用户正在搜索的内容的大量信息。如果用户正在搜索医疗信息并且查询泄露了关于用户患有某种疾病的信息，这可能会特别有害。 • **压缩数据：** 如果在加密之前对明文进行了压缩，那么即使只加密定长数据，也可能会泄露关于明文的信息。 （这样的加密方案因此不满足定义3.8。） 例如，一个短的压缩明文将表明原始（未压缩）明文具有大量冗余。 如果攻击者能够控制被加密的一部分内容，那么这种漏洞可以使攻击者了解关于明文的额外信息；已经证明，可以使用完全是这种类型的攻击（CRIME攻击）来对抗加密的HTTP流量以泄露秘密会话cookie。

在使用加密时，应该确定泄露明文长度是否是一个问题，如果是，则应采取措施，通过在加密前将所有消息填充到某个预定长度来减轻或防止这种泄露。

3.2.2 *语义安全性

我们对安全加密的定义是基于这样一个观点的：攻击者应该不可能从密文中获取关于明文的任何部分信息。然而，不可区分性（indistinguishability）的定义看起来大不相同。正如我们提到的，定义 3.8 等价于一个称为*语义安全*（semantic security）的定义，它正式地表达了“无法获取部分信息”这一概念。我们通过讨论两种较弱的概念并证明它们都蕴含于不可区分性来建立这个定义。

我们首先证明不可区分性意味着密文不会泄露关于明文单个比特的任何信息。形式上，假设加密方案 $(\mathrm{Enc}, \mathrm{Dec})$ 是 EAV-secure 的（回想一下，当 Gen 被省略时，密钥是一个均匀的 n 比特字符串），并且 $m \in \{0,1\}^{\ell}$ 是均匀分布的。然后我们证明，对于任何索引 i，攻击者要从 $\mathrm{Enc}_k(m)$ 中猜测 m_i（在本节中，m_i 表示 m 的第 i 个比特）的概率要远优于 $1/2$ 是不可行的。

定理 3.10 设 $\Pi = (\mathrm{Enc}, \mathrm{Dec})$ 是一个用于长度为 ℓ 的消息的固定长度私钥加密方案，并且在窃听者存在的情况下具有不可区分的加密。那么对于所有概率多项式时间（ppt）攻击者 $\mathcal{A}$ 和任何 $i \in \{1, \ldots, \ell\}$，都存在一个可忽略函数 negl，使得

$$\Pr[\mathcal{A}(1^n, \mathrm{Enc}_k(m)) = m_i] \leq 1/2 + \mathrm{negl}(n),$$

其中概率是针对均匀分布的 $m \in \{0,1\}^\ell$ 和 $k \in \{0,1\}^n$、$\mathcal{A}$ 的随机性以及 Enc 的随机性来计算的。

证明 这个定理背后的思想是，如果可以从 $\mathrm{Enc}_k(m)$ 中猜测 m 的第 i 个比特，那么也就可以区分 i 个比特不同的消息 m_0 和 m_1 的加密。我们通过归约证明来形式化这一点，其中我们展示了如何使用任何高效攻击者 $\mathcal{A}$ 来构造一个高效攻击者 $\mathcal{A}'$：如果 $\mathcal{A}$ 违反了该定理对 Π 的安全概念，那么 $\mathcal{A}'$ 将违反 Π 的不可区分性定义。（参见 3.3.2 节。）由于 Π 具有不可区分的加密，因此它在定理的意义上也必然是安全的。

固定一个任意的 ppt 攻击者 $\mathcal{A}$ 和 $i \in \{1, \ldots, \ell\}$。设 $\mathcal{I}_0 \subset \{0,1\}^\ell$ 是所有第 i 个比特为 0 的字符串的集合，设 $\mathcal{I}_1 \subset \{0,1\}^\ell$ 是所有第 i 个比特为 1 的字符串的集合。我们有

$$\Pr[\mathcal{A}(1^n, \mathrm{Enc}_k(m)) = m_i] = 1/2 \cdot \Pr_{m_0 \leftarrow \mathcal{I}_0}[\mathcal{A}(1^n, \mathrm{Enc}_k(m_0)) = 0] + 1/2 \cdot \Pr_{m_1 \leftarrow \mathcal{I}_1}[\mathcal{A}(1^n, \mathrm{Enc}_k(m_1$$

构造如下的窃听攻击者 $\mathcal{A}'$：

攻击者 $\mathcal{A}'$：

1. 均匀选择 $m_0 \in \mathcal{I}_0$ 和 $m_1 \in \mathcal{I}_1$。输出 m_0, m_1。
2. 在观察到密文 c 后，调用 $\mathcal{A}(1^n, c)$。如果 $\mathcal{A}$ 输出 0，则输出 $b' = 0$；否则，输出 $b' = 1$。

$\mathcal{A}'$ 运行时间为多项式时间，因为 $\mathcal{A}$ 是多项式时间算法。

根据实验 $\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A}',\Pi}(n)$ 的定义，我们有 $\mathcal{A}'$ 成功的条件当且仅当 $\mathcal{A}$ 在接收到 $\mathrm{Enc}_k(m_b)$ 时输出 b。因此

$$\Pr[\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A}',\Pi}(n) = 1] = \Pr[\mathcal{A}(1^n, \mathrm{Enc}_k(m_b)) = b] = 1/2 \cdot \Pr_{m_0 \leftarrow \mathcal{I}_0}[\mathcal{A}(1^n, \mathrm{Enc}_k(m_0)) = 0] + 1/$$

根据 Π 具有在窃听者存在的情况下不可区分的加密这一假设，存在一个可忽略函数 negl，使得 $\Pr[\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A}',\Pi}(n) = 1] \leq 1/2 + \mathrm{negl}(n)$。我们得出结论

$$\Pr[\mathcal{A}(1^n, \mathrm{Enc}_k(m)) = m_i] \leq 1/2 + \mathrm{negl}(n),$$

完成证明。

接下来我们大致主张，不可区分性意味着在给定密文的情况下，无论消息的分布如何，任何 ppt 攻击者都无法获取关于明文的任何函数信息。这旨在捕捉这样一个观点，即所得密文不会泄露关

于底层明文的任何信息。然而，这个要求形式化起来并不简单。原因在于，即使对于上面考虑的情况，如果 m 是从所有第 i 个比特为 0 的字符串集合中均匀选择的（而不是从 $\{0,1\}^\ell$ 中均匀选择），那么计算 m 的第 i 个比特也是很容易的。因此，我们真正想说的是，如果存在任何攻击者在给定 $\mathrm{Enc}_k(m)$ 的情况下以某种概率正确计算了 $f(m)$，那么也存在一个攻击者可以在完全不给定密文的情况下（仅知道 m 的分布）以相同的概率正确计算 $f(m)$。在下文中，我们将重点关注 m 从某个集合 $\mathcal{S} \subseteq \{0,1\}^\ell$ 中均匀选择的情况。

定理 3.11 设 $(\mathrm{Enc}, \mathrm{Dec})$ 是一个用于长度为 ℓ 的消息的固定长度私钥加密方案，并且在窃听者存在的情况下具有不可区分的加密。那么对于任何 ppt 算法 $\mathcal{A}$，都存在一个 ppt 算法 $\mathcal{A}'$，使得对于任何集合 $\mathcal{S} \subseteq \{0,1\}^\ell$ 和任何函数 $f : \{0,1\}^\ell \to \{0,1\}$，都存在一个可忽略函数 negl，使得

$$|\Pr[\mathcal{A}(1^n, \mathrm{Enc}_k(m)) = f(m)] - \Pr[\mathcal{A}'(1^n) = f(m)]| \leq \mathrm{negl}(n),$$

其中第一个概率是针对 $k \in \{0,1\}^n$ 的均匀选择、 $m \in \mathcal{S}$ 的均匀选择、$\mathcal{A}$ 的随机性以及 Enc 的随机性来计算的，而第二个概率是针对 $m \in \mathcal{S}$ 的均匀选择和 $\mathcal{A}'$ 的随机性来计算的。

证明概要 $(\mathrm{Enc}, \mathrm{Dec})$ 是 EAV-安全的意味着，对于任何 $\mathcal{S} \subseteq \{0,1\}^\ell$，没有 ppt 攻击者能够区分 $\mathrm{Enc}_k(m)$（对于均匀的 $m \in \mathcal{S}$）和 $\mathrm{Enc}_k(1^\ell)$。现在考虑 $\mathcal{A}$ 在给定 $\mathrm{Enc}_k(m)$ 的情况下成功计算 $f(m)$ 的概率。我们主张 $\mathcal{A}$ 应该在给定 $\mathrm{Enc}_k(1^\ell)$ 的情况下以几乎相同的概率成功计算 $f(m)$；否则，$\mathcal{A}$ 就可以被用来区分 $\mathrm{Enc}_k(m)$ 和 $\mathrm{Enc}_k(1^\ell)$。区分者很容易构造：选择均匀的 $m \in \mathcal{S}$，并输出 $m_0 = m, m_1 = 1^\ell$。当给定密文 c 是 m_0 或 m_1 的加密时，调用 $\mathcal{A}(1^n, c)$，当且仅当 $\mathcal{A}$ 输出 $f(m)$ 时输出 0。如果 $\mathcal{A}$ 在给定 m 的加密时输出 $f(m)$ 的概率与在给定 1^ℓ 的加密时输出 $f(m)$ 的概率存在显著差异，那么所描述的区分者将违反定义 3.9。

上面所述表明，如下的算法 $\mathcal{A}'$（它没有接收到 $c = \mathrm{Enc}_k(m)$）却能以与 $\mathcal{A}$ 几乎相同的效果计算 $f(m)$：$\mathcal{A}'(1^n)$ 选择一个均匀的密钥 $k \in \{0,1\}^n$，在 $c \leftarrow \mathrm{Enc}_k(1^\ell)$ 上调用 $\mathcal{A}$，并输出 $\mathcal{A}$ 的任何结果。根据上面所述，当 $\mathcal{A}'$ 作为子例程运行时，$\mathcal{A}$ 输出 $f(m)$ 的概率与它接收到 $\mathrm{Enc}_k(m)$ 时的概率几乎相同。因此，$\mathcal{A}'$ 满足了该主张所要求的性质。

语义安全。 语义安全的完整定义比定理 3.11 中考虑的性质提供了多得多的保证。该定义允许明文的长度取决于安全参数，并允许对明文进行基本上任意的分布。（实际上，我们只允许*高效可采样的分布*。这意味着存在一个概率多项式时间算法 Samp，使得 $\mathrm{Samp}(1^n)$ 根据该分布输出消息。）该定义还考虑了关于明文的任意“外部”信息 $h(m)$，这些信息可能通过其他方式泄露给攻击者（例如，因为相同的消息 m 被用于其他目的）。

定义 3.12 私钥加密方案 $(\mathrm{Enc}, \mathrm{Dec})$ 在窃听者存在的情况下是*语义安全*的，如果对于所有 ppt 算法 $\mathcal{A}$，都存在一个 ppt 算法 $\mathcal{A}'$，使得对于任何 ppt 算法 Samp 和任何多项式时间可

计算函数 f 和 h，以下表达式是可忽略的：

$$|\Pr[\mathcal{A}(1^n, \mathrm{Enc}_k(m), h(m)) = f(m)] - \Pr[\mathcal{A}'(1^n, |m|, h(m)) = f(m)]| \leq \mathrm{negl}(n),$$

其中第一个概率是针对 $k \in \{0,1\}^n$ 的均匀选择、$\mathrm{Samp}(1^n)$ 输出的 m、$\mathcal{A}$ 的随机性以及 Enc 的随机性来计算的，而第二个概率是针对 $\mathrm{Samp}(1^n)$ 输出的 m 和 $\mathcal{A}'$ 的随机性来计算的。

攻击者 $\mathcal{A}$ 被给予密文 $\mathrm{Enc}_k(m)$ 以及外部信息 $h(m)$，并试图猜测 $f(m)$ 的值。算法 $\mathcal{A}'$ 也试图猜测 $f(m)$ 的值，但只被给予 $h(m)$ 和 m 的长度。安全要求指出，$\mathcal{A}$ 正确猜测 $f(m)$ 的概率与 $\mathcal{A}'$ 的概率大致相同。那么从直觉上讲，密文 $\mathrm{Enc}_k(m)$ 没有泄露关于 $f(m)$ 值的任何额外信息。

定义 3.12 构成了加密方案应提供的安全保证的非常强大且令人信服的表述。然而，与不可区分性（定义 3.8）的定义相比，它的使用更为困难。幸运的是，这两个定义是等价的：

定理 3.13 私钥加密方案在窃听者存在的情况下具有不可区分的加密，当且仅当它在窃听者存在的情况下是语义安全的。

展望未来，在本章以及第 11 章中介绍的所有定义中，语义安全和不可区分性之间都存在类似的等价性。因此，我们可以使用不可区分性作为我们的工作定义，同时确信所获得的保证是语义安全所要求的。

3.3 构建安全加密方案

在定义了安全加密方案的含义之后，读者可能期望我们立即转向构建安全加密方案。 然而，在此之前，我们需要引入**伪随机生成器 (PRGs)** 和**流密码**的概念，它们是私钥加密的重要构建块。 这些反过来又会引出**伪随机性**的讨论，它在整个密码学，特别是私钥加密中起着基础性的作用。

3.3.1 伪随机生成器和流密码

伪随机生成器 G 是一个高效的、确定性算法，用于将一个**短的、均匀的**字符串（称为**种子**）转换成一个**更长**的、“看起来均匀的”（或“伪随机的”）输出字符串。 换句话说，伪随机生成器使用少量**真随机性**来生成大量**伪随机性**。 当需要大量随机（-看起来）比特时，这是很有用的，因为生成真随机比特是困难且缓慢的。 （参见第2章开头的讨论。） 事实上，伪随机生成器自至少1940年代以来就被研究，当时它们被提议用于运行统计模拟。 在那种情况下，研究人员提出了各种统计测试，一个伪随机生成器应该通过这些测试才能被认为是“好的”。 作为一个简单的

例子，伪随机生成器输出的第一个比特应该以**非常接近** $1/2$ 的概率等于1（其中概率是根据种子的均匀选择来计算的），因为一个均匀字符串的第一个比特以**恰好** $1/2$ 的概率等于1。 事实上，输出比特的任何固定子集的**奇偶校验**也应该以**非常接近** $1/2$ 的概率为1。 也可以考虑更复杂的统计测试。

这种确定候选伪随机生成器质量的历史方法是**特设的**，并且不清楚通过某些统计测试集是否足以保证在某些应用中使用候选伪随机生成器的可靠性。 （特别是，可能存在另一个统计测试，它可以成功地区分生成器输出与真随机比特的差异。） 当将伪随机生成器用于密码学应用时，历史方法甚至更成问题；在这种情况下，如果攻击者能够区分生成器的输出与均匀分布，安全性可能会受到威胁，而我们事先不知道攻击者可能会使用什么策略。

上述考虑促成了在1980年代定义伪随机生成器的密码学方法。 基本的认识是，一个好的伪随机生成器应该通过**所有（高效的）统计测试**。 也就是说，对于任何高效统计测试（或**区分器**）D，给定伪随机生成器输出时 D 返回1的概率应该**接近**给定相同长度的均匀字符串时 D 返回1的概率。 因此，非正式地说，伪随机生成器的输出应该对任何**高效的观察者**来说**“看起来像”**一个均匀字符串。

（我们强调，形式上，说任何**固定的字符串**是**“伪随机的”**是没有意义的，就像说任何固定的字符串是“随机的”**是没有意义的一样。 相反，伪随机性是字符串分布的一种属性。 尽管如此，我们有时非正式地称根据均匀分布抽样的字符串为“均匀字符串”，并称伪随机生成器输出的字符串为“伪随机字符串”**。）

另一个视角是通过定义**分布**是伪随机的来获得的。 令 $Dist$ 是一个关于 l 比特字符串的分布。（这意味着 $Dist$ 为 $\{0,1\}^l$ 中的每个字符串分配了某个概率；从 $Dist$ 抽样意味着我们根据这个概率分布选择一个 l 比特字符串。） 非正式地，如果从 $Dist$ 抽样的字符串的实验与抽样长度为 l 的均匀字符串的实验**不可区分**，则 $Dist$ 是伪随机的。 （严格来说，由于我们处于渐近设置中，我们需要讨论一系列分布 $Dist = \{Dist_n\}$ 的伪随机性，其中分布 $Dist_n$ 用于安全参数 n。 我们在当前的讨论中忽略了这一点。） 更精确地说，对于任何**多项式时间算法**来说，应该**不可行**去判断它是被赋予了一个根据 $Dist$ 抽样的字符串，还是被赋予了一个均匀的 l 比特字符串（比猜测更好）。 这意味着，只要我们只考虑**多项式时间观察者**，伪随机字符串就与均匀字符串一样好。 正如不可区分性是完美保密性的**计算放松**一样，伪随机性是**真随机性**的**计算放松**。

现在令 $G : \{0,1\}^n \to \{0,1\}^l$ 是一个函数，并定义 $Dist$ 为通过选择一个均匀的 $s \in \{0,1\}^n$ 并输出 $G(s)$ 获得的 l 比特字符串上的分布。 那么，如果且仅如果分布 $Dist$ 是伪随机的，则 G 是一个伪随机生成器。

形式化定义。 如上所述，如果没有任何**高效区分器**能够检测到它是被赋予了 G 输出的字符串还是一个均匀随机选择的字符串，则 G 是一个伪随机生成器。 与定义3.9中一样，这通过要求每个高效算法在给定 $G(s)$（对于均匀种子 s）或均匀字符串时，以**几乎相同**的概率输出1来形式化。 （对于一个与定义3.8类似的等价定义，参见练习3.5。） 我们在渐近设置中获得一个定义，通过让安全参数 n 确定种子的长度。 然后，我们坚持 G 必须可以通过一个高效算法计算。 作为一个技术细节，我们还要求 G 的输出**比其输入长**；否则， G 就没有太大用处或意义。

定义 3.14 令 l 是一个多项式，并令 G 是一个确定性多项式时间算法，使得对于任何 n 和任何输入 $s \in \{0,1\}^n$，结果 $G(s)$ 的长度为 $l(n)$。 我们称 G 是一个 **伪随机生成器 (pseudorandom generator)**，如果满足以下条件：

1. **（扩展性：Expansion）** 对于每个 n，都有 $l(n) > n$ 成立。
2. **（伪随机性：Pseudorandomness）** 对于任何 **ppt** 算法 D，存在一个可忽略函数 $negl$ 使得

$$|\Pr[D(G(s)) = 1] - \Pr[D(r) = 1]| \leq negl(n),$$

   其中第一个概率是根据 $s \in \{0,1\}^n$ 的均匀选择和 D 的随机性来计算的，第二个概率是根据 $r \in \{0,1\}^{l(n)}$ 的均匀选择和 D 的随机性来计算的。

我们称 l 为 G 的**扩展因子**。

我们给出一个不安全的伪随机生成器的例子，以熟悉该定义。

例 3.15 定义 $G(s)$ 输出 s 后面跟着 $\bigoplus_{i=1}^{n} s_i$，因此 G 的扩展因子为 $l(n) = n + 1$。 G 的输出可以很容易地与均匀分布区分开来。 考虑以下高效区分器 D：给定一个字符串 w 作为输入，当且仅当 w 的最后一个比特等于 w 的所有前导比特的异或时，输出1。 由于 G 输出的所有字符串都具有此属性，我们有 $\Pr[D(G(s)) = 1] = 1$。 另一方面，如果 w 是均匀分布的，则 w 的最后一个比特是均匀分布的，因此 $\Pr[D(w) = 1] = 1/2$。 量 $|1/2 - 1|$ 是常数，不可忽略，因此这个 G 不是一个伪随机生成器。 （请注意， D 并非总是“正确”的，因为它在给定均匀字符串时有时也会输出1。 这不改变 D 是一个好的区分器的这一事实。）

讨论。 伪随机生成器 G 的输出分布**远非均匀**。 为了看到这一点，考虑 $l(n) = 2n$ 的情况，因此 G 将其输入长度加倍。 在 $\{0,1\}^{2n}$ 上的均匀分布下， 2^{2n} 个可能的字符串中的每一个都以恰好 2^{-2n} 的概率被选择。 相比之下，考虑 G 的输出分布（当 G 在均匀种子上运行时）。 当 G 接收长度为 n 的输入时， G 的范围内的不同字符串的数量至多为 2^n。 因此，长度为 $2n$ 的字符串中在 G 范围内的分数至多为 $2^n/2^{2n} = 2^{-n}$，我们看到绝大多数长度为 $2n$ 的字符串不

是 G 的输出。

这特别意味着，给定**无限时间**，区分随机字符串和伪随机字符串是微不足道的。 令 G 如上所述，并考虑按以下方式工作的**指数时间区分器** D：当且仅当存在一个 $s \in \{0,1\}^n$ 使得 $G(s) = w$ 时， $D(w)$ 输出1。 （这种计算是通过穷举计算每个 $s \in \{0,1\}^n$ 的 $G(s)$ 在指数时间内完成的。 回想一下，根据**柯克霍夫原则**， G 的规格是为 D 所知的。） 现在，如果 w 是由 G 输出的，那么 D 以1的概率输出1。 相比之下，如果 w 是在 $\{0,1\}^{2n}$ 中均匀分布的，那么存在 s 使得 $G(s) = w$ 的概率至多为 2^{-n}，因此在这种情况下 D 以至多 2^{-n} 的概率输出1。 因此，

$$|\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| \geq 1 - 2^{-n},$$

这是一个很大的值。 这只是暴力搜索攻击的另一个例子，并且不与 G 的伪随机性相矛盾，因为该攻击不是**高效的**。

种子及其长度。 伪随机生成器的种子类似于加密方案中使用的**密码密钥**，种子必须**均匀选择**并对任何攻击者**保密**。 另一个重要点，从上述暴力搜索攻击的讨论中可以看出，s **必须足够长**，以至于枚举所有可能的种子是不可行的。 在渐近意义上，这通过将种子的长度设置为安全参数来解决，以便对所有可能的种子进行穷举搜索需要指数时间。 在实践中，种子必须足够长，以便在指定的时间限制内尝试所有可能的种子是不可能的。

伪随机生成器的存在性。 伪随机生成器存在吗？ 它们看起来肯定很难构造，人们可能会问是否存在满足定义3.14的算法。 虽然我们不知道如何**无条件**地证明伪随机生成器的存在，但我们有充分的理由相信它们存在。 一方面，它们可以在**相当弱**的假设下构造，即**单向函数**存在（如果某些问题如分解大数是**困难的**，则这是成立的）； 我们将在第7章详细讨论这一点。 我们还有几种候选伪随机生成器的**实用构造**，称为**流密码**，目前还没有已知的有效区分器。 （稍后，我们将介绍更强的原语，称为**分组密码**。） 接下来，我们给出流密码的高级概述，并在第6章讨论具体的流密码。

流密码

我们对伪随机生成器的定义在两个方面是有限的：扩展因子是固定的，并且生成器在“一次性”中产生其**全部**输出。 在实践中用于实例化伪随机生成器的**流密码**工作方式有所不同。 流密码的伪随机输出比特是**逐步**和**按需**产生的，因此应用程序可以请求恰好所需的伪随机比特数量。 这提高了效率（因为应用程序如果足够的话可以请求更少的比特）和灵活性（因为可以请求的比特数量没有上限）。

形式上，我们将流密码视为一对确定性算法 $(Init, GetBits)$，其中： • $Init$ 以一个**种子** s

和一个可选的**初始化向量** IV 为输入，并输出一个**初始状态** st_0。 • $GetBits$ 以**状态信息** st_i 为输入，并输出一个比特 y 和**更新后的状态** st_{i+1}。 （在实践中， y 是一块包含几个比特的分组；为了通用性和简单性，我们这里将 y 视为单个比特。）

给定一个流密码和任何期望的扩展因子 l，我们可以定义一个算法 G_l，它将长度为 n 的输入映射到长度为 $l(n)$ 的输出。 该算法简单地运行 $Init$，然后重复运行 $GetBits$ 总共 l 次。

算法 3.16 从 $(Init, GetBits)$ 构造 G_l 输入： 种子 s 和可选初始化向量 IV **输出：** $y_1, \ldots, y_l$ $st_0 := Init(s, IV)$ **for** $i = 1$ **to** l: $(y_i, st_i) := GetBits(st_{i-1})$ **return** $y_1, \ldots, y_l$

如果一个流密码在**没有** IV 的基本意义上是安全的，并且对于任何多项式 l 来说，上面构造的函数 G_l 是一个扩展因子为 l 的伪随机生成器。 我们将在第3.6.1节中简要讨论流密码使用 IV 的一种可能的安全概念。

3.3.2 规约证明

如果我们想要证明一个给定的构造是**计算安全**的，那么我们必须依赖于**未经证明的假设** （除非该方案在信息理论上是安全的）。 我们的策略是**假设**某个数学问题是困难的，或者某个低级密码原语是安全的，然后**证明**在这个假设下，一个给定的基于该问题/原语的构造是安全的。 在第1.4.2节中，我们已经非常详细地解释了为什么这种方法更可取，因此我们在此不再重复这些论点。

一个密码学构造是安全的，只要某个基础问题是困难的**证明**通常是通过呈现一个**显式规约**来进行的，该规约展示了如何将任何成功“破解”该构造的有效攻击者 A 转换为一个解决被假定为困难的问题的高效算法 A'。 由于这一点非常重要，我们将详细介绍这种证明的步骤的高级概述。（我们将在本书中看到许多具体的例子，从定理3.18的证明开始。） 我们从假设某个问题 X 不能被任何多项式时间算法解决（在某种精确定义的意义上），除了以可忽略的概率之外开始。我们想要证明某个密码学构造 Π 是安全的（同样，在某种精确定义的意义上）。 一个证明通过以下步骤进行（另见图3.1）：

1. **固定**某个高效（即概率多项式时间）攻击者 A 攻击 Π。 用 $\varepsilon(n)$ 表示该攻击者的成功概率。
2. **构造**一个高效算法 A'，称为**“规约”**，**它尝试使用攻击者 A 作为子程序来解决问题 X。这里一个重要的点是 A' 对 A 的工作方式**一无所知**； A' 唯一知道的是 A 期望攻击 Π 。 因此，给定问题 X 的某个输入实例 x，我们的算法 A' 将为 A **模拟** Π 的一个实例，使得： (a) 就 A 所知，当 A' 作为子程序运行时， A 的视角应该与 A 与 Π 本身交互时的视角**分布相同**（或至少**接近**）。 (b) 如果 A 成功“破解”了由 A' 模拟的 Π 实例，那么 A'

应该能够以至少 $1/p(n)$ 的**逆多项式概率**解决它被给予的实例 x。

图 3.1： 安全性证明的规约的高级概述。 3. 综合起来，2(a)和2(b)意味着 A' 以 $\varepsilon(n)/p(n)$ 的概率解决 X。 如果 $\varepsilon(n)$ 不是可忽略的，那么 $\varepsilon(n)/p(n)$ 也不是可忽略的。 此外，如果 A 是高效的，那么我们获得了以非可忽略概率解决 X 的高效算法 A'，这与最初的假设相矛盾。 4. 鉴于我们关于 X 的假设，我们得出结论，**没有高效攻击者** A 能够以非可忽略概率成功破解 Π。 换句话说， Π 在计算上是安全的。

在接下来的部分，我们将精确地阐述上述想法：我们将展示如何使用任何伪随机生成器 G 来构造一个加密方案；我们通过展示任何能够“破解”该加密方案的攻击者都可以用来区分 G 的输出与均匀字符串，来证明该加密方案是安全的。 那么，在 G 是一个伪随机生成器的假设下，加密方案是安全的。

3.3.3 一个安全定长加密方案

伪随机生成器提供了一种自然的方式来构造一个密钥比消息短的**安全定长加密方案**。 回想一下，在**一次性密码本**中（参见第2.2节），加密是通过将**随机密码本**与消息进行异或来完成的。我们的洞察是我们可以使用**伪随机密码本**来代替。 然而，发送方和接收方可以**共享一个种子**，而不是共享这个长的、伪随机的密码本，该种子在需要时用于生成密码本（参见图3.2）；这个种子将比密码本短，因此也比消息短。 至于安全性，直觉是**伪随机字符串对于任何多项式时间攻击者来说“看起来是随机的”**，因此计算受限的窃听者无法区分使用一次性密码本加密的消息与使用这个“伪-”一次性密码本加密方案加密的消息。

图 3.2： 使用伪随机生成器进行加密。

加密方案。 固定某个消息长度 l，并令 G 是一个扩展因子为 l 的伪随机生成器（即 $|G(s)| = l(|s|)$）。 回想一下，加密方案由三个算法定义：密钥生成算法 Gen、加密算法 Enc 和解密算法 Dec。 密钥生成算法是平凡的： $Gen(1^n)$ 简单地输出长度为 n 的均匀密钥 k。 加密的工作原理是对密钥（用作种子）应用 G，以获得一个密码本，然后将该密码本与明文进行异或。 解密应用 G 到密钥，并将得到的密码本与密文进行异或以恢复消息。 该方案在**构造 3.17** 中正式描述。 在第3.6.1节中，我们将描述流密码在实践中如何用于实现该方案的一个变体。

构造 3.17 令 G 是一个扩展因子为 l 的伪随机生成器。 定义一个用于加密长度为 l 的消息的私钥加密方案如下： • **Gen：** 输入 1^n，选择均匀的 $k \in \{0,1\}^n$ 并输出它作为密钥。 • **Enc：** 输入密钥 $k \in \{0,1\}^n$ 和消息 $m \in \{0,1\}^{l(n)}$，输出密文

$$c := G(k) \oplus m.$$

• **Dec：** 输入密钥 $k \in \{0,1\}^n$ 和密文 $c \in \{0,1\}^{l(n)}$，输出消息

$$m := G(k) \oplus c.$$

一个基于任何伪随机生成器的私钥加密方案。

定理 3.18 如果 G 是一个伪随机生成器，则**构造 3.17** 是一个**在窃听者存在下具有不可区分加密性的定长私钥加密方案**。

证明 令 Π 表示**构造 3.17**。 我们证明 Π 满足定义3.8。 即，我们证明对于任何概率多项式时间攻击者 A 来说，存在一个可忽略函数 $negl$ 使得

$$\Pr\left[PrivK^{\text{eav}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + negl(n). \tag{3.2}$$

直觉是，如果 Π 使用**均匀密码本**代替伪随机密码本 $G(k)$，则得到的方案将与**一次性密码本加密方案**相同，且 A 无法以任何优于 $1/2$ 的概率正确猜出哪个消息被加密。 因此，如果公式(3.2)不成立，则 A 必然隐式地区分 G 的输出与随机字符串。 我们通过展示一个**规约**来明确这一点；即，通过展示如何使用 A 来构造一个高效区分器 D， D 区分 G 的输出与均匀字符串的能力与 A 确定哪个消息被 Π 加密的能力直接相关。 那么 G 的安全性意味着 Π 的安全性。

令 A 是一个任意 **ppt** 攻击者。 我们构造一个区分器 D，它将字符串 w 作为输入，其目标是确定 w 是均匀选择的（即 w 是一个**“随机字符串”）还是通过选择一个均匀 k 并计算 $w := G(k)$ 生成的（即 w 是一个“伪随机字符串”**）。 我们构造 D 以使其模拟 A 的窃听实验，如下所述，并观察 A 是否成功。 如果 A 成功，则 D 猜测 w 必是伪随机字符串，而如果 A 不成功，则 D 猜测 w 是随机字符串。 详细地说：

区分器 D： D 获得一个字符串 $w \in \{0,1\}^{l(n)}$ 作为输入。 （我们假设可以从 $l(n)$ 确定 n。）

1. 运行 $A(1^n)$ 以获得一对消息 $m_0, m_1 \in \{0,1\}^{l(n)}$。
2. 选择一个均匀比特 $b \in \{0,1\}$。 设置 $c := w \oplus m_b$。
3. 将 c 提供给 A，并获得输出 b'。 如果 $b' = b$，则输出1，否则输出0。

D 显然在多项式时间内运行（假设 A 如此）。

在分析 D 的行为之前，我们定义一个修改后的加密方案 $\overline{\Pi} = (\overline{Gen}, \overline{Enc}, \overline{Dec})$，它恰好是**一次性密码本**加密方案，除了我们现在引入了一个安全参数来确定要加密的消息的长度。 也就是说， $\overline{Gen}(1^n)$ 输出一个长度为 $l(n)$ 的均匀密钥 k，并且使用密钥 $k \in \{0,1\}^{l(n)}$ 加密消息 $m \in 2^{l(n)}$ 的密文 $c := k \oplus m$。 （解密可以照常进行，但在后续内容中是不必要的。） 一次

性密码本的完美保密性意味着

$$\Pr\left[PrivK^{\text{eav}}_{A,\Pi}(n)=1\right]=\frac{1}{2}. \tag{3.3}$$

要分析 D 的行为，主要观察结果是：

1. **如果 w 是在 $\{0,1\}^{l(n)}$ 中均匀选择的**，则 A 作为子程序在 D 运行时看到的视图与 A 在实验 $PrivK^{\text{eav}}_{A,\Pi}(n)$ 中看到的视图**分布相同**。 这是因为在这种情况下， A 被赋予了一个密文 $c=w\oplus m_b$，其中 $w\in\{0,1\}^{l(n)}$ 是均匀分布的。 由于 D 仅在 A 成功进行窃听实验时输出1，因此我们有（参见公式(3.3)）

$$\Pr_{w\leftarrow\{0,1\}^{l(n)}}[D(w)=1]=\Pr\left[PrivK^{\text{eav}}_{A,\Pi}(n)=1\right]=\frac{1}{2}. \tag{3.4}$$

   （第一个概率的下标只是明确说明 w 是在 $\{0,1\}^{l(n)}$ 中均匀选择的。）
2. **如果 w 是通过选择均匀 $k\in\{0,1\}^n$ 然后设置 $w:=G(k)$ 生成的**，则 A 作为子程序在 D 运行时看到的视图与 A 在实验 $PrivK^{\text{eav}}_{A,\Pi}(n)$ 中看到的视图**分布相同**。 这是因为 A 作为子程序在 D 运行时，现在被赋予了一个密文 $c=w\oplus m_b$，其中 $w=G(k)$ 对于均匀 $k\in\{0,1\}^n$。 因此，

$$\Pr_{k\leftarrow\{0,1\}^{n}}[D(G(k))=1]=\Pr\left[PrivK^{\text{eav}}_{A,\Pi}(n)=1\right]. \tag{3.5}$$

由于 G 是一个伪随机生成器（并且由于 D 在多项式时间内运行），我们知道存在一个可忽略函数 $negl$ 使得

$$\left|\Pr_{w\leftarrow\{0,1\}^{l(n)}}[D(w)=1]-\Pr_{k\leftarrow\{0,1\}^{n}}[D(G(k))=1]\right|\leq negl(n).$$

因此，使用公式(3.4)和(3.5)，我们看到

$$\left|\frac{1}{2}-\Pr\left[PrivK^{\text{eav}}_{A,\Pi}(n)=1\right]\right|\leq negl(n),$$

这蕴含着 $\Pr[PrivK^{\text{eav}}_{A,\Pi}(n)=1]\leq 1/2+negl(n)$。 由于 A 是一个任意 **ppt** 攻击者，这完成了 Π 在窃听者存在下具有不可区分加密性的证明。

很容易迷失在证明的细节中，并想知道与一次性密码本相比是否获得了什么； 毕竟，一次性密码本也通过将其与一个 l 比特字符串进行异或来加密一个 l 比特消息！ 当然，该构造的重点是， l 比特字符串 $G(k)$ 可以**比共享密钥 k 长得多**。 特别是，使用上述方案，可以用一个128

比特的密钥安全地加密一个1 Mb的文件。 通过依赖**计算安全性**，我们因此规避了定理2.10的不可实现性结果，该定理指出任何完美保密加密方案都必须使用**至少与消息一样长**的密钥。

规约——讨论。 我们**不能无条件**地证明**构造 3.17** 是安全的。 相反，我们证明了它在 G 是一个伪随机生成器的**假设下**是安全的。 这种将**更高级别构造**的安全性**规约**到**更低级别原语**的安全性**方法**具有许多优点（如第1.4.2节所讨论的）。 其中一个优点是，通常来说，设计低级别原语比高级别原语更容易；一般来说，直接分析算法 G 相对于低级别定义比分析更复杂的方案 Π 相对于高级别定义更容易。 这并不意味着构造伪随机生成器是“容易的”，只是说它比从头开始构造一个加密方案更容易。 （在当前情况下，加密方案除了将伪随机生成器的输出与消息异或之外没有做任何其他事情，因此这并不是真的。 然而，我们将看到更复杂的构造，在这些情况下，将任务规约为更简单的任务的能力是非常重要的。） 另一个优点是，一旦构造了合适的 G，它就可以用作各种其他方案的组件。

具体安全性。 尽管定理3.18及其证明是在渐近设置中，但我们可以轻松地调整该证明，以根据 G 的**具体安全性**来限定加密方案的**具体安全性**。 固定 n 的某个值以进行本讨论的其余部分，并令 Π 现在表示使用该 n 值的**构造 3.17**。 假设 G 是 (t,ε)**-伪随机**的（对于给定 n 值），这意味着对于所有运行时间**至多**为 t 的区分器 D，我们有

$$|\Pr[D(r)=1]-\Pr[D(G(s))=1]|\leq\varepsilon. \tag{3.6}$$

（想象 $t\approx 2^{80}$ 且 $\varepsilon\approx 2^{-60}$，尽管精确值与我们的讨论无关。） 我们声称 Π 是 $(t-c,\varepsilon)$**-安全**的，其中 c 是某个（小的）常数，其含义是对于所有运行时间**至多**为 $t-c$ 的 A 来说，我们有

$$\Pr\left[PrivK^{\text{eav}}_{A,\Pi}=1\right]\leq\frac{1}{2}+\varepsilon. \tag{3.7}$$

（请注意，由于我们不在此处于渐近设置中，上述是固定数字，而不是 n 的函数。） 要看到这一点，令 A 是一个任意的攻击者，其运行时间**至多**为 $t-c$。 如定理3.18证明中构造的区分器 D 在运行 A 之外只有很少的开销；适当设置 c 确保 D 运行时间**至多**为 t。 那么我们关于 G 的具体安全性的假设蕴含公式(3.6)；继续完全按照定理3.18的证明进行，我们获得了公式(3.7)。

3.4 更强的安全概念

到目前为止，我们只考虑了相对较弱的安全性定义，其中攻击者只**被动地窃听**诚实方之间发送的**单个密文**。 在本节中，我们考虑两个更强的安全概念。 回想一下，安全定义指定了一个安全目标和一个攻击模型。 在定义第一个新安全概念时，我们**修改**了安全目标；对于第二个，我们**加**

强了攻击模型。

3.4.1 多次加密的安全性

定义3.8处理了通信方传输**单个密文**并被窃听者观察到的情况。 然而，如果通信方可以互相发送**多个密文**——所有这些都使用相同的密钥生成——即使窃听者可能观察到所有这些密文，那也将很方便。 对于此类应用，我们需要一个对**多次消息加密**都安全的加密方案。

我们从针对此设置的适当安全定义开始。 与定义3.8一样，我们首先引入为任何加密方案 Π、攻击者 A 和安全参数 n 定义的适当实验：

多消息窃听实验 $PrivK^{\text{mult}}_{A,\Pi}(n)$：

1. 攻击者 A 获得输入 1^n，并输出一对**等长列表**的消息 $M_0 = (m_{0,1}, \ldots, m_{0,t})$ 和 $M_1 = (m_{1,1}, \ldots, m_{1,t})$，其中对所有 i 都有 $|m_{0,i}| = |m_{1,i}|$。
2. 通过运行 $Gen(1^n)$ 生成一个密钥 k，并选择一个**均匀比特** $b \in \{0,1\}$。 对于所有 i，计算密文 $c_i \leftarrow Enc_k(m_{b,i})$，并将列表 $C = (c_1, \ldots, c_t)$ 提供给 A。
3. A 输出一个比特 b'。
4. 实验的输出定义为如果 $b' = b$，则为1，否则为0。

安全定义与之前相同，只是它现在指向上述实验。

定义 3.19 一个私钥加密方案 $\Pi = (Gen, Enc, Dec)$ 具有**在窃听者存在下的不可区分多重加密性**，如果对于所有概率多项式时间攻击者 A 来说，存在一个**可忽略函数** $negl$ 使得

$$\Pr\left[PrivK^{\text{mult}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + negl(n),$$

其中概率是根据 A 使用的随机性以及实验中使用的随机性来计算的。

任何在窃听者存在下具有不可区分多重加密性的方案显然也满足定义3.8，因为实验 $PrivK^{\text{eav}}$ 对应于攻击者输出的两个列表仅包含单个消息的 $PrivK^{\text{mult}}$ 特殊情况。 事实上，正如以下所示，我们的新定义**严格强于**定义3.8。

命题 3.20 存在一个私钥加密方案，它在窃听者存在下具有**不可区分加密性**，但**不**具有在窃听者存在下不可区分的**多重加密性**。

证明 我们不需要走远就能找到一个满足该命题的加密方案的例子。 **一次性密码本**是完美保密的，因此在窃听者存在下也具有不可区分加密性。 我们证明它在定义3.19的意义上是**不安全**的。 （我们已经在第2章讨论了这种攻击；在这里，我们只是根据定义3.19分析该攻击。） 具

体来说，考虑攻击该方案的以下攻击者 A（在实验 $PrivK^{\text{mult}}$ 定义的意义上）： A 输出 $M_0 = (0^l, 0^l)$ 和 $M_1 = (0^l, 1^l)$。 （第一个列表包含相同的明文两次，而第二个列表包含两个不同的消息。） 令 $C = (c_1, c_2)$ 是 A 接收到的密文列表。 如果 $c_1 = c_2$，则 A 输出 $b' = 0$；否则， A 输出 $b' = 1$。 我们现在分析 $b' = b$ 的概率。 **关键点**是**一次性密码本是确定性的**，因此使用相同的密钥加密相同的消息两次总是产生相同的密文。 因此，如果 $b = 0$，那么我们**必须有** $c_1 = c_2$ 并且 A 在这种情况下输出0。 另一方面，如果 $b = 1$，那么每次加密不同的消息；因此 $c_1 \neq c_2$ 并且 A 输出1。 我们得出结论， A 以1的概率正确输出 $b' = b$，因此该加密方案在定义3.19方面是**不安全**的。

概率加密的必要性。 上述内容可能表明，使用**任何**加密方案都**不可能**实现定义3.19。 但事实上，只有当加密方案是**确定性**的，并且因此使用相同的密钥多次加密相同的消息总是产生相同的结果时，这才是正确的。 这一点非常重要，值得作为一个定理来陈述。

定理 3.21 如果 Π 是一个（**无状态**）加密方案，其中 Enc 是密钥和消息的确定性函数，则 Π **不可能**在窃听者存在下具有**不可区分多重加密性**。

这不应被理解为定义3.19过于强大。 的确，将两个加密的消息泄漏给窃听者，即它们是相同的，可能是一个**重大的安全漏洞**。 （例如，考虑一个学生加密一系列是非题答案的场景！） 为了构建一个对加密多条消息安全的方案，我们必须设计一个加密是**随机化**的方案，以便当相同的消息被多次加密时，可以产生**不同的密文**。 这似乎是不可能的，因为解密必须始终能够恢复消息。 然而，我们很快就会看到如何实现它。

3.4.2 选择明文攻击和CPA-安全性

选择明文攻击捕获了攻击者对诚实方加密的内容行使（部分）控制的能力。 我们设想一个场景，其中两个诚实方共享一个密钥 k，攻击者可以影响这些方加密消息 $m_1, m_2, \ldots$（使用 k）并将产生的密文发送到攻击者可以观察到的信道上。 在稍后的某个时间点，攻击者观察到对应于某个**未知消息** m 的密文，该密文是使用相同的密钥 k 加密的；我们甚至假设攻击者知道 m 是 m_0, m_1 两种可能性之一。 针对选择明文攻击的安全性意味着，即使在这种情况下，攻击者也不能以明显优于随机猜测的概率区分这两个消息中哪个被加密了。 （现在我们回到窃听者只被赋予单个未知消息的加密的情况。 很快，我们将回到考虑**多消息**的情况。）

现实世界中的选择明文攻击。 选择明文攻击是一个现实的担忧吗？ 首先，请注意选择明文攻击也涵盖了**已知明文攻击**——其中攻击者知道哪些消息正在被加密，即使它没有机会选择它们——作为一种特殊情况。 此外，有几种现实世界的场景中，攻击者可能对哪些消息被加密具有显著影响。 一个简单的例子是攻击者在终端上打字，终端反过来使用与远程服务器共享（且对攻击者未知）的密钥加密和发送攻击者输入的所有内容。 在这里，攻击者**精确地控制**了哪些内容被

加密，但是加密方案在被用于加密**另一个用户**的数据时——使用**相同的密钥**——应该保持安全。

有趣的是，选择明文攻击在历史上也成功地被用作破解军事加密方案的一部分。 例如，在第二次世界大战期间，英国在某些地点布设了水雷，知道德国人——当他们发现这些水雷时——会加密这些地点并将其发送回总部。 这些加密消息被布莱切利公园的密码分析师用来破解德国加密方案。

另一个例子是由中途岛战役的著名故事所提供的。 1942年5月，美国海军密码分析师截获了日本人的一条加密消息，他们能够部分解码。 结果表明日本人正计划攻击AF，其中AF是一个密文片段，美国无法解码。 由于其他原因，美国相信中途岛是目标。 不幸的是，他们试图说服华盛顿计划者的努力是徒劳的；普遍的信念是中途岛不可能成为目标。 海军密码分析师设计了以下计划：他们指示中途岛的美军发送一条假消息，称他们的淡水供应不足。 日本人截获了这条消息并立即向他们的上级报告说“AF淡水不足”。 海军密码分析师现在有了他们的证据，证明AF对应于中途岛，美国向该地点派出了三艘航空母舰。 结果是中途岛得以保全，日本人在太平洋战争中遭受了重大损失。

在这里，海军密码分析师进行了**选择明文攻击**，因为他们能够（尽管是以迂回的方式）影响日本人加密“中途岛”这个词。 如果日本人的加密方案对选择明文攻击是安全的，美国密码分析师的这种策略就不会奏效（历史可能会因此而大不相同）！

CPA-安全性。 在形式化定义中，我们通过赋予攻击者 A 访问一个**加密预言机**来建模选择明文攻击，该预言机被视为一个**“黑箱”，它使用一个对 A 未知的密钥 k 来加密 A 选择的消息。也就是说，我们设想 A 可以访问一个**“预言机”** $Enc_k(\cdot)$；当 A 通过提供消息 m 作为输入来查询这个预言机时，预言机返回密文 $c \leftarrow Enc_k(m)$ 作为答复。 （当 Enc 是随机化时，预言机在每次回答查询时都使用**新的随机性**。） 攻击者被允许以**自适应**的方式与加密预言机交互，次数不限。

考虑为任何加密方案 $\Pi = (Gen, Enc, Dec)$、攻击者 A 和安全参数值 n 定义的以下实验：

CPA 不可区分性实验 $PrivK^{\text{cpa}}_{A,\Pi}(n)$：

1. 通过运行 $Gen(1^n)$ 生成一个密钥 k。
2. 攻击者 A 获得输入 1^n 和对 $Enc_k(\cdot)$ 的**预言机访问权限**，并输出一对**等长**的消息 m_0, m_1。
3. 选择一个均匀比特 $b \in \{0,1\}$，然后计算密文 $c \leftarrow Enc_k(m_b)$ 并将其提供给 A。 我们称 c 为**挑战密文**。
4. 攻击者 A **继续**拥有对 $Enc_k(\cdot)$ 的预言机访问权限，并输出一个比特 b'。
5. 实验的输出定义为如果 $b' = b$，则为1，否则为0。 在前一种情况下，我们说 A **成功**。

定义 3.22 一个私钥加密方案 $\Pi = (Gen, Enc, Dec)$ 具有**在选择明文攻击下的不可区分加密性**，或简称 **CPA-安全的**，如果对于所有概率多项式时间攻击者 A 来说，存在一个**可忽略函数** $negl$ 使得

$$\Pr\left[PrivK^{\text{cpa}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + negl(n),$$

其中概率是根据 A 使用的随机性以及实验中使用的随机性来计算的。

多次加密的 CPA-安全性

可以像将定义3.8扩展为定义3.19一样，将定义3.22扩展到多次加密的情况，即使用消息列表。 在这里，我们采用了一种略微简单的方法，它具有建模攻击者可以**自适应地**选择要加密的明文的优点，即使在观察了以前的密文之后也是如此。 在当前的定义中，我们赋予攻击者访问一个**“左或右”预言机** $LR_{k,b}$，它以一对等长消息 m_0, m_1 为输入，计算密文 $c \leftarrow Enc_k(m_b)$ 并返回 c。 也就是说，如果 $b = 0$，则攻击者接收到“左”明文的加密，如果 $b = 1$，则接收到“右”明文的加密。 在这里， b 是在实验开始时选择的一个随机比特，与以前的定义一样，攻击者的目标是猜测 b。 这推广了先前**多消息安全性**的定义（定义3.19），因为攻击者现在可以**顺序地**查询 $LR_{k,b}(m_{0,1}, m_{1,1}), \ldots, LR_{k,b}(m_{0,t}, m_{1,t})$，而不是输出列表 $(m_{0,1}, \ldots, m_{0,t})$ 和 $(m_{1,1}, \ldots, m_{1,t})$，其中之一的消息将被加密。 这也包含了攻击者对加密预言机的访问权限，因为攻击者可以简单地查询 $LR_{k,b}(m, m)$ 来获得 $Enc_k(m)$。

我们现在正式定义这个实验，称为 **LR-预言机实验**。 令 Π 是一个加密方案， A 是一个攻击者， n 是安全参数：

LR-预言机实验 $PrivK^{\text{LR-cpa}}_{A,\Pi}(n)$：

1. 通过运行 $Gen(1^n)$ 生成一个密钥 k。
2. 选择一个均匀比特 $b \in \{0, 1\}$。
3. 攻击者 A 获得输入 1^n 和对 $LR_{k,b}(\cdot, \cdot)$ 的预言机访问权限，如上定义。
4. 攻击者 A 输出一个比特 b'。
5. 实验的输出定义为如果 $b' = b$，则为1，否则为0。 在前一种情况下，我们说 A **成功**。

定义 3.23 私钥加密方案 Π 具有**在选择明文攻击下的不可区分多重加密性**，或简称 **CPA-安全多重加密**，如果对于所有概率多项式时间攻击者 A 来说，存在一个**可忽略函数** $negl$ 使得

$$\Pr\left[PrivK^{\text{LR-cpa}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + negl(n),$$

其中概率是根据 A 使用的随机性以及实验中使用的随机性来计算的。

我们之前的讨论表明，**CPA-安全多重加密**至少与我们所有以前的定义一样强大。 特别是，如果一个私钥加密方案是 **CPA-安全多重加密** 的，那么它显然也是 **CPA-安全** 的。 重要的是，**反之亦然**；也就是说，**CPA-安全性**意味着 **CPA-安全多重加密**。 （这与窃听者攻击的情况形成对比；参见命题3.20。） 我们在此不加证明地陈述以下定理；公钥设置中的类似结果将在第11.2.2节中证明。

定理 3.24 任何 CPA-安全私钥加密方案也对**多次加密**是 CPA-安全的。

这是 **CPA-安全性** 的一个重要技术优势：只需证明一个方案是 CPA-安全的（针对单个加密），然后我们就可以**“免费”**获得它对多次加密也是 CPA-安全的。

针对选择明文攻击的安全性是加密方案应满足的**最低**安全概念，尽管现在更常见的要求是满足第4.5节中讨论的更强安全属性。

定长消息 vs. 任意长度消息。 使用 **CPA-安全性** 定义的另一个优势是，它允许我们**不失一般性地**处理**定长加密方案**。 特别是，给定任何 CPA-安全定长加密方案 $\Pi = (Gen, Enc, Dec)$，可以很容易地构造一个用于**任意长度消息**的 CPA-安全加密方案 $\Pi' = (Gen', Enc', Dec')$。为简单起见，假设 Π 加密长度为 n 比特的消息（尽管我们所说的任何内容都以自然的方式扩展，无论 Π 支持的消息长度如何）。 保持 Gen' 与 Gen 相同。 对于任何消息 m（具有任意长度 l），定义 Enc'_k 为 $Enc'_k(m) = Enc_k(m_1), \ldots, Enc_k(m_l)$，其中 m_i 表示 m 的第 i 个比特。 解密以自然方式完成。 如果 Π 是 CPA-安全的，那么 Π' 也是；一个证明由定理3.24得出。

有比以这种方式调整定长加密方案**更有效**地加密任意长度消息的方法。 我们将在第3.6节中进一步探讨这一点。

3.5 构建 CPA-安全加密方案

在构建对**选择明文攻击**安全的加密方案之前，我们首先介绍一个重要的概念：**伪随机函数**。

3.5.1 伪随机函数和分组密码

伪随机函数 (PRFs) 推广了伪随机生成器的概念。 现在，我们不再考虑**“看起来随机的”**字符串**，而是考虑**“看起来随机的”**函数**。 正如我们之前对伪随机性的讨论一样，说任何**固定函数** $f : \{0,1\}^* \to \{0,1\}^*$ 是伪随机的，没有太大意义（就像说任何固定函数是随机的没有意义一样）。 因此，我们必须转而参考**函数分布**的伪随机性，这是由**密钥函数**自然引起的，定义如下。

一个**密钥函数** $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ 是一个双输入函数，其中第一个输入称为**密钥**并表示为 k。 如果存在一个多项式时间算法，该算法在给定 k 和 x 时计算 $F(k,x)$，则我们称 F 是**高效的**。 （我们只对高效的密钥函数感兴趣。） 在典型用法中，密钥 k 被选择并固定，然后我们对**单输入函数** $F_k : \{0,1\}^* \to \{0,1\}^*$ 感兴趣，它由 $F_k(x) = F(k,x)$ 定义。 安全参数 n 决定了密钥长度、输入长度和输出长度。 也就是说，我们将 F 与三个函数 l_{key}, l_{in}, l_{out} 关联起来；对于任何密钥 $k \in \{0,1\}^{l_{key}(n)}$，函数 F_k 仅对输入 $x \in \{0,1\}^{l_{in}(n)}$ 有定义，在这种情况下 $F_k(x) \in \{0,1\}^{l_{out}(n)}$。 除非另有说明，我们假设 F 是**长度保持**的，这意味着 $l_{key}(n) = l_{in}(n) = l_{out}(n) = n$。 也就是说，通过固定一个密钥 $k \in \{0,1\}^n$，我们获得了将 n 比特输入字符串映射到 n 比特输出字符串的函数 F_k。

一个密钥函数 F 通过选择一个**均匀密钥** $k \in \{0,1\}^n$ 然后考虑得到的**单输入函数** F_k，自然地在函数上引发一个分布。 如果函数 F_k（对于均匀密钥 k）与从具有相同域和范围的**所有函数**集合中均匀选择的函数 f **不可区分**，则我们称 F 是**伪随机的**；也就是说，如果没有任何高效攻击者能够区分——在下面我们将更仔细地定义这种意义——它是在与 F_k（对于均匀 k）交互，还是在与 f（其中 f 是从将 n 比特输入映射到 n 比特输出的**所有函数**集合中均匀选择的）交互。

由于随机选择函数不如随机选择字符串直观，因此值得花更多时间讨论这个想法。 考虑所有将 n 比特字符串映射到 n 比特字符串的函数集合 $Func_n$。 这个集合是**有限的**，选择一个均匀函数 $f \in Func_n$ 意味着从这个集合中**均匀选择**一个元素。 $Func_n$ 有多大？ 一个函数 f 是通过给出它在其域的每个点上的值来指定的。 我们可以将任何函数（在有限域上）视为一个**大查找表**，它存储 $f(x)$ 在由 x 标记的行中。 对于 $f \in Func_n$， f 的查找表有 2^n 行（域 $\{0,1\}^n$ 的每个点一行），每行包含一个 n 比特字符串（因为 f 的范围是 $\{0,1\}^n$）。 连接表的所有条目，我们看到 $Func_n$ 中的任何函数都可以用长度为 $2^n \cdot n$ 的字符串表示。 此外，这种对应关系是**一一对应**的，因为长度为 $2^n \cdot n$ 的每个字符串（即包含 2^n 个长度为 n 的条目的每个表）都定义了 $Func_n$ 中一个唯一的函数。 因此， $Func_n$ 的大小恰好是长度为 $n \cdot 2^n$ 的字符串的数量，即 $|Func_n| = 2^{n \cdot 2^n}$。

将函数视为查找表提供了另一种思考均匀函数 $f \in Func_n$ 的选择的有用方式：这与均匀选择 f 查找表中的每一行是**完全等价**的。 这意味着，特别是，值 $f(x)$ 和 $f(y)$（对于任何两个输入 $x \neq y$）是**均匀且独立**的。 我们可以将这个查找表被**提前**随机条目填充，**在 f 被评估于任何输入之前**，或者我们可以将表的条目在 f 被评估于**以前从未评估过**的新输入上时，**按需均匀选择**。

回到我们关于伪随机函数的讨论，回想一下，一个伪随机函数是一个**密钥函数** F 使得 F_k（对于均匀随机选择的 $k \in \{0,1\}^n$）与 f（对于均匀随机选择的 $f \in Func_n$）是**不可区分的**。 前者是从一个（至多） 2^n 个不同函数的分布中选择的，而后者是从 $Func_n$ 中的所有 $2^{n \cdot 2^n}$ 个

函数中选择的。 尽管如此，这些函数的“行为”对任何**多项式时间区分器**来说必须看起来相同。

形式化伪随机函数概念的第一次尝试将以与定义3.14相同的方式进行。 也就是说，我们可以要求每个**多项式时间区分器** D 接收伪随机函数 F_k 的描述时，输出1的概率“几乎”与它接收随机函数 f 的描述时的概率相同。 然而，这个定义是**不恰当**的，因为随机函数的描述具有**指数长度**（由其 $n \cdot 2^n$ 长度的查找表给出），而 D 被限制在**多项式时间**内运行。 因此， D 甚至没有足够的时间来检查其**整个输入**。

因此，该定义赋予 D **访问一个预言机** O 的权限，该预言机要么等于 F_k（对于均匀 k），要么等于 f（对于均匀函数 f）。 区分器 D 可以在任何点 x 查询其预言机，作为回应，预言机返回 $O(x)$。 我们像在选择明文攻击的定义中向攻击者提供加密算法的预言机访问权限一样，将预言机视为一个**黑箱**。 然而，这里的预言机计算一个**确定性函数**，因此如果对相同的输入查询两次，则返回相同的结果。 （因此，我们可以假设不失一般性， D 永远不会对相同的输入查询预言机两次。） D 可以自由地与其预言机交互，根据所有以前的输出来**自适应地**选择其查询。 然而，由于 D 在多项式时间内运行，它只能询问**多项式次数**的查询。

我们现在提出形式化的定义。 （该定义为简单起见假定 F 是**长度保持**的。）

定义 3.25 令 $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ 是一个**高效**、**长度保持**的密钥函数。 如果对于所有**概率多项式时间区分器** D，存在一个可忽略函数 $negl$ 使得：

$$\left| \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \right| \leq negl(n),$$

其中第一个概率是根据 $k \in \{0,1\}^n$ 的均匀选择和 D 的随机性来计算的，第二个概率是根据 $f \in Func_n$ 的均匀选择和 D 的随机性来计算的。

一个重要的点是 D **没有被给予密钥** k。 如果 k 是已知的，要求 F_k 是伪随机的是**没有意义的**，因为给定 k，区分 F_k 的预言机和 f 的预言机是微不足道的。 （区分器所需要做的就是在任何点 x 查询预言机以获得答案 y，然后将其与它使用已知值 k **自己计算**的结果 $y' := F_k(x)$ 进行比较。 F_k 的预言机将返回 $y = y'$，而随机函数的预言机将以仅 2^{-n} 的概率返回 $y = y'$ 。） 这意味着，如果 k 被泄露，关于 F_k 伪随机性的任何声明都不再成立。 举一个具体的例子，如果 F 是一个伪随机函数，那么给定 F_k 的预言机访问权限（对于均匀 k），找到一个 x 使得 $F_k(x) = 0^n$ 必须是困难的（因为对于一个真正随机的函数 f，找到这样的输入也是困难的）。 但是如果 k 是已知的，找到这样的输入可能很容易。

例 3.26 像往常一样，我们可以通过看一个不安全的例子来熟悉该定义。 定义密钥的、长度保持函数 F 为 $F(k,x) = k \oplus x$。 对于任何输入 x， $F_k(x)$ 的值是均匀分布的（当 k 均匀时）。 然而， F 不是伪随机的，因为它的值在任何两个点上是**相关**的。 具体来说，考虑区分

器 D，它在任意两个**不同点** x_1, x_2 上查询其预言机 O，以获得值 $y_1 = O(x_1)$ 和 $y_2 = O(x_2)$，并**当且仅当** $y_1 \oplus y_2 = x_1 \oplus x_2$ 时输出1。 如果 $O = F_k$，则对于任何 k， D 输出1。 另一方面，如果 $O = f$ 且 f 是从 $Func_n$ 中均匀选择的，则 $f(x_1) \oplus f(x_2) = x_1 \oplus x_2$ 的概率恰好是 2^{-n}， D 以此概率输出1。 差异是 $|1 - 2^{-n}|$，这是**不可忽略的**。

伪随机置换/分组密码

令 $Perm_n$ 是 $\{0,1\}^n$ 上的**所有置换**（即双射）的集合。 像以前一样将任何 $f \in Perm_n$ 视为查找表，我们现在有了额外的约束，即任何两行中的条目必须**不同**。 我们在表的第一行有 2^n 种不同的选择；一旦我们固定了该条目，我们只剩下 $2^n - 1$ 种第二行选择，依此类推。 因此我们看到 $Perm_n$ 的大小是 $(2^n)!$。

令 F 是一个密钥函数。 我们称 F 是一个**密钥置换**，如果 $l_{in} = l_{out}$，并且对于所有 $k \in \{0,1\}^{l_{key}(n)}$ 来说，函数 $F_k : \{0,1\}^{l_{in}(n)} \to \{0,1\}^{l_{in}(n)}$ 是**一一对应**的（即 F_k 是一个**置换**）。 像以前一样，除非另有说明，我们假设 F 是**长度保持**的，因此 $l_{key}(n) = l_{in}(n) = n$。 如果存在一个**多项式时间算法**，用于在给定 k 和 x 时计算 $F_k(x)$，以及一个**多项式时间算法**用于在给定 k 和 y 时计算 $F_k^{-1}(y)$，则密钥置换是**高效的**。 也就是说，给定 k， F_k 必须是既高效可计算又高效可逆的。

关于一个高效密钥置换 F 是一个**伪随机置换**的定义与定义3.25完全类似，唯一的区别是**我们现在要求 F_k 与均匀置换不可区分，而不是与均匀函数不可区分**。 也就是说，我们要求没有高效算法能够区分访问 F_k（对于均匀密钥 k）和访问 f（对于均匀 $f \in Perm_n$）。 事实证明，这仅仅是一个**美学上的选择**，因为只要分组长度足够长，随机置换本身与随机函数是**不可区分**的。 从直觉上说，这是因为一个均匀函数 f 看起来与均匀置换相同，除非发现**不同**的值 x 和 y 使得 $f(x) = f(y)$，因为在这种情况下，该函数不可能是置换。 使用多项式次数的查询找到这些值 x, y 的概率是可忽略的。 （这由附录A.4的结果得出。）

命题 3.27 如果 F 是一个**伪随机置换**并且此外 $l_{in}(n) \geq n$，则 F 也是一个**伪随机函数**。

如果 F 是一个密钥置换，那么基于 F 的密码方案可能要求诚实方除了计算 F_k 本身之外，还需要计算**逆函数** F_k^{-1}。 这可能会引入新的安全问题。 特别是，现在可能需要施加更强的要求，即 F_k 与均匀置换不可区分，即使攻击者被**额外赋予**了对置换**逆函数**的预言机访问权限。 如果 F 具有此属性，我们称其为**强伪随机置换**。

定义 3.28 令 $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ 是一个**高效**、**长度保持**的密钥置换。 如果对于所有**概率多项式时间区分器** D，存在一个可忽略函数 $negl$ 使得：

$$\left| \Pr[D^{F_k(\cdot), F_k^{-1}(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot), f^{-1}(\cdot)}(1^n) = 1] \right| \leq negl(n),$$

其中第一个概率是根据 $k \in \{0,1\}^n$ 的均匀选择和 D 的随机性来计算的，第二个概率是根据 $f \in Perm_n$ 的均匀选择和 D 的随机性来计算的。

当然，任何**强伪随机置换**也是一个**伪随机置换**。

分组密码。 在实践中，**分组密码**被设计为**（强）伪随机置换**的**安全实例化，具有某些**固定密钥长度**和**分组长度**。** 我们在第6章讨论了构建分组密码的方法，以及一些流行的候选分组密码。 对于本章的目的，这些构造的细节并不重要，我们现在简单地假设（强）伪随机置换存在。

伪随机函数和伪随机生成器。 正如人们所期望的那样，伪随机函数和伪随机生成器之间存在密切的关系。 从伪随机函数 F 构造伪随机生成器 G 相当容易，只需对一系列不同的输入进行评估； 例如，对于任何期望的 l，我们可以定义 $G(s) \stackrel{\text{def}}{=} F_s(1)F_s(2)\cdots F_s(l)$。 如果用均匀函数 f 代替 F_s，则 G 的输出将是均匀的；因此，当使用 F 代替时，输出是伪随机的。 要求你正式证明这一点，见练习3.14。

更一般地，我们可以使用上述想法构造一个接受**初始化向量** IV 的**流密码** $(Init, GetBits)$。（参见第3.3.1节。） 唯一的区别是，不是对固定输入序列 $1, 2, 3, \ldots$ 评估 F_s，而是对输入 $IV + 1, IV + 2, \ldots$ 评估 F。

构造 3.29 令 F 是一个伪随机函数。 定义一个流密码 $(Init, GetBits)$，其中每次调用 $GetBits$ 输出 n 个比特，如下： • **Init：** 输入 $s \in \{0,1\}^n$ 和 $IV \in \{0,1\}^n$，设置 $st_0 := (s, IV)$。 • **GetBits：** 输入 $st_i = (s, IV)$，计算 $IV' := IV + 1$ 并设置 $y := F_s(IV')$ 和 $st_{i+1} := (s, IV')$。 输出 (y, st_{i+1})。 **一个从任何伪随机函数/分组密码构造的流密码。**

尽管流密码可以从分组密码构造出来，但在实践中使用的**专用流密码**通常具有更好的性能，特别是在资源受限的环境中。 另一方面，流密码（在实践中）似乎不如分组密码理解得那么好，并且对其安全性的信心较低。 因此，**建议**尽可能使用**分组密码**（可能首先将它们转换为流密码模式）。

考虑另一个方向，一个伪随机生成器 G 立即给出了一个**分组长度小**的伪随机函数 F。 具体地说，假设 G 的扩展因子为 $n \cdot 2^{t(n)}$。 我们可以定义密钥函数 $F : \{0,1\}^n \times \{0,1\}^{t(n)} \to \{0,1\}^n$ 如下：要计算 $F_k(i)$，首先计算 $G(k)$ 并将结果解释为一个查找表，其中包含 $2^{t(n)}$ 行，每行包含 n 个比特；输出第 i 行。 这仅在 $t(n) = O(\log n)$ 时在多项式时间内运行。 尽管更困难，但可以从伪随机生成器构造**分组长度大**的伪随机函数；这在第7.5节中有所展示。

3.5.2 从伪随机函数构造 CPA-安全加密

我们这里专注于构造一个 **CPA-安全**、**定长**的加密方案。 根据我们在第3.4.2节末尾所说的，这意味着存在一个用于**任意长度消息**的 CPA-安全加密方案。 在第3.6节中，我们将讨论**更有效**地加密任意长度消息的方法。

构造一个安全加密方案的**朴素尝试**是定义 $Enc_k(m) = F_k(m)$。 尽管我们期望这“不会泄露关于 m 的任何信息”（因为，如果 f 是一个均匀函数，那么 $f(m)$ 只是一个均匀的 n 比特字符串），这种加密方法是**确定性**的，因此**不可能**是 CPA-安全的。 特别是，将相同的明文加密两次将产生相同的密文。

我们的安全构造是**随机化**的。 具体来说，我们通过对**随机值** r 应用伪随机函数（而不是消息）并将结果与明文进行**异或**来加密。 （参见图3.3和**构造 3.30**。） 这可以再次被视为将**伪随机密码本**与明文异或的实例，主要的区别在于**每次**都使用**新的**伪随机密码本。 （事实上，如果伪随机函数被应用于一个**“新”值**，伪随机密码本就是“新的”，这个值以前从未被应用过。 虽然 r 随机值可能与以前选择的某个 r 值相等，但这仅以**可忽略的概率**发生。）

图 3.3： 使用伪随机函数进行加密。

基于伪随机函数的安全性证明。 在转向证明上述构造是 CPA-安全的之前，我们强调一个**通用模板**，该模板被大多数基于伪随机函数的构造的安全性证明（甚至在加密的上下文之外）所使用。这些证明的第一步是考虑构造的**假设版本**，其中伪随机函数被**随机函数**替换。 然后，使用**规约证明**来论证这种修改不会显著影响攻击者的成功概率。 然后，我们只剩下分析一个使用**完全随机函数**的方案。 此时，证明的其余部分通常依赖于**概率分析**，并且不依赖于任何计算假设。 我们将在本章和下一章多次利用这个证明模板。

构造 3.30 令 F 是一个伪随机函数。 定义一个用于加密长度为 n 的消息的私钥加密方案如下： • **Gen：** 输入 1^n，选择均匀的 $k \in \{0,1\}^n$ 并输出它。 • **Enc：** 输入密钥 $k \in \{0,1\}^n$ 和消息 $m \in \{0,1\}^n$，选择**均匀** $r \in \{0,1\}^n$ 并输出密文

$$c := \langle r, F_k(r) \oplus m \rangle.$$

• **Dec：** 输入密钥 $k \in \{0,1\}^n$ 和密文 $c = \langle r, s \rangle$，输出消息

$$m := F_k(r) \oplus s.$$

一个从任何伪随机函数构造的 CPA-安全加密方案。

定理 3.31 如果 F 是一个伪随机函数，则**构造 3.30** 是一个用于加密长度为 n 的消息的

CPA-安全私钥加密方案。

证明 令 $\bar{\Pi} = (\overline{Gen}, \overline{Enc}, \overline{Dec})$ 是一个加密方案，它与**构造 3.30** 中的 $\Pi = (Gen, Enc, Dec)$ **完全相同**，除了用**真正随机的函数** f 代替伪随机函数 F_k。 也就是说，$\overline{Gen}(1^n)$ 选择一个**均匀函数** $f \in Func_n$，并且 $\overline{Enc}$ 的加密方式与 Enc 完全相同，只是使用 f 代替 F_k。 （这个修改后的加密方案是**不高效**的。 但是我们可以仍然为了涉及 $\bar{\Pi}$ 的实验的目的而将其定义为一个**假设**的加密方案，这对证明的正确性**没有影响**。）

固定一个任意 **ppt** 攻击者 A，并令 $q(n)$ 是 $A(1^n)$ 对其加密预言机进行查询次数的上限。（请注意，q 必须以某个多项式为上限。） 作为证明的第一步，我们证明存在一个**可忽略函数** $negl$ 使得

$$\left|\Pr\left[PrivK^{\text{cpa}}_{A,\Pi}(n) = 1\right] - \Pr\left[PrivK^{\text{cpa}}_{A,\bar{\Pi}}(n) = 1\right]\right| \leq negl(n). \tag{3.8}$$

我们通过**规约**来证明这一点。 我们使用 A 来构造一个针对伪随机函数 F 的区分器 D。 区分器 D 被赋予对某个函数 O 的预言机访问权限，其目标是确定这个函数是“伪随机的”（即等于均匀 $k \in \{0,1\}^n$ 的 F_k）还是“随机的”（即等于均匀 $f \in Func_n$ 的 f）。 为此， D 如下所述地模拟 A 的 $PrivK^{\text{cpa}}$ 实验，并观察 A 是否成功。 如果 A 成功，则 D 猜测其预言机必是伪随机函数，而如果 A 不成功，则 D 猜测其预言机必是随机函数。 详细地说：

区分器 D： D 获得输入 1^n 和对预言机 $O : \{0,1\}^n \to \{0,1\}^n$ 的访问权限。

1. 运行 $A(1^n)$。 每当 A 查询其加密预言机关于消息 $m \in \{0,1\}^n$ 时，以以下方式回答此查询： (a) 选择均匀 $r \in \{0,1\}^n$。 (b) 查询 $O(r)$ 并获得响应 y。 (c) 将密文 $\langle r, y \oplus m\rangle$ 返回给 A。
2. 当 A 输出消息 $m_0, m_1 \in \{0,1\}^n$ 时，选择一个均匀比特 $b \in \{0,1\}$ 然后： (a) 选择均匀 $r \in \{0,1\}^n$。 (b) 查询 $O(r)$ 并获得响应 y。 (c) 将挑战密文 $\langle r, y \oplus m_b\rangle$ 返回给 A。
3. 继续回答 A 的加密预言机查询，直到 A 输出一个比特 b'。 如果 $b' = b$，则输出1，否则输出0。

D 在多项式时间内运行，因为 A 如此。 关键点如下：

1. **如果 D 的预言机是一个伪随机函数**，那么 A 作为子程序在 D 运行时看到的视图与 A 在实验 $PrivK^{\text{cpa}}_{A,\Pi}(n)$ 中看到的视图**分布相同**。 这是因为，在这种情况下，均匀随机选择一个密钥 k，然后每次加密都是通过选择一个均匀 r，计算 $y := F_k(r)$，并设置密文等于 $\langle r, y \oplus m\rangle$ 来执行的，这与**构造 3.30** 中完全相同。 因此，

$$\Pr_{k\leftarrow\{0,1\}^n}\left[D^{F_k(\cdot)}(1^n)=1\right]=\Pr\left[PrivK^{\mathrm{cpa}}_{A,\bar{\Pi}}(n)=1\right], \tag{3.9}$$

其中我们强调 k 是在左侧均匀选择的。

2. **如果 D 的预言机是一个随机函数**，那么 A 作为子程序在 D 运行时看到的视图与 A 在实验 $PrivK^{\mathrm{cpa}}_{A,\bar{\Pi}}(n)$ 中看到的视图**分布相同**。 这可以像上面一样精确地看到，唯一的区别是用均匀函数 $f\in Func_n$ 代替 F_k。 因此，

$$\Pr_{f\leftarrow Func_n}\left[D^{f(\cdot)}(1^n)=1\right]=\Pr\left[PrivK^{\mathrm{cpa}}_{A,\bar{\Pi}}(n)=1\right], \tag{3.10}$$

其中 f 是在左侧从 $Func_n$ 中均匀选择的。

根据 F 是一个伪随机函数的假设（并且由于 D 是高效的），存在一个可忽略函数 $negl$ 使得

$$\left|\Pr\left[D^{F_k(\cdot)}(1^n)=1\right]-\Pr\left[D^{f(\cdot)}(1^n)=1\right]\right|\leq negl(n).$$

将上述内容与公式(3.9)和(3.10)结合起来，得到公式(3.8)。

对于证明的第二部分，我们证明

$$\Pr\left[PrivK^{\mathrm{cpa}}_{A,\bar{\Pi}}(n)=1\right]\leq\frac{1}{2}+\frac{q(n)}{2^n}. \tag{3.11}$$

（回想一下， $q(n)$ 是 A 对加密预言机进行查询次数的上限。 上述内容即使我们不对 A 施加计算限制也成立。） 要看到公式(3.11)成立，观察到每当在实验 $PrivK^{\mathrm{cpa}}_{A,\bar{\Pi}}(n)$ 中加密消息 m 时（无论是通过加密预言机还是计算挑战密文时），都会选择一个**均匀** $r\in\{0,1\}^n$，并将密文设置为 $\langle r, f(r)\oplus m\rangle$。 令 r^* 表示用于生成挑战密文 $\langle r^*, f(r^*)\oplus m_b\rangle$ 的随机字符串。 有两种可能性：

1. **值 r^* 从未用于回答 A 的任何加密预言机查询：** 在这种情况下， A 从其与加密预言机的交互中没有学到关于 $f(r^*)$ 的任何信息（因为 f 是一个真正随机的函数）。 这意味着，就 A 而言，与 m_b 异或的值 $f(r^*)$ 在实验的其余部分中是**均匀分布且独立**的，因此在这种情况下 A 输出 $b'=b$ 的概率恰好是 $1/2$（如**一次性密码本**的情况）。
2. **值 r^* 用于回答 A 的至少一个加密预言机查询：** 在这种情况下， A 可以很容易地确定 m_0 或 m_1 是否被加密。 这是因为如果加密预言机返回密文 $\langle r^*, s\rangle$ 以响应加密消息 m 的请求，攻击者就会知道 $f(r^*)=s\oplus m$。 然而，由于 A 最多对其加密预言机进行 $q(n)$ 次查询（因此在回答 A 的加密预言机查询时最多使用 $q(n)$ 个 r 值），并且由于 r^* 是在 $\{0,1\}^n$ 中均匀选择的，因此发生此事件的概率**至多**为 $q(n)/2^n$。

令 $Repeat$ 表示 r^* 被加密预言机用于回答 A 的至少一个查询的事件。 正如刚刚讨论的，$Repeat$ 的概率至多为 $q(n)/2^n$，并且如果 $Repeat$ 不发生， A 在 $PrivK^{\text{cpa}}_{A,\bar{\Pi}}$ 中成功的概率恰好是 $1/2$。 因此：

$$\Pr\left[PrivK^{\text{cpa}}_{A,\bar{\Pi}}(n)=1\right]=\Pr\left[PrivK^{\text{cpa}}_{A,\bar{\Pi}}(n)=1\wedge\overline{Repeat}\right]+\Pr\left[PrivK^{\text{cpa}}_{A,\bar{\Pi}}(n)=1\wedge Repe$$

将上述内容与公式(3.8)结合起来，我们看到存在一个可忽略函数 $negl'$ 使得 $\Pr[PrivK^{\text{cpa}}_{A,\Pi}(n)=1]\leq 1/2+q(n)/2^n+negl(n)$。 由于 q 是多项式的， $q(n)/2^n$ 是可忽略的。 此外，两个可忽略函数的和是可忽略的，因此存在一个可忽略函数 $negl''$ 使得 $\Pr[PrivK^{\text{cpa}}_{A,\Pi}(n)=1]\leq 1/2+negl''(n)$，从而完成了证明。

具体安全性。 上述证明表明

$$\Pr\left[PrivK^{\text{cpa}}_{A,\Pi}(n)=1\right]\leq\frac{1}{2}+\frac{q(n)}{2^n}+negl(n)$$

对于某个可忽略函数 $negl$ 来说。 最后一项取决于 F 作为伪随机函数的安全性；它是算法 D 的区分概率的界限（其运行时间大致与攻击者 A 相同）。 项 $q(n)/2^n$ 表示用于加密挑战密文的值 r^* 被用于加密其他消息的概率的界限。

3.6 运行模式

运行模式提供了一种使用流密码或分组密码**安全地**（且高效地）加密**长消息**的方法。

3.6.1 流密码运行模式

构造 3.17 提供了一种使用伪随机生成器构造加密方案的方法。 该方案有两个主要缺点。 首先，如所呈现的，要加密的消息的长度必须是固定的并事先已知。 其次，该方案仅是 EAV-安全，而不是 CPA-安全。

流密码可以被视为灵活的伪随机生成器，可用于解决这些缺点。 在实践中，流密码以两种方式用于加密：**同步模式**和**非同步模式**。

同步模式。 在这种**有状态**加密方案中，发送方和接收方必须**同步**，即他们知道到目前为止已经加密（或解密）了多少明文。 同步模式通常用于**单个**通信会话中（参见第4.5.3节），其中有状态是可以接受的，并且消息是按顺序接收而不会丢失。 这里的直觉是生成了一个**长伪随机流**，并且它的**不同部分**用于加密每条消息。 需要同步以确保正确解密（即，接收方知道流的哪一部分用于加密下一条消息），并确保流的任何部分不会被**重复使用**。 接下来，我们将详细描述这

种模式。

图 3.4： 同步模式和非同步模式。

我们已经在**算法 3.16** 中看到流密码可以用于构造具有任何期望扩展因子 l 的伪随机生成器 G_l。 我们可以很容易地修改该算法，以获得具有**可变输出长度**的伪随机生成器 G_∞。 G_∞ 接受两个输入：一个种子 s 和一个期望的输出长度 1^l（我们以一元形式指定，因为 G_∞ 将在 l 的多项式时间内运行）。 与**算法 3.16** 中一样， $G_\infty(s, 1^l)$ 运行 $Init(s)$，然后重复运行 $GetBits$ 总共 l 次。

我们可以使用 G_∞ 在**构造 3.17** 中处理任意长度消息的加密：使用密钥 k 加密消息 m 是通过计算密文 $c := G_\infty(k, 1^{|m|}) \oplus m$ 完成的；使用密钥 k 解密密文 c 是通过计算消息 $m := G_\infty(k, 1^{|c|}) \oplus c$ 完成的。 对**定理 3.18** 证明的微小修改表明，如果流密码是安全的，则该加密方案在**窃听者存在下**具有不可区分加密性。

稍加思索就会发现，如果通信双方愿意**维护状态**，他们就可以使用相同的密钥来加密多条消息。（参见图3.4。） 概念上的洞察是，各方可以将多条消息 $m_1, m_2, \ldots$ 视为**单个、长消息**；此外，**构造 3.17**（以及上一段中修改后的版本）具有这样的属性：即使消息的其余部分未知，消息的初始部分也可以被加密和传输。 具体来说，通信双方共享一个密钥 k，并都通过计算 $st_0 := Init(k)$ 开始。 为了加密长度为 l_1 的第一条消息 m_1，发送方从 st_0 开始，重复运行 $GetBits$ 总共 l_1 次，以获得比特流 $pad_1 \stackrel{\text{def}}{=} y_1, \ldots, y_{l_1}$ 以及更新后的状态 st_1；然后发送 $c_1 := pad_1 \oplus m_1$。 接收到 c_1 后，另一方重复运行 $GetBits$ 总共 l_1 次，以获得相同的值 pad_1 和 st_1；它使用 pad_1 来恢复 $m_1 := pad_1 \oplus c_1$。 随后，为了加密长度为 l_2 的第二条消息 m_2，发送方将从 st_1 开始，重复运行 $GetBits$ 总共 l_2 次，以获得 $pad_2 \stackrel{\text{def}}{=} y_{l_1+1}, \ldots, y_{l_1+l_2}$ 以及更新后的状态 $st_{l_1+l_2}$，然后计算密文 $c_2 := pad_2 \oplus m_2$，依此类推。这可以无限期地继续下去，允许双方发送无限数量的任意长度消息。 我们指出，在这种模式下，流密码**不需要**使用 IV。

这种加密多条消息的方法要求通信双方**维护同步状态**，这解释了**“同步模式”**这一术语。 **因此，这种方法适用于**两个方在单个“会话”内通信**的情况，但对于**零星通信**或当一方可能随着时间的推移从**不同设备**通信时，效果不佳。 （在不同位置保持固定密钥的副本相对容易；在多个位置维护同步状态则更困难。） 此外，如果双方的同步出现偏差（例如，由于一方的传输被丢弃），解密将返回不正确的结果。 重新同步是可能的，但会增加额外的开销。

非同步模式。 对于 $Init$ 函数接受**初始化向量** IV 作为输入的流密码，我们可以实现**无状态**的任意长度消息 CPA-安全加密。 在这里，我们修改 G_∞ 以接受三个输入：一个种子 s，一个初始化向量 IV，和一个期望的输出长度 1^l。 现在，这个算法首先计算 $st_0 := Init(s, IV)$，

然后重复运行 $GetBits$ 总共 l 次。 然后可以使用**构造 3.30** 的变体来执行加密：使用密钥 k 加密消息 m 是通过选择**均匀初始化向量** $IV \in \{0,1\}^n$ 并计算密文 $\langle IV, G_\infty(s, IV, 1^{|m|}) \oplus m \rangle$ 来完成的；解密以自然方式进行。 （参见图3.4。） 如果流密码现在具有更强的属性，即对于任何多项式 l 来说，由 $F_k(IV) \stackrel{\text{def}}{=} G_\infty(k, IV, 1^l)$ 定义的函数 F 是一个**伪随机函数**，则该方案是 CPA-安全的。 （事实上， F 只需要在评估于均匀输入时是伪随机的即可。 具有这种较弱属性的密钥函数被称为**弱伪随机函数**。）

3.6.2 分组密码运行模式

我们已经看到了一个基于伪随机函数/分组密码的 CPA-安全加密方案的构造。 但是**构造 3.30**（以及在第3.4.2节末尾讨论的它对任意长度消息的扩展）的缺点是密文的长度是明文长度的**两倍**。 **分组密码运行模式**提供了一种使用**更短密文**加密任意长度消息的方法。

在本节中，令 F 是一个分组长度为 n 的分组密码。 我们假设要加密的所有消息 m 的长度都是 n 的倍数，并写 $m = m_1, m_2, \ldots, m_l$ 使得每个 $m_i \in \{0,1\}^n$ 表示一个明文块。 不为 n 的倍数的消息总是可以通过在后面附加一个1和足够多的0来明确地填充到 n 的倍数，因此这个假设不会失去太多的一般性。

已知几种分组密码运行模式；我们介绍四种最常见的模式并讨论它们的安全性。

电子密码本 (ECB) 模式。 这是一种**朴素**的运行模式，其中密文是通过将分组密码**直接**应用于每个明文块来获得的。 也就是说， $c := F_k(m_1), F_k(m_2), \ldots, F_k(m_l)$；参见图3.5。 解密以显而易见的方式完成，利用 F_k^{-1} 是高效可计算的事实。

ECB 模式是确定性的，因此不可能是 CPA-安全的。 更糟糕的是，ECB 模式加密甚至**不**具有在**窃听者存在下的不可区分加密性**。 这是因为如果明文中的一个块重复出现，它将导致密文中的一个重复块。 因此，很容易区分由两个相同块组成的明文的加密与由两个不同块组成的明文的加密。 这不仅仅是一个理论问题。 考虑加密一张图像，其中小块像素对应一个明文块。 使用ECB模式加密可能会泄露关于图像中模式的大量信息，这是在使用安全加密方案时不应该发生的事情。 图3.6演示了这一点。

出于这些原因，**不应该使用 ECB 模式**。 （我们包含它只是因为它具有历史意义。）

图 3.5： 电子密码本 (ECB) 模式。

图 3.6： 使用 ECB 模式的危险性图示。 中间的图是使用 ECB 模式加密左侧图像的密文；右侧的图是使用安全模式加密同一图像的密文。 （取自 http://en.wikipedia.org 并源自 Larry Ewing (lewing@isc.tamu.edu) 使用 The GIMP 创建的图像。）

密码块链接 (CBC) 模式。 要使用这种模式进行加密，首先选择一个长度为 n 的均匀**初始化向量 (IV)**。 然后，通过对**当前明文块**和**前一个密文块**的**异或**应用分组密码来生成密文块。 也就是说，设置 $c_0 := IV$，然后对于 $i = 1$ 到 l，设置 $c_i := F_k(c_{i-1} \oplus m_i)$。 最终的密文是 $\langle c_0, c_1, \ldots, c_l \rangle$。 （参见图3.7。） 密文 $\langle c_0, \ldots, c_l \rangle$ 的解密是通过计算 $m_i := F_k^{-1}(c_i) \oplus c_{i-1}$，对于 $i = 1, \ldots, l$ 来完成的。 我们强调 IV 包含在密文中；这对于解密能够进行至关重要。

重要的是，CBC 模式的加密是**概率性**的，并且已经证明，如果 F 是一个伪随机置换，那么 CBC 模式加密是 **CPA-安全** 的。 这种模式的主要缺点是加密必须**顺序**进行，因为需要前一个密文块 c_{i-1} 才能加密明文块 m_i。 因此，如果可以进行并行处理，CBC 模式加密可能不是最有效的选择。

人们可能会倾向于认为，为每次加密使用**不同的** IV（而不是随机的 IV）就足够了，例如，首先使用 $IV = 1$，然后每次加密消息时将 IV 增加1。 在练习3.20中，我们要求您证明这种 CBC 模式加密的变体是**不安全**的。

有人可能还会考虑 CBC 模式加密的**有状态**变体——称为**链式 CBC 模式**——其中**前一个密文的最后一个块**被用作加密下一条消息的 IV。 这减少了带宽，因为**不需要**每次都发送 IV。 参见图3.8，其中初始消息 m_1, m_2, m_3 使用随机 IV 加密，然后第二条消息 m_4, m_5 使用 c_3 作为 IV 加密。 （相比之下，使用无状态 CBC 模式加密在加密第二条消息时会生成**新的** IV。） 链式 CBC 模式用于 SSL 3.0 和 TLS 1.0。

链式 CBC 模式似乎与 CBC 模式一样安全，因为链式 CBC 对 m_1, m_2, m_3 的加密，后面跟着对 m_4, m_5 的加密，产生的密文块与 CBC 模式对（单个）消息 m_1, m_2, m_3, m_4, m_5 的加密相同。 然而，链式 CBC 模式**易受选择明文攻击**。 攻击的基础是攻击者事先知道将用于第二条加密消息的**“初始化向量”**。 我们根据图3.8非正式地描述攻击。 假设攻击者知道 $m_1 \in \{m'_1, m''_1\}$，并观察到第一个密文 $\langle IV, c_1, c_2, c_3 \rangle$。 然后攻击者请求加密第二条消息 m_4, m_5，其中 $m_4 = IV \oplus m'_1 \oplus c_3$，并观察第二个密文 $\langle c_4, c_5 \rangle$。 可以验证 $m_1 = m'_1$ 当且仅当 $c_4 = c_1$。 这个例子应该作为一个强烈的警告，即对密码方案进行任何修改都可能看起来是无害的。

图 3.7： 密码块链接 (CBC) 模式。

图 3.8： 链式 CBC。

输出反馈 (OFB) 模式。 我们介绍的第三种模式可以被视为一种**非同步流密码模式**，其中流密码是通过底层分组密码以特定方式构造的。 我们直接描述这种模式。 首先，选择一个均匀的 $IV \in \{0,1\}^n$。 然后，通过以下方式从 IV 生成一个伪随机流：定义 $y_0 := IV$，并设置流

的第 i 个块 y_i 为 $y_i := F_k(y_{i-1})$。 每个明文块通过将其与流的相应块进行异或来加密；也就是说， $c_i := y_i \oplus m_i$。 （参见图3.9。） 与 CBC 模式一样， IV 作为密文的一部分包含在内以启用解密。 然而，与 CBC 模式相反，这里**不要求 F 是可逆的**。 （事实上，它甚至不需要是置换。） 此外，与流密码模式一样，这里**不需要**明文长度是分组长度的倍数。 相反，生成的流可以截断到恰好是明文长度。 OFB 模式的另一个优势是其有状态变体（其中用于加密某个消息的最终值 y_l 被用作加密下一条消息的 IV）是**安全**的。 这种有状态变体等价于同步流密码模式，流密码以刚刚描述的特定方式从分组密码构造。

如果 F 是一个**伪随机函数**，则 OFB 模式可以被证明是 **CPA-安全** 的。 尽管加密和解密都必须**顺序**进行，但与 CBC 模式相比，这种模式的优势在于**大部分计算**（即伪随机流的计算）可以**独立于**要加密的实际消息。 因此，可以在**事先**使用预处理生成伪随机流，在此之后，明文的加密（一旦已知）会**非常快**。

图 3.9： 输出反馈 (OFB) 模式。

图 3.10： 计数器 (CTR) 模式。

计数器 (CTR) 模式。 计数器模式也可以被视为一种**非同步流密码模式**，其中流密码是根据**构造 3.29** 中的分组密码构造的。 我们在这里给出一个**自包含**的描述。 要使用 CTR 模式加密，首先选择一个均匀值 $ctr \in \{0,1\}^n$。 然后通过计算 $y_i := F_k(ctr + i)$ 来生成一个伪随机流，其中 ctr 和 i 被视为整数，并且加法是在模 2^n 下完成的。 第 i 个密文块是 $c_i := y_i \oplus m_i$，并且 IV 再次作为密文的一部分发送；参见图3.10。 请再次注意，解密**不要求 F 可逆**，甚至不需要是置换。 与 OFB 模式一样，另一种**“流密码”模式**下，**生成的流可以截断到恰好是明文长度，可以使用预处理在消息已知**之前**生成伪随机流，并且 CTR 模式的有状态变体是**安全**的。

与前面讨论的所有安全模式相比，CTR 模式的优势在于**加密和解密可以完全并行化**，因为伪随机流的所有块都可以彼此独立地计算。 与 OFB 相比，它还可以仅使用**一次 F 的评估**来解密密文的第 i 个块。 这些特性使 CTR 模式成为一个有吸引力的选择。

CTR 模式也相当容易分析：

定理 3.32 如果 F 是一个伪随机函数，则 CTR 模式是 **CPA-安全** 的。

证明 我们遵循与**定理 3.31** 证明相同的模板：我们首先用随机函数 f 替换 F，然后分析得到的方案。 令 $\overline{\Pi} = (\overline{Gen}, \overline{Enc}, \overline{Dec})$ 是（无状态的）CTR 模式加密方案，并令 $\overline{\Pi}' = (\overline{Gen}', \overline{Enc}', \overline{Dec}')$ 是与 Π 相同，只是用一个**真正随机的函数** f 代替 F_k 的加密方案。 也就是说， $\overline{Gen}'(1^n)$ 选择一个均匀函数 $f \in Func_n$，并且 $\overline{Enc}'$ 的加密方式与 Enc 完全相

同，只是使用 f 代替 F_k。 （同样， $\overline{Gen}'$ 和 $\overline{Enc}'$ 都不是高效的，但这对于涉及 $\bar{\Pi}'$ 的实验的目的并不重要。）

固定一个任意 **ppt** 攻击者 A，并令 $q(n)$ 是 $A(1^n)$ 对加密预言机的查询次数的上限，以及任何此类查询中的最大块数和 m_0 和 m_1 中的最大块数。 作为证明的第一步，我们声称存在一个**可忽略函数** $negl$ 使得

$$\left|\Pr\left[PrivK^{\text{cpa}}_{A,\Pi}(n)=1\right]-\Pr\left[PrivK^{\text{cpa}}_{A,\bar{\Pi}'}(n)=1\right]\right|\leq negl(n). \qquad (3.12)$$

这通过类似于**定理 3.31** 证明中类似步骤的**规约**来证明，并留作读者的练习。 我们接下来声称

$$\Pr\left[PrivK^{\text{cpa}}_{A,\bar{\Pi}'}(n)=1\right]<\frac{1}{2}+\frac{2q(n)^2}{2^n}. \qquad (3.13)$$

结合公式(3.12)，这意味着

$$\Pr\left[PrivK^{\text{cpa}}_{A,\Pi}(n)=1\right]<\frac{1}{2}+\frac{2q(n)^2}{2^n}+negl(n).$$

由于 q 是多项式的， $2q(n)^2/2^n$ 是可忽略的，这完成了证明。

我们现在证明公式(3.13)。 固定安全参数的某个值 n。 令 $l^*\leq q(n)$ 表示实验 $PrivK^{\text{cpa}}_{A,\bar{\Pi}'}(n)$ 中 A 输出的消息 m_0,m_1 的长度（以块计），并令 ctr^* 表示生成挑战密文时使用的初始值。 类似地，令 $l_i\leq q(n)$ 是 A 进行的第 i 个加密预言机查询的长度（以块计），并令 ctr_i 表示回答此查询时使用的初始值。 当回答第 i 个加密预言机查询时， f 被应用于值 $ctr_i+1,\ldots,ctr_i+l_i$。 当加密挑战密文时， f 被应用于 $ctr^*+1,\ldots,ctr^*+l^*$，第 i 个密文块是通过将 $f(ctr^*+i)$ 与第 i 个消息块异或来计算的。 有两种情况：

1. **不存在任何 $i,j,j^*\geq 1$（其中 $j\leq l_i$ 且 $j^*\leq l^*$）使得 $ctr_i+j=ctr^*+j^*$：** 在这种情况下，用于加密挑战密文的值 $f(ctr^*+1),\ldots,f(ctr^*+l^*)$ 是**均匀分布**的，并且**独立于**实验的其余部分，因为 f 未应用于回答攻击者预言机查询的任何这些输入。 这意味着挑战密文是通过将均匀比特流与消息 m_b 异或来计算的，因此 A 输出 $b'=b$ 的概率恰好是 $1/2$（如**一次性密码本**的情况）。
2. **存在 $i,j,j^*\geq 1$（其中 $j\leq l_i$ 且 $j^*\leq l^*$）使得 $ctr_i+j=ctr^*+j^*$：** 我们将此事件表示为 $Overlap$。 在这种情况下， A 可能会确定哪个消息被加密以给出挑战密文，因为 A 可以从其第 i 个预言机查询的答案中推导出 $f(ctr_i+j)=f(ctr^*+j^*)$ 的值。

让我们分析 $Overlap$ 发生的概率。 如果 l^* 和每个 l_i 都尽可能大，使得 $l^*=l_i=q(n)$，则概率最大化。 令 $Overlap_i$ 表示序列 $ctr_i+1,\ldots,ctr_i+q(n)$ 与序列 ctr^*+

$1, \ldots, ctr^* + q(n)$ **重叠**的事件；则 $Overlap$ 是 $Overlap_i$ 发生于某个 i 的事件。 由于最多有 $q(n)$ 个预言机查询，并集界（参见命题A.7）给出

$$\Pr[Overlap] \leq \sum_{i=1}^{q(n)} \Pr[Overlap_i]. \tag{3.14}$$

固定 ctr^*，事件 $Overlap_i$ 恰好在 ctr_i 满足

$$ctr^* - q(n) + 1 \leq ctr_i \leq ctr^* + q(n) - 1.$$

时发生。 由于有 $2q(n) - 1$ 个 ctr_i 值会发生 $Overlap_i$，并且 ctr_i 是从 $\{0,1\}^n$ 中均匀选择的，我们看到

$$\Pr[Overlap_i] = \frac{2q(n) - 1}{2^n} < \frac{2q(n)}{2^n}.$$

结合公式(3.14)，这给出了 $\Pr[Overlap] < q(n) \cdot \frac{2q(n)}{2^n} = \frac{2q(n)^2}{2^n}$。 鉴于上述，我们可以很容易地限制 A 的成功概率：

$$\Pr\left[PrivK^{\text{cpa}}_{A,\bar{\Pi}'}(n) = 1\right] = \Pr\left[PrivK^{\text{cpa}}_{A,\bar{\Pi}'}(n) = 1 \wedge \overline{Overlap}\right] + \Pr\left[PrivK^{\text{cpa}}_{A,\bar{\Pi}'}(n) = 1 \wedge Ov$$

这证明了公式(3.13)并完成了证明。

运行模式和消息篡改。 在许多教科书中，运行模式也根据它们如何防止攻击者对密文的修改来比较。 我们**不**在这里进行这种比较，因为**消息完整性**或**消息认证**的问题应与**加密**分开处理，我们将在下一章中这样做。 上述模式都没有实现我们将在下一章中定义的消息完整性。 我们推迟到下一章再进一步讨论。

关于不同模式在存在**“良性”**（即非恶意）传输错误时的行为，请参见练习3.21和3.22。 然而，我们指出，通常可以使用标准技术（例如，错误纠正或重传）来解决此类错误。

分组长度和具体安全性。 CBC、OFB 和 CTR 模式都使用**随机** IV。 这具有随机化加密过程的效果，并确保（以高概率）底层分组密码始终在**新鲜**（即新的）输入上进行评估。 这一点很重要，因为正如我们在**定理 3.31** 和**定理 3.32** 的证明中看到的那样，如果分组密码的一个输入被多次使用，那么安全性就会被破坏。

分组密码的分组长度因此对基于该密码的加密方案的**具体安全性**有重大影响。 例如，考虑使用分组长度为 l 的分组密码 F 的 CTR 模式。 那么 IV 是一个均匀的 l 比特字符串，我们预计在加密约 $2^{l/2}$ 条消息后 IV 会重复出现（参见附录A.4）。 如果 l 太短，那么——即使 F 作为

伪随机置换是安全的——由此产生的具体安全界限对于实际应用来说也会太弱。 具体来说，如果 $l = 64$，就像 DES（我们将在第6章研究的分组密码）的情况一样，那么在约 $2^{32} \approx 4,300,000,000$ 次加密后——或大约34千兆字节的明文——预计会发生重复的 IV。 虽然这看起来是很多数据，但它小于现代硬盘的容量。

IV 误用。 在我们对各种（安全）模式的描述和讨论中，我们假设**每次加密消息时**都会选择一个**随机** IV。 如果这个假设出了问题，例如由于随机性生成不佳或错误的实现，会发生什么？ 当然，我们不能再保证定义3.22意义上的安全性了。 然而，从实际的角度来看，“流密码模式”（OFB和CTR）比CBC模式要**差得多**。 如果在前者使用时 IV 重复，攻击者可以对结果的两个密文进行异或，并了解两个加密消息的**全部内容**的大量信息（正如我们在**一次性密码本**的上下文中看到的那样，如果密钥被重复使用）。 然而，使用 CBC 模式，在仅几个块之后，分组密码的输入很可能会“发散”，攻击者将无法了解超出前几个消息块的任何信息。

解决潜在 IV 误用的一种方法是使用**有状态加密**，如在 OFB 和 CTR 模式的上下文中讨论的那样。 如果**无状态加密不可能**，并且担心潜在的 IV 误用，那么由于上述原因，**推荐**使用 **CBC 模式**。

3.7 选择密文攻击

3.7.1 定义 CCA-安全性

到目前为止，我们已经定义了针对两种类型攻击的安全性：**被动窃听**和**选择明文攻击**。 **选择密文攻击**甚至更强大。 在选择密文攻击中，攻击者不仅能够获得它选择的消息的加密（如在选择明文攻击中），而且还能够获得它选择的**密文的解密**（有一个技术限制，稍后讨论）。 形式上，我们赋予攻击者除了**加密预言机**之外，还可以访问**解密预言机**。 我们介绍形式化定义，并推迟进一步讨论。

考虑为任何私钥加密方案 $\Pi = (Gen, Enc, Dec)$、攻击者 A 和安全参数值 n 定义的以下实验。

CCA 不可区分性实验 $PrivK^{\text{cca}}_{A,\Pi}(n)$：

1. 通过运行 $Gen(1^n)$ 生成一个密钥 k。
2. 攻击者 A 获得输入 1^n 和对 $Enc_k(\cdot)$ 和 $Dec_k(\cdot)$ 的**预言机访问权限**。 它输出一对**等长**的消息 m_0, m_1。
3. 选择一个均匀比特 $b \in \{0,1\}$，然后计算密文 $c \leftarrow Enc_k(m_b)$ 并将其提供给 A。 我们称 c 为**挑战密文**。
4. 攻击者 A **继续**拥有对 $Enc_k(\cdot)$ 和 $Dec_k(\cdot)$ 的预言机访问权限，但是**不允许**查询后者关于

挑战密文 c 本身。 最终， A 输出一个比特 b'。
5. 实验的输出定义为如果 $b' = b$，则为1，否则为0。 如果实验的输出为1，我们说 A **成功**。

定义 3.33 一个私钥加密方案 Π 具有**在选择密文攻击下的不可区分加密性**，或简称 **CCA-安全**，如果对于所有概率多项式时间攻击者 A 来说，存在一个**可忽略函数** $negl$ 使得

$$\Pr\left[PrivK^{\text{cca}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + negl(n),$$

其中概率是根据实验中使用的所有随机性来计算的。

为了完整性，我们指出定理3.24的自然类比也适用于 CCA-安全性。 （即，如果一个方案在选择密文攻击下具有不可区分加密性，那么它在选择密文攻击下也具有不可区分多重加密性，适当定义。）

在上述实验中，攻击者对解密预言机的访问权限是**无限的**，除了**限制**攻击者不能请求挑战密文本身的解密。 这个限制是**必要**的，否则任何加密方案都显然没有希望满足该定义。

此时，您可能想知道**选择密文攻击**是否真实地建模了任何现实世界的攻击。 与选择明文攻击的情况一样，我们不期望诚实方解密攻击者选择的**任意**密文。 然而，可能存在攻击者能够影响哪些内容被解密并了解关于结果的一些部分信息的场景。 例如：

1. 在第3.4.2节的中途岛示例中，可以设想美国密码分析师可能也曾试图向日本人发送加密消息，然后监视他们的行为。 这种行为（例如，部队调动等）可能提供了关于底层明文的重要信息。
2. 想象一个用户向她的银行发送加密消息。 攻击者可能能够代表该用户发送密文；银行将解密这些密文，攻击者可能会了解关于结果的一些信息。 例如，如果一个密文对应于一个**格式不正确**的明文（例如，一个不可理解的消息，或者只是一个格式不正确的消息），攻击者可能能够从银行的反应（即，后续通信的模式）中推断出这一点。 下面第3.7.2节中介绍了这种类型的实用攻击。
3. 加密通常用于更高级别的协议中；例如，加密方案可能被用作**认证协议**的一部分，其中一方将密文发送给另一方，另一方解密它并返回结果。 在这种情况下，其中一个诚实方正好扮演一个**解密预言机**的角色。

我们研究过的方案的安全性。 到目前为止，我们看到的加密方案中**没有CCA-安全**的。 我们通过**构造 3.30** 来证明这一点，其中加密计算为 $Enc_k(m) = \langle r, F_k(r) \oplus m \rangle$。 考虑一个运行 CCA 不可区分性实验的攻击者 A，她选择 $m_0 = 0^n$ 和 $m_1 = 1^n$。 然后，在收到密文 $c = \langle r, s \rangle$ 后，攻击者可以翻转 s 的第一个比特并请求对得到的密文 c' 进行解密。 由于 $c' \neq c$，

允许进行此查询，解密预言机将返回 10^{n-1}（在这种情况下，显然 $b = 0$）或 01^{n-1}（在这种情况下， $b = 1$）。 这个例子表明 CCA-安全性是**相当严格**的。 任何允许密文以任何可控方式**“被操纵”**的加密方案都不可能是 **CCA-安全的。 因此，CCA-安全性意味着一个非常重要的属性，称为**不可篡改性**。 粗略地说，**不可篡改加密方案**具有以下属性：如果攻击者尝试修改给定的密文，结果要么是**无效密文**，要么是**解密为与原始明文没有任何关系**的明文。 这对于用于复杂密码协议的方案来说是一个非常有用的属性。

构造 CCA-安全加密方案。 我们将在第4.5.4节中展示如何构造一个 CCA-安全加密方案。 之所以在那里介绍该构造，是因为它使用了我们在第4章中开发的工具。

3.7.2 填充预言机攻击

上一节中描述的针对**构造 3.30** 的选择密文攻击有点**做作**，因为它假设攻击者可以获得修改后的密文的**完整解密**。 虽然这种攻击在定义3.33下是允许的，但尚不清楚它是否代表了实践中的严重担忧。 我们在这里展示一个针对**自然且广泛使用**的加密方案的选择密文攻击；此外，该攻击**仅需要**攻击者确定修改后的密文是否**正确解密**的能力。 这种信息通常很容易获得，因为，例如，如果服务器收到一个解密不正确（或不按预期格式化）的密文，它可能会请求重传或终止会话，并且这些事件中的任何一个都会在观察到的流量中产生明显的**变化**。 这种攻击已被证明在各种部署的协议中实际可行；我们在本节末尾给出一个具体的例子。

如前所述，当使用 CBC 模式时，明文的长度必须是**分组长度的倍数**；如果明文不满足此属性，则必须在加密前进行**填充**。 我们将**原始明文**称为**消息**，填充后的结果称为**编码数据**。 我们使用的填充方案必须允许接收方从编码数据中**明确**确定消息的结束位置。 一种流行且标准化的方法是使用 **PKCS #5 填充**。 假设原始消息具有整数个字节，并令 L 表示正在使用的分组密码的分组长度（以字节为单位）。 令 b 表示需要附加到消息末尾的字节数，以便得到的编码数据的总长度是分组长度的倍数。 这里， b 是一个介于1和 L 之间（含）的整数。 （我们不能有 $b = 0$，因为这会导致填充模糊不清。 因此，如果消息长度已经是分组长度的倍数，则会附加 L 个字节的填充。） 然后，我们将包含整数 b 的字符串（以1字节或2个十六进制数字表示）重复 b 次附加到消息末尾。 也就是说，如果需要1字节的填充，则附加字符串 $0\text{x}01$（以十六进制写入）；如果需要4字节的填充，则附加十六进制字符串 $0\text{x}04040404$；等等。 然后，使用常规的 CBC 模式加密来加密编码数据。

在解密时，接收方首先像往常一样应用 CBC 模式解密以恢复编码数据，然后检查编码数据是否正确填充。 （这很容易做到：只需读取最后一个字节的值 b，然后验证结果的最后 b 个字节是否都具有值 b。） 如果正确，则剥离填充并返回原始消息。 否则，标准程序是返回一个**“错误填充”错误**（例如，在Java中，标准异常称为 $javax.crypto.\text{BadPaddingException}$）。 存在这样的错误消息为攻击者提供了一个**部分解密预言机**。 也就是说，攻击者可以向服务器发

送任何密文，并根据是否返回“错误填充”错误来了解底层编码数据是否以正确的格式填充。 尽管这看起来似乎是**无关紧要**的信息，但我们证明它使攻击者能够**完全恢复**它选择的任何密文的原始消息。

为简单起见，我们描述对一个3块密文的攻击。 令 $\langle IV, c_1, c_2\rangle$ 是攻击者观察到的密文，令 m_1, m_2 是底层编码数据（攻击者未知），它对应于如上所述的填充消息。 （每个块长 L 字节。） 请注意

$$m_2 = F_k^{-1}(c_2) \oplus c_1, \tag{3.15}$$

其中 k 是诚实方使用的密钥（当然，攻击者不知道）。 第二个块 m_2 结束于 $\underbrace{0\text{x}b \cdots 0\text{x}b}_{b \text{ times}}$，其中我们令 $0\text{x}b$ 表示整数 b 的1字节表示。 攻击中使用的**关键属性**是密文的某些更改会导致 CBC 模式解密后底层编码数据发生可预测的更改。 具体来说，令 c_1' 与 c_1 相同，除了最后一个字节的修改。 考虑解密修改后的密文 $\langle IV, c_1', c_2\rangle$。 这将导致编码数据 m_1', m_2'，其中 $m_2' = F_k^{-1}(c_2) \oplus c_1'$。 与公式(3.15)相比，我们看到 m_2' 将与 m_2 相同，除了最后一个字节的修改。 （ m_1' 的值是不可预测的，但这不会对攻击产生不利影响。） 类似地，如果 c_1' 与 c_1 相同，除了第 i 个字节的更改，则 $\langle IV, c_1', c_2\rangle$ 的解密将导致 m_1', m_2'，其中 m_2' 与 m_2 相同，除了第 i 个字节的更改。 更一般地，如果 $c_1' = c_1 \oplus \Delta$ 对于任何字符串 Δ，则 $\langle IV, c_1', c_2\rangle$ 的解密将产生 m_1', m_2'，其中 $m_2' = m_2 \oplus \Delta$。 结果是**攻击者可以对编码数据的最后一个块行使显著的控制**。

作为热身，让我们看看攻击者如何利用这一点来了解 b，即填充量。 （这泄露了原始消息的长度。） 回想一下，在解密时，接收方查看编码数据的第二个块的最后一个字节的值 b，然后验证结果的最后 b 个字节是否都具有相同的值。 攻击者首先修改 c_1 的第一个字节，并将得到的密文 $\langle IV, c_1', c_2\rangle$ 发送给接收方。 如果解密失败（即接收方返回错误），则**必须**是接收方正在检查 m_2' 的所有 L 个字节，因此 $b = L$！ 否则，攻击者得知 $b < L$，然后它可以对第二个字节重复该过程，依此类推。 导致解密失败的**最左侧修改字节**精确地泄露了接收方正在检查的**最左侧字节**，从而精确地泄露了 b。

知道了 b，攻击者可以继续**逐字节**地学习消息的字节。 我们以消息的最后一个字节（我们用 B 表示）为例说明这个想法。 攻击者知道 m_2 以 $0\text{x}b \cdots 0\text{x}b$ 结尾（其中 $0\text{x}b$ 重复 b 次），并希望了解 B。 定义

$$\Delta_i \stackrel{\text{def}}{=} \underbrace{0\text{x}00 \cdots 0\text{x}00}_{L-b-1 \text{ times}} 0\text{x}i \underbrace{0\text{x}(b+1) \cdots 0\text{x}(b+1)}_{b \text{ times}} \oplus \underbrace{0\text{x}00 \cdots 0\text{x}000}_{L-b-1 \text{ times}} 0\text{x}00 \underbrace{0\text{x}b \cdots 0\text{x}b}_{b \text{ times}}$$

对于 $0 \le i < 2^8$；也就是说， Δ_i 的最后 $b+1$ 个字节包含整数 i（以十六进制表示）后面跟

着值 $(b+1) \oplus b$（以十六进制表示）重复 b 次的字符串。 如果攻击者向接收方提交密文 $\langle IV, c_1 \oplus \Delta_i, c_2 \rangle$，那么在 CBC 模式解密后，得到的编码数据将等于 $0\text{x}(B \oplus i)0\text{x}(b+1) \cdots 0\text{x}(b+1)$（其中 $0\text{x}(b+1)$ 重复 b 次）。 解密将失败，除非 $0\text{x}(B \oplus i) = 0\text{x}(b+1)$。 攻击者只需尝试最多 2^8 个值 $\Delta_0, \ldots, \Delta_{2^8-1}$ 直到解密成功，此时她可以推导出 $B = 0\text{x}(b+1) \oplus i$。 我们将**完整描述**如何扩展此攻击以了解**整个明文**——而不仅仅是最后一个块——留作练习。

对 CAPTCHA 的填充预言机攻击。 CAPTCHA 是一个失真的图像，例如，一个英文单词，它对人类来说容易阅读，但对计算机来说难以处理。 CAPTCHA 用于确保正在与网页交互的是**人类用户**——而不是一些**自动化软件**。 CAPTCHA 被使用，例如，在线电子邮件服务用于确保人类开设账户；这对于防止垃圾邮件发送者自动开设数千个账户并使用它们发送垃圾邮件非常重要。

CAPTCHA 可以配置为在独立服务器上运行的单独服务。 为了了解这是如何工作的，我们用 S_W 表示网络服务器，用 S_C 表示 CAPTCHA 服务器，用 U 表示用户。 当用户 U 加载由 S_W 服务的一个网页时，发生以下事件： S_W 使用最初在 S_W 和 S_C 之间共享的密钥 k 加密一个随机英文单词 w，并将得到的密文发送给用户（连同网页）。 U 将密文转发给 S_C， S_C 解密它，获得 w，并向 U 渲染 w 的失真图像。 最后， U 将单词 w 发送回 S_W 进行验证。 这里有趣的观察是 S_C 解密它从 U 收到的任何密文，如果解密失败，它将发出一个“错误填充”错误消息，如前所述。 这为 U 提供了一个机会来执行上述填充预言机攻击，从而**自动解决** CAPTCHA（即，确定 w）而**无需任何人为干预**，使得 CAPTCHA 失效。 虽然可以使用**特设措施**（例如，让 S_C 返回一个随机图像而不是解密错误），但真正需要的是使用一个 **CCA-安全** 的加密方案。
