第七章 对称密钥原语的理论构建

在前面的章节中，我们介绍了伪随机性（pseudorandomness）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（PRG）、伪随机函数（PRF）和伪随机置换（PRP）。我们在第三章和第四章中看到，这些原语是对称密钥密码学所有内容的基础构建模块。因此，从理论角度理解这些原语的存在性是非常重要的。在本章中，我们将正式引入单向函数（one-way functions）的概念——非正式地说，它们是易于计算但难以求逆的函数——并展示如何仅在**假设单向函数存在**的前提下，构建伪随机生成器、伪随机函数和伪随机置换。此外，我们将看到单向函数的存在性对于“非平凡的”对称密钥密码学是**必要且充分**的：单向函数与其他（非平凡的）对称密钥密码学的存在性是等价的。这是现代密码学的主要贡献之一。

本章展示的构造应该被视为对前一章讨论的流密码和分组密码构造的补充。前一章的重点是这些密码学原语在实践中如何实现，并介绍了所使用的基本方法和设计原则。然而，有些令人失望的是，我们展示的构造都无法基于任何更弱的（即，更合理的）假设来证明其安全性。相比之下，在本章中，我们将证明可以从**仅仅假设单向置换存在**的前提下，构建伪随机置换。这个假设比“假设AES是伪随机置换”的假设更温和，因为前者是一个性质上更弱的假设，而且我们拥有一些基于数论的、经过多年研究的候选单向函数（详见本章开头部分对这一点的进一步讨论）。然而，其缺点是，本章展示的构造效率远低于第六章的构造，因此在实践中并未被使用。弥合这一差距，开发出可证明安全、效率与现有流密码和分组密码相媲美的伪随机生成器、函数和置换的构造，仍然是密码学界面临的一项重要挑战。

抗碰撞哈希函数。 与前一章不同，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的方法是未知的，事实上，有证据表明这种构造是不可能的。我们将在§8.4.2中转向基于特定的数论假设的、可证明安全的抗碰撞哈希函数构造。

关于本章的说明。 本章的内容比本书的其余部分稍微高级一些。因为本章的内容在其他地方没有被明确使用，所以如果愿意，可以跳过本章。尽管如此，我们力求以一种（通过努力）可被高级本科生或初级研究生理解的方式来呈现这些材料。我们鼓励所有读者至少阅读§7.1和§7.2，它们介绍了单向函数并概述了本章的其余部分。我们认为，熟悉这里涵盖的一些主题非常重要，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是满足该定义的候选函数。接下来，我们将介绍硬核谓词（hard-core predicates）的概念，它可以被视为封装了单向函数求逆的困难性，并将在后续章节的构造中被广泛使用。

7.1.1 定义

一个单向函数 $f : \{0,1\}^* \to \{0,1\}^*$ 必须满足两个条件：易于计算，但难以求逆。第一个条件很容易形式化：我们将要求 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一个密码系统，使概率多项式时间（PPT）的攻击者无法以可忽略的概率破坏它，我们将要求第二个条件形式化为：任何概率多项式时间算法都无法以可忽略的概率求逆 f——即，找到给定值 y 的一个原像。一个技术细节是，这个概率是在这样一个实验中计算的：y 是通过从 f 的定义域中均匀选取一个元素 x，然后计算 $y := f(x)$ 而生成的（而不是通过从 f 的值域中均匀选取 y）。这样做的原因将在本章后续构造中变得清晰。

设 $f : \{0,1\}^* \to \{0,1\}^*$ 是一个函数。考虑为给定算法 A 和参数 n 定义的以下实验：

> **求逆实验 $\mathrm{Invert}_{A,f}(n)$：**
>
> 1. 均匀选取 $x \in \{0,1\}^n$，并计算 $y := f(x)$。
> 2. A 输入 1^n 和 y，输出 x'。
> 3. 如果 $f(x') = y$，则实验输出 1，否则输出 0。

我们强调 A 不需要找到原始的原像 x；A 只要找到任何满足 $f(x') = y = f(x)$ 的值 x' 就足够了。我们在第二步中给 A 输入 1^n，是为了强调 A 的运行时间可以是以安全参数 n 为多项式的，而不论 y 的长度是多少。

我们现在可以定义函数 f 是单向的意味着什么。

> **定义 7.1** 函数 $f : \{0,1\}^* \to \{0,1\}^*$ 是**单向的**，如果满足以下两个条件：
>
> 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. （难于求逆）：对于所有概率多项式时间算法 A，存在一个可忽略函数 negl，使得
>
> $$\Pr[\mathrm{Invert}_{A,f}(n) = 1] \leq \mathrm{negl}(n)$$

记法。 在本章中，我们经常通过对概率符号（probability notation）加下标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：对于所有概率多项式时间算法 A，存在一个可忽略函数 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 $x \leftarrow \{0,1\}^n$ 表示 x 是从 $\{0,1\}^n$ 中均匀选取的。）上述概率也包括了 A 所使用的随机性，但这里被隐去了。

单向函数的成功求逆。 一个**非**单向函数不一定在所有时候（甚至不一定“经常”）容易求逆。相反，定义 7.1 的第二个条件的逆是：存在一个概率多项式时间算法 A 和一个不可忽略函数 γ，使得 A 以至少 $\gamma(n)$ 的概率求逆 $f(x)$（其中概率是针对 $x \in \{0,1\}^n$ 的均匀选择和 A 的随机性）。这意味着存在一个正多项式 $p(\cdot)$，使得对于无穷多个 n 值，算法 A 以至少 $1/p(n)$ 的概率求逆 f。因此，如果存在一个算法 A，它对于所有偶数 n，以 n^{-10} 的概率求逆 f（但当 n 是奇数时总是失败），那么 f 就不是单向的——即使 A 仅在**一半**的 n 值上成功，并且仅以 n^{-10} 的概率成功（对于它成功的 n 值）。

单向置换。 我们通常对具有额外结构属性的单向函数感兴趣。我们说函数 f 是**保长**的，如果对于所有 x，$|f(x)| = |x|$。一个保长且一对一的单向函数称为**单向置换**。如果 f 是一个单向置换，那么对于任何值 y，它都有一个唯一的原像 $x = f^{-1}(y)$。尽管如此，在多项式时间内找到 x 仍然是困难的。

单向函数/置换族。 上述单向函数和单向置换的定义涉及一个在无限定义域和值域上的单一函数。然而，大多数候选单向函数和置换并不能完全符合这种框架。相反，通常存在一个算法 Gen，它生成定义函数 f_I 的一些参数 I；这里的单向性意味着 f_I 应该是单向的，除非以可忽略的概率选取 I。因为每个 I 值定义了一个不同的函数，所以我们现在称之为**单向函数族**（或**单向置换族**）。我们现在给出定义，并参考下一节了解一个具体例子（另见§8.4.1）。

> **定义 7.2** 由概率多项式时间算法组成的元组 $\Pi = (\mathrm{Gen}, \mathrm{Samp}, f)$ 是一个**函数族**，如果满足以下条件：
>
> 1. **参数生成算法 Gen**：输入 1^n，输出参数 I，且 $|I| \geq n$。Gen 输出的每个 I 值定义了一个集合 D_I 和 R_I，它们分别是函数 f_I 的定义域和值域。
> 2. **采样算法 Samp**：输入 I，输出 D_I 中的一个均匀分布的元素。
> 3. **确定性求值算法 f**：输入 I 和 $x \in D_I$，输出元素 $y \in R_I$。我们记为 $y := f_I(x)$。
>
> 如果对于 $\mathrm{Gen}(1^n)$ 输出的每个 I 值，$D_I = R_I$ 且函数 $f_I : D_I \to D_I$ 是一个双射（bijection），则 Π 是一个**置换族**。

设 Π 是一个函数族。下面是前面介绍的求逆实验的自然对应版本：

> **求逆实验 $\mathrm{Invert}_{A,\Pi}(n)$：**
>
> 1. 运行 $\mathrm{Gen}(1^n)$ 得到 I，然后运行 $\mathrm{Samp}(I)$ 得到 D_I 中的一个均匀 x。最后，计算 $y := f_I(x)$。
> 2. A 输入 I 和 y，输出 x'。
> 3. 如果 $f_I(x') = y$，则实验输出 1。

定义 7.3 函数/置换族 $\Pi = (\mathrm{Gen}, \mathrm{Samp}, f)$ 是**单向的**，如果对于所有概率多项式时间算法 A，存在一个可忽略函数 negl，使得

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

在整章中，我们主要使用**在无限定义域上的单向函数/置换**（如定义 7.1）来工作，而不是单向函数/置换族。这主要是为了方便，并不会显著影响任何结果。（见习题 7.7）。

7.1.2 候选单向函数

单向函数只有在它们存在时才有意义。我们不知道如何**无条件地**证明它们的存在性（这将是计算复杂性理论中的一个重大突破），所以我们必须猜想或假设它们的存在。这种猜想是基于这样一个事实：一些自然的计算问题受到了广泛关注，但至今仍没有找到解决它们的**多项式时间**算法。

最著名的此类问题可能是**整数分解**（integer factorization），即找到一个大整数的质因数。将两个数字相乘并得到它们的乘积是容易的，但从一个数中找出它的因子是困难的。这启发我们定义函数 $f_{\mathrm{mult}}(x, y) = x \cdot y$。如果我们不对 x 和 y 的长度施加任何限制，那么 f_{mult} 很容易求逆：以很高的概率，$x \cdot y$ 是偶数，在这种情况下，$(2, xy/2)$ 就是一个逆。这个问题可以通过限制 f_{mult} 的定义域为长度相等的质数 x 和 y 来解决。我们将在§8.2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**（subset-sum problem）定义的：

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

其中每个 x_i 是一个解释为整数的 n 位字符串，J 是一个解释为指定 $\{1, \ldots, n\}$ 的某个子集的 n 位字符串。对输出 $(x_1, \ldots, x_n, y)$ 求逆 f_{ss} 需要找到一个子集 $J' \subseteq \{1, \ldots, n\}$ 使得 $[\sum_{j \in J'} x_j \bmod 2^n] = y$。熟悉 NP-完全性的学生可能记得，这个问题是 NP-完全的。但即使 $P \neq NP$ 也不能直接证明 f_{ss} 是单向的：$P \neq NP$ 意味着所有多项式时间算法都会在**至少一个**输入上无法解决子集和问题，而要让 f_{ss} 是单向函数，则要求所有多项式时间算法几乎在所有时候都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：**即使在随机输入上**，也没有已知的算法能够以“较小”的概率解决这个问题，而不仅仅是基于该问题是 NP-完全的。

最后，我们通过展示一个被认为是单向的置换族来总结。设 Gen 是一个概率多项式时间算法，输入 1^n，输出一个 n 位质数 p 和一个特殊元素 $g \in \{2, \ldots, p-1\}$。（元素 g 应该是一个 Z_p^* 的生成元，详见§8.3.3）。设 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

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

（$f_{p,g}$ 可以被高效计算，详见附录§B.2.3）。可以证明这个函数是一对一的，因此是一个置换。该函数难以求逆的猜想是基于**离散对数问题**的困难性；我们将在§8.3中详细讨论。

最后，我们提醒，通过假设 SHA-1 是抗碰撞的或 AES 是伪随机置换，可以从实际的密码学构造中获得非常高效的单向函数（分别见习题 7.4和7.5）。（从技术上讲，它们不能满足单向性的定义，因为它们的输入/输出长度是固定的，所以我们无法观察它们的渐近行为。尽管如此，猜想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，一个单向函数是难以求逆的。换句话说：给定 $y = f(x)$，任何多项式时间算法都无法（除非以可忽略的概率）计算出 x 的全部。有人可能会认为，这意味着从 $f(x)$ 中无法在多项式时间内确定任何关于 x 的信息。但情况不一定如此。事实上，即使 f 是单向的，$f(x)$ 仍然可能"泄露"大量关于 x 的信息。举一个简单的例子，设 g 是一个单向函数，并定义 $f(x_1, x_2) := (x_1, g(x_2))$，其中 $|x_1| = |x_2|$。很容易证明 f 也是一个单向函数（见习题），尽管它泄露了其输入的一半！

对于我们的应用，我们需要确定一个关于 x 的特定信息位，它被 $f(x)$ 完美地"隐藏"了。这引出了**硬核谓词**（hard-core predicate）的概念。函数 f 的硬核谓词 $hc : \{0,1\}^* \to \{0,1\}$ 具有以下属性：给定 $f(x)$，计算 $hc(x)$ 的概率难以显著优于 $1/2$。（由于 hc 是一个布尔函数，通过随机猜测，总是可以以 $1/2$ 的概率计算出 $hc(x)$。）形式上：

> **定义 7.4** 函数 $hc : \{0,1\}^* \to \{0,1\}$ 是函数 f 的**硬核谓词**，如果 hc 可以在多项式时间内计算，并且对于所有概率多项式时间算法 A，存在一个可忽略函数 negl，使得
>
> $$\Pr_{x \leftarrow \{0,1\}^n}[A(1^n, f(x)) = hc(x)] \leq \frac{1}{2} + \mathrm{negl}(n)$$
>
> 其中概率是针对 x 在 $\{0,1\}^n$ 中的均匀选择和 A 的随机性。

我们强调，给定 x，$hc(x)$ 是可高效计算的（因为函数 hc 是在多项式时间内计算的）；定义要求**给定** $f(x)$，计算 $hc(x)$ 是困难的。上述定义没有要求 f 是单向的；然而，如果 f 是一个置换，那么除非 f 是单向的，否则它不能拥有一个硬核谓词。（见习题 7.13）。

简单的想法不起作用。 考虑谓词 $hc(x) := \bigoplus_{i=1}^{n} x_i$，其中 $x_1, \ldots, x_n$ 表示 x 的位。有人可能希望，对于任何单向函数 f，这都是一个硬核谓词：如果 f 不能求逆，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 的所有位的异或和 $hc(x)$ 是难以计算的。尽管这个

论点很吸引人，但它是**不正确**的。要看清这一点，设 g 是一个单向函数，并定义 $f(x) := (g(x), \bigoplus_{i=1}^{n} x_i)$。很容易证明 f 也是单向的。然而，$f(x)$ 显然没有隐藏 $hc(x) = \bigoplus_{i=1}^{n} x_i$ 的值，因为这是它的输出的一部分；因此，$hc(x)$ 不是 f 的硬核谓词。推而广之，可以证明对于任何固定的谓词 hc，都存在一个单向函数 f，使得 hc 不是 f 的硬核谓词。

平凡的硬核谓词。 有些函数拥有“平凡”的硬核谓词。例如，设 f 是丢弃其输入最后一位的函数（即 $f(x_1 \cdots x_n) = x_1 \cdots x_{n-1}$）。给定 $f(x)$，确定 x_n 是困难的，因为 x_n 独立于输出；因此，$hc(x) = x_n$ 是 f 的硬核谓词。然而，f 不是单向的。当我们用硬核谓词进行构造时，就会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仅基于任何单向函数/置换的假设来构建伪随机生成器、函数和置换。在本节中，我们概述了这些构造。详细内容将在后续章节给出。

从任何单向函数构造硬核谓词。 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，目前尚不清楚这是否成立；我们将证明一个较弱但足以满足我们需求的结果。即，我们将证明给定一个单向函数 f，我们可以构造另一个单向函数 g 以及 g 的一个硬核谓词。

> **定理 7.5 (Goldreich–Levin定理)** 假设单向函数（或置换）存在。那么存在一个单向函数（或置换）g 和 g 的一个硬核谓词 hc。

设 f 是一个单向函数。函数 g 和 hc 构造如下：设 $g(x, r) := (f(x), r)$，其中 $|x| = |r|$。并定义

$$hc(x, r) := \bigoplus_{i=1}^{n} x_i \cdot r_i$$

其中 $x = x_1 \cdots x_n$ 且 $r = r_1 \cdots r_n$。注意如果 r 是均匀的，$hc(x, r)$ 输出的是 x 的一个随机子集的位的异或和。（当 $r_i = 1$ 时，位 x_i 被包含在异或和中，否则不包含。）Goldreich–Levin定理本质上表明，如果 f 是一个单向函数，那么 $f(x)$ 隐藏了 x 的一个随机子集的位的异或和。

从单向置换构造伪随机生成器。 下一步是证明如何利用单向置换的硬核谓词来构造一个伪随机生成器。（众所周知，单向函数的硬核谓词也足够，但证明极其复杂，超出了本书的范围。）具体来说，我们证明：

> **定理 7.6** 设 f 是一个单向置换，hc 是 f 的一个硬核谓词。那么 $G(s) := f(s) \parallel hc(s)$

> 是一个具有膨胀因子 $\ell(n) = n + 1$ 的伪随机生成器。

关于为什么定理中定义的 G 构成一个伪随机生成器的直觉是：首先，$f(s)$ 的前 n 位输出（即 $f(s)$ 的位）是真正均匀分布的，因为 f 是一个置换。其次，由于 hc 是 f 的硬核谓词，这意味着 $hc(s)$ 看起来是"随机的"——即伪随机的——即使给定 $f(s)$（同样假设 s 是均匀的）。综合这些观察，我们看到 G 的整个输出是伪随机的。

具有任意膨胀因子的伪随机生成器。 即使存在一个将种子拉伸一位的伪随机生成器（如我们刚刚看到的），也已经是非常非平凡的了。但是对于实际应用（例如，为了高效地加密大消息，如§3.3），我们需要一个具有更大膨胀因子的伪随机生成器。幸运的是，我们可以获得任何我们想要的**多项式**膨胀因子：

> **定理 7.7** 如果存在一个膨胀因子为 $\ell(n) = n + 1$ 的伪随机生成器 G，那么对于任何多项式 poly，存在一个膨胀因子为 $\mathrm{poly}(n)$ 的伪随机生成器 $\hat{G}$。

我们得出结论：可以从任何单向置换构建具有任意（多项式）膨胀因子的伪随机生成器。

从伪随机生成器构造伪随机函数/置换。 伪随机生成器足以构建 EAV-安全的私钥加密方案。但是，为了实现 CPA-安全的私钥加密（更不用说消息认证码），我们需要依赖伪随机函数。以下结果表明，后者可以从前者构造：

> **定理 7.8** 如果存在一个膨胀因子为 $\ell(n) = 2n$ 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置换。

综合上述所有定理以及第三章和第四章的结果，我们得到以下推论：

> **推论 7.10** 假设单向置换存在，那么存在具有任何多项式膨胀因子的伪随机生成器、伪随机函数和强伪随机置换。

> **推论 7.11** 假设单向置换存在，那么存在 CPA-安全的私钥加密方案和安全的消息认证码。

如前所述，有可能仅基于单向函数的存在来获得所有这些结果。

7.3 从单向函数构造硬核谓词

在本节中，我们通过证明以下定理来证明定理 7.5：

定理 7.12 设 f 是一个单向函数，并定义 g 为 $g(x,r) := (f(x),r)$，其中 $|x| = |r|$。定义 $hc(x,r) := \bigoplus_{i=1}^{n} x_i \cdot r_i$，其中 $x = x_1 \cdots x_n$ 且 $r = r_1 \cdots r_n$。那么 hc 是 g 的一个硬核谓词。

由于证明的复杂性，我们逐步证明三个渐进增强的结果，最终达到定理中声称的结果。

7.3.1 一个简单情况

我们首先证明，如果存在一个多项式时间算法 A，它**总是**正确计算给定 $g(x,r) = (f(x),r)$ 的 $hc(x,r)$，那么就可以在多项式时间内求逆 f。根据 f 是单向函数的假设，可以得出不存在这样的算法 A。

命题 7.13 设 f 和 hc 如定理 7.12所述。如果存在一个多项式时间算法 A，使得对于所有 n 和所有 $x,r \in \{0,1\}^n$，$A(f(x),r) = hc(x,r)$，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。$A'(1^n, y)$ 对于 $i = 1, \ldots, n$ 计算 $x_i := A(y, e_i)$，其中 e_i 表示第 i 位是 1 而其他位都是 0 的 n 位字符串。然后 A' 输出 $x = x_1 \cdots x_n$。显然 A' 在多项式时间内运行。

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

$$x_i = A(f(\hat{x}), e_i) = hc(\hat{x}, e_i) = \bigoplus_{j=1}^{n} \hat{x}_j \cdot e_{i,j} = \hat{x}_i$$

因此，对于所有 i，$x_i = \hat{x}_i$，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不可忽略的概率求逆 f。因此，我们得出结论：不存在一个多项式时间算法可以**总是**根据 $(f(x),r)$ 正确计算 $hc(x,r)$。这是一个相当弱的结果，距离我们最终的目标——证明给定 $(f(x),r)$，计算 $hc(x,r)$ 的概率难以显著优于 $1/2$——还很遥远。

7.3.2 一个更复杂的情况

我们现在证明，对于任何概率多项式时间算法 A，根据 $(f(x),r)$ 计算 $hc(x,r)$ 的概率难以显著优于 $3/4$。我们再次证明，任何这样的 A 将意味着存在一个概率不可忽略的多项式时间算法 A' 来求逆 f。

注意，在命题 7.13的证明中，我们前面使用的策略在这里失败了，因为 A 可能在 $r = e_i$ 时从

未成功（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 $A(f(x),r)$ 是否等于 $hc(x,r)$；A' 唯一知道的是，算法 A 以高概率是正确的。这使证明更加复杂。

命题 7.14 设 f 和 hc 如定理 7.12所述。如果存在一个概率多项式时间算法 A 和一个多项式 $p(\cdot)$，使得对于无穷多个 n 值，

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

那么存在一个概率多项式时间算法 A' 和一个多项式 $p'(\cdot)$，使得对于无穷多个 n 值，

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

证明 支撑本命题证明的核心观察是：对于任何 $r\in\{0,1\}^n$，可以利用 $hc(x,r)$ 和 $hc(x,r\oplus e_i)$ 一起计算 x 的第 i 位。（回想 e_i 表示 n 位字符串，其第 i 位是 1。）这成立是因为

$$hc(x,r)\oplus hc(x,r\oplus e_i)$$

$$=\left(\bigoplus_{j=1}^{n}x_j\cdot r_j\right)\oplus\left(\bigoplus_{j=1}^{n}x_j\cdot(r_j\oplus e_{i,j})\right)=x_i\cdot r_i\oplus x_i\cdot\bar{r}_i=x_i$$

其中第二个等式是由于对于 $j\neq i$，$x_j\cdot r_j$ 出现在两个求和中，因此被抵消了。

上述表明，如果 A 在 $(f(x),r)$ 和 $(f(x),r\oplus e_i)$ 上都回答正确，那么 A' 就可以正确计算 x_i。不幸的是，A' 不知道 A 何时回答正确，何时不正确；A' 唯一知道的是 A 回答正确的概率很高。因此，A' 将使用多个随机的 r 值，用每个值来估计 x_i，然后将出现次数最多的估计值作为对 x_i 的最终猜测。

作为预备步骤，我们首先证明对于许多 x 值，当 r 是均匀选取时，A 在 $(f(x),r)$ 和 $(f(x),r\oplus e_i)$ 上都回答正确的概率足够高。这使得我们能够固定 x，然后仅关注 r 的均匀选择，从而使分析更容易。

断言 7.15 设 n 满足 $\Pr_{x,r\leftarrow\{0,1\}^n}[A(f(x),r)=hc(x,r)]\geq\frac{3}{4}+\frac{1}{p(n)}$。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 $x\in S_n$，有

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

证明 设 $\epsilon(n) = 1/p(n)$，并定义 $S_n \subseteq \{0,1\}^n$ 为所有满足 $\Pr_{r \leftarrow \{0,1\}^n}[A(f(x), r) = hc(x, r)] \geq \frac{3}{4} + \frac{\epsilon(n)}{2}$ 的 x 的集合。我们有

$$\Pr_{x,r \leftarrow \{0,1\}^n}[A(f(x), r) = hc(x, r)] = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} \Pr_{r \leftarrow \{0,1\}^n}[A(f(x), r) = hc(x, r)]$$

$$= \frac{1}{2^n} \sum_{x \in S_n} \Pr_{r \leftarrow \{0,1\}^n}[\cdots] + \frac{1}{2^n} \sum_{x \notin S_n} \Pr_{r \leftarrow \{0,1\}^n}[\cdots]$$

$$\leq \frac{|S_n|}{2^n} \cdot 1 + \frac{1}{2^n} \sum_{x \notin S_n} \left(\frac{3}{4} + \frac{\epsilon(n)}{2} \right) \leq \frac{|S_n|}{2^n} + \left(\frac{3}{4} + \frac{\epsilon(n)}{2} \right)$$

由于 $\frac{3}{4} + \epsilon(n) \leq \Pr_{x,r \leftarrow \{0,1\}^n}[A(f(x), r) = hc(x, r)]$，简单的代数运算得到 $|S_n| \geq \frac{\epsilon(n)}{2} \cdot 2^n$。

下面是前面断言的一个简单推论。

> **断言 7.16** 设 n 满足 $\Pr_{x,r \leftarrow \{0,1\}^n}[A(f(x), r) = hc(x, r)] \geq \frac{3}{4} + \frac{1}{p(n)}$。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 $x \in S_n$ 和所有 i，有
>
> $$\Pr_{r \leftarrow \{0,1\}^n}[A(f(x), r) = hc(x, r) \wedge A(f(x), r \oplus e_i) = hc(x, r \oplus e_i)] \geq \frac{1}{2} + \frac{1}{p(n)}$$

证明 设 $\epsilon(n) = 1/p(n)$，取 S_n 为前一断言保证的集合。我们知道对于任何 $x \in S_n$，$\Pr_{r \leftarrow \{0,1\}^n}[A(f(x), r) \neq hc(x, r)] \leq \frac{1}{4} - \frac{\epsilon(n)}{2}$。

固定 $i \in \{1, \ldots, n\}$。如果 r 是均匀分布的，那么 $r \oplus e_i$ 也是均匀分布的；因此，$\Pr_{r \leftarrow \{0,1\}^n}[A(f(x), r \oplus e_i) \neq hc(x, r \oplus e_i)] \leq \frac{1}{4} - \frac{\epsilon(n)}{2}$。

我们感兴趣的是 A 在 $hc(x, r)$ 和 $hc(x, r \oplus e_i)$ 上都回答正确的概率；等价地，我们想要限制 A 在任何一种情况下未能回答正确的概率。注意 r 和 $r \oplus e_i$ 并非独立，但我们可以应用**联合界**（联合上限，见命题 A.7）来对失败的概率求和。也就是说，A 在 $hc(x, r)$ 或 $hc(x, r \oplus e_i)$ 上回答不正确的概率至多为

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

因此，A 在 $hc(x, r)$ 和 $hc(x, r \oplus e_i)$ 上都回答正确的概率至少为 $\frac{1}{2} + \epsilon(n)$。

对于证明的其余部分，我们设 $\epsilon(n) = 1/p(n)$，并仅考虑满足 $\Pr_{x,r \leftarrow \{0,1\}^n}[A(f(x), r) =$

$hc(x,r)] \geq \frac{3}{4} + \epsilon(n)$ 的 n 值。前面的断言指出，对于 $\frac{\epsilon(n)}{2}$ 比例的输入 x，对于任何 i，算法 A 在 r 的均匀选择上，在 $(f(x), r)$ 和 $(f(x), r \oplus e_i)$ 上都正确回答的概率至少为 $\frac{1}{2} + \epsilon(n)$。从现在开始，我们只关注这样的 x 值。我们构造一个概率多项式时间算法 A'，当 $x \in S_n$ 时，它以至少 $1/2$ 的概率求逆 $f(x)$。这足以证明命题 7.14，因为在这种情况下，对于无穷多个 n 值，

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

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

$$\geq \frac{1}{2} \cdot \frac{\epsilon(n)}{2} = \frac{1}{4p(n)}$$

算法 A' 输入 1^n 和 y，运行如下：

1. 对于 $i = 1, \ldots, n$：
   - 重复选择均匀 $r \in \{0,1\}^n$，并计算 $A(y, r) \oplus A(y, r \oplus e_i)$ 作为对原像的第 i 位的“估计”。在进行足够多次后（见下文），设 x_i 为出现次数最多的“估计”值。
2. 输出 $x = x_1 \cdots x_n$。

我们简要分析 A' 正确求逆其给定输入 y 的概率。（我们将允许自己略微简略，因为一个更困难情况的完整证明将在下一节给出。）设 $y = f(\hat{x})$，我们假设 n 满足方程 (7.1) 且 $\hat{x} \in S_n$。固定某个 i。前面的断言表明，估计值 $A(y, r) \oplus A(y, r \oplus e_i)$ 等于 $hc(\hat{x}, e_i)$ 的概率至少为 $\frac{1}{2} + \epsilon(n)$。通过获得足够多的估计值并取 x_i 为多数值， A' 可以确保 x_i 等于 $hc(\hat{x}, e_i)$ 的概率至少为 $1 - \frac{1}{2n}$。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 $\epsilon(n) = 1/p(n)$ 且对于获得每个估计值都使用了独立的 r 值，切尔诺夫界（见命题 A.14）表明多项式数量的估计值就足够了。

总结来说，我们有 A' 计算的每个 x_i 不正确的概率至多为 $\frac{1}{2n}$。应用联合界，我们看到 A' 对于某些 i 不正确的概率至多为 $n \cdot \frac{1}{2n} = \frac{1}{2}$。也就是说，$A'$ 对于所有 i 都是正确的（因此正确求逆 y）的概率至少为 $1 - \frac{1}{2} = \frac{1}{2}$。这完成了命题 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，给定 $(f(x), r)$， A 正确猜测 $hc(x, r)$ 的概率至多比 $3/4$ 多一个可忽略函数。

7.3.3 完整的证明

我们假设读者熟悉前面几节中的简化证明，并在此基础上发展。我们将依赖于附录§A.3中讨论的概率论中的一些术语和标准结果。

我们证明以下命题，它蕴含了定理 7.12：

命题 7.17 设 f 和 hc 如定理 7.12所述。如果存在一个概率多项式时间算法 A 和一个多项式 $p(\cdot)$，使得对于无穷多个 n 值，

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

那么存在一个概率多项式时间算法 A' 和一个多项式 $p'(\cdot)$，使得对于无穷多个 n 值，

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

证明 我们再次设 $\epsilon(n)=1/p(n)$，并仅考虑满足 $\Pr_{x,r\leftarrow\{0,1\}^n}[A(f(x),r)=hc(x,r)]\geq\frac{1}{2}+\epsilon(n)$ 的 n 值。

断言 7.18 设 n 满足 $\Pr_{x,r\leftarrow\{0,1\}^n}[A(f(x),r)=hc(x,r)]\geq\frac{1}{2}+\epsilon(n)$。那么存在一个集合 $S_n\subseteq\{0,1\}^n$，其大小至少为 $\frac{\epsilon(n)}{2}\cdot 2^n$，使得对于 $x\in S_n$，有

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

如果试图证明断言 7.16的一个类比结果，我们在这里能声称的最好结果是：当 $x\in S_n$ 时，$\Pr_{r\leftarrow\{0,1\}^n}[A(f(x),r)=hc(x,r)\wedge A(f(x),r\oplus e_i)=hc(x,r\oplus e_i)]\geq\epsilon(n)$ 对于所有 i 成立。因此，如果我们试图使用 $A(f(x),r)\oplus A(f(x),r\oplus e_i)$ 作为对 x_i 的估计，我们能声称的只是这个估计正确的概率至少为 $\epsilon(n)$，这可能不比随机猜测好！（因为 $\epsilon(n)$ 可能很小）。我们不能声称翻转结果会得到一个好的估计。

相反，我们设计 A'，使其仅通过调用 A **一次**来计算 $hc(x,r)$ 和 $hc(x,r\oplus e_i)$。我们通过让 A' 运行 $A(x,r\oplus e_i)$，并让 A' **猜测** $hc(x,r)$ 的值来实现。这样做的一个问题是，如果我们像前面那样独立地选择多项式数量的 r，那么所有猜测都正确的概率将是可忽略的。

本证明的关键观察是：A' 可以**成对独立地**生成 r 值并以特定的方式进行猜测，从而确保所有猜测都正确的概率是**不可忽略**的。具体来说，为了生成 m 个 r 值，我们让 A' 选择 $\ell=\lceil\log(m+1)\rceil$ 个独立均匀的字符串 $s_1,\ldots,s_\ell\in\{0,1\}^n$。然后，对于 $\{1,\ldots,\ell\}$ 的每个非空子集 I，我们设 $r_I:=\bigoplus_{i\in I}s_i$。由于有 $2^\ell-1\geq m$ 个非空子集，这定义了 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**成对独立**的。要看清这一点，注意对于任何两个不同的子集 $I\neq J$，存在一个索引 $j\in I\cup J$ 使得 $j\notin I\cap J$。不妨设 $j\notin I$。那么 s_j 的值是均匀的，且独立于 r_I 的值。由于 s_j 被包含在定义 r_J 的异或和中，这表明 r_J 也是均匀的，且独立于 r_I。

现在我们有了两个重要的观察：

1. 给定 $hc(x, s_1), \ldots, hc(x, s_\ell)$，对于 $\{1, \ldots, \ell\}$ 的每个子集 I，可以计算 $hc(x, r_I)$。这成立是因为

$$hc(x, r_I) = hc(x, \bigoplus_{i \in I} s_i) = \bigoplus_{i \in I} hc(x, s_i)$$

2. 如果 A' 简单地通过选择一个均匀的位来**猜测** $hc(x, s_1), \ldots, hc(x, s_\ell)$ 的值，那么所有这些猜测都正确的概率为 $1/2^\ell$。如果 m 是安全参数 n 的多项式，那么 $1/2^\ell$ 是不可忽略的，因此 A' 以**不可忽略的概率**正确猜出所有值 $hc(x, s_1), \ldots, hc(x, s_\ell)$。

综合上述观察，我们得到了一种以不可忽略的概率获得 $m = \mathrm{poly}(n)$ 个均匀且成对独立的字符串 $\{r_I\}$ 以及 $\{hc(x, r_I)\}$ 的正确值的方法。然后，这些值可以用来像命题 7.14的证明中那样计算 x_i。细节如下。

求逆算法 A'。 我们给出算法 A' 的完整描述，它输入 1^n 和 y，并尝试计算 y 的一个逆。算法运行如下：

1. 设 $\ell := \lceil \log(2n/\epsilon(n)^2 + 1) \rceil$。
2. 选择 ℓ 个均匀且独立的 $s_1, \ldots, s_\ell \in \{0,1\}^n$ 和 $\sigma_1, \ldots, \sigma_\ell \in \{0,1\}$。
3. 对于每个非空子集 $I \subseteq \{1, \ldots, \ell\}$，计算 $r_I := \bigoplus_{i \in I} s_i$ 和 $\sigma_I := \bigoplus_{i \in I} \sigma_i$。
4. 对于 $i = 1, \ldots, n$：
   - 对于每个非空子集 $I \subseteq \{1, \ldots, \ell\}$，设 $x_I^{(i)} := \sigma_I \oplus A(y, r_I \oplus e_i)$。
   - 设 $x_i := \mathrm{majority}_I \{x_I^{(i)}\}$（即，取上一步中出现次数最多的位）。
5. 输出 $x = x_1 \cdots x_n$。

现在我们计算 A' 输出 $x \in f^{-1}(y)$ 的概率。像命题 7.14的证明中一样，我们只关注 n 满足断言 7.18条件的那些值，并假设 $y = f(\hat{x})$ 且 $\hat{x} \in S_n$。设 E_σ 为所有猜测 σ_i 都是正确的事件。如前所述，$\Pr[E_\sigma] \geq 1/2^\ell$，这是不可忽略的。以 E_σ 为条件，$\sigma_I = hc(\hat{x}, r_I)$ 对于所有 I 成立。

固定一个索引 $i \in \{1, \ldots, n\}$。对于任何非空 I，估计值 $A(y, r_I \oplus e_i)$ 等于 $hc(\hat{x}, r_I \oplus e_i)$ 的概率至少为 $\frac{1}{2} + \frac{\epsilon(n)}{2}$（在 r_I 的选择上），这成立是因为 $\hat{x} \in S_n$ 且 $r_I \oplus e_i$ 是均匀分布的。因此，$\Pr[x_I^{(i)} = \hat{x}_i \mid E_\sigma] \geq \frac{1}{2} + \frac{\epsilon(n)}{2}$。

由于 $\{x_I^{(i)}\}_{I \subseteq \{1,\ldots,\ell\}}$ 是成对独立的（因为 $\{r_I \oplus e_i\}_{I \subseteq \{1,\ldots,\ell\}}$ 是成对独立的），我们可以应用命题 A.13：

$$\Pr[x_i \neq \hat{x}_i \mid E_\sigma] \leq \frac{1}{4 \cdot (\epsilon(n)/2)^2 \cdot (2^\ell - 1)} \leq \frac{1}{4 \cdot (\epsilon(n)/2)^2 \cdot (2n/\epsilon(n)^2)} = \frac{1}{4n}$$

上述对所有 i 都成立，所以应用联合界，我们看到 A' 输出不正确的概率至多为 $n \cdot \frac{1}{4n} = \frac{1}{4}$。也就是说，在事件 E_σ 发生的条件下，$x = \hat{x}$ 的概率至少为 $1 - \frac{1}{4} = \frac{3}{4}$。

综合所有：设 n 满足断言 7.18的条件，$y = f(\hat{x})$。$\hat{x} \in S_n$ 的概率至少为 $\epsilon(n)/2$。所有猜测 σ_i 都正确的概率至少为 $1/2^\ell$。**如果** $\hat{x} \in S_n$ **且** E_σ 发生，则 A' 以至少 $3/4$ 的概率输出 $x = \hat{x}$。

A' 正确求逆其输入的总概率因此至少为

$$\Pr[x = \hat{x}] \geq \Pr[x = \hat{x} \mid \hat{x} \in S_n \wedge E_\sigma] \cdot \Pr[\hat{x} \in S_n \wedge E_\sigma] \geq \frac{3}{4} \cdot \frac{\epsilon(n)}{2} \cdot \frac{1}{2^\ell}$$

$$\geq \frac{3}{4} \cdot \frac{\epsilon(n)}{2} \cdot \frac{1}{2 \cdot (2n/\epsilon(n)^2 + 1)} \approx \frac{3}{4} \cdot \frac{\epsilon(n)}{2} \cdot \frac{\epsilon(n)^2}{4n} = \frac{3\epsilon(n)^3}{32n}$$

由于 $3\epsilon(n)^3/(32n)$ 是一个多项式的逆，这就证明了命题 7.17。

7.4 构造伪随机生成器

我们首先展示如何仅在假设单向置换存在的前提下，构造将输入拉伸一位的伪随机生成器。然后我们展示如何扩展它以获得任意多项式膨胀因子。

7.4.1 具有最小膨胀的伪随机生成器

设 f 是一个具有硬核谓词 hc 的单向置换。这意味着，给定 $f(s)$，$hc(s)$ 看起来是“随机的”（假设 s 是均匀的）。此外，由于 f 是一个置换， $f(s)$ 本身就是均匀分布的。（对均匀分布的值应用置换，结果仍然是均匀分布的。）因此，如果 s 是一个均匀的 n 位字符串，那么 $(n+1)$ 位字符串 $f(s) \parallel hc(s)$ 由一个均匀的 n 位字符串加上一个附加位组成，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，定义 $G(s) = f(s) \parallel hc(s)$ 的算法是一个伪随机生成器。

> **定理 7.19** 设 f 是一个单向置换，hc 是 f 的一个硬核谓词。那么定义 $G(s) = f(s) \parallel hc(s)$ 的算法 G 是一个具有膨胀因子 $\ell(n) = n + 1$ 的伪随机生成器。

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

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

类似的论证表明存在一个可忽略函数 negl'，使得

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

这就完成了证明。

首先观察

$$\Pr_{r\leftarrow\{0,1\}^{n+1}}[D(r)=1]=\Pr_{r\leftarrow\{0,1\}^n,r'\leftarrow\{0,1\}}[D(r\parallel r')=1]$$

$$=\Pr_{s\leftarrow\{0,1\}^n,r'\leftarrow\{0,1\}}[D(f(s)\parallel r')=1]$$

$$=\frac{1}{2}\cdot\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)\parallel hc(s))=1]+\frac{1}{2}\cdot\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)\parallel \overline{hc(s)})=1]$$

其中第二个等式是由于 f 是置换，第三个等式是由于均匀位 r' 等于 $hc(s)$ 的概率恰好是 $1/2$。由于 $\Pr_{s\leftarrow\{0,1\}^n}[D(G(s))=1]=\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)\parallel hc(s))=1]$（根据 G 的定义），这意味着证明上式等价于证明存在可忽略函数 negl'' 使得

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

考虑以下算法 A，它输入一个值 $y=f(s)$ 并尝试预测 $hc(s)$：

1. 选择均匀 $r'\in\{0,1\}$。
2. 运行 $D(y\parallel r')$。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

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

$$=\frac{1}{2}\cdot\Pr_{s\leftarrow\{0,1\}^n}[A(f(s))=hc(s)\mid r'=hc(s)]+\frac{1}{2}\cdot\Pr_{s\leftarrow\{0,1\}^n}[A(f(s))=hc(s)\mid r'\neq hc(s)]$$

$$=\frac{1}{2}\cdot\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)\parallel hc(s))=0]+\frac{1}{2}\cdot\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)\parallel \overline{hc(s)})=1]$$

$$=\frac{1}{2}\cdot\left(1-\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)\parallel hc(s))=1]\right)+\frac{1}{2}\cdot\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)\parallel \overline{hc(s)})=1]$$

$$=\frac{1}{2}+\frac{1}{2}\cdot\left(\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)\parallel \overline{hc(s)})=1]-\Pr_{s\leftarrow\{0,1\}^n}[D(f(s)\parallel hc(s))=1]\right)$$

由于 hc 是 f 的硬核谓词，因此存在一个可忽略函数 negl，使得

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

这就证明了上述等式成立。

7.4.2 增加膨胀因子

我们现在证明，伪随机生成器的膨胀因子可以增加任意**多项式**数量。这意味着前一个构造（膨胀因子为 $n+1$）足以构造具有任意（多项式）膨胀因子的伪随机生成器。

> **定理 7.20** 如果存在一个膨胀因子为 $n+1$ 的伪随机生成器 G，那么对于任何多项式 poly，存在一个膨胀因子为 $\mathrm{poly}(n)$ 的伪随机生成器 $\hat{G}$。

证明 我们首先考虑构造一个输出 $n+2$ 位的伪随机生成器 $\hat{G}$。$\hat{G}$ 的工作方式如下：给定一个初始种子 $s \in \{0,1\}^n$，它计算 $t_1 := G(s)$ 得到 $n+1$ 个伪随机位。t_1 的前 n 位被再次用作 G 的种子；将 G 得到的 $n+1$ 位与 t_1 的最后一位连接起来，就得到了 $n+2$ 位的输出。（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。我们给出的安全性证明表明，这并不影响输出的伪随机性。

我们现在证明 $\hat{G}$ 是一个伪随机生成器。定义三个分布序列 $\{H_n^0\}_{n=1,\ldots}, \{H_n^1\}_{n=1,\ldots}, \{H_n^2\}_{n=1,\ldots}$，其中 H_n^0, H_n^1 和 H_n^2 都是 $n+2$ 位字符串的分布。在分布 H_n^0 中，选择一个均匀字符串 $t_0 \in \{0,1\}^n$，输出 $\hat{G}(t_0)$。在分布 H_n^1 中，选择一个均匀字符串 $t_1 \in \{0,1\}^{n+1}$，并将其解析为 $s_1 \parallel \sigma_1$（其中 s_1 是 t_1 的前 n 位，σ_1 是最后一位）。输出是 $t_2 := G(s_1) \parallel \sigma_1$。在分布 H_n^2 中，输出是一个均匀字符串 $t_2 \in \{0,1\}^{n+2}$。我们用 $t_2 \leftarrow H_n^i$ 表示根据分布 H_n^i 生成一个 $(n+2)$ 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。我们首先证明存在一个可忽略函数 negl_0，使得

$$\left| \Pr_{t_2 \leftarrow H_n^0}[D(t_2) = 1] - \Pr_{t_2 \leftarrow H_n^1}[D(t_2) = 1] \right| \leq \mathrm{negl}_0(n)$$

要看到这一点，考虑以下多项式时间区分器 D'：输入 $t_1 \in \{0,1\}^{n+1}$，将其解析为 $s_1 \parallel \sigma_1$，其中 $|s_1| = n$，计算 $t_2 := G(s_1) \parallel \sigma_1$，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，则 D' 生成的 t_2 的分布恰好是分布 H_n^1。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 $s \in \{0,1\}^n$，则 D' 生成的 t_2 的分布恰好是分布 H_n^0。即

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

G 的伪随机性意味着存在一个可忽略函数 negl_0，使得

$$\left|\Pr_{s \leftarrow \{0,1\}^n}[D'(G(s)) = 1] - \Pr_{t_1 \leftarrow \{0,1\}^{n+1}}[D'(t_1) = 1]\right| \leq \mathrm{negl}_0(n)$$

这就得到了上述等式。

接下来，我们证明存在一个可忽略函数 negl_{00}，使得

$$\left|\Pr_{t_2 \leftarrow H_n^1}[D(t_2) = 1] - \Pr_{t_2 \leftarrow H_n^2}[D(t_2) = 1]\right| \leq \mathrm{negl}_{00}(n)$$

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

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

和以前一样，G 的伪随机性蕴含了上述等式。

综合所有，我们有

$$\begin{aligned}
&\left|\Pr_{s \leftarrow \{0,1\}^n}[D(\hat{G}(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{n+2}}[D(r) = 1]\right| \\
&= \left|\Pr_{t_2 \leftarrow H_n^0}[D(t_2) = 1] - \Pr_{t_2 \leftarrow H_n^2}[D(t_2) = 1]\right| \\
&\leq \left|\Pr_{t_2 \leftarrow H_n^0}[D(t_2) = 1] - \Pr_{t_2 \leftarrow H_n^1}[D(t_2) = 1]\right| + \left|\Pr_{t_2 \leftarrow H_n^1}[D(t_2) = 1] - \Pr_{t_2 \leftarrow H_n^2}[D(t_2) = 1]\right| \\
&\leq \mathrm{negl}_0(n) + \mathrm{negl}_{00}(n)
\end{aligned}$$

使用上述两个等式。由于 D 是任意概率多项式时间区分器，这证明了 $\hat{G}$ 是一个伪随机生成器。

一般情况。 可以迭代应用上述想法来生成所需数量的伪随机位。形式上，设我们希望构造一个膨胀因子为 $n + p(n)$ 的伪随机生成器 $\hat{G}$，其中 p 是某个多项式。输入 $s \in \{0,1\}^n$，算法 $\hat{G}$ 运行如下（参照图 7.1）：

1. 设 $t_0 := s$。对于 $i = 1, \ldots, p(n)$：
   - 设 s'_{i-1} 是 t_{i-1} 的前 n 位，σ_{i-1} 是剩下的 $i-1$ 位。（当 $i = 1$ 时，$s'_0 = t_0$ 且 σ_0

是空串。）
- 设 $t_i := G(s'_{i-1}) \parallel \sigma_{i-1}$。

2. 输出 $t_{p(n)}$。

我们证明 $\hat{G}$ 是一个伪随机生成器。该证明使用了**混合论证**（hybrid argument）这一常用技术。（实际上，即使是前面 $p(n) = 2$ 的情况，也使用了简单的混合论证。）与前一个证明的主要区别是技术上的：我们不能像以前那样定义和明确地处理三个分布序列 $\{H_n^0\}, \{H_n^1\}, \{H_n^2\}$。

对于任何 n 和 $0 \le j \le p(n)$，设 H_n^j 是 $(n + p(n))$ 位字符串的分布，定义如下：选择均匀 $t_j \in \{0,1\}^{n+j}$，然后从第 $j+1$ 次迭代开始运行 $\hat{G}$，输出 $t_{p(n)}$。（当 $j = p(n)$ 时，这意味着我们只是选择均匀 $t_{p(n)} \in \{0,1\}^{n+p(n)}$ 并输出它。）关键的观察是 H_n^0 对应于输出 $\hat{G}(s)$，其中 $s \in \{0,1\}^n$ 是均匀的，而 $H_n^{p(n)}$ 对应于输出一个均匀的 $(n + p(n))$ 位字符串。固定任何多项式时间区分器 D，这意味着

$$\left|\Pr_{s \leftarrow \{0,1\}^n}[D(\hat{G}(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{n+p(n)}}[D(r) = 1]\right|$$

$$= \left|\Pr_{t \leftarrow H_n^0}[D(t) = 1] - \Pr_{t \leftarrow H_n^{p(n)}}[D(t) = 1]\right|$$

我们证明上述差值是可忽略的，因此 $\hat{G}$ 是一个伪随机生成器。

固定 D 并考虑以下区分器 D'：输入 $w \in \{0,1\}^{n+1}$，它运行如下：

1. 选择均匀 $j \in \{1, \ldots, p(n)\}$。
2. 选择均匀且独立的 $\sigma'_0 \in \{0,1\}^{j-1}$。（当 $j = 1$ 时，σ'_0 是空串。）
3. 设 $t_j := w \parallel \sigma'_0$。然后从第 $j+1$ 次迭代开始运行 $\hat{G}$ 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，但基本思想是相同的。固定 n 并设 D' 选择 $j = j^*$。如果 w 是均匀的，则 t_{j^*} 是均匀的，因此 $t = t_{p(n)}$ 的分布恰好是分布 $H_n^{j^*}$。因此

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

另一方面，设 D' 选择 $j = j^*$ 且 $w = G(s)$ 对于均匀 $s \in \{0,1\}^n$。设 $t_{j^*-1} = s \parallel \sigma'_0$。那么 t_{j^*-1} 是均匀的，因此 $t = t_{p(n)}$ 的分布恰好是分布 $H_n^{j^*-1}$。因此

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

我们现在可以分析 D' 区分 G 的输出和随机串的能力：

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

其中第二个等式成立是因为除了左边求和的第一项和右边求和的最后一项外，每个求和中都包含了相同的项。由于 G 是一个伪随机生成器，上式左侧必须是可忽略的；因为 $p(n)$ 是多项式，这蕴含了 $\left|\Pr_{t\leftarrow H_n^0}[D(t)=1]-\Pr_{t\leftarrow H_n^{p(n)}}[D(t)=1]\right|$ 也是可忽略的。这就完成了证明。

总结。 设 f 是一个单向置换。我们采用定理 7.19中具有 $n+1$ 膨胀因子的伪随机生成器，并使用定理 7.20的证明中方法将其膨胀因子增加到 $n+\ell$。我们得到以下伪随机生成器 $\hat{G}$：

$$\hat{G}(s)=f^{(\ell)}(s)\parallel hc(f^{(\ell-1)}(s))\parallel\cdots\parallel hc(s)$$

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

与流密码的联系。 回想§3.3.1中的流密码（没有 IV）是由算法 $(\mathrm{Init},\mathrm{GetBits})$ 定义的，其中 Init 输入一个种子 $s\in\{0,1\}^n$ 并返回初始状态 st，$\mathrm{GetBits}$ 输入当前状态 st 并输出一位 σ 和更新后的状态 st'。前一个证明中得到的构造 $\hat{G}$ 非常符合这种范式：取 Init 为输出 $\mathrm{st}=s$ 的平凡算法，并定义 $\mathrm{GetBits}(\mathrm{st})$ 来计算 $G(\mathrm{st})$，将其解析为 $\mathrm{st}'\parallel\sigma$，其中 $|\mathrm{st}'|=n$，并输出位 σ 和更新后的状态 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，那么我们得到的恰好是 $\hat{G}(s)$ 的最后 $p(n)$ 位，但顺序是反的。）前面的证明表明，这构成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，就会用到它。非正式地说，该技术通过定义一系列中间的“混合分布”，在两个我们希望证明其不可区分的“极端分布”之间架起一座桥梁。（在上面的证明中，这些极端分布对应于 $\hat{G}$ 的输出和一个随机字符串。）为了应用该证明技术，需要满足三个条件。首先，极端分布应该与最初感兴趣的情况相匹配。（在上面的证明中，H_n^0 等于 $\hat{G}$ 引起的分布，而 $H_n^{p(n)}$ 是

均匀分布。）其次，必须能够将区分连续混合分布的能力转化为破坏某个基础假设。（在上面，我们本质上证明了区分 H_n^i 和 H_n^{i+1} 等价于区分 G 的输出和随机串。）最后，混合分布的数量应该是多项式的。（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（长度翻倍）伪随机生成器构造一个伪随机函数。回想§3.5.1中的伪随机函数是一个可高效计算的、带密钥的函数 F，它与一个真正随机函数不可区分。为了简单起见，我们在此将范围限制在**保长**的伪随机函数上，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（保长）伪随机函数可以被非正式地视为一个膨胀因子为 $n \cdot 2^n$ 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；这里有指数多的块，我们需要一种计算第 i 个块的方法，而不需要计算所有其他块。

我们将通过沿着二叉树向下遍历来计算输出的“块”。我们首先展示一个接受 2 位输入的伪随机函数的构造示例。设 G 是一个膨胀因子为 $2n$ 的伪随机生成器。如果我们像定理 7.20的证明中那样使用 G，我们可以得到一个使用三次 G 调用的膨胀因子为 $4n$ 的伪随机生成器 $\hat{G}$。如果我们定义 $F'_k(i)$（其中 $0 \leq i < 4$ 且 i 被编码为一个 2 位二进制字符串）为 $\hat{G}(k)$ 的第 i 个块，那么计算 $F'_k(3)$ 将需要计算整个 $\hat{G}$ 的输出，因此需要三次调用 G。我们将展示如何对于任何输入，只使用两次调用 G 来构造伪随机函数 F。

设 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 $G(k) = G_0(k) \,\|\, G_1(k)$ 且 $|G_0(k)| = |G_1(k)| = |k|$。定义 F 如下：

$$F_k(00) = G_0(G_0(k)) \quad F_k(10) = G_0(G_1(k))$$

$$F_k(01) = G_1(G_0(k)) \quad F_k(11) = G_1(G_1(k))$$

我们声称上述四个字符串即使放在一起看也是伪随机的。（这足以证明 F 是伪随机的。）直觉上，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 $k_0 \,\|\, k_1$ 不可区分。但是，

$$G_0(G_0(k)) \,\|\, G_1(G_0(k)) \,\|\, G_0(G_1(k)) \,\|\, G_1(G_1(k))$$

将与

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

不可区分。由于 G 是一个伪随机生成器，上述结果与一个均匀的 $4n$ 位字符串不可区分。一个形式化的证明使用了混合论证。

推广这个想法，我们可以通过定义 $F_k(x) = G_{x_n}(\cdots G_{x_1}(k)\cdots)$ 来获得一个在 n 位输入上的伪随机函数，其中 $x = x_1 \cdots x_n$。见构造 7.21。直觉上，这个函数是伪随机的，原因与前面相同，但由于要考虑**指数级**数量的输入，形式化证明变得复杂。

构造 7.21 设 G 是一个膨胀因子为 $\ell(n) = 2n$ 的伪随机生成器，并如正文所述定义 G_0, G_1。对于 $k \in \{0,1\}^n$，定义函数 $F_k : \{0,1\}^n \to \{0,1\}^n$ 如下：

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

从伪随机生成器构造伪随机函数。

将此构造视为对于每个密钥 $k \in \{0,1\}^n$ 定义了一个深度为 n 的**完全二叉树**，其中每个节点包含一个 n 位值。（见图 7.2，其中 $n = 3$。）根节点的值为 k，对于每个值为 k' 的内部节点，其左子节点的值为 $G_0(k')$，右子节点的值为 $G_1(k')$。对于 $x = x_1 \cdots x_n$，结果 $F_k(x)$ 被定义为通过按照 x 的位遍历该树到达的叶节点的值，其中 $x_i = 0$ 表示“向左走”， $x_i = 1$ 表示“向右走”。（该函数仅对长度为 n 的输入有定义，因此只输出叶节点的值。）树的大小是 n 的指数函数。尽管如此，计算 $F_k(x)$ 只需要 n 次调用 G，而不需要构造或存储整个树。

定理 7.22 如果 G 是一个膨胀因子为 $\ell(n) = 2n$ 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 $t(n)$ 个均匀的 $2n$ 位字符串和 $t(n)$ 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：

$$\left|\Pr\left[A(r_1 \parallel \cdots \parallel r_{t(n)}) = 1\right] - \Pr\left[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)})) = 1\right]\right|$$

其中第一个概率是关于 $r_1, \ldots, r_{t(n)} \in \{0,1\}^{2n}$ 的均匀选择，第二个概率是关于 $s_1, \ldots, s_{t(n)} \in \{0,1\}^n$ 的均匀选择。证明通过混合论证完成。固定多项式 t 和 ppt 算法 A，并考虑以下算法 A'：

区分器 A'： A' 输入 $w \in \{0,1\}^{2n}$。

1. 选择均匀 $j \in \{1, \ldots, t(n)\}$。
2. 选择均匀且独立的 $r_1, \ldots, r_{j-1} \in \{0,1\}^{2n}$ 和 $s_{j+1}, \ldots, s_{t(n)} \in \{0,1\}^n$。
3. 输出 $A(r_1 \parallel \cdots \parallel r_{j-1} \parallel w \parallel G(s_{j+1}) \parallel \cdots \parallel G(s_{t(n)}))$。

A' 在多项式时间内运行。如前面所示，如果 G 是伪随机生成器，那么上述差值是可忽略的。

现在转向证明 F（如构造 7.21所述）是伪随机函数的核心。我们使用另一个混合论证。在这里，我们定义一个深度为 n 的完全二叉树的叶节点值上的分布序列。通过将每个叶节点与一个长度为 n 的字符串关联起来，就像在构造 7.21中一样，我们可以将其视为**定义域为 n 位输入，值域为 n 位输出**的函数上的分布。对于任何 n 和 $0 \leq i \leq n$，设 H_n^i 是深度为 n 的二叉树的叶节点值上的以下分布：首先从 $\{0,1\}^n$ 中独立均匀地选择第 i 级节点的值。然后对于第 i 级或以下，任何值为 k' 的节点，其左子节点的值为 $G_0(k')$，右子节点的值为 $G_1(k')$。注意 H_n^n 对应于叶节点的所有值都是独立均匀选取的分布，因此对应于从 Func_n 中均匀选取一个函数，而 H_n^0 对应于从构造 7.21中均匀选取一个密钥 k，因为在这种情况下只有根节点（在第 0 级）是均匀选取的。即

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

$$=\left|\Pr_{f\leftarrow H_n^0}[D^{f(\cdot)}(1^n)=1]-\Pr_{f\leftarrow H_n^n}[D^{f(\cdot)}(1^n)=1]\right|$$

我们证明上述差值是可忽略的，从而完成了证明。

设 $t=t(n)$ 是区分器 D 在输入 1^n 时对谕言的查询次数的（多项式）上界。定义区分器 A 如下，它试图区分 $t(n)$ 个均匀的 $2n$ 位字符串和 $t(n)$ 个伪随机字符串：

> **区分器 A：** A 输入一个 $2n\cdot t(n)$ 位字符串 $w_1 \parallel \cdots \parallel w_{t(n)}$。
>
> 1. 选择均匀 $j \in \{0,\ldots,n-1\}$。在以下步骤中，A（隐式地）维护一个深度为 n 的二叉树，其中深度为 $j+1$ 及以下的内部节点处有 n 位值。
> 2. 运行 $D(1^n)$。当 D 查询谕言 $x=x_1\cdots x_n$ 时，查看前缀 $x_1\cdots x_j$。有两种情况：* 如果 D 以前从未查询过具有此前缀的查询，则使用 $x_1\cdots x_j$ 到达树的第 j 级节点 v。取下一个未使用的 $2n$ 位字符串 w 并将 w 的左半部分设置为节点 v 左子节点的值，右半部分设置为右子节点的值。 * 如果 D 以前查询过具有前缀 $x_1\cdots x_j$ 的查询，则节点 $x_1\cdots x_{j+1}$ 的值已经被分配。 * 使用节点 $x_1\cdots x_{j+1}$ 的值，像构造 7.21中那样计算对应于 $x_1\cdots x_n$ 的叶节点的值，并将此值返回给 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。这里一个重要的点是 A 不需要存储指数级大小的整个二叉树。相反，它只需要“填充”树中最多 $2\cdot t(n)$ 个节点的值。设 A 选择 $j=j^*$。观察到：

1. 如果 A 的输入是一个均匀的 $2n\cdot t(n)$ 位字符串，那么它给 D 的答案的分布恰好是 D 与从分布 $H_n^{j^*+1}$ 中选取的函数进行交互的分布。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 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 的答案的分布恰好是 D 与从分布 $H_n^{j^*}$ 中选取的函数进行交互的分布。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。

像以前一样，可以证明

$$\begin{aligned}&\left|\Pr\left[A(r_1 \parallel \cdots \parallel r_{t(n)}) = 1\right] - \Pr\left[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)})) = 1\right]\right| \\ &= \frac{1}{n} \cdot \left|\Pr_{f \leftarrow H_n^0}[D^{f(\cdot)}(1^n) = 1] - \Pr_{f \leftarrow H_n^n}[D^{f(\cdot)}(1^n) = 1]\right|\end{aligned}$$

我们已经证明上述左侧必须是可忽略的。因此，这也蕴含了
$\left|\Pr_{f \leftarrow H_n^0}[D^{f(\cdot)}(1^n) = 1] - \Pr_{f \leftarrow H_n^n}[D^{f(\cdot)}(1^n) = 1]\right|$ 必须是可忽略的。

7.6 构造（强）伪随机置换

我们现在展示如何从任何伪随机函数构造伪随机置换和强伪随机置换。

Feistel网络回顾。 Feistel网络提供了一种从一组任意函数构造可逆函数的方法。Feistel网络在多轮中运行。第 i 轮的输入是一个 $2n$ 位字符串，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”）。该轮的输出 (L_i, R_i) 是一个 $2n$ 位字符串，其中

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

其中 f_i 是某个可高效计算的（但不一定是可逆的）函数，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。我们在§6.2.2中看到，$\mathrm{Feistel}_{f_1,\ldots,f_r}$ 是一个可高效求逆的置换，无论 $\{f_i\}$ 如何。

我们可以通过使用一个**依赖于密钥**的 Feistel 网络来定义一个带密钥的置换。例如，设 $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ 是一个伪随机函数，并定义带密钥的置换 $F^{(1)}$ 为

$$F_k^{(1)}(x) := \mathrm{Feistel}_{F_k}(x)$$

（注意 $F_k^{(1)}$ 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？经过一番思考，可以发现它**绝不是**。对于任何密钥 $k \in \{0,1\}^n$，$F_k^{(1)}$ 的输出的前 n 位（即 L_1）等于输入的后 n 位（即 R_0），这对于一个随机函数来说发生的概率是可忽略的。

再试一次，定义 $F^{(2)} : \{0,1\}^{2n} \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ 如下：

$$F_{k_1,k_2}^{(2)}(x) := \mathrm{Feistel}_{F_{k_1},F_{k_2}}(x)$$

（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，见习题 7.16。

鉴于此，三轮 Feistel 网络是伪随机的可能有些令人惊讶。定义带密钥的置换 $F^{(3)}$，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：

$$F^{(3)}_{k_1,k_2,k_3}(x) := \mathrm{Feistel}_{F_{k_1},F_{k_2},F_{k_3}}(x)$$

其中 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 $F^{(3)}$ 是一个伪随机置换。

证明 按照标准方法，我们可以用均匀选取的函数代替构造 $F^{(3)}$ 中使用的伪随机函数。F 的伪随机性意味着，这种修改对任何与 $F^{(3)}$ 交互的概率多项式时间区分器的成功概率影响是可忽略的。我们留下了证明中的细节作为习题。

设 D 是一个概率多项式时间区分器。在本证明的其余部分，我们证明以下差值是可忽略的：

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

其中第一个概率是关于 f_1, f_2, f_3 从 Func_n 中的独立均匀选择，第二个概率是关于 π 从 Perm_{2n} 中的均匀选择。固定某个安全参数 n，设 $q = q(n)$ 是 D 在输入 1^n 时对谕言的查询次数的（多项式）上界。我们假设 D 从不查询相同的谕言两次。关注 D 与 $\mathrm{Feistel}_{f_1,f_2,f_3}(\cdot)$ 的交互，设 (L_i^0, R_i^0) 是 D 对谕言的第 i 次查询，设 $(L_i^1, R_i^1), (L_i^2, R_i^2)$ 和 (L_i^3, R_i^3) 分别是该查询在第 $1, 2, 3$ 轮之后产生的中间值。（见图 7.3。）注意 D 选择 (L_i^0, R_i^0) 并看到结果 (L_i^3, R_i^3)，但不能直接观察 (L_i^1, R_i^1) 或 (L_i^2, R_i^2)。

我们说在 R^1 处发生了碰撞，如果存在不同的 i, j 使得 $R_i^1 = R_j^1$。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 $i \neq j$。如果 $R_i^0 = R_j^0$，那么 $L_i^0 \neq L_j^0$，但

$$R_i^1 = L_i^0 \oplus f_1(R_i^0) \neq L_j^0 \oplus f_1(R_j^0) = R_j^1$$

如果 $R_i^0 \neq R_j^0$，那么 $f_1(R_i^0)$ 和 $f_1(R_j^0)$ 是均匀且独立的，因此

$$\Pr[L_i^0 \oplus f_1(R_i^0) = L_j^0 \oplus f_1(R_j^0)] = \Pr[f_1(R_j^0) = L_i^0 \oplus f_1(R_i^0) \oplus L_j^0] = 2^{-n}$$

对所有不同的 i, j 应用联合界，表明在 R^1 处发生碰撞的概率至多为 $q^2/2^n$。

接下来我们证明在 R^2 处发生碰撞的概率（以没有在 R^1 处发生碰撞为条件）是可忽略的。分析类似于前面：考虑任意固定的 i, j，注意如果没有在 R^1 处发生碰撞，则 $R_i^1 \neq R_j^1$。因此 $f_2(R_i^1)$ 和 $f_2(R_j^1)$ 是均匀且独立的，因此

$$\Pr[L_i^1 \oplus f_2(R_i^1) = L_j^1 \oplus f_2(R_j^1) \mid \text{没有碰撞在 } R^1] = 2^{-n}$$

对所有不同的 i, j 应用联合界，得到 $\Pr[\text{碰撞在 } R^2 \mid \text{没有碰撞在 } R^1] \leq q^2/2^n$。

注意 $L_i^3 = R_i^2 = L_i^1 \oplus f_2(R_i^1)$；因此，以没有在 R^1 处发生碰撞为条件，值 $L_1^3, \ldots, L_q^3$ 都是独立均匀分布的 n 位值。如果我们进一步以没有在 R^2 处发生碰撞为条件，那么值 $L_1^3, \ldots, L_q^3$ 是在 $\{0,1\}^n$ 的所有序列中均匀分布的（但必须是 q 个不同的值）。类似地，$R_i^3 = L_i^2 \oplus f_3(R_i^2)$；因此，以没有在 R^2 处发生碰撞为条件，值 $R_1^3, \ldots, R_q^3$ 都是均匀分布的 n 位值，且彼此独立，也独立于 $L_1^3, \ldots, L_q^3$。

总结：当在 q 个不同的输入上查询 $F^{(3)}$（使用均匀的轮函数）时，除了以可忽略的概率失败外，输出值 $(L_1^3, R_1^3), \ldots, (L_q^3, R_q^3)$ 的分布是：$\{L_i^3\}$ 是均匀、独立、但不同的 n 位值，而 $\{R_i^3\}$ 是均匀、独立的 n 位值。相比之下，当查询一个随机置换时，输出值 $(L_1^3, R_1^3), \ldots, (L_q^3, R_q^3)$ 是均匀、独立、但不同的 $2n$ 位值。 D 的最佳区分攻击将是：如果存在不同的 i, j 使得 $L_i^3 = L_j^3$，则猜测它与一个随机置换交互。但即使在这种情况下，该事件发生的概率也是可忽略的。这可以转化为一个形式化的证明。

$F^{(3)}$ 不是一个强伪随机置换，见习题 7.17。幸运的是，增加第四轮确实可以得到一个强伪随机置换。详见构造 7.24。

构造 7.24 设 F 是一个带密钥的保长函数。定义带密钥的置换 $F^{(4)}$ 如下：

- **输入**：密钥 $k = (k_1, k_2, k_3, k_4)$，其中 $|k_i| = n$，以及输入 $x \in \{0,1\}^{2n}$，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)。 **从任何伪随机函数构造强伪随机置换。**

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 $2n$ 位输出的强伪随机置换（使用 $4n$ 位密钥）。

7.7 对称密钥密码学的假设

我们已经证明：(1) 如果存在单向置换，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；(3) 如果存在伪随机函数，那么存在（强）伪随机置换。尽管我们没有在

本章中证明，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置换也存在。

我们在第三章和第四章中研究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 CPA-安全的私钥加密方案和安全的消息认证码也存在。

也就是说，单向函数足以构建所有对称密钥密码学。

现在，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，那么存在一个单向函数。

证明 设 G 是一个膨胀因子为 $\ell(n) = 2n$ 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了具有此膨胀因子的生成器的存在。）我们证明 G 本身是单向的。可高效计算性是显然的（因为 G 可以在多项式时间内计算）。我们证明求逆 G 的能力可以转化为区分 G 的输出和均匀分布的能力。直觉上，这成立是因为求逆 G 的能力蕴含了找到生成器所使用的种子的能力。

设 A 是一个任意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（参照定义 7.1）。要看到这一点，考虑以下 ppt 区分器 D：输入 $w \in \{0,1\}^{2n}$，运行 $A(w)$ 得到输出 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 D 输出 1 的概率，当其输入 w 是均匀的。由于 G 的值域中至多有 2^n 个值（即 $\{G(s)\}_{s\in\{0,1\}^n}$），w 位于 G 的值域中的概率至多为 $2^n/2^{2n} = 2^{-n}$。当 w 不在 G 的值域中时，A 不可能计算出 w 的逆，因此 D 也不可能输出 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 $w = G(s)$ 对于均匀选取的种子 $s \in \{0,1\}^n$，那么根据定义，A 以恰好 $\Pr[\mathrm{Invert}_{A,G}(n) = 1]$ 的概率计算出正确的逆（因此 D 输出 1）。因此

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

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

非平凡私钥加密蕴含单向函数。 命题 7.28并没有直接蕴含构建安全的私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建后者。此外，可以构建完美安全的加密方案（见第二章），只要密文不长于密钥。因此，证明安全私钥加密蕴含单向函数需要更加小心。

> **命题 7.29** 如果存在一个加密消息长度是密钥长度两倍的 EAV-安全私钥加密方案，那么存在一个单向函数。

证明 设 $\Pi = (\mathrm{Enc}, \mathrm{Dec})$ 是一个具有不可区分加密的私钥加密方案，它在密钥长度为 n 时加密长度为 $2n$ 的消息。（为简化，我们假设密钥是均匀选取的。）我们假设当使用 n 位密钥时，Enc 最多使用 $\ell(n)$ 位随机性。用 $\mathrm{Enc}_k(m; r)$ 表示使用密钥 k 和随机性 r 加密消息 m。

定义以下函数 f：

$$f(k, m, r) := \mathrm{Enc}_k(m; r) \parallel m$$

其中 $|k| = n, |m| = 2n, |r| = \ell(n)$。我们声称 f 是一个单向函数。显然它可以高效计算；我们证明它难以求逆。设 A 是一个任意 ppt 算法，我们证明 $\Pr[\mathrm{Invert}_{A,f}(n) = 1]$ 是可忽略的（参照定义 7.1）。

考虑以下攻击私钥加密方案 Π 的概率多项式时间攻击者 A'（即，在实验 $\mathrm{PrivK}^{\mathrm{eav}}_{\Pi,A'}(n)$ 中）：

> **攻击者 A' (1^n):**
>
> 1. 选择均匀 $m_0, m_1 \leftarrow \{0,1\}^{2n}$ 并输出。接收挑战密文 c。
> 2. 运行 $A(c \parallel m_0)$ 得到 (k', m', r')。如果 $f(k', m', r') = c \parallel m_0$，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， $c \parallel m_0$ 的分布恰好是 $f(k, m_0, r)$ 的分布，其中 k, m_0, r 是均匀的。因此，A 以恰好 $\Pr[\mathrm{Invert}_{A,f}(n) = 1]$ 的概率输出一个正确的逆（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（每个可能的密钥对应一个）是 c 的原像。由于 m_0 是一个均匀的 $2n$ 位字符串，这意味着存在一个密钥 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 能够输出 $c \parallel m_0$ 的有效逆的概率的一个上限，因此也是 A' 输出 0 的概率的一个上限。

综合上述，我们有：

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

Π 的安全性意味着 $\Pr[\mathrm{PrivK}^{\mathrm{eav}}_{\Pi,A'}(n) = 1] \leq 1/2 + \mathrm{negl}(n)$，其中 negl 是某个可忽略函数。这又蕴含了 $\Pr[\mathrm{Invert}_{A,f}(n) = 1]$ 是可忽略的，完成了 f 是单向的证明。

消息认证码蕴含单向函数。 满足定义 4.2的消息认证码也蕴含单向函数的存在。像私钥加密一样，证明这个事实有点微妙，因为当事先知道消息认证次数的上限时，无条件的消息认证码是存在的（见§4.6）。因此，证明依赖于这样一个事实：定义 4.2要求即使攻击者看到任意多项式数量的消息认证标签，仍然必须满足安全性。该证明比较复杂，我们在此不予给出。

讨论。 我们得出结论：单向函数的存在性对于所有（非平凡的）对称密钥密码学都是**必要和充分**的。换句话说，就对称密钥密码学而言，单向函数是一个**最小**的假设。有趣的是，对于哈希函数和公钥加密，情况似乎并非如此，单向函数被证明是必要的，但尚未被证明（或相信）是充分的。

7.8 计算不可区分性

计算不可区分性（computational indistinguishability）的概念是密码学理论的核心，也是我们第三章和本章中看到的大部分内容的基础。非正式地，如果任何高效算法都无法区分（或辨别）两个概率分布，那么这两个分布就是计算不可区分的。更详细地说，考虑两个在长度为 ℓ 的字符串上的分布 X 和 Y；即 X 和 Y 各自为 $\{0,1\}^{\ell}$ 中的每个字符串分配了一个概率。当我们说某个算法 D 无法区分这两个分布时，我们的意思是 D 无法判断它得到的字符串样本是根据分布 X 采样的还是根据分布 Y 采样的。换句话说，如果 D 认为它的输入是根据 X 采样时输出 0，而认为是根据 Y 采样时输出 1，那么无论 D 得到的是 X 的样本还是 Y 的样本，它输出 1 的概率应该大致相同。即，我们希望

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

是小的。

这应该让人想起我们定义伪随机生成器的方式，事实上，我们很快就会用这个术语来形式化地重新定义伪随机生成器的概念。

计算不可区分性的形式化定义涉及**概率集成**（probability ensembles），即无限的概率分布序列。（这种形式化对于有意义的渐近方法是必要的。）虽然这个概念可以推广，但对于我们的目的，我们只考虑底层分布由自然数索引的概率集成。如果对于每个自然数 n，我们都有一个分布 X_n，那么 $X = \{X_n\}_{n\in\mathbb{N}}$ 就是一个概率集成。在很多情况下，$X_n = Y_{t(n)}$ 对于某个函数 t 成立，在这种情况下，我们用 $\{Y_{t(n)}\}_{n\in\mathbb{N}}$ 代替 $\{X_n\}_{n\in\mathbb{N}}$。我们只对**可高效采样**的概率集成感兴趣。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，那么集成 $X = \{X_n\}_{n\in\mathbb{N}}$ 是可高效采样的。也就是说，算法 S 是采样 X 的一种高效方法。

我们现在可以形式化地定义两个集成是计算不可区分的。

> **定义 7.30** 两个概率集成 $X = \{X_n\}_{n\in\mathbb{N}}$ 和 $Y = \{Y_n\}_{n\in\mathbb{N}}$ 是**计算不可区分的**，记作 $X \approx_c Y$，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：
>
> $$|\Pr_{x\leftarrow X_n}[D(1^n, x) = 1] - \Pr_{y\leftarrow Y_n}[D(1^n, y) = 1]| \leq \mathrm{negl}(n)$$

在定义中，D 得到了**一元输入** 1^n，因此它的运行时间可以是 n 的多项式。当 X_n 和 Y_n 的输出长度可能小于 n 时，这一点很重要。在概率表达式中，我们有时将 X 写成一个占位符，表示来自分布 X 的随机样本。也就是说，我们用 $\Pr[D(1^n, X_n) = 1]$ 代替 $\Pr_{x\leftarrow X_n}[D(1^n, x) = 1]$。

计算不可区分性的关系是**可传递的**：如果 $X \approx_c Y$ 且 $Y \approx_c Z$，那么 $X \approx_c Z$。

伪随机性和伪随机生成器。 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，设 U_ℓ 表示 $\{0,1\}^\ell$ 上的均匀分布。我们可以如下定义伪随机生成器：

> **定义 7.31** 设 $\ell(\cdot)$ 是一个多项式，G 是一个（确定性）多项式时间算法，使得对于所有 s，$|G(s)| = \ell(|s|)$。我们说 G 是一个**伪随机生成器**，如果满足以下两个条件：
>
> 1. （膨胀）：对于所有 n，$\ell(n) > n$。
> 2. （伪随机性）：集成 $\{G(U_n)\}_{n\in\mathbb{N}}$ 与集成 $\{U_{\ell(n)}\}_{n\in\mathbb{N}}$ 是计算不可区分的。

本书中的许多其他定义和假设也可以被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，一个重要的定理是：**多项式数量**的（可高效采样的）计算不

可区分集成也是计算不可区分的。

> **定理 7.32** 设 X 和 Y 是计算不可区分的可高效采样概率集成。那么，对于所有多项式 p，集成 $\mathcal{X} = \{(X_n^{(1)}, \ldots, X_n^{(p(n))})\}_{n\in\mathbb{N}}$ 与集成 $\mathcal{Y} = \{(Y_n^{(1)}, \ldots, Y_n^{(p(n))})\}_{n\in\mathbb{N}}$ 是计算不可区分的。

例如，设 G 是一个膨胀因子为 $2n$ 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。我们在定理 7.22的证明中证明了，对于任何多项式 t，集成

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

也是计算不可区分的。定理 7.32是通过完全相同的混合论证证明的。