第 7 章

对称密钥原语的理论构造

在第 3 章中，我们介绍了伪随机性的概念，并定义了一些基本的密码学原语，包括伪随机生成器、函数和排列。我们在第 3 章和第 4 章中表明，这些原语是对称密钥密码学的所有构建模块。因此，从理论角度理解这些原语至关重要。在本章中，我们将正式引入**单向函数**的概念——这些函数非正式地易于计算但难以反转——并展示如何仅在单向函数存在[1]的唯一假设下构造伪随机生成器、函数和排列。此外，我们将看到单向函数对于“非平凡”的对称密钥密码学是必要的。也就是说：**单向函数的存在等价于所有（非平凡）对称密钥密码学的存在**。这是现代密码学的主要贡献之一。

我们在本章中展示的构造应被视为对上一章中讨论的流密码和分组密码构造的补充。上一章的重点是各种密码学原语如何在实践中实现，其目的是介绍一些基本方法和设计原则。然而，令人有些失望的是，我们展示的构造都不能被证明是安全的，其基础是任何较弱（即更合理）的假设。相比之下，在本章中，我们将证明从**单向函数**存在的非常温和的假设开始，可以构造伪随机排列。这个假设比假设 AES 是伪随机排列等更易于接受，原因有二：一是它在性质上是一个更弱的假设，二是我们有许多经过多年研究的候选数论单向函数，甚至在密码学出现之前就已经存在。（有关此点的进一步讨论，请参阅第 6 章的开头部分。）然而，不利的一面是，我们在此处展示的构造都比第 6 章中的构造效率低得多，因此实际上并未被使用。对于密码学家来说，如何“弥合这一差距”并开发可证明安全的伪随机生成器、函数和排列的构造，使其效率与最佳流密码和分组密码相当，仍然是一个重要的挑战。

> [1]严格来说，这不是完全正确的，因为在本章中，我们主要依赖于**单向排列**。但已知单向函数就足够了。

抗碰撞哈希函数

与上一章不同，我们在本章中不考虑抗碰撞哈希函数。原因在于，从单向函数构造此类哈希函数尚不清楚，事实上，有证据表明此类构造是不可能的。我们将在第 8.4.2 节中转向基于特定的数论假设的抗碰撞哈希函数的可证明构造。

关于本章的说明

本章的内容比本书其余部分的内容稍微高级一些。这些材料没有在其他地方明确使用，因此如果需要，可以跳过本章。话虽如此，我们已努力以一种方式呈现材料，使其对高年级本科生或初级

研究生（付出努力后）是可理解的。我们鼓励所有读者通读介绍单向函数并概述本章其余部分的第 7.1 节和第 7.2 节。我们相信，熟悉此处涵盖的至少一些主题是非常重要的，值得付出努力。

7.1 单向函数

在本节中，我们将正式定义单向函数，然后简要讨论一些被广泛认为满足此定义的候选函数。（我们将在第 8 章中看到更多推测的单向函数示例。）接下来，我们将介绍**硬核谓词**的概念，它可以被视为封装了反转单向函数的难度，并将广泛用于后续章节中的构造。

7.1.1 定义

单向函数 $f : \{0,1\}^* \to \{0,1\}^*$ 易于计算，但难以反转。第一个条件很容易形式化：我们只需要求 f 是可在**多项式时间**内计算的。由于我们最终感兴趣的是构建难以被**概率多项式时间敌手**以**可忽略的概率**破解的密码方案，我们将通过要求任何概率多项式时间算法反转 f（即，找到给定值 y 的原像）是**不可行**的，除非具有可忽略的概率，来形式化第二个条件。一个技术要点是，这个概率是在一个实验中计算的，其中 y 是通过从 f 的域中选择一个均匀元素 x 并设置 $y := f(x)$ 来生成的（而不是从 f 的范围中均匀选择 y）。这样做的原因应该从我们将在本章其余部分看到的构造中变得清晰。

令 $f : \{0,1\}^* \to \{0,1\}^*$ 是一个函数。考虑为任何算法 A 和安全参数的任何值 n 定义的以下实验：

反转实验 $\mathrm{Invert}_{A,f}(n)$

1. 选择均匀 $x \in \{0,1\}^n$，并计算 $y := f(x)$。
2. A 被给予 1^n 和 y 作为输入，并输出 x'。
3. 如果 $f(x') = y$，则实验的输出定义为 1，否则为 0。

我们强调 A **不需要**找到原始原像 x；A 找到任何满足 $f(x') = y = f(x)$ 的值 x' 就足够了。我们在第二步中向 A 提供了安全参数 1^n，以强调 A 可以在多项式时间内运行，与 y 的长度无关。

我们现在可以定义函数 f 成为单向函数的含义。

定义 7.1 如果函数 $f : \{0,1\}^* \to \{0,1\}^*$ 满足以下两个条件，则称其为**单向函数**：

1. **（易于计算：）**存在一个多项式时间算法 M_f 计算 f；即，对于所有 x， $M_f(x) = f(x)$。
2. **（难以反转：）**对于每个概率多项式时间算法 A，存在一个**可忽略的函数** negl，使得

$$\Pr[\mathrm{Invert}_{A,f}(n) = 1] \leq \mathrm{negl}(n)$$

记法。 在本章中，我们将经常通过在概率记法中添加下标（部分）来使概率空间更加明确。例如，我们可以简洁地表达上述定义中的第二个要求如下：对于每个概率多项式时间算法 A，存在一个可忽略的函数 negl，使得

$$\Pr_{x \leftarrow \{0,1\}^n}[A(1^n, f(x)) \in f^{-1}(f(x))] \leq \mathrm{negl}(n)$$

（回想一下，$x \leftarrow \{0,1\}^n$ 意味着 x 是从 $\{0,1\}^n$ 中均匀选择的。）上述概率也计算了 A 使用的随机性，此处未明确指出。

单向函数的成功反转。 一个**非**单向函数不一定总是（甚至"经常"）易于反转。相反，定义 7.1 中第二个条件的逆是，存在一个概率多项式时间算法 A 和一个**非可忽略的函数** γ，使得 A 以至少 $\gamma(n)$ 的概率反转 $f(x)$（其中概率是针对 $x \in \{0,1\}^n$ 的均匀选择和 A 的随机性计算的）。反过来，这意味着存在一个正多项式 $p(\cdot)$，使得对于**无限多**个 n 值，算法 A 以至少 $1/p(n)$ 的概率反转 f。因此，如果存在一个 A 算法，对于所有偶数 n 值，它以 n^{-10} 的概率反转 f（但当 n 是奇数时总是无法反转 f），那么 f 就不是单向函数——尽管 A 仅在 n 值的一半上成功，并且仅以 n^{-10} 的概率成功（对于它成功的 n 值）。

指数时间反转。 任何单向函数都可以在**指数时间**内反转任意点 y，只需简单地尝试所有值 $x \in \{0,1\}^n$ 直到找到满足 $f(x) = y$ 的值 x。因此，单向函数的存在本质上是关于**计算复杂度**和**计算难度**的假设。也就是说，它涉及一个原则上可以解决但被假定难以有效解决的问题。

单向排列。 我们经常对具有附加结构属性的单向函数感兴趣。如果对于所有 x 都有 $|f(x)| = |x|$，则称函数 f 是**保持长度**的。保持长度且是一对一的单向函数称为**单向排列**。如果 f 是一个单向排列，那么任何值 y 都有一个唯一的原像 $x = f^{-1}(y)$。然而，在多项式时间内找到 x 仍然很困难。

单向函数/排列族。 上述单向函数和排列的定义很方便，因为它们只考虑一个作用于无限域和范围的函数。然而，大多数候选单向函数和排列并不完全符合这个框架。相反，存在一个算法生成一组参数 I，这些参数定义了一个函数 f_I；这里的单向性本质上意味着 f_I 应该是单向的，除了在选择 I 时具有可忽略的概率。由于 I 的每个值定义了一个不同的函数，我们现在称之为**单向函数族**（或排列族）。我们现在给出定义，并请读者参考下一节中的具体示例。（另请参阅第 8.4.1 节。）

定义 7.2 如果以下条件成立，则称概率多项式时间算法的元组 $\Pi = (\mathrm{Gen}, \mathrm{Samp}, f)$ 是一个**函数族**：

1. **参数生成算法** Gen，输入 1^n，输出参数 I，其中 $|I| \geq n$。Gen 输出的每个 I 值都定义了构成函数 f_I 的域和范围的集合 D_I 和 R_I。
2. **采样算法** Samp，输入 I，输出 D_I 的一个均匀分布元素。
3. **确定性求值算法** f，输入 I 和 $x \in D_I$，输出 R_I 的一个元素 y。我们将其写为 $y := f_I(x)$。

如果对于 $\mathrm{Gen}(1^n)$ 输出的每个 I 值，都有 $D_I = R_I$ 且函数 $f_I : D_I \to D_I$ 是一个双射，则 Π 是一个**排列族**。

令 Π 是一个函数族。下面是先前引入的实验的自然类比。

反转实验 $\mathrm{Invert}_{A,\Pi}(n)$**：**

1. 运行 $\mathrm{Gen}(1^n)$ 以获得 I，然后运行 $\mathrm{Samp}(I)$ 以获得均匀 $x \in D_I$。最后，计算 $y := f_I(x)$。
2. A 被给予 I 和 y 作为输入，并输出 x'。
3. 如果 $f_I(x') = y$，则实验的输出为 1。

定义 7.3 如果对于所有概率多项式时间算法 A，存在一个可忽略的函数 negl，使得

$$\Pr[\mathrm{Invert}_{A,\Pi}(n) = 1] \leq \mathrm{negl}(n)$$

则称函数/排列族 $\Pi = (\mathrm{Gen}, \mathrm{Samp}, f)$ 是**单向的**。

在本章中，我们使用作用于无限域的单向函数/排列（如定义 7.1 中所示），而不是使用单向函数/排列族。这主要是为了方便，并且不会显著影响任何结果。（参见练习 7.7。）

7.1.2 候选单向函数

单向函数只有在存在时才具有意义。我们不知道如何**无条件地**证明它们存在（这将是计算复杂性理论的重大突破），所以我们必须**推测**或**假设**它们的存在。这种推测是基于这样一个事实：几个自然的计算问题受到了广泛关注，但仍然没有多项式时间算法来解决它们。也许最著名的此类问题是**整数分解**，即找到一个大整数的素因数。很容易将两个数相乘并获得它们的积，但很难将一个数分解并找到它的因数。这导致我们定义函数 $f_{\mathrm{mult}}(x, y) = x \cdot y$。如果我们不对 x 和 y 的长度施加任何限制，那么 f_{mult} 就很容易反转：以高概率 $x \cdot y$ 将是偶数，在这种情况下 $(2, xy/2)$ 是一个逆。这个问题可以通过将 f_{mult} 的域限制为等长素数 x 和 y 来解决。我们将

在第 8.2 节中回到这个想法。

另一个不直接依赖于数论的候选单向函数是基于**子集和问题**，定义如下：

$$f_{\text{ss}}(x_1, \ldots, x_n, J) = \left(x_1, \ldots, x_n, \left[\sum_{j \in J} x_j \bmod 2^n \right] \right),$$

其中每个 x_i 是一个解释为整数的 n 位字符串，而 J 是一个解释为指定 $\{1, \ldots, n\}$ 子集的 n 位字符串。反转 f_{ss} 在输出 $(x_1, \ldots, x_n, y)$ 上需要找到一个子集 $J' \subseteq \{1, \ldots, n\}$，使得 $\sum_{j \in J'} x_j = y \bmod 2^n$。学过 NP 完备性的学生可能会回想起这个问题是 **NP 完备**的。但即使 $P \neq NP$ 也不意味着 f_{ss} 是单向的：$P \neq NP$ 将意味着每个多项式时间算法都无法解决至少一个输入上的子集和问题，而对于 f_{ss} 要成为一个单向函数，则要求每个多项式时间算法**几乎总是**无法解决子集和问题（至少对于某些参数）。因此，我们相信上述函数是单向的，是基于这样一个事实：没有已知的算法能够在随机输入上以“小”概率解决这个问题，而不仅仅是基于这个问题是 NP 完备的。

我们最后展示一个被认为是单向的排列族。令 Gen 是一个概率多项式时间算法，输入 1^n，输出一个 n 位素数 p 以及一个特殊元素 $g \in \{2, \ldots, p-1\}$。（元素 g 应该是 $\mathbb{Z}_p^*$ 的一个生成元；参见第 8.3.3 节。）令 Samp 是一个算法，给定 p 和 g，输出一个均匀整数 $x \in \{1, \ldots, p-1\}$。最后，定义

$$f_{p,g}(x) = [g^x \bmod p]$$

（$f_{p,g}$ 可以高效计算的事实源于附录 B.2.3 中的结果。）可以证明这个函数是一对一的，因此是一个排列。反转这个函数的假定困难性是基于**离散对数问题**的推测困难性；我们将在第 8.3 节中对此进行更多讨论。

最后，我们指出，可以从实用的密码学构造（如 SHA-1 或 AES）中获得非常高效的单向函数，前提是它们分别具有抗碰撞性或伪随机排列性；参见练习 7.4 和 7.5。（从技术上讲，它们不能满足单向性的定义，因为它们具有固定长度的输入/输出，因此我们不能观察它们的渐近行为。然而，推测它们在具体意义上是单向的是合理的。）

7.1.3 硬核谓词

根据定义，单向函数难以反转。换句话说：给定 $y = f(x)$，任何多项式时间算法都不能（除非以可忽略的概率；我们在此忽略这一点）完全计算出值 x。人们可能会产生这样一种印象：在多项式时间内无法从 $f(x)$ 中确定关于 x 的任何信息。情况不一定如此。事实上，即使 f 是单

向的， $f(x)$ 仍然可能“泄露”关于 x 的大量信息。举一个平凡的例子，令 g 是一个单向函数，并定义 $f(x_1, x_2) \stackrel{\text{def}}{=} (x_1, g(x_2))$，其中 $|x_1| = |x_2|$。很容易证明 f 也是一个单向函数（这留作练习），尽管它泄露了其输入的一半。

对于我们的应用，我们需要识别关于 x 的特定信息，这些信息被 $f(x)$“隐藏”。这激发了**硬核谓词**的概念。函数 f 的硬核谓词 $hc : \{0,1\}^* \to \{0,1\}$ 具有这样的性质：给定 $f(x)$，计算 $hc(x)$ 的概率显著好于 $1/2$ 是困难的。（由于 hc 是一个布尔函数，因此通过随机猜测总是可以以 $1/2$ 的概率计算 $hc(x)$。）形式上：

定义 7.4 如果函数 $hc : \{0,1\}^* \to \{0,1\}$ 可以在多项式时间内计算，并且对于每个概率多项式时间算法 A，存在一个可忽略的函数 negl，使得

$$\Pr_{x \leftarrow \{0,1\}^n}[A(1^n, f(x)) = hc(x)] \leq \frac{1}{2} + \text{negl}(n),$$

其中概率是针对 $\{0,1\}^n$ 中 x 的均匀选择和 A 的随机性计算的，则称 hc 是函数 f 的**硬核谓词**。

我们强调 $hc(x)$ 是给定 x 时可有效计算的（因为 hc 可以在多项式时间内计算）；定义要求 $hc(x)$ 在给定 $f(x)$ 时难以计算。上述定义不要求 f 是单向的；然而，如果 f 是一个排列，那么除非 f 是单向的，否则它不能具有硬核谓词。（参见练习 7.13。）

简单的想法不起作用。 考虑谓词 $hc(x) \stackrel{\text{def}}{=} \bigoplus_{i=1}^n x_i$，其中 $x_1, \ldots, x_n$ 表示 x 的位。人们可能希望这是任何单向函数 f 的硬核谓词：如果 f 不能被反转，那么 $f(x)$ 必须隐藏其原像 x 的至少一位 x_i，这似乎意味着 x 的所有位的异或难以计算。尽管这个论点很有吸引力，但它是**不正确**的。要看到这一点，令 g 是一个单向函数，并定义 $f(x) \stackrel{\text{def}}{=} (g(x), \bigoplus_{i=1}^n x_i)$。不难证明 f 是单向的。然而，很明显 $f(x)$ 并没有隐藏 $hc(x) = \bigoplus_{i=1}^n x_i$ 的值，因为这是其输出的一部分；因此，$hc(x)$ 不是 f 的硬核谓词。扩展这个想法，可以证明对于任何固定的谓词 hc，存在一个单向函数 f 使 hc 不是 f 的硬核谓词。

平凡的硬核谓词。 一些函数具有“平凡”的硬核谓词。例如，令 f 是一个丢弃其输入最后一位的函数（即，$f(x_1 \ldots x_n) = x_1 \ldots x_{n-1}$）。给定 $f(x)$ 很难确定 x_n，因为 x_n 与输出无关；因此，$hc(x) = x_n$ 是 f 的硬核谓词。然而，f 不是单向的。当我们使用硬核谓词进行构造时，将清楚地看到这种平凡的硬核谓词是毫无用处的。

7.2 从单向函数到伪随机性

本章的目标是展示如何基于任何单向函数/排列构造伪随机生成器、函数和排列。在本节中，我们概述这些构造。细节将在后续章节中给出。

来自任何单向函数的硬核谓词。 第一步是证明对于任何单向函数都存在一个硬核谓词。实际上，这是否属实仍然是开放的；我们展示了一个更弱但足以满足我们目标的结论。即，我们证明给定一个单向函数 f，我们可以构造一个不同的单向函数 g 以及 g 的硬核谓词 hc。

定理 7.5 (Goldreich-Levin 定理) 假设单向函数（或排列）存在。那么存在一个单向函数（或排列）g 和 g 的硬核谓词 hc。

令 f 是一个单向函数。函数 g 和 hc 构造如下：设置 $g(x,r) \stackrel{\text{def}}{=} (f(x),r)$，其中 $|x| = |r|$，并定义

$$hc(x,r) \stackrel{\text{def}}{=} \bigoplus_{i=1}^{n} x_i r_i,$$

其中 x_i（或 r_i）表示 x（或 r）的第 i 位。请注意，如果 r 是均匀的，那么 $hc(x,r)$ 输出 x 的随机子集位的异或。（当 $r_i = 1$ 时，位 x_i 被包含在异或中，否则不包含。）Goldreich-Levin 定理本质上指出，如果 f 是一个单向函数，那么 $f(x)$ 会隐藏 x 的随机子集位的异或。

来自单向排列的伪随机生成器。 下一步是展示如何使用单向排列的硬核谓词来构造伪随机生成器。（已知单向函数的硬核谓词就足够了，但证明极其复杂，超出了本书的范围。）具体来说，我们展示：

定理 7.6 令 f 是一个单向排列，令 hc 是 f 的硬核谓词。那么，$G(s) \stackrel{\text{def}}{=} f(s)||hc(s)$ 是一个扩展因子 $l(n) = n+1$ 的伪随机生成器。

关于定理中定义的 G 为何构成伪随机生成器的直觉是：首先，请注意，当 s 均匀分布时，由于 f 是一个排列， $G(s)$ 输出的初始 n 位（即 $f(s)$ 的位）是**真正均匀分布**的。其次， hc 是 f 的硬核谓词这一事实意味着 $hc(s)$ “看起来是随机的”——即，它是**伪随机**的——即使给定 $f(s)$（再次假设 s 是均匀的）。将这些观察结果放在一起，我们看到 G 的整个输出是伪随机的。

具有任意扩展的伪随机生成器。 存在一个将其种子扩展哪怕一位（如我们刚刚看到的）的伪随机生成器，这已经是高度非平凡的。但对于应用（例如，如第 3.3 节中高效加密大型消息），我们需要具有更大扩展的伪随机生成器。幸运的是，我们可以获得我们想要的任何多项式扩展因子：

定理 7.7 如果存在一个扩展因子 $l(n) = n + 1$ 的伪随机生成器，那么对于任何多项式 poly，存在一个扩展因子 $\mathrm{poly}(n)$ 的伪随机生成器。

我们得出结论：具有任意（多项式）扩展的伪随机生成器可以由任何单向排列构造。

来自伪随机生成器的伪随机函数/排列。 伪随机生成器足以构造 EAV 安全的对称密钥加密方案。然而，为了实现 CPA 安全的对称密钥加密（更不用说消息认证码），我们依赖于伪随机函数。以下结果表明后者可以由前者构造：

定理 7.8 如果存在一个扩展因子 $l(n) = 2n$ 的伪随机生成器，那么存在一个伪随机函数。

事实上，我们可以做得更多：

定理 7.9 如果存在一个伪随机函数，那么存在一个**强伪随机排列**。

结合上述所有定理，以及第 3 章和第 4 章的结果，我们得到以下推论：

推论 7.10 假设单向排列存在，那么存在具有任何多项式扩展因子的伪随机生成器、伪随机函数和强伪随机排列。

推论 7.11 假设单向排列存在，那么存在 CCA 安全的对称密钥加密方案和安全的消息认证码。

如前所述，仅基于单向函数的存在就可以获得所有这些结果。

7.3 从单向函数到硬核谓词

在本节中，我们通过证明以下内容来证明定理 7.5：

定理 7.12 令 f 是一个单向函数，并通过 $g(x, r) \stackrel{\text{def}}{=} (f(x), r)$ 定义函数 g，其中 $|x| = |r|$。定义 $gl(x, r) \stackrel{\text{def}}{=} \bigoplus_{i=1}^{n} x_i r_i$，其中 $x = x_1 \dots x_n$ 和 $r = r_1 \dots r_n$。那么 gl 是 g 的硬核谓词。

由于证明的复杂性，我们证明了三个逐步增强的结果，最终得出定理中声称的内容。

7.3.1 一个简单案例

我们首先证明，如果存在一个多项式时间敌手 A，它在给定 $g(x, r) = (f(x), r)$ 时**总是**正确计算 $gl(x, r)$，那么就可以在多项式时间内反转 f。给定 f 是单向函数的假设，可以得出不存在这样的敌手 A。

命题 7.13 令 f 和 gl 如定理 7.12 中所定义。如果存在一个多项式时间算法 A 使得对于所有 n 和所有 $x, r \in \{0,1\}^n$， $A(f(x), r) = gl(x, r)$，那么存在一个多项式时间算法 A' 使得对于所有 n 和所有 $x \in \{0,1\}^n$， $A'(1^n, f(x)) = x$。

证明 我们构造 A' 如下。$A'(1^n, y)$ 计算 $x_i := A(y, e^i)$，其中 e^i 是一个 n 位字符串，在第 i 个位置为 1，其他地方为 0。然后 A' 输出 $\hat{x} = \hat{x}_1 \dots \hat{x}_n$。显然 A' 在多项式时间内运行。

在 $A'(1^n, f(x))$ 的执行中， A' 计算的值 $\hat{x}_i$ 满足

$$\hat{x}_i = A(f(x), e^i) = gl(x, e^i) = \bigoplus_{j=1}^{n} x_j e^i_j = x_i.$$

因此，对于所有 i 都有 $\hat{x}_i = x_i$，所以 A' 输出正确的逆 $x = \hat{x}$。

如果 f 是单向的，那么任何概率多项式时间算法都不可能以**非可忽略的概率**反转 f。因此，我们得出结论，不存在一个**总是**从 $(f(x), r)$ 正确计算 $gl(x, r)$ 的多项式时间算法。这是一个相当弱的结果，距离我们最终的目标（证明给定 $(f(x), r)$ 时， $gl(x, r)$ 不能以**显著好于 $1/2$ 的概率**计算）还很远。

7.3.2 一个更复杂的情况

我们现在证明，任何概率多项式时间算法 A 很难以显著好于 $3/4$ 的概率从 $(f(x), r)$ 计算 $gl(x, r)$。我们将再次证明，任何这样的 A 将意味着存在一个以**非可忽略的概率**反转 f 的多项式时间算法 A'。请注意，命题 7.13 证明中的策略在这里失败了，因为它可能是 A 在 $r = e^i$ 时从不成功（尽管它可能在所有其他 r 值上成功）。此外，在当前情况下， A' 不知道结果 $A(f(x), r)$ 是否等于 $gl(x, r)$； A' 唯一知道的是，以高概率，算法 A 是正确的。这进一步使证明复杂化。

命题 7.14 令 f 和 gl 如定理 7.12 中所定义。如果存在一个概率多项式时间算法 A 和一个多项式 $p(\cdot)$，使得

$$\Pr_{x,r \leftarrow \{0,1\}^n}[A(f(x), r) = gl(x, r)] \geq \frac{3}{4} + \frac{1}{p(n)}$$

对于**无限多**个 n 值成立，那么存在一个概率多项式时间算法 A'，使得

$$\Pr_{x \leftarrow \{0,1\}^n}[A'(1^n, f(x)) \in f^{-1}(f(x))] \geq \frac{1}{4 \cdot p(n)}$$

对于**无限多**个 n 值成立。

证明 这个命题证明的关键观察是：对于每个 $r \in \{0,1\}^n$， $gl(x, r \oplus e^i)$ 和 $gl(x, r)$ 的值可以一起用于计算 x 的第 i 位。（回想一下，e^i 表示一个 n 位字符串，除第 i 个位置为 1 外，其他地方都为 0。）这是因为

$$gl(x, r) \oplus gl(x, r \oplus e^i) = \left(\bigoplus_{j=1}^{n} x_j r_j \right) \oplus \left(\bigoplus_{j=1}^{n} x_j (r_j \oplus e^i_j) \right) = \bigoplus_{j=1}^{n} x_j (r_j \oplus (r_j \oplus e^i_j)) = x_i$$

其中 $r_j \oplus (r_j \oplus e^i_j)$ 是 r_j 的补码，第二个等式成立是因为对于 $j \neq i$，值 $x_j r_j$ 出现在两个和中，因此被抵消了。

上述内容表明，如果 A 对 $(f(x), r)$ 和 $(f(x), r \oplus e^i)$ 都回答正确，那么 A' 可以正确计算 x_i。不幸的是， A' 不知道 A 何时回答正确，何时回答不正确； A' 只知道算法 A 以“高”概率是正确的。因此， A' 将使用多个随机值 r，使用每个 r 来获得 x_i 的估计值，然后将出现次数最多的估计值作为对 x_i 的最终猜测。

作为一个初步步骤，我们证明对于许多 x 值，当 r 是均匀时， A 对 $(f(x), r)$ 和 $(f(x), r \oplus e^i)$ 都回答正确的概率足够高。这允许我们固定 x，然后只关注 r 的均匀选择，这使得分析更容易。

断言 7.15 令 n 使得

$$\Pr_{x, r \leftarrow \{0,1\}^n}[A(f(x), r) = gl(x, r)] \geq \frac{3}{4} + \frac{1}{p(n)}.$$

那么存在一个集合 $S_n \subseteq \{0,1\}^n$，其大小至少为 $\frac{1}{2p(n)} \cdot 2^n$，使得对于每个 $x \in S_n$，成立

$$\Pr_{r \leftarrow \{0,1\}^n}[A(f(x), r) = gl(x, r)] \geq \frac{3}{4} + \frac{1}{2p(n)}.$$

证明 令 $\varepsilon(n) = 1/p(n)$，并定义 $S_n \subseteq \{0,1\}^n$ 为所有 x 的集合，使得

$$\Pr_{r \leftarrow \{0,1\}^n}[A(f(x), r) = gl(x, r)] \geq \frac{3}{4} + \frac{\varepsilon(n)}{2}.$$

我们有

$$\begin{aligned}\Pr_{x,r\leftarrow\{0,1\}^n}[A(f(x),r)=gl(x,r)] &= \frac{1}{2^n}\sum_{x\in\{0,1\}^n}\Pr_{r\leftarrow\{0,1\}^n}[A(f(x),r)=gl(x,r)] \\ &= \frac{1}{2^n}\sum_{x\in S_n}\Pr_{r\leftarrow\{0,1\}^n}[A(f(x),r)=gl(x,r)] \\ &\quad + \frac{1}{2^n}\sum_{x\notin S_n}\Pr_{r\leftarrow\{0,1\}^n}[A(f(x),r)=gl(x,r)] \\ &\le \frac{|S_n|}{2^n}\cdot 1 + \frac{2^n-|S_n|}{2^n}\cdot\left(\frac{3}{4}+\frac{\varepsilon(n)}{2}\right) \\ &= \frac{|S_n|}{2^n}\cdot\left(\frac{1}{4}-\frac{\varepsilon(n)}{2}\right)+\left(\frac{3}{4}+\frac{\varepsilon(n)}{2}\right).\end{aligned}$$

由于 $\frac{3}{4}+\varepsilon(n)\le \Pr_{x,r\leftarrow\{0,1\}^n}[A(f(x),r)=gl(x,r)]$，直接代数运算给出 $|S_n|\ge \frac{\varepsilon(n)}{2}\cdot 2^n$。

下面的内容是紧随其后的一个简单推论。

断言 7.16 令 n 使得

$$\Pr_{x,r\leftarrow\{0,1\}^n}[A(f(x),r)=gl(x,r)]\ge \frac{3}{4}+\frac{1}{p(n)}.$$

那么存在一个集合 $S_n\subseteq\{0,1\}^n$，其大小至少为 $\frac{1}{2p(n)}\cdot 2^n$，使得对于每个 $x\in S_n$ 和每个 i，成立

$$\Pr_{r\leftarrow\{0,1\}^n}[A(f(x),r)=gl(x,r)\wedge A(f(x),r\oplus e^i)=gl(x,r\oplus e^i)]\ge \frac{1}{2}+\frac{1}{p(n)}.$$

证明 令 $\varepsilon(n)=1/p(n)$，并取 S_n 为前一个断言保证的集合。我们知道对于任何 $x\in S_n$，我们有

$$\Pr_{r\leftarrow\{0,1\}^n}[A(f(x),r)\ne gl(x,r)]\le 1-\left(\frac{3}{4}+\frac{\varepsilon(n)}{2}\right)=\frac{1}{4}-\frac{\varepsilon(n)}{2}.$$

固定 $i\in\{1,\dots,n\}$。如果 r 是均匀分布的，那么 $r\oplus e^i$ 也是；因此

$$\Pr_{r\leftarrow\{0,1\}^n}[A(f(x),r\oplus e^i)\ne gl(x,r\oplus e^i)]\le \frac{1}{4}-\frac{\varepsilon(n)}{2}.$$

我们感兴趣于下界 A 输出 $gl(x,r)$ 和 $gl(x,r\oplus e^i)$ 两者正确答案的概率；等价地，我们想上

界 A 在这两种情况中**任一**失败的概率。请注意，r 和 $r \oplus e^i$ 不是独立的，因此我们不能简单地将失败的概率相乘。然而，我们可以应用并集界（参见命题 A.7）并求和失败的概率。也就是说， A 在 $gl(x, r)$ 或 $gl(x, r \oplus e^i)$ 上不正确的概率至多为

$$\left(\frac{1}{4} - \frac{\varepsilon(n)}{2}\right) + \left(\frac{1}{4} - \frac{\varepsilon(n)}{2}\right) = \frac{1}{2} - \varepsilon(n),$$

因此 A 在 $gl(x, r)$ 和 $gl(x, r \oplus e^i)$ 上都正确的概率至少为 $1 - (\frac{1}{2} - \varepsilon(n)) = \frac{1}{2} + \varepsilon(n)$。这证明了断言。

对于证明的其余部分，我们设置 $\varepsilon(n) = 1/p(n)$，并且只考虑满足以下条件的 n 值：

$$\Pr_{x,r \leftarrow \{0,1\}^n}[A(f(x), r) = gl(x, r)] \geq \frac{3}{4} + \varepsilon(n). \tag{7.1}$$

前一个断言表明，对于 $\varepsilon(n)/2$ 分数的输入 x，以及任何 i，算法 A 在 r 的均匀选择上，以至少 $1/2 + \varepsilon(n)$ 的概率对 $(f(x), r)$ 和 $(f(x), r \oplus e^i)$ 都回答正确，从现在开始，我们只关注这些 x 值。我们构造一个概率多项式时间算法 A'，它以至少 $1/2$ 的概率反转 $f(x)$，当 $x \in S_n$ 时。这足以证明命题 7.14，因为这样，对于无限多 n 值，

$$\begin{aligned}\Pr_{x \leftarrow \{0,1\}^n}[A'(1^n, f(x)) \in f^{-1}(f(x))] &\geq \Pr_{x \leftarrow \{0,1\}^n}[A'(1^n, f(x)) \in f^{-1}(f(x)) \mid x \in S_n] \cdot \Pr_{x \leftarrow \{0,1\}^n}[x \\ &\geq \frac{1}{2} \cdot \frac{\varepsilon(n)}{2} = \frac{1}{4p(n)}.\end{aligned}$$

算法 A'，给定输入 1^n 和 y，工作如下：

1. 对于 $i = 1, \ldots, n$ 执行：
   - 重复选择均匀 $r \in \{0,1\}^n$，并计算 $A(y, r) \oplus A(y, r \oplus e^i)$ 作为 y 的原像的第 i 位的“估计”。在进行足够多次（见下文）后，令 $\hat{x}_i$ 是出现次数最多的“估计”。
2. 输出 $\hat{x} = \hat{x}_1 \ldots \hat{x}_n$。

我们概述 A' 正确反转给定输入 y 的概率分析。（我们允许自己稍微简略，因为在下一节中给出了一个更困难案例的完整证明。）假设 $y = f(x)$，并回想我们假设 n 满足等式 (7.1) 且 $x \in S_n$。固定某个 i。前一个断言意味着估计 $A(y, r) \oplus A(y, r \oplus e^i)$ 等于 $gl(x, e^i) = x_i$ 的概率至少为 $1/2 + \varepsilon(n)$，取决于 r 的选择。通过获得足够多的估计值并令 $\hat{x}_i$ 是多数值，A' 可以确保 $\hat{x}_i$ 等于 x_i 的概率至少为 $1 - 1/(2n)$。当然，我们需要确保多项式数量的估计值就足够了。幸运的是，由于 $\varepsilon(n) = 1/p(n)$ 对于某个多项式 p，并且为获得每个估计值都使用了一个独立的 r 值，Chernoff 界（参见命题 A.14）表明多项式数量的估计值就足够了。

总结一下，我们有对于每个 i， A' 计算的值 $\hat{x}_i$ 不正确的概率至多为 $1/(2n)$。因此，并集界表明 A' 对于某些 i 不正确的概率至多为 $n \cdot 1/(2n) = 1/2$。也就是说， A' 对于所有 i 都是正确的——因此正确反转 y——的概率至少为 $1 - 1/2 = 1/2$。这就完成了命题 7.14 的证明。

命题 7.14 的一个推论是，如果 f 是一个单向函数，那么对于任何多项式时间算法 A，在给定 $(f(x), r)$ 时 A 正确猜测 $gl(x, r)$ 的概率至多是 $3/4$ 加上一个可忽略的量。

7.3.3 完整的证明

我们假设读者熟悉前面章节中的简化证明，并在此基础上建立。我们依赖于附录 A.3 中讨论的一些概率论术语和标准结果。

我们证明以下命题，它蕴含了定理 7.12：

命题 7.17 令 f 和 gl 如定理 7.12 中所定义。如果存在一个概率多项式时间算法 A 和一个多项式 $p(\cdot)$，使得

$$\Pr_{x,r \leftarrow \{0,1\}^n}[A(f(x), r) = gl(x, r)] \geq \frac{1}{2} + \frac{1}{p(n)}$$

对于**无限多**个 n 值成立，那么存在一个概率多项式时间算法 A' 和一个多项式 $p'(\cdot)$，使得

$$\Pr_{x \leftarrow \{0,1\}^n}[A'(1^n, f(x)) \in f^{-1}(f(x))] \geq \frac{1}{p'(n)}$$

对于**无限多**个 n 值成立。

证明 我们再次设置 $\varepsilon(n) = 1/p(n)$，并且只考虑满足 $\Pr_{x,r \leftarrow \{0,1\}^n}[A(f(x), r) = gl(x, r)] \geq 1/2 + \varepsilon(n)$ 的 n 值。以下内容类似于断言 7.15，并以相同的方式证明。

断言 7.18 令 n 使得

$$\Pr_{x,r \leftarrow \{0,1\}^n}[A(f(x), r) = gl(x, r)] \geq \frac{1}{2} + \varepsilon(n).$$

那么存在一个集合 $S_n \subseteq \{0,1\}^n$，其大小至少为 $\frac{\varepsilon(n)}{2} \cdot 2^n$，使得对于每个 $x \in S_n$，成立

$$\Pr_{r \leftarrow \{0,1\}^n}[A(f(x), r) = gl(x, r)] \geq \frac{1}{2} + \frac{\varepsilon(n)}{2}. \tag{7.2}$$

如果我们尝试证明一个类似于断言 7.16 的结论，我们在这里能声称的最佳结果是，当 $x \in S_n$ 时，我们有

$$\Pr_{r \leftarrow \{0,1\}^n}[A(f(x), r) = gl(x, r) \wedge A(f(x), r \oplus e^i) = gl(x, r \oplus e^i)] \geq \varepsilon(n)$$

对于任何 i。因此，如果我们尝试使用 $A(f(x), r) \oplus A(f(x), r \oplus e^i)$ 作为 x_i 的估计，我们能声称的最佳结果是这个估计正确的概率至少为 $\varepsilon(n)$，这可能不比随机猜测好！我们也无法声称翻转结果会给出一个好的估计。

相反，我们设计 A' 使得它只调用 A 一次来计算 $gl(x, r)$ 和 $gl(x, r \oplus e^i)$。我们通过让 A' 运行 $A(x, r \oplus e^i)$，并让 A' 简单地"猜测" $gl(x, r)$ 的值。像以前一样，选择独立的 r 值，并让 A' 对每个 r 值独立猜测 $gl(x, r)$ 的天真方法，然后所有这些猜测都正确的概率（正如我们将看到的，如果 A' 要输出正确的逆，这是必要的）将是可忽略的，因为使用了多项式数量的 r 。

当前证明的关键观察是， A' 可以以**两两独立**的方式生成 r 值，并以特定的方式进行猜测，使得所有猜测都以**非可忽略的概率**正确。具体来说，为了生成 m 个 r 值， A' 选择 $l = \lceil \log(m+1) \rceil$ 个独立且均匀分布的字符串 $s^1, \ldots, s^l \in \{0,1\}^n$。然后，对于每个非空子集 $I \subseteq \{1, \ldots, l\}$，我们设置 $r^I := \bigoplus_{i \in I} s^i$。由于有 $2^l - 1$ 个非空子集，这定义了一个包含 $2^{\lceil \log(m+1) \rceil} - 1 \geq m$ 个字符串的集合。每个这样的字符串都是均匀分布的。这些字符串不是独立的，但它们是**两两独立**的。要看到这一点，请注意对于任意两个子集 $I \neq J$，存在一个索引 $j \in I \cup J$ 使得 $j \notin I \cap J$。不失一般性，假设 $j \notin I$。那么 s^j 的值是均匀的，并且独立于 r^I 的值。由于 s^j 包含在定义 r^J 的异或中，这意味着 r^J 也是均匀的，并且独立于 r^I。

我们现在有两个重要的观察结果：

1. 给定 $gl(x, s^1), \ldots, gl(x, s^l)$，可以计算每个子集 $I \subseteq \{1, \ldots, l\}$ 的 $gl(x, r^I)$。这是因为

$$gl(x, r^I) = gl(x, \bigoplus_{i \in I} s^i) = \bigoplus_{i \in I} gl(x, s^i).$$

2. 如果 A' 简单地通过为每个值选择一个均匀位来猜测 $gl(x, s^1), \ldots, gl(x, s^l)$ 的值，那么所有这些猜测都将以 $1/2^l$ 的概率正确。如果 m 在安全参数 n 中是多项式的，那么 $1/2^l$ 是**非可忽略的**，因此以**非可忽略的概率**， A' 正确猜测了所有值 $gl(x, s^1), \ldots, gl(x, s^l)$ 。

结合上述观察结果，我们得到了一个以**非可忽略的概率**获得 $m = \text{poly}(n)$ 个均匀且两两独立

字符串 $\{r^I\}$ 以及 $\{gl(x, r^I)\}$ 的正确值的方法。然后可以以与命题 7.14 证明中相同的方式使用这些值来计算 x_i。细节如下。

反转算法 A'。 我们现在提供算法 A' 的完整描述，该算法接收输入 $1^n, y$ 并尝试计算 y 的逆。算法执行如下：

1. 设置 $l := \lceil \log(2n/\varepsilon(n)^2 + 1) \rceil$。
2. 选择均匀、独立的 $s^1, \ldots, s^l \in \{0,1\}^n$ 和 $\sigma^1, \ldots, \sigma^l \in \{0,1\}$。
3. 对于每个非空子集 $I \subseteq \{1, \ldots, l\}$，计算 $r^I := \bigoplus_{i \in I} s^i$ 和 $\sigma^I := \bigoplus_{i \in I} \sigma^i$。
4. 对于 $i = 1, \ldots, n$ 执行：
   (a) 对于每个非空子集 $I \subseteq \{1, \ldots, l\}$，设置
   $$x_i^I := \sigma^I \oplus A(y, r^I \oplus e^i).$$
   (b) 设置 $\hat{x}_i := \mathrm{majority}_I\{x_i^I\}$（即，取在上一步中出现次数占多数的位）。
5. 输出 $\hat{x} = \hat{x}_1 \ldots \hat{x}_n$。

剩下的就是计算 A' 输出 $x \in f^{-1}(y)$ 的概率。与命题 7.14 的证明一样，我们只关注断言 7.18 中的 n，并假设 $y = f(x)$ 对于某个 $x \in S_n$。每个 σ^i 代表对 $gl(x, s^i)$ 值的“猜测”。如前所述，以非可忽略的概率所有这些猜测都是正确的；我们证明，在这个事件发生的情况下，A' 以至少 $1/2$ 的概率输出 $\hat{x} = x$。

假设 $\sigma^i = gl(x, s^i)$ 对于所有 i。那么 $\sigma^I = gl(x, r^I)$ 对于所有 I。固定一个索引 $i \in \{1, \ldots, n\}$，并考虑 A' 获得正确值 $\hat{x}_i = x_i$ 的概率。对于任何非空 I，我们有 $A(y, r^I \oplus e^i) = gl(x, r^I \oplus e^i)$ 的概率至少为 $1/2 + \varepsilon(n)/2$，取决于 r^I 的选择；这成立是因为 $x \in S_n$ 且 $r^I \oplus e^i$ 是均匀分布的。因此，对于任何非空子集 I，我们有 $\Pr[x_i^I = x_i] \geq 1/2 + \varepsilon(n)/2$。此外，$\{x_i^I\}_{I \subseteq \{1,\ldots,l\}}$ 是**两两独立**的，因为 $\{r^I\}_{I \subseteq \{1,\ldots,l\}}$（以及因此 $\{r^I \oplus e^i\}_{I \subseteq \{1,\ldots,l\}}$）是两两独立的。由于 $\hat{x}_i$ 被定义为在 $\{x_i^I\}_{I \subseteq \{1,\ldots,l\}}$ 中出现次数占多数的值，我们可以应用命题 A.13 来获得

$$\Pr[\hat{x}_i \neq x_i] < \frac{1}{4 \cdot (\varepsilon(n)/2)^2 \cdot (2^l - 1)} \leq \frac{1}{4 \cdot (\varepsilon(n)/2)^2 \cdot (2n/\varepsilon(n)^2)} = \frac{1}{2n}.$$

上述内容对于所有 i 都成立，因此通过应用并集界，我们看到 $\hat{x}_i \neq x_i$ 对于某些 i 的概率至多为 $1/2$。也就是说，$\hat{x}_i = x_i$ 对于所有 i（因此 $\hat{x} = x$）的概率至少为 $1/2$。

把所有东西放在一起：令 n 如断言 7.18 中所述，且 $y = f(x)$。以至少 $\varepsilon(n)/2$ 的概率，我们有 $x \in S_n$。所有猜测 σ^i 都是正确的概率至少为

$$\frac{1}{2^l} \geq \frac{1}{2 \cdot (2n/\varepsilon(n)^2 + 1)} > \frac{\varepsilon(n)^2}{5n}$$

对于足够大的 n。在上述两种情况都成立的条件下， A' 以至少 $1/2$ 的概率输出 $\hat{x} = x$。因此， A' 反转其输入的总概率至少为 $\varepsilon(n)^3/(20n) = 1/(20np(n)^3)$ 对于无限多 n 值。由于 $20np(n)^3$ 在 n 中是多项式的，这证明了命题 7.17。

7.4 构造伪随机生成器

我们首先展示如何构造将输入扩展一位的伪随机生成器，假设单向排列存在。然后我们展示如何将其扩展到获得任何多项式扩展因子。

7.4.1 具有最小扩展的伪随机生成器

令 f 是一个具有硬核谓词 hc 的单向排列。这意味着给定 $f(s)$ 时， $hc(s)$ “看起来是随机的”，当 s 是均匀时。此外，由于 f 是一个排列， $f(s)$ 本身是均匀分布的。（将排列应用于均匀分布的值会产生均匀分布的值。）因此，如果 s 是一个均匀的 n 位字符串，那么 $(n+1)$ 位字符串 $f(s)||hc(s)$ 由一个均匀的 n 位字符串加上一位附加位组成，该附加位即使以初始 n 位为条件也看起来是均匀的；换句话说，这个 $(n+1)$ 位字符串是伪随机的。因此，由 $G(s) = f(s)||hc(s)$ 定义的算法是一个伪随机生成器。

定理 7.19 令 f 是一个具有硬核谓词 hc 的单向排列。那么由 $G(s) = f(s)||hc(s)$ 定义的算法 G 是一个扩展因子 $l(n) = n + 1$ 的伪随机生成器。

证明 令 D 是一个概率多项式时间算法。我们证明存在一个可忽略的函数 negl，使得

$$\Pr_{r \leftarrow \{0,1\}^{n+1}}[D(r) = 1] - \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 1] \leq \mathrm{negl}(n). \tag{7.3}$$

类似的论证表明存在一个可忽略的函数 negl'，使得

$$\Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{n+1}}[D(r) = 1] \leq \mathrm{negl}'(n),$$

这就完成了证明。

首先观察到

$$\begin{aligned}\Pr_{r\leftarrow\{0,1\}^{n+1}}[D(r)=1] &= \Pr_{r\leftarrow\{0,1\}^n,r'\leftarrow\{0,1\}}[D(r||r')=1]\\ &= \Pr_{s\leftarrow\{0,1\}^n,r'\leftarrow\{0,1\}}[D(f(s)||r')=1]\\ &= \frac{1}{2}\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||hc(s))=1]+\frac{1}{2}\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||\overline{hc(s)})=1],\end{aligned}$$

其中第二个等式使用了 f 是一个排列的事实，第三个等式使用了均匀位 r' 以恰好 $1/2$ 的概率等于 $hc(s)$ 的事实。

由于

$$\Pr_{s\leftarrow\{0,1\}^n}[D(G(s))=1]=\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||hc(s))=1]$$

（根据 G 的定义），这意味着等式 (7.3) 等价于

$$\frac{1}{2}\left(\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||\overline{hc(s)})=1]-\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||hc(s))=1]\right)\le \mathrm{negl}(n).$$

考虑以下算法 A，它以值 $y=f(s)$ 为输入，并尝试预测 $hc(s)$ 的值：

1. 选择均匀 $r'\in\{0,1\}$。
2. 运行 $D(y||r')$。如果 D 输出 0，则输出 r'；否则输出 $\overline{r'}$。

显然 A 在多项式时间内运行。根据 A 的定义，我们有

$$\begin{aligned}\Pr_{s\leftarrow\{0,1\}^n}[A(f(s))=hc(s)] &= \frac{1}{2}\Pr_{s\leftarrow\{0,1\}^n}[A(f(s))=hc(s)\mid r'=hc(s)]+\frac{1}{2}\Pr_{s\leftarrow\{0,1\}^n}[A(f(s))=\\ &= \frac{1}{2}\left(\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||hc(s))=0]+\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||\overline{hc(s)})=1]\right)\\ &= \frac{1}{2}\left(\left(1-\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||hc(s))=1]\right)+\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||\overline{hc(s}\right)\\ &= \frac{1}{2}+\frac{1}{2}\left(\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||\overline{hc(s)})=1]-\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||hc(s))=\right.\end{aligned}$$

由于 hc 是 f 的硬核谓词，因此存在一个可忽略的函数 negl，使得

$$\frac{1}{2}\left(\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||\overline{hc(s)})=1]-\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)||hc(s))=1]\right)\le \mathrm{negl}(n),$$

如所期望的那样。

7.4.2 增加扩展因子

我们现在展示，伪随机生成器的扩展因子可以增加到任何期望的（多项式）量。这意味着前面的构造，扩展因子 $l(n) = n + 1$，足以构造具有任意（多项式）扩展因子的伪随机生成器。

定理 7.20 如果存在一个扩展因子 $n + 1$ 的伪随机生成器 G，那么对于任何多项式 poly，存在一个扩展因子 $\mathrm{poly}(n)$ 的伪随机生成器 $\tilde{G}$。

证明 我们首先考虑构造一个输出 $n + 2$ 位的伪随机生成器 $\tilde{G}$。 $\tilde{G}$ 的工作如下：给定初始种子 $s \in \{0,1\}^n$，它计算 $t_1 := G(s)$ 以获得 $n + 1$ 个伪随机位。然后使用 t_1 的初始 n 位作为 G 的种子；由此产生的 $n + 1$ 位，与 t_1 的最后一位串联起来，得到 $(n + 2)$ 位输出。（参见图 7.1。） G 的第二次应用使用了伪随机种子而不是随机种子。我们接下来给出的安全性证明表明这不会影响输出的伪随机性。

我们现在证明 $\tilde{G}$ 是一个伪随机生成器。定义三种分布序列 $\{H_n^0\}_{n=1,\ldots}$、 $\{H_n^1\}_{n=1,\ldots}$ 和 $\{H_n^2\}_{n=1,\ldots}$，其中 H_n^0, H_n^1 和 H_n^2 都是长度为 $n + 2$ 的字符串上的分布。在分布 H_n^0 中，选择均匀字符串 $t_0 \in \{0,1\}^n$，输出为 $G(t_0)$。在分布 H_n^1 中，选择均匀字符串 $t_1 \in \{0,1\}^{n+1}$，并将其解析为 $s_1||\sigma_1$（其中 s_1 是 t_1 的初始 n 位， σ_1 是最后一位）。输出为 $t_2 := G(s_1)||\sigma_1$。在分布 H_n^2 中，输出是均匀字符串 $t_2 \in \{0,1\}^{n+2}$。我们用 $t_2 \leftarrow H_n^j$ 表示根据分布 H_n^j 生成 $(n + 2)$ 位字符串 t_2 的过程。

固定一个任意的概率多项式时间区分器 D。我们首先声称存在一个可忽略的函数 negl'，使得

$$|\Pr_{t_2 \leftarrow H_n^1}[D(t_2) = 1] - \Pr_{t_2 \leftarrow H_n^2}[D(t_2) = 1]| \leq \mathrm{negl}'(n). \tag{7.4}$$

要看到这一点，考虑多项式时间区分器 D'，它在输入 $t_1 \in \{0,1\}^{n+1}$ 时，将 t_1 解析为 $s_1||\sigma_1$ 且 $|s_1| = n$，计算 $t_2 := G(s_1)||\sigma_1$，并输出 $D(t_2)$。显然 D' 在多项式时间内运行。观察到：

1. 如果 t_1 是均匀的，那么 D' 生成的 t_2 的分布恰好是 H_n^1 的分布。因此，

$$\Pr_{t_1 \leftarrow \{0,1\}^{n+1}}[D'(t_1) = 1] = \Pr_{t_2 \leftarrow H_n^1}[D(t_2) = 1].$$

2. 如果 $t_1 = G(s)$ 对于均匀 $s \in \{0,1\}^n$，那么 D' 生成的 t_2 的分布恰好是 H_n^0 的分布。也就是说，

$$\Pr_{s\leftarrow\{0,1\}^n}[D'(G(s))=1]=\Pr_{t_2\leftarrow H_n^0}[D(t_2)=1].$$

G 的伪随机性意味着存在一个可忽略的函数 negl'，使得

$$|\Pr_{s\leftarrow\{0,1\}^n}[D'(G(s))=1]-\Pr_{t_1\leftarrow\{0,1\}^{n+1}}[D'(t_1)=1]|\leq \mathrm{negl}'(n).$$

等式 (7.4) 成立。

接下来我们声称存在一个可忽略的函数 negl''，使得

$$|\Pr_{t_2\leftarrow H_n^0}[D(t_2)=1]-\Pr_{t_2\leftarrow H_n^1}[D(t_2)=1]|\leq \mathrm{negl}''(n). \tag{7.5}$$

要看到这一点，考虑多项式时间区分器 D''，它在输入 $w\in\{0,1\}^{n+1}$ 时，选择均匀 $\sigma_1\in\{0,1\}$，设置 $t_2:=w||\sigma_1$，并输出 $D(t_2)$。如果 w 是均匀的，那么 t_2 也是；因此，

$$\Pr_{w\leftarrow\{0,1\}^{n+1}}[D''(w)=1]=\Pr_{t_2\leftarrow H_n^2}[D(t_2)=1].$$

另一方面，如果 $w=G(s)$ 对于均匀 $s\in\{0,1\}^n$，那么 t_2 的分布恰好是 H_n^0 的分布，因此

$$\Pr_{s\leftarrow\{0,1\}^n}[D''(G(s))=1]=\Pr_{t_2\leftarrow H_n^0}[D(t_2)=1].$$

如前所述， G 的伪随机性意味着等式 (7.5) 成立。

把所有东西放在一起，我们有

$$\begin{aligned}
&|\Pr_{s\leftarrow\{0,1\}^n}[D(\tilde{G}(s))=1]-\Pr_{r\leftarrow\{0,1\}^{n+2}}[D(r)=1]|\\
&=|\Pr_{t_2\leftarrow H_n^0}[D(t_2)=1]-\Pr_{t_2\leftarrow H_n^2}[D(t_2)=1]|\\
&\leq|\Pr_{t_2\leftarrow H_n^0}[D(t_2)=1]-\Pr_{t_2\leftarrow H_n^1}[D(t_2)=1]|+|\Pr_{t_2\leftarrow H_n^1}[D(t_2)=1]-\Pr_{t_2\leftarrow H_n^2}[D(t_2)=1]|\\
&\leq \mathrm{negl}'(n)+\mathrm{negl}''(n),
\end{aligned} \tag{7.6}$$

使用了等式 (7.4) 和 (7.5)。由于 D 是一个任意的多项式时间区分器，这证明了 $\tilde{G}$ 是一个伪随机生成器。

一般情况。 与上述相同的思想可以迭代应用，以生成任何数量的伪随机位。形式上，假设我们希望构造一个扩展因子 $n+p(n)$ 的伪随机生成器 $\tilde{G}$，其中 p 是某个多项式。在输入 $s\in\{0,1\}^n$ 时，算法 $\tilde{G}$ 执行（参见图 7.1）：

1. 设置 $t_0 := s$。对于 $i = 1, \ldots, p(n)$ 执行：(a) 令 s_{i-1} 是 t_{i-1} 的前 n 位，令 σ_{i-1} 表示其余的 $i-1$ 位。（当 $i = 1$ 时，$s_0 = t_0$，σ_0 是空字符串。）(b) 设置 $t_i := G(s_{i-1})||\sigma_{i-1}$。
2. 输出 $t_{p(n)}$。

我们证明 $\tilde{G}$ 是一个伪随机生成器。证明使用了称为**混合论证**的常用技术。（实际上，上面 $p(n) = 2$ 的情况也使用了简单的混合论证。）主要的区别在于技术细节。以前，我们可以定义并明确使用三个分布序列 $\{H_n^0\}, \{H_n^1\}, \{H_n^2\}$。在这里，这是不可能的，因为要考虑的分布数量随 n 增长。

对于任何 n 和 $0 \leq j \leq p(n)$，令 H_n^j 是长度为 $n + p(n)$ 的字符串上的分布，定义如下：选择均匀 $t_j \in \{0,1\}^{n+j}$，然后从迭代 $j+1$ 开始运行 $\tilde{G}$ 并输出 $t_{p(n)}$。（当 $j = p(n)$ 时，这意味着我们简单地选择均匀 $t_{p(n)} \in \{0,1\}^{n+p(n)}$ 并输出它。）关键的观察是 H_n^0 对应于输出 $G(s)$ 对于均匀 $s \in \{0,1\}^n$，而 $H_n^{p(n)}$ 对应于输出均匀 $(n + p(n))$ 位字符串。固定一个多项式时间区分器 D，这意味着

$$\left|\Pr_{s \leftarrow \{0,1\}^n}[D(\tilde{G}(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{n+p(n)}}[D(r) = 1]\right| = \left|\Pr_{t \leftarrow H_n^0}[D(t) = 1] - \Pr_{t \leftarrow H_n^{p(n)}}[D(t) = 1]\right|. \tag{7.7}$$

我们证明上述是可忽略的，因此 $\tilde{G}$ 是一个伪随机生成器。

固定上述 D，并考虑区分器 D'，它在给定输入 $w \in \{0,1\}^{n+1}$ 时执行以下操作：

1. 选择均匀 $j \in \{1, \ldots, p(n)\}$。
2. 选择均匀 $\sigma'_j \in \{0,1\}^{j-1}$。（当 $j = 1$ 时，σ'_j 是空字符串。）
3. 设置 $t_j := w||\sigma'_j$。然后从迭代 $j+1$ 开始运行 $\tilde{G}$ 以计算 $t_{p(n)} \in \{0,1\}^{n+p(n)}$。输出 $D(t_{p(n)})$。

显然 D' 在多项式时间内运行。分析 D' 的行为比以前更复杂，尽管基本思想是相同的。固定 n，并假设 D' 选择 $j = j^*$。如果 w 是均匀的，那么 t_{j^*} 是均匀的，因此 $t \stackrel{\text{def}}{=} t_{p(n)}$ 的分布恰好是 $H_n^{j^*}$ 的分布。也就是说，

$$\Pr_{w \leftarrow \{0,1\}^{n+1}}[D'(w) = 1 \mid j = j^*] = \Pr_{t \leftarrow H_n^{j^*}}[D(t) = 1].$$

由于每个 j 值都是以相等的概率选择的，

$$\Pr_{w \leftarrow \{0,1\}^{n+1}}[D'(w) = 1] = \frac{1}{p(n)} \sum_{j^*=1}^{p(n)} \Pr_{w \leftarrow \{0,1\}^{n+1}}[D'(w) = 1 \mid j = j^*] = \frac{1}{p(n)} \sum_{j^*=1}^{p(n)} \Pr_{t \leftarrow H_n^{j^*}}[D(t) = 1]. \tag{7.8}$$

另一方面，假设 D' 选择 $j = j^*$ 且 $w = G(s)$ 对于均匀 $s \in \{0,1\}^n$。定义 $t_{j^*-1} = s||\sigma'_{j^*}$，我们看到 t_{j^*-1} 是均匀的，因此涉及 D' 的实验等价于从迭代 j^* 开始运行 $\tilde{G}$ 以计算 $t_{p(n)}$。也就是说，$t \stackrel{\text{def}}{=} t_{p(n)}$ 的分布现在恰好是 $H_n^{j^*-1}$ 的分布，因此

$$\Pr_{s \leftarrow \{0,1\}^n}[D'(G(s)) = 1 \mid j = j^*] = \Pr_{t \leftarrow H_n^{j^*-1}}[D(t) = 1].$$

因此，

$$\Pr_{s \leftarrow \{0,1\}^n}[D'(\tilde{G}(s)) = 1] = \frac{1}{p(n)} \sum_{j^*=1}^{p(n)} \Pr_{s \leftarrow \{0,1\}^n}[D'(\tilde{G}(s)) = 1 \mid j = j^*] = \frac{1}{p(n)} \sum_{j^*=1}^{p(n)} \Pr_{t \leftarrow H_n^{j^*-1}}[D(t) = 1] \tag{7.9}$$

我们现在可以分析 D' 区分 $\tilde{G}$ 的输出与随机的好坏程度：

$$\begin{aligned} &\left| \Pr_{s \leftarrow \{0,1\}^n}[D'(\tilde{G}(s)) = 1] - \Pr_{w \leftarrow \{0,1\}^{n+1}}[D'(w) = 1] \right| \\ &= \frac{1}{p(n)} \left| \sum_{j^*=0}^{p(n)-1} \Pr_{t \leftarrow H_n^{j^*}}[D(t) = 1] - \sum_{j^*=1}^{p(n)} \Pr_{t \leftarrow H_n^{j^*}}[D(t) = 1] \right| \\ &= \frac{1}{p(n)} \left| \Pr_{t \leftarrow H_n^0}[D(t) = 1] - \Pr_{t \leftarrow H_n^{p(n)}}[D(t) = 1] \right|, \end{aligned} \tag{7.10}$$

依赖于等式 (7.8) 和 (7.9) 来进行第一个等式。（第二个等式成立是因为每个和中都包含相同的项，除了左边和的第一个项和右边和的最后一个项。）由于 G 是一个伪随机生成器，等式 (7.10) 左侧的项是**可忽略的**；因为 p 是多项式的，这意味着等式 (7.7) 是可忽略的，从而完成了 $\tilde{G}$ 是伪随机生成器的证明。

综合起来。 令 f 是一个单向排列。取定理 7.19 中扩展因子为 $n+1$ 的伪随机生成器，并使用定理 7.20 证明中的方法将扩展因子增加到 $n+l$，我们得到以下伪随机生成器 $\tilde{G}$：

$$\tilde{G}(s) = f^{(l)}(s)||hc(f^{(l-1)}(s))|| \ldots ||hc(s),$$

其中 $f^{(i)}(s)$ 表示 f 的 i 次迭代。注意 $\tilde{G}$ 使用 l 次 f 的求值，并使用硬核谓词 hc 为每次求值生成一位伪随机位。

与流密码的联系。 回想一下第 3.3.1 节，流密码（没有 IV）由算法 $(\text{Init}, \text{GetBits})$ 定义，其中 Init 接受一个种子 $s \in \{0,1\}^n$ 并返回初始状态 st，GetBits 接受当前状态 st 并输出一位 σ 和更新状态 st'。前面的证明中的构造 $\tilde{G}$ 非常适合这个范式：取 Init 为输出 $st = s$ 的平凡算法，并定义 $\text{GetBits}(st)$ 来计算 $G(st)$，将结果解析为 $st'||\sigma$ 且 $|st'| = n$，并输出位 σ

和更新状态 st'。（如果我们使用这个流密码从种子 s 开始生成 $p(n)$ 个输出位，那么我们得到的正是 $\tilde{G}(s)$ 的最后 $p(n)$ 位，按相反的顺序。）前面的证明表明这产生了一个伪随机生成器。

混合论证。 混合论证是证明**不可区分性**的基本工具，当一个基本原语（或几个不同的原语）被多次应用时。非正式地说，该技术通过定义一系列中间的“混合分布”来桥接我们希望证明**不可区分**的两个“极端分布”。（在上面的证明中，这些极端分布对应于 $\tilde{G}$ 的输出和一个随机字符串。）要应用该证明技术，需要满足三个条件。首先，极端分布应该与感兴趣的原始情况匹配。（在上面的证明中，H_n^0 等于 $\tilde{G}$ 诱导的分布，而 $H_n^{p(n)}$ 是均匀分布。）其次，将区分连续混合分布的能力转化为打破某个潜在假设的能力必须是可能的。（上面，我们本质上表明区分 H_n^j 和 H_n^{j+1} 等价于区分 G 的输出与随机。）最后，混合分布的数量应该是多项式的。另见定理 7.32。

7.5 构造伪随机函数

我们现在展示如何从任何（长度加倍的）伪随机生成器构造伪随机函数。回想一下，伪随机函数是一个可高效计算的带密钥函数 F，它与一个真正随机函数在第 3.5.1 节所述的意义上是**不可区分**的。为简单起见，我们在此将注意力限制在 F 保持长度的情况，这意味着对于 $k \in \{0,1\}^n$，函数 F_k 将 n 位输入映射到 n 位输出。一个（保持长度的）伪随机函数可以非正式地视为扩展因子为 $n \cdot 2^n$ 的伪随机生成器；给定这样一个伪随机生成器 G，我们可以定义 $F_k(i)$（对于 $0 \leq i < 2^n$）为 $G(k)$ 的第 i 个 n 位块。这种方法行不通的原因是 F 必须是**可高效计算**的；有指数数量的块，我们需要一种方法来计算第 i 个块，而无需计算所有其他块。

我们将通过沿着**二叉树**向下走来计算输出的“块”。我们首先通过展示一个接受 2 位输入的伪随机函数的构造来举例说明。令 G 是一个扩展因子为 $2n$ 的伪随机生成器。如果我们在定理 7.20 的证明中使用 G，我们可以获得一个扩展因子为 $4n$ 的伪随机生成器 $\tilde{G}$，它使用三次调用 G。（每次应用 G 时，我们产生 n 个额外的伪随机位。）如果我们定义 $F_k(i)$（其中 $0 \leq i < 4$ 且 i 被编码为 2 位二进制字符串）为 $\tilde{G}(k)$ 的第 i 个块，那么 $F_k(3)$ 的计算将需要计算所有 $\tilde{G}$ 的输出，因此需要三次调用 G。我们展示如何使用仅**两次**调用 G 来构造伪随机函数 F 在任何输入上。

令 G_0 和 G_1 是表示 G 输出的前半部分和后半部分的函数；即 $G(k) = G_0(k)||G_1(k)$，其中 $|G_0(k)| = |G_1(k)| = |k|$。定义 F 如下：

$$F_k(00) = G_0(G_0(k))$$
$$F_k(01) = G_1(G_0(k))$$
$$F_k(10) = G_0(G_1(k))$$
$$F_k(11) = G_1(G_1(k))$$

我们声称上述四个字符串即使**一起**查看也是伪随机的。（这足以证明 F 是伪随机的。）直观上，这是因为 $G_0(k)||G_1(k) = G(k)$ 是伪随机的，因此与均匀的 $2n$ 位字符串 $k_0||k_1$ 是**不可区分**的。但随后

$$G_0(G_0(k))||G_1(G_0(k))||G_0(G_1(k))||G_1(G_1(k))$$

与

$$G_0(k_0)||G_1(k_0)||G_0(k_1)||G_1(k_1) = G(k_0)||G(k_1)$$

是不可区分的。由于 G 是一个伪随机生成器，上述内容与均匀的 $4n$ 位字符串是**不可区分**的。正式证明使用了混合论证。

推广这个思想，我们可以通过定义以下内容获得作用于 n 位输入的伪随机函数：

$$F_k(x) = G_{x_n}(\dots(G_{x_1}(k))\dots),$$

其中 $x = x_1 \dots x_n$；参见构造 7.21。这个函数为什么是伪随机的直觉与以前相同，但正式证明由于现在需要考虑**指数数量**的输入而变得复杂。

构造 7.21

令 G 是一个扩展因子 $l(n) = 2n$ 的伪随机生成器，并定义 G_0, G_1 如正文所述。对于 $k \in \{0,1\}^n$，定义函数 $F_k : \{0,1\}^n \to \{0,1\}^n$ 为：

$$F_k(x_1 x_2 \dots x_n) = G_{x_n}(\dots(G_{x_2}(G_{x_1}(k)))\dots).$$

来自伪随机生成器的伪随机函数。

将此构造视为定义一个深度为 n 的完全二叉树是有用的，其中每个节点包含一个 n 位值，对于每个密钥 $k \in \{0,1\}^n$。（参见图 7.2，其中 $n = 3$。）根节点的值为 k，对于每个内部节点，其值为 k'，其左子节点的值为 $G_0(k')$，其右子节点的值为 $G_1(k')$。然后 $F_k(x)$ 对于 $x = x_1 \dots x_n$ 的结果被定义为通过根据 x 的位遍历树到达的叶节点上的值，其中 $x_i = 0$ 意味着“向左走”， $x_i = 1$ 意味着“向右走”。（该函数仅对长度为 n 的输入定义，因此仅输出叶节点上的值。）树的大小在 n 中是指数级的。然而，要计算 $F_k(x)$ 不需要构造或存储整个树；

只需要 n 次 G 的求值。

图 7.2： 构造伪随机函数。

定理 7.22 如果 G 是一个扩展因子 $l(n) = 2n$ 的伪随机生成器，那么构造 7.21 是一个伪随机函数。

证明 我们首先证明对于任何多项式 t，区分 $t(n)$ 个均匀 $2n$ 位字符串与 $t(n)$ 个伪随机字符串是不可行的；即，对于任何多项式 t 和任何 PPT 算法 A，以下是可忽略的：

$$|\Pr[A(r_1||\dots||r_{t(n)}) = 1] - \Pr[A(G(s_1)||\dots||G(s_{t(n)})) = 1]|,$$

其中第一个概率是关于均匀选择 $r_1, \dots, r_{t(n)} \in \{0,1\}^{2n}$，第二个概率是关于均匀选择 $s_1, \dots, s_{t(n)} \in \{0,1\}^n$。

证明使用了混合论证。固定一个多项式 t 和一个 PPT 算法 A，并考虑以下算法 A'：

区分器 A'： A' 被给定输入一个字符串 $w \in \{0,1\}^{2n}$。

1. 选择均匀 $j \in \{1, \dots, t(n)\}$。
2. 选择均匀、独立的值 $r_1, \dots, r_{j-1} \in \{0,1\}^{2n}$ 和 $s_{j+1}, \dots, s_{t(n)} \in \{0,1\}^n$。
3. 输出 $A(r_1||\dots||r_{j-1}||w||G(s_{j+1})||\dots||G(s_{t(n)}))$。

对于任何 n 和 $0 \le i \le t(n)$，令 G_n^i 表示长度为 $2nt(n)$ 的字符串上的分布，其中前 i 个长度为 $2n$ 的“块”是均匀的，其余 $t(n) - i$ 个块是伪随机的。请注意，$G_n^{t(n)}$ 对应于所有 $t(n)$ 个块都是均匀的分布，而 G_n^0 对应于所有 $t(n)$ 个块都是伪随机的分布。也就是说，

$$|\Pr_{y \leftarrow G_n^{t(n)}}[A(y) = 1] - \Pr_{y \leftarrow G_n^0}[A(y) = 1]| = |\Pr[A(r_1||\dots||r_{t(n)}) = 1] - \Pr[A(G(s_1)||\dots||G(s_{t(n)})) = 1]|.$$

假设 A' 选择 $j = j^*$。如果其输入 w 是一个均匀 $2n$ 位字符串，那么 A 在根据 $G_n^{j^*}$ 分布的输入上运行。另一方面，如果 $w = G(s)$ 对于均匀 s，那么 A 在根据 $G_n^{j^*-1}$ 分布的输入上运行。这意味着

$$\Pr_{r \leftarrow \{0,1\}^{2n}}[A'(r) = 1] = \frac{1}{t(n)} \sum_{j=1}^{t(n)} \Pr_{y \leftarrow G_n^j}[A(y) = 1]$$

和

$$\Pr_{s\leftarrow\{0,1\}^n}[A'(G(s))=1]=\frac{1}{t(n)}\sum_{j=0}^{t(n)-1}\Pr_{y\leftarrow G_n^j}[A(y)=1].$$

因此，

$$\begin{aligned}&\left|\Pr_{r\leftarrow\{0,1\}^{2n}}[A'(r)=1]-\Pr_{s\leftarrow\{0,1\}^n}[A'(G(s))=1]\right|\\&=\frac{1}{t(n)}\left|\Pr_{y\leftarrow G_n^{t(n)}}[A(y)=1]-\Pr_{y\leftarrow G_n^0}[A(y)=1]\right|.\end{aligned}\tag{7.12}$$

由于 G 是一个伪随机生成器且 A' 在多项式时间内运行，我们知道等式 (7.12) 的左侧必须是可忽略的；因为 $t(n)$ 是多项式的，这意味着等式 (7.11) 的左侧也是可忽略的。

转到证明的核心，我们现在证明构造 7.21 中的 F 是一个伪随机函数。令 D 是一个任意的 PPT 区分器，它被给定 1^n 作为输入。我们证明 D 无法区分以下两种情况：当它被给定对等于 F_k 的函数的预言访问，其中 k 是均匀的，或者一个从 Func_n 中均匀选择的函数。（参见第 3.5.1 节。）为此，我们使用另一个混合论证。在这里，我们定义了深度为 n 的完全二叉树叶节点值的分布序列。通过将每个叶节点与构造 7.21 中长度为 n 的字符串关联起来，我们可以等效地将这些视为映射 n 位输入到 n 位输出的函数上的分布。对于任何 n 和 $0\leq i\leq n$，令 H_n^i 是深度为 n 的二叉树叶节点值上的以下分布：首先为第 i 级节点独立且均匀地选择 $\{0,1\}^n$ 中的值。然后，对于第 i 级或以下的每个节点，其值为 k，其左子节点被给定值 $G_0(k)$，其右子节点被给定值 $G_1(k)$。请注意，H_n^n 对应于叶节点上的所有值都是均匀且独立选择的分布，因此对应于从 Func_n 中选择一个均匀函数，而 H_n^0 对应于在构造 7.21 中选择一个均匀密钥 k，因为在这种情况下，只有根节点（在第 0 级）是均匀选择的。也就是说，

$$\left|\Pr_{k\leftarrow\{0,1\}^n}[D^{F_k(\cdot)}(1^n)=1]-\Pr_{f\leftarrow\mathrm{Func}_n}[D^{f(\cdot)}(1^n)=1]\right|=\left|\Pr_{f\leftarrow H_n^0}[D^{f(\cdot)}(1^n)=1]-\Pr_{f\leftarrow H_n^n}[D^{f(\cdot)}(1^n)=1]\right|\tag{7.13}$$

我们证明等式 (7.13) 是可忽略的，完成证明。

令 $t=t(n)$ 是 D 对其预言在输入 1^n 时发出的查询次数的多项式上界。定义一个区分器 A，它试图区分 $t(n)$ 个均匀 $2n$ 位字符串与 $t(n)$ 个伪随机字符串，如下所示：

区分器 A： A 被给定输入一个 $2nt(n)$ 位字符串 $w_1||\ldots||w_{t(n)}$。

1. 选择均匀 $j\in\{0,\ldots,n-1\}$。在下文中，A （隐式地）维护一个深度为 n 的二叉树，其内部节点（子集）在深度 $j+1$ 及以下具有 n 位值。
2. 运行 $D(1^n)$。当 D 发出预言查询 $x=x_1\ldots x_n$ 时，查看前缀 $x_1\ldots x_j$。有两种情况：

- 如果 D 以前从未对这个前缀发出查询，那么使用 $x_1 \ldots x_j$ 到达树的第 j 级的节点 v。取下一个未使用的 $2n$ 位字符串 w 并设置节点 v 的左子节点的值为 w 的左半部分，右子节点的值为 w 的右半部分。
- 如果 D 以前对前缀 $x_1 \ldots x_j$ 发出过查询，那么节点 $x_1 \ldots x_{j+1}$ 已经被分配了一个值。 使用节点 $x_1 \ldots x_{j+1}$ 处的值，计算对应于 $x_1 \ldots x_n$ 的叶节点的值，如构造 7.21 中所示，并将该值返回给 D。

3. 当 D 执行完毕后，输出 D 返回的位。

A 在多项式时间内运行。重要的是 A 不需要存储指数大小的整个二叉树。相反，它“填充”了树中至多 $2t(n)$ 个节点的值。假设 A 选择 $j = j^*$。观察到：

1. 如果 A's 的输入是一个均匀 $2nt(n)$ 位字符串，那么它给 D 的答案分布恰好如同 D 正在与从分布 $H_n^{j^*+1}$ 中选择的函数交互一样。这成立是因为树的第 $j^* + 1$ 级节点的值是均匀且独立的。
2. 如果 A's 的输入由 $t(n)$ 个伪随机字符串组成——即 $w_i = G(s_i)$ 对于均匀种子 s_i——那么它给 D 的答案分布恰好如同 D 正在与从分布 $H_n^{j^*}$ 中选择的函数交互一样。这成立是因为树的第 j^* 级节点的值（即 s 值）是均匀且独立的。（这些 s 值对 A 来说是未知的，但这没有区别。）

像以前一样，可以证明

$$\begin{aligned} &|\Pr[A(r_1||\ldots||r_{t(n)}) = 1] - \Pr[A(G(s_1)||\ldots||G(s_{t(n)})) = 1]| \\ &= \frac{1}{n}\left|\Pr_{f \leftarrow H_n^{j^*}}[D^{f(\cdot)}(1^n) = 1] - \Pr_{f \leftarrow H_n^{j^*+1}}[D^{f(\cdot)}(1^n) = 1]\right|. \end{aligned} \tag{7.14}$$

我们前面已经证明等式 (7.14) 必须是可忽略的。因此，上述内容意味着等式 (7.13) 也必须是可忽略的。

7.6 构造（强）伪随机排列

接下来我们展示如何从任何伪随机函数构造伪随机排列和强伪随机排列。回想一下第 3.5.1 节，伪随机排列是一个也是可高效反转的伪随机函数，而强伪随机排列是即使敌手被给定对排列及其**逆**的预言访问，也难以与随机排列区分开的排列。

Feistel 网络回顾。 Feistel 网络在第 6.2.2 节中介绍，它提供了一种从任意函数集构造可反转函数的方法。Feistel 网络在一系列轮中操作。第 i 轮的输入是一个长度为 $2n$ 的字符串，分

为两个 n 位半部分 L_{i-1} 和 R_{i-1}（“左半部分”和“右半部分”）。第 i 轮的输出是 $2n$ 位字符串 (L_i, R_i)，其中

$$L_i := R_{i-1} \quad \text{和} \quad R_i := L_{i-1} \oplus f_i(R_{i-1})$$

对于某些可高效计算（但不一定是可反转）的函数 f_i，它将 n 位输入映射到 n 位输出。我们用 $\mathrm{Feistel}_{f_1,\ldots,f_r}$ 表示使用函数 $f_1, \ldots, f_r$ 的 r 轮 Feistel 网络。（也就是说，$\mathrm{Feistel}_{f_1,\ldots,f_r}(L_0, R_0)$ 输出 $2n$ 位字符串 (L_r, R_r)。）我们在第 6.2.2 节中看到，无论 $\{f_i\}$ 如何，$\mathrm{Feistel}_{f_1,\ldots,f_r}$ 都是一个可高效反转的排列。

我们可以通过在 Feistel 网络中使用依赖于密钥的函数来定义带密钥排列。例如，令 F：$\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ 是一个伪随机函数，并定义带密钥排列 $F^{(1)}$ 为

$$F^{(1)}_k(x) \stackrel{\text{def}}{=} \mathrm{Feistel}_{F_k}(x).$$

（请注意，$F^{(1)}_k$ 有一个 n 位密钥，并将 $2n$ 位输入映射到 $2n$ 位输出。） $F^{(1)}$ 是伪随机的吗？稍加思考就会发现它肯定不是。对于任何密钥 $k \in \{0,1\}^n$， $F^{(1)}_k$ 输出的前 n 位（即 L_1）等于输入的后 n 位（即 R_0），这对于随机函数来说发生的概率是可忽略的。

再次尝试，定义 $F^{(2)} : \{0,1\}^{2n} \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ 如下：

$$F^{(2)}_{k_1,k_2}(x) \stackrel{\text{def}}{=} \mathrm{Feistel}_{F_{k_1},F_{k_2}}(x). \tag{7.15}$$

（请注意，k_1 和 k_2 是独立的密钥。）不幸的是， $F^{(2)}$ 也不是伪随机的，正如你在练习 7.16 中被要求证明的那样。

鉴于此，三轮 Feistel 网络是伪随机的可能有些令人惊讶。定义带密钥排列 $F^{(3)}$，它接受长度为 $3n$ 的密钥并将 $2n$ 位输入映射到 $2n$ 位输出，如下所示：

$$F^{(3)}_{k_1,k_2,k_3}(x) \stackrel{\text{def}}{=} \mathrm{Feistel}_{F_{k_1},F_{k_2},F_{k_3}}(x) \tag{7.16}$$

其中，再次地，k_1, k_2 和 k_3 是独立的。我们有：

定理 7.23 如果 F 是一个伪随机函数，那么 $F^{(3)}$ 是一个伪随机排列。

证明 按照标准方式，我们可以用均匀随机选择的函数替换构造 $F^{(3)}$ 中使用的伪随机函数。 F 的伪随机性意味着这只会对与 $F^{(3)}$ 作为预言交互的任何概率多项式时间区分器的输出产生可忽略的影响。我们将细节留作练习。

令 D 是一个概率多项式时间区分器。在证明的其余部分，我们证明以下是可忽略的：

$$|\Pr[D^{\mathrm{Feistel}_{f_1,f_2,f_3}(\cdot)}(1^n)=1]-\Pr[D^{\pi(\cdot)}(1^n)=1]|,$$

其中第一个概率是关于均匀且独立选择 f_1, f_2, f_3 从 Func_n 中，第二个概率是关于均匀选择 π 从 Perm_{2n} 中。固定安全参数 n 的某个值，令 $q = q(n)$ 表示 D 对预言发出的查询次数的多项式上界。我们假设不失一般性， D 从不发出相同的预言查询两次。关注 D 与 $\mathrm{Feistel}_{f_1,f_2,f_3}(\cdot)$ 的交互，令 (L_0^i, R_0^i) 表示 D 对其预言发出的第 i 次查询，令 $(L_1^i, R_1^i), (L_2^i, R_2^i)$ 和 (L_3^i, R_3^i) 分别表示由此查询产生的第 1、2 和 3 轮后的中间值。（参见图 7.3。）请注意， D 选择 (L_0^i, R_0^i) 并看到结果 (L_3^i, R_3^i)，但不能直接观察 (L_1^i, R_1^i) 或 (L_2^i, R_2^i)。

图 7.3： 三轮 Feistel 网络，用于从伪随机函数构造伪随机排列。

我们称在 R_1 处发生**碰撞**，如果 $R_1^i = R_1^j$ 对于某些不同的 i, j。我们首先证明在 R_1 处发生碰撞的概率是可忽略的。考虑任意固定的、不同的 i, j。如果 $R_0^i \neq R_0^j$，那么 $f_1(R_0^i)$ 和 $f_1(R_0^j)$ 是均匀且独立的，所以

$$\Pr[L_0^i \oplus f_1(R_0^i) = L_0^j \oplus f_1(R_0^j)] = \Pr[f_1(R_0^i) = L_0^i \oplus L_0^j \oplus f_1(R_0^j)] = 2^{-n}.$$

对所有不同的 i, j 应用并集界，表明在 R_1 处发生碰撞的概率至多为 $q^2/2^n$。

假设在 R_2 处发生碰撞，如果 $R_2^i = R_2^j$ 对于某些不同的 i, j。我们证明以 R_1 处没有碰撞为条件，在 R_2 处发生碰撞的概率是可忽略的。分析与上述类似：考虑任意固定的 i, j，并注意如果 R_1 处没有碰撞，那么 $R_1^i \neq R_1^j$。因此 $f_2(R_1^i)$ 和 $f_2(R_1^j)$ 是均匀且独立的，因此

$$\Pr[L_1^i \oplus f_2(R_1^i) = L_1^j \oplus f_2(R_1^j) \mid \text{no collision at } R_1] = 2^{-n}.$$

（请注意 f_2 独立于 f_1，这使得上述计算变得容易。）对所有不同的 i, j 应用并集界，得到

$$\Pr[\text{collision at } R_2 \mid \text{no collision at } R_1] \leq q^2/2^n.$$

请注意 $L_3^i = R_2^i = L_1^i \oplus f_2(R_1^i)$；因此，以 R_1 处没有碰撞为条件，值 $L_3^1, \ldots, L_3^q$ 在 $\{0,1\}^n$ 中是独立且均匀分布的。如果我们在事件“ R_2 处没有碰撞”上也设置条件，那么值 $L_3^1, \ldots, L_3^q$ 在 $\{0,1\}^n$ 中的所有 q 个不同值序列中是均匀分布的。类似地， $R_3^i = L_2^i \oplus f_3(R_2^i)$；因此，以 R_2 处没有碰撞为条件，值 $R_3^1, \ldots, R_3^q$ 在 $\{0,1\}^n$ 中是均匀分布的，彼此独立，也独立于 $L_3^1, \ldots, L_3^q$。

总结：当查询 $F^{(3)}$ （具有均匀轮函数）在一系列 q 个不同输入上时，除了可忽略的概率外，输出值 $(L_3^1, R_3^1), \ldots, (L_3^q, R_3^q)$ 的分布是这样的： $\{L_3^i\}$ 是均匀且独立的 n 位值，但**不同**，而 $\{R_3^i\}$ 是均匀且独立的 n 位值。相比之下，当查询一个随机排列在一系列 q 个不同输入上时，

输出值 $(L_3^1, R_3^1), \ldots, (L_3^q, R_3^q)$ 是均匀且独立的 $2n$ 位值，但**不同**。那么，对于 D 来说最好的区分攻击是猜测它正在与一个随机排列交互，如果 $L_3^i = L_3^j$ 对于某些不同的 i, j。但是即使在这种情况下，该事件发生的概率也是可忽略的。这可以转化为正式证明。

$F^{(3)}$ **不是强伪随机排列**，正如你在练习 7.17 中被要求证明的那样。幸运的是，增加第四轮确实产生了强伪随机排列。细节在构造 7.24 中给出。

定理 7.25 如果 F 是一个伪随机函数，那么构造 7.24 是一个将 $2n$ 位输入映射到 $2n$ 位输出的**强伪随机排列**（并使用 $4n$ 位密钥）。

构造 7.24 令 F 是一个带密钥、保持长度的函数。定义带密钥排列 $F^{(4)}$ 如下：

- **输入：** 一个密钥 $k = (k_1, k_2, k_3, k_4)$，其中 $|k_i| = n$，和一个输入 $x \in \{0,1\}^{2n}$，解析为 (L_0, R_0)，其中 $|L_0| = |R_0| = n$。
- **计算：**
  i. 计算 $L_1 := R_0$ 和 $R_1 := L_0 \oplus F_{k_1}(R_0)$。
  ii. 计算 $L_2 := R_1$ 和 $R_2 := L_1 \oplus F_{k_2}(R_1)$。
  iii. 计算 $L_3 := R_2$ 和 $R_3 := L_2 \oplus F_{k_3}(R_2)$。
  iv. 计算 $L_4 := R_3$ 和 $R_4 := L_3 \oplus F_{k_4}(R_3)$。
  v. 输出 (L_4, R_4)。

来自任何伪随机函数的强伪随机排列。

7.7 对称密钥密码学的假设

我们已经证明：(1) 如果存在单向排列，那么存在伪随机生成器；(2) 如果存在伪随机生成器，那么存在伪随机函数；以及 (3) 如果存在伪随机函数，那么存在（强）伪随机排列。虽然我们没有在这里证明，但可以从单向函数构造伪随机生成器。因此，我们有以下基本定理：

定理 7.26 如果单向函数存在，那么伪随机生成器、伪随机函数和强伪随机排列也存在。

我们在第 3 章和第 4 章中研究过的所有对称密钥方案都可以从伪随机生成器/函数构造。因此我们有：

定理 7.27 如果单向函数存在，那么 CCA 安全的对称密钥加密方案和安全的消息认证码也存在。

也就是说，单向函数足以满足所有对称密钥密码学的需求。在这里，我们证明单向函数也是必要的。

伪随机性蕴含单向函数。 我们首先证明伪随机生成器蕴含单向函数的存在：

命题 7.28 如果存在一个伪随机生成器，那么也存在一个单向函数。

证明 令 G 是一个扩展因子 $l(n) = 2n$ 的伪随机生成器。（根据定理 7.20，我们知道伪随机生成器的存在蕴含着具有这个扩展因子的生成器的存在。）我们证明 G 本身是单向的。高效可计算性是直接的（因为 G 可以在多项式时间内计算）。我们证明反转 G 的能力可以转化为区分 G 的输出与均匀随机的能力。直观上，这成立是因为反转 G 的能力蕴含着找到生成器使用的种子的能力。

令 A 是一个任意的概率多项式时间算法。我们证明 $\Pr[\mathrm{Invert}_{A,G}(n) = 1]$ 是可忽略的（参见定义 7.1）。要看到这一点，考虑以下 PPT 区分器 D：在输入字符串 $w \in \{0,1\}^{2n}$ 时，运行 $A(w)$ 以获得输出 s。如果 $G(s) = w$ 则输出 1；否则输出 0。

我们现在分析 D 的行为。首先考虑当其输入字符串 w 是均匀时 D 输出 1 的概率。由于 G 的范围中至多有 2^n 个值（即值集合 $\{G(s)\}_{s\in\{0,1\}^n}$），w 在 G 的范围中的概率至多为 $2^n/2^{2n} = 2^{-n}$。当 w 不在 G 的范围中时，A 不可能计算 w 的逆，因此 D 不可能输出 1。我们得出结论

$$\Pr_{w\leftarrow\{0,1\}^{2n}}[D(w) = 1] \leq 2^{-n}.$$

另一方面，如果 $w = G(s)$ 对于均匀选择的种子 $s \in \{0,1\}^n$，那么根据定义，A 以恰好等于 $\Pr[\mathrm{Invert}_{A,G}(n) = 1]$ 的概率计算出一个正确的逆（因此 D 输出 1）。因此，

$$\Pr_{w\leftarrow\{0,1\}^{2n}}[D(w) = 1] - \Pr_{s\leftarrow\{0,1\}^{n}}[D(G(s)) = 1] \geq \Pr[\mathrm{Invert}_{A,G}(n) = 1] - 2^{-n}.$$

由于 G 是一个伪随机生成器，上述差异必须是可忽略的。由于 2^{-n} 是可忽略的，这意味着 $\Pr[\mathrm{Invert}_{A,G}(n) = 1]$ 也是可忽略的，因此 G 是单向的。

非平凡的对称密钥加密蕴含单向函数。 命题 7.28 并不意味着构造安全的对称密钥加密方案需要单向函数，因为它可能无需依赖伪随机生成器就能构造后者。此外，可以构造**完美安全**的加密方案（参见第 2 章），只要明文不长于密钥即可。因此，证明安全的对称密钥加密蕴含单向函数需要更多的谨慎。

命题 7.29 如果存在一个 EAV 安全的对称密钥加密方案，它可以加密长度是密钥两倍的消息，

那么存在一个单向函数。

证明 令 $\Pi = (\mathrm{Enc}, \mathrm{Dec})$ 是一个具有窃听者存在下不可区分加密的对称密钥加密方案，并且当密钥长度为 n 时，它可以加密长度为 $2n$ 的消息。（我们假设密钥是均匀选择的，以简化。）假设当使用 n 位密钥时， Enc 使用至多 $l(n)$ 位随机性。用 $\mathrm{Enc}_k(m;r)$ 表示使用密钥 k 和随机性 r 对消息 m 的加密。

定义以下函数 f：

$$f(k, m, r) \stackrel{\mathrm{def}}{=} \mathrm{Enc}_k(m;r)||m,$$

其中 $|k| = n, |m| = 2n, |r| = l(n)$。我们声称 f 是一个单向函数。显然它可以高效计算；我们证明它难以反转。令 A 是一个任意的 PPT 算法，我们证明 $\Pr[\mathrm{Invert}_{A,f}(n) = 1]$ 是可忽略的（参见定义 7.1）。

考虑以下概率多项式时间敌手 A' 攻击对称密钥加密方案 Π （即在实验 $\mathrm{PrivK}^{\mathrm{eav}}_{\Pi,A'}(n)$ 中）：

敌手 $A'(1^n)$

1. 选择均匀 $m_0, m_1 \leftarrow \{0,1\}^{2n}$ 并输出它们。返回一个挑战密文 c。
2. 运行 $A(c||m_0)$ 以获得 (k', m', r')。如果 $f(k', m', r') = c||m_0$，则输出 0；否则输出 1。

我们现在分析 A' 的行为。当 c 是 m_0 的加密时， $c||m_0$ 的分布恰好如同 $f(k, m_0, r)$ 对于均匀 k, m_0 和 r 一样。因此， A 以恰好等于 $\Pr[\mathrm{Invert}_{A,f}(n) = 1]$ 的概率输出 $c||m_0$ 的有效逆（因此 A' 输出 0）。

另一方面，当 c 是 m_1 的加密时， c 独立于 m_0。对于挑战密文 c 的任何固定值，至多有 2^n 个可能的消息（每个可能的密钥一个）与 c 对应。由于 m_0 是一个均匀 $2n$ 位字符串，这意味着存在某个密钥 k 使得 $\mathrm{Dec}_k(c) = m_0$ 的概率至多为 $2^n/2^{2n} = 2^{-n}$。这给出了 A 可能输出 $c||m_0$ 在 f 下的有效逆的概率的上界，因此也给出了 A' 在那种情况下输出 0 的概率的上界。

将上述内容放在一起，我们有：

$$\begin{aligned}
\Pr[\mathrm{PrivK}^{\mathrm{eav}}_{\Pi,A'}(n) = 1] &= \frac{1}{2} \cdot \Pr[A' \text{ outputs } 0 \mid b = 0] + \frac{1}{2} \cdot \Pr[A' \text{ outputs } 1 \mid b = 1] \\
&\geq \frac{1}{2} \cdot \Pr[\mathrm{Invert}_{A,f}(n) = 1] + \frac{1}{2} \cdot (1 - 2^{-n}) \\
&= \frac{1}{2} + \frac{1}{2} \cdot (\Pr[\mathrm{Invert}_{A,f}(n) = 1] - 2^{-n}).
\end{aligned}$$

Π 的安全性意味着 $\Pr[\mathrm{PrivK}^{\mathrm{eav}}_{\Pi,A'}(n) = 1] \leq 1/2 + \mathrm{negl}(n)$ 对于某个可忽略的函数 negl。反过来，这意味着 $\Pr[\mathrm{Invert}_{A,f}(n) = 1]$ 是可忽略的，完成了 f 是单向函数的证明。

消息认证码蕴含单向函数。 消息认证码满足定义 4.2 也蕴含单向函数的存在。与对称密钥加密的情况一样，这个事实的证明有些微妙，因为**无条件**消息认证码确实存在，当认证的消息数量有**先验**界限时。（参见第 4.6 节。）因此，证明依赖于这样一个事实：定义 4.2 要求即使敌手看到**任意**（多项式）数量的消息的认证标签，安全性仍然成立。证明有些复杂，所以我们在此不给出。

讨论。 我们得出结论，单向函数的存在对于所有（非平凡的）对称密钥密码学是**必要且充分**的。换句话说，就对称密钥密码学而言，单向函数是**最小的假设**。有趣的是，对于哈希函数和公钥加密而言，情况似乎并非如此，单向函数已知是必要的，但尚未知（或不被相信）是充分的。

7.8 计算不可区分性

计算不可区分性的概念是密码学理论的核心，也是我们在第 3 章和本章中看到的大部分内容的基础。非正式地，如果两个概率分布没有高效算法可以区分它们，则称它们是**计算不可区分**的。更详细地说，考虑两个分布 X 和 Y 在某些长度 l 的字符串上；也就是说， X 和 Y 都为 $\{0,1\}^l$ 中的每个字符串分配一些概率。当我们说某个算法 D 不能区分这两个分布时，我们指的是 D 无法判断它被给定的是根据分布 X 采样得到的字符串，还是根据分布 Y 采样得到的字符串。换句话说，如果我们想象 D 在认为其输入是根据 X 采样时输出“0”，而在认为其输入是根据 Y 采样时输出“1”，那么 D 输出“1”的概率应该大致相同，无论 D 是被提供了来自 X 的样本还是来自 Y 的样本。换句话说，我们希望

$$|\Pr_{s \leftarrow X}[D(s) = 1] - \Pr_{s \leftarrow Y}[D(s) = 1]|$$

很小。

这应该让人想起我们定义伪随机生成器的方式，事实上，我们很快将用这个术语正式重新定义伪随机生成器的概念。

计算不可区分性的正式定义指的是**概率集成**，它们是概率分布的无限序列。（这种形式主义对于有意义的渐近方法是必要的。）虽然这个概念可以推广，但为了我们的目的，我们考虑其中潜在分布由自然数索引的概率集成。如果对于每个自然数 n，我们有一个分布 X_n，那么 $\mathcal{X} = \{X_n\}_{n \in \mathbb{N}}$ 是一个概率集成。通常情况是 $X_n = Y_{l(n)}$ 对于某个函数 l，在这种情况下，我们用

$\{Y_{l(n)}\}_{n\in\mathbb{N}}$ 代替 $\{X_n\}_{n\in\mathbb{N}}$。

我们将只对**可高效采样**的概率集成感兴趣。集成 $\mathcal{X}=\{X_n\}_{n\in\mathbb{N}}$ 是可高效采样的，如果存在一个概率多项式时间算法 S 使得随机变量 $S(1^n)$ 和 X_n 具有相同的分布。也就是说，算法 S 是采样 $\mathcal{X}$ 的高效方法。

我们现在可以正式定义两个集成是**计算不可区分**的含义。

定义 7.30 如果对于每个概率多项式时间区分器 D，存在一个可忽略的函数 negl，使得

$$|\Pr_{x\leftarrow X_n}[D(1^n,x)=1]-\Pr_{y\leftarrow Y_n}[D(1^n,y)=1]|\leq \mathrm{negl}(n).$$

则称两个概率集成 $\mathcal{X}=\{X_n\}_{n\in\mathbb{N}}$ 和 $\mathcal{Y}=\{Y_n\}_{n\in\mathbb{N}}$ 是**计算不可区分**的，记为 $\mathcal{X}\stackrel{c}{\equiv}\mathcal{Y}$。

在定义中， D 被给定一元输入 1^n，因此它可以在多项式时间内运行。当 X_n 和 Y_n 的输出长度小于 n 时，这很重要。作为概率表达式的简写，我们有时会写 $\mathcal{X}$ 作为来自分布 X 的随机样本的占位符。也就是说，我们写 $\Pr[D(1^n,X_n)=1]$ 代替 $\Pr_{x\leftarrow X_n}[D(1^n,x)=1]$。

计算不可区分性的关系是**传递性**的：如果 $\mathcal{X}\stackrel{c}{\equiv}\mathcal{Y}$ 且 $\mathcal{Y}\stackrel{c}{\equiv}\mathcal{Z}$，那么 $\mathcal{X}\stackrel{c}{\equiv}\mathcal{Z}$。

伪随机性与伪随机生成器。 伪随机性只是计算不可区分性的一个特例。对于任何整数 l，令 U_l 表示 $\{0,1\}^l$ 上的均匀分布。我们可以定义伪随机生成器如下：

定义 7.31 令 $l(\cdot)$ 是一个多项式，令 G 是一个（确定性）多项式时间算法，其中对于所有 s 都有 $|G(s)|=l(|s|)$。如果以下两个条件成立，则称 G 是一个**伪随机生成器**：

1. **（扩展：）** 对于每个 n，$l(n)>n$ 成立。
2. **（伪随机性：）** 集成 $\{G(U_n)\}_{n\in\mathbb{N}}$ 与集成 $\{U_{l(n)}\}_{n\in\mathbb{N}}$ 是**计算不可区分**的。

本书中的许多其他定义和假设也可以视为计算不可区分性的特例或变体。

多个样本。 关于计算不可区分性的一个重要定理是：多项式数量的（可高效采样）计算不可区分集成的样本也是计算不可区分的。

定理 7.32 令 $\mathcal{X}$ 和 $\mathcal{Y}$ 是可高效采样的概率集成，它们是**计算不可区分**的。那么，对于每个多项式 p，集成 $\mathcal{X}^p=\{(X_n^{(1)},\ldots,X_n^{(p(n))})\}_{n\in\mathbb{N}}$ 与集成 $\mathcal{Y}^p=\{(Y_n^{(1)},\ldots,Y_n^{(p(n))})\}_{n\in\mathbb{N}}$ 是**计算不可区分**的。

例如，令 G 是一个扩展因子 $2n$ 的伪随机生成器，在这种情况下，集成 $\{G(U_n)\}_{n\in\mathbb{N}}$ 和 $\{U_{2n}\}_{n\in\mathbb{N}}$ 是计算不可区分的。在定理 7.22 的证明中，我们证明了对于任何多项式 t，集成

$$\{(G(U_n),\ldots,G(U_n))\}_{n\in\mathbb{N}}^{t(n)} \quad \text{和} \quad \{(U_{2n},\ldots,U_{2n})\}_{n\in\mathbb{N}}^{t(n)}$$

也是计算不可区分的。定理 7.32 以完全相同的方式通过混合论证证明。

参考文献和额外阅读

单向函数的概念最早由 Diffie 和 Hellman [58] 提出，后来由 Yao [179] 正式化。硬核谓词由 Blum 和 Micali [37] 引入，而对于每个单向函数都存在硬核谓词的事实由 Goldreich 和 Levin [79] 证明。第一个伪随机生成器的构造（基于特定的数论困难假设）由 Blum 和 Micali [37] 给出。从任何单向排列构造伪随机生成器的构造由 Yao [179] 给出，而伪随机生成器可以从任何单向函数构造的结果由 Håstad 等人 [85] 证明。伪随机函数由 Goldreich, Goldwasser 和 Micali [78] 定义和构造，并且它们到（强）伪随机排列的扩展由 Luby 和 Rackoff [116] 证明。单向函数是对称密钥密码学的大部分工作所必需的假设，这一点由 [93] 证明。命题 7.29 的证明来自 [72]。

我们的介绍受到 Goldreich 的著作 [75] 的强烈影响，对于那些有兴趣更深入地探索本章主题的人，强烈推荐这本书。

练习

7.1 证明如果存在一个单向函数，那么存在一个单向函数 f 使得对于每个 n， $f(0^n)=0^n$。请注意，对于无限多 y 值，很容易计算 $f^{-1}(y)$。为什么这与单向性不矛盾?

7.2 证明如果 f 是一个单向函数，那么由 $g(x_1,x_2)\stackrel{\text{def}}{=}(f(x_1),x_2)$ 定义的函数 g，其中 $|x_1|=|x_2|$，也是一个单向函数。观察到 g 泄露了其输入的一半，但仍然是单向的。

7.3 证明如果存在一个单向函数，那么存在一个**保持长度**的单向函数。 **提示：** 令 f 是一个单向函数，令 $p(\cdot)$ 是一个多项式，使得 $|f(x)|\leq p(|x|)$ 成立。（证明存在这样的 p。）定义 $f'(x)\stackrel{\text{def}}{=}f(x)||1^{p(|x|)-|f(x)|}$。进一步修改 f' 以获得一个保持长度的函数，它仍然是单向的。

7.4 令 (Gen,H) 是一个抗碰撞哈希函数，其中 H 将长度为 $2n$ 的字符串映射到长度为 n 的字符串。证明函数族 $(\mathrm{Gen},\mathrm{Samp},H)$ 是单向的（参见定义 7.3），其中 Samp 是采样长度为 $2n$ 的均匀字符串的平凡算法。 **提示：** 选择均匀 $x\in\{0,1\}^{2n}$ 并找到 $y=H_I(x)$ 的逆并不能保证碰撞。但它确实以大部分时间产生碰撞。

7.5 令 F 是一个（保持长度的）伪随机排列。 (a) 证明函数 $f(x,y)=F_x(y)$ 不是单向的。

(b) 证明函数 $f(y) = F_{0^n}(y)$（其中 $n = |y|$）不是单向的。(c) 证明函数 $f(x) = F_x(0^n)$（其中 $n = |x|$）是单向的。

7.6 令 f 是一个保持长度的单向函数，令 hc 是 f 的硬核谓词。定义 G 为 $G(x) = f(x)||hc(x)$。G 一定是伪随机生成器吗？证明你的答案。

7.7 证明当且仅当存在单向函数族时，存在单向函数。讨论你的证明为什么不能推广到单向排列的情况。

7.8 令 f 是一个单向函数。$g(x) \stackrel{\text{def}}{=} f(f(x))$ 一定是单向函数吗？$g'(x) \stackrel{\text{def}}{=} f(x)||f(f(x))$ 呢？证明你的答案。

7.9 令 $\Pi = (\mathrm{Gen}, \mathrm{Samp}, f)$ 是一个函数族。一个函数 $hc : \{0,1\}^* \to \{0,1\}$ 是 Π 的**硬核谓词**，如果它可以高效计算，并且对于每个 PPT 算法 A，存在一个可忽略的函数 negl，使得

$$\Pr_{I \leftarrow \mathrm{Gen}(1^n), x \leftarrow \mathrm{Samp}(I)}[A(I, f_I(x)) = hc(I, x)] \leq \frac{1}{2} + \mathrm{negl}(n).$$

证明 Goldreich-Levin 定理的这个版本对于这个设定：即，如果存在一个单向函数（或排列）族 Π，那么存在一个单向函数（或排列）族 Π' 和 Π' 的硬核谓词 hc。

7.10 展示从任何单向排列族构造伪随机生成器的方法。你可以使用上一个练习的结果。

7.11 这个练习是为学习过计算复杂性理论或熟悉 NP 完备性的学生准备的。(a) 证明单向函数的存在蕴含 $P \neq NP$。(b) 假设 $P \neq NP$。证明存在一个函数 f，它满足：(1) 在多项式时间内可计算，(2) 在最坏情况下难以反转（即，对于所有概率多项式时间 A，$\Pr_{x \leftarrow \{0,1\}^n}[f(A(f(x))) = f(x)] \neq 1$），但 (3) 不是单向的。

7.12 令 $x \in \{0,1\}^n$，表示 $x = x_1 \dots x_n$。证明如果存在一个单向函数，那么存在一个单向函数 f，使得对于每个 i，存在一个算法 A_i 使得

$$\Pr_{x \leftarrow \{0,1\}^n}[A_i(f(x)) = x_i] \geq \frac{1}{2} + \frac{1}{2n}.$$

（这个练习表明，声称每个单向函数都隐藏了输入的至少一个**特定位**是不可能的。）

7.13 证明如果一个一对一函数 f 有一个硬核谓词，那么 f 是单向的。

7.14 证明如果构造 7.21 以自然的方式修改，使得 $F_k(x)$ 对于长度至多为 n 的每个非空字符串 x 定义，那么该构造不再是伪随机函数。

7.15 证明如果存在一个伪随机函数，它使用长度为 n 的密钥，将 n 位输入映射到单一位输出，那么存在一个伪随机函数，它将 n 位输入映射到 n 位输出。 **提示：** 使用长度为 n^2 的密钥，并使用混合论证证明你的构造是安全的。

7.16 证明使用伪随机轮函数的两轮 Feistel 网络（如等式 (7.15) 中）不是伪随机的。

7.17 证明使用伪随机轮函数的三轮 Feistel 网络（如等式 (7.16) 中）不是**强**伪随机的。 **提示：** 这比上一个练习困难得多。使用一个区分器，它对排列发出两次查询，并对其逆发出一次查询。

7.18 考虑由

$$F_k^*(x) \stackrel{\text{def}}{=} \text{Feistel}_{F_k,F_k,F_k}(x)$$

定义的带密钥排列 F^*。（请注意，在每一轮中都使用相同的密钥。）证明 F^* 不是伪随机的。

7.19 令 G 是一个扩展因子 $l(n) = n + 1$ 的伪随机生成器。证明 G 是一个单向函数。

7.20 令 $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ 是概率集成。证明如果 $\mathcal{X} \stackrel{c}{\equiv} \mathcal{Y}$ 且 $\mathcal{Y} \stackrel{c}{\equiv} \mathcal{Z}$，那么 $\mathcal{X} \stackrel{c}{\equiv} \mathcal{Z}$。

7.21 证明定理 7.32。

7.22 令 $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ 和 $\mathcal{Y} = \{Y_n\}_{n\in\mathbb{N}}$ 是计算不可区分的概率集成。证明对于任何概率多项式时间算法 A，集成 $\{A(X_n)\}_{n\in\mathbb{N}}$ 和 $\{A(Y_n)\}_{n\in\mathbb{N}}$ 是计算不可区分的。