Part I Setting the Stage

第 1 章 数字签名：背景和定义

1.1 数字签名方案：快速入门

一般来说，数字签名提供了手写签名的加密模拟，但实际上，它提供更强大的安全保证。数字签名是一种强大的工具，并且现在被一些国家接受具有法律约束力，可以用来证明合同的真实性和公证文件，确认个人或公司的真实性，以及用作更复杂协议的一部分。数字签名还可以保证公钥的安全分发和传输，确切地讲，它是所有公钥密码学的基础。

数字签名方案，通常是由一个签名者和多个验证者使用。（我们这里讨论的相对不正式，稍后我们给出正式定义）签名者首先通过运行密钥生成算法，产生一对密钥（pk, sk），其中 pk 被称为签名者的公钥，sk 被称为签名者的私钥（有时候也被称为秘密密钥）。然后签名者公开公钥，我们假设所有潜在的验证者拥有（或者得到）签名者的公钥 pk 的真实副本。我们将不关注签名者如何分发其公钥的确切细节，具体来说，想象一下这里有一个公共目录，链接签名者到他们的公钥，并且这个目录的管理模式使得某人以别人的名字注册公钥是不可行的。我们强调，通常很多签名者，每个签名者有自己的公钥，任意潜在的验证者不仅必须知道一系列有效的公钥，同样必须知道这些公钥属于哪个它有兴趣验证的签名的签名者。

一旦签名者已经建立上述讨论的公钥 pk，数字签名方案允许签名者以某种方式去“认证”（或者签名）消息，任何知道 pk 的用户都可以验证消息的来源于签名者且未进行任何修改。更详细的，对于消息 m（我们只是将其看作一个比特串），签名者用私钥 sk 将签名算法作用于 m；签名的结果 σ 可以被任何知道公钥 pk 的验证者使用相应的验证算法进行验证。

在这一点上，考虑数字签名方案的典型用法将非常有用：考虑一家软件公司，该公司希望以经过认证的方式发布软件补丁/更新；也就是说，当公司需要发布软件补丁时，任何客户都应该有可能认识到该软件补丁是经过认证的，并且恶意的第三方永远不能愚弄客户接受公司实际上未发布的补丁。为此，公司可以生成公钥 pk 和私钥 sk，然后以某种可靠的方式将 pk 分发给其客户（也许将公钥与软件的初始分发捆绑在一起）。当发布补丁 m 时，公司可以使用其私钥 sk 在 m 上计算数字签名 σ，然后在其网页上发布（m，σ）。每个用户在下载补丁 m 之前，通过检查签名 σ 是补丁 m 相对应的公钥 pk 的一个合法签名来验证补丁 m 的真实性。

恶意用户可能通过假冒公司的网页并发布（m'，σ'）来尝试发布假的补丁，其中 m' 代表公司从未发布过的补丁。该 m'可能是某些先前补丁 m 的修改版本，或者它可能是全新的并且与先前的补丁无关。如果一个签名方案是安全的（从某种意义上说，我们将尽快更仔细地定义），然后，当用户尝试验证 σ'时，它将发现相对于 pk 的 m'是无效签名，因此将拒绝该签名。请注意，在此应用中，即使伪造的补丁 m'与真正的补丁 m 仅有少许改动，客户也必须拒绝。

以上不仅是数字签名的理论应用，而且是当今广泛使用的一种。（例如，Microsoft 在发布其 Windows 操作系统的更新时完全使用这种方法。）

用户能够获得签名者公钥的合法副本的假设意味着，签名者能够以可靠且经过验证的方式传输至少一条消息（即 pk 本身）。鉴于此，人们可能想知道为什么需要签名方案！核心问题是是可靠地分发 pk 是一项艰巨的任务，但是使用签名方案意味着只需要执行一次，此后便可以可靠地发送无限数量的消息。此外，签名方案本身用于确保其他公钥的可靠分发，是公钥基础设施（PKI）的一部分。

1.1.1 数字签名的性质

我们刚刚看到，数字签名提供了一种对通过公共信道发送的消息进行认证的方法。签名方案也具有更强的特性，我们通过与消息认证码（数字签名的对称密钥模拟）的比较来阐明这些特性。

消息认证代码的实例是由在（单个）发送者和（单个）接收者之间共享的秘密密钥 s 定义的。签名者可以通过共享密钥 s 对消息 m 应用消息认证算法验证消息 m；其结果是“标记” t。给定 m 和 t 的接收者可以使用相应的验证算法以及相同的密钥 s 来验证 m 的真实性。与数字签名一样，消息认证码提供的安全保证是不知道 s 的恶意第三方不能在未经发送方明确认证的消息 m′上伪造一个有效的查找标记 t′。（我们请读者参考[72]，以更深入地讨论消息认证码。）

因此，消息认证码和数字签名方案都能保证传输消息的完整性（真实性）。然而，一个明显的区别是在最初的密钥建立阶段。数字签名属于公钥密码学的范畴，其中一方（即本例中的签名者）只需在一个公开的但已授权的信道上分发一些密钥。另一方面，对于消息认证码，发送方必须通过秘密的且经过认证的通道共享密钥。当发送者想要向多个接收者发送同一个消息时，签名方案也有一个决定性的优势。当使用数字签名方案时，这将通过分发单个公钥并计算可由任何接收者验证的单个签名来实现；与此相反，使用消息认证码，发送者必须与每个可能的接收者建立一个单独的密钥，对于每个收件者必须单独计算的密钥标记（与相应的共享密钥相关）。

与消息认证代码相比，数字签名具备的质的优势是其签名是可公开验证的。这意味着，如果接收方对给定消息上的签名验证为合法，那么可以保证收到此签名消息的所有其他参与方也将验证它是合法的。消息认证码无法实现此功能，消息认证码中的签名者与每个接收者共享一个单独的密钥：在这种情况下，恶意发送方可能计算关于接收者 A 的共享密钥的正确标签，但计算针对不同用户 B 的共享密钥的不正确标签。在这种情况下，A 知道他收到了来自发送者的真实消息，但不能保证其他接受者会同意。

公开可验证性意味着签名是可转让的：特定签名者 S 在消息 m 上的签名 σ 可以展示给第三方，然后第三方可以验证 σ 是 m 相对于 S 的公钥的合法签名（此处我们假设此第三方也知道 S 的公钥）。通过复制签名，第三方可以将签名展示给另一方，并让他们知道 S 已认证 m。可转让性和公共可验证性对于将数字签名应用于证书和公钥基础结构至关重要。

数字签名方案还具有非常重要的不可否认性。也就是说，假设签名者 S 首先公开了他的公钥，一旦签名者 S 在一条信息上签名，那么以后他就不能否认。在接受者需要向第三方（例如法官）证明签名人确实“证明”了某一特定消息（例如合同）的情况下，数字签名的这一方面至关重要：假设 S 的公钥为法官所知，或者是公开的，那么消息上的有效签名就足以让法官相信 S 确实签署了该消息。消息认证码根本无法提供此功能。为此，假设用户 S 和 R 共享密钥 S_{SR}，S 向 R 发送消息 m 和使用 S_{SR} 计算的（有效）MAC 标签 t。因为法官不知道 S_{SR}（实际上，此密钥由 S 和 R 保密），因此法官无法确定 t 是否为有效标签。如果 R 向法官透露密钥 S_{SR}，法官仍然无法知道这是否是 S 和 R 共享的“实际”密钥，或是 R 在事后创造的一些“假”密钥。我们假设即使法官获得了实际的密钥 S_{SR}，并且能以某种方式相信这个事实，这仍然不会提供不可否认性，因为 R 无法证明是 S 生成的 t，事实上消息认证码是对称的（因此 S 可以生成，R 也可以生成），这意味着 R 可以自己生成 t，因此法官无法区分在两个参与者的动作。

由于数字签名具有不可否认且可公开验证，因此在互联网上经常使用数字签名来签署合同，公证文件等，并且在许多国家/地区已获得法律效力。

当然，与消息认证码相比，数字签名的一个缺点是消息认证码的效率要比数字签名高 2-

3 个数量级。因此，在不需要公开可验证性，可转让性和/或不可否认性的情况下，发送者将主要与单个接受者（可以共享密钥）进行通信，使用消息认证码是首选。我们指出，在某些情况下，特别需要不可抵赖性和可传递性：例如，当签名人 S 希望确保特定接受者确信 S 已认证了一个消息，但又不希望该接受者向其他人证明这一事实。（这有时称为可拒绝性的属性。）在这种情况下，必须使用消息认证码（或更复杂的密码原语）。

1.2 计算安全性

消息认证码和数字签名方案的另一个区别是，当数量有限的消息被认证时，消息认证码是无条件安全的。另一方面，数字签名方案的安全性本质上是计算性的(即使我们限制了被签名的消息的数量)。具体来说，没有一个签名方案能够安全地抵抗全能型敌手全能对手是指拥有无限计算能力的人，或者，等价地说，没有时间限制的人)。实际上，考虑这样一个敌手，给定签名者的公钥 pk 和消息 m，尝试所有可能的 σ 值，直到找到一个 Vrfypk（m，σ）=1 的值。（敌手知道 pk，因此可以根据自己选择的输入计算 Vrfypk（·，·)。现在，至少存在一个签名 σ 满足 $Vrfy_{pk}(m, \sigma) = 1$，因为合法签名者必须能够在 m 上生成有效的签名。但这样一来，所描述的敌手将最终找到这样一个签名 σ 并成功伪造签名。

第二个发现是，即使面对非常“弱”的敌手，签名方案也不可能是完全安全的。这一点的说明比以前更简单：假设一个敌手只是随机选择了 σ。显然，这样选择的值满足 $Vrfy_{pk}(m, \sigma) = 1$ 的概率是非零的(同样，利用至少有一个满足这个条件的 σ 存在的事实)。这个敌手甚至不需要知道签名者的公钥。

以上是否表明安全的签名方案是不可能构造的? 不是的。值得庆幸的是，如果人们愿意放松安全需求，并考虑安全性的计算概念而不是完美的概念，那么希望并没有破灭。也就是说，我们不是要求（如上文所述)任何敌手都不可能伪造签名，而是要求，除了小概率以外，任何高效（例如，计算受限的）敌手伪造签名是不可能的；特别要注意的是，这排除了上述两种攻击。在这本书中，我们以标准的方式对这些概念进行了形式化（进一步讨论见[72]），为了完整起见，我们现在简要回顾一下细节。

1.2.1 计算安全性概念

在计算环境中，我们引入了安全参数 $k \in N$，用于参数化敌手和签名方案本身；该安全性参数可以看作是量化由该方案的特定实例获得的安全性级别（尽管这在形式上并不完全正确）。更详细地说，我们将签名者视为在为方案生成密钥时选择一个安全参数 k；安全参数将作为输入传递给密钥生成算法，并且公钥和私钥的长度取决于 k。遵循理论密码学中的标准惯例，我们将“高效敌手”等同于在概率多项式时间内运行的算法（其中，运行时间是 k 的函数）。由于敌手将被限制在多项式时间内运行，因此要求诚实方执行的所有算法（例如签名、验证）也应在多项式时间内运行才是公平的。我们将“概率多项式时间”简称为 PPT。

如果事件发生的概率在 k 中可以忽略不计，则事件发生的概率是“小概率”的，其正式定义如下：

定义 1.1 如果对于所有 $c \geq 0$，存在 $k_c \geq 0$，使得 $k > k^c$，$\varepsilon(k) < 1/k^c$，则函数 $\varepsilon: N \rightarrow [0, 1]$ 可忽略不计。

换句话说，满足计算安全性概念的签名方案具备性质：每个概率多项式时间的敌手仅以可忽略的概率成功地伪造了一个签名（相对于该方案）。换句话说，固定某个在多项式时间内运行的敌手 A，并让 ε_A（k）表示 A 对某个签名方案伪造有效签名的概率。（当我们引入正式的安全定义时，我们将更仔细地说明这个实验。）然后，如果 ε_A（k）在 k 中可忽略，则该方案是安全的，这意味着随着签名者增加 k 的值，敌手 A 的“成功”概率迅速减少。需要

强调的是，对于特定的 k 值，这并没有说明方案的“绝对”安全性；给出的所有保证只是关于该方案的渐进性能。从某种意义上说，k 值越大，在实际中就会产生一个“更安全”的方案；然而，正式地讲，如果不参考 k 的任何特定值，则方案要么是安全的，要么是不安全的。

通过一个示例可以很好地说明这一点。

范例 1.1 想象一个安全的签名方案，其中私钥是长度为 k 的均匀随机字符串，再考虑一下对对密钥执行 k^5 步暴力破解搜索天真敌手 A。（我们假设敌手可以识别匹配给定公钥的私钥。）A 找到正确私钥的概率为 $k^5/2^k$。对于 $k=30$，敌手运行 $2.4*10^7$ 步，并以 1/50 的概率找到私钥；在实际中，这可能不会被视为可接受的安全级别（尽管从渐进的角度来看，该方案本身仍然是“安全的”）。通过将安全参数“调节”为 $k=60$，我们已经获得了合理的安全保证：即使 A 运行 $7.7*10^8$ 步，它也只能以 $6.7*10^{-10}$ 的概率找到私钥。

上面的示例说明了具体的安全性分析的重要性，即量化任何敌手运行指定时间并使用安全参数的特定值攻击一个方案的最大成功概率的一种分析——除了在安全性的正式定义中使用的渐近式外。如前所述，使用渐进安全性分析被证明安全的方案只能保证——随着安全性参数的增加，任何多项式时间的敌手“破坏”方案的可能性都可以忽略不计。它没有说在实际中使用什么安全参数值来确保针对在特定时间段内运行的敌手的特定安全级别。该问题将在第 7 章中重新讨论，其中将考虑对某些实际方案进行具体的安全分析。

1.2.2 记号

在理解签名方案的安全性时，我们将对分析概率实验感兴趣。我们引入一些记号（见下文[61]），它们为安全性分析供有用的速记。设 A 为概率图灵机。则 A(x;r)为 A 在输入 x 和(足够长)随机磁带 r 上运行时的输出。如果 A 接受多个输入 x_1，…，x_n(当然，总是可以编码为单个输入)，那么 A(x_1，…，x_n；r)表示在这些输入和随机磁带 r 上运行时 A 的输出。

在写一个多阶段实验时，符号 y:=A(x；r)仅仅意味着变量 y 被赋予了值 A(x；r)。如果 S 是一个有限集，那么 y←S 表示 y 是从 S 中均匀选择的元素。符号 y1，y2←S 被认为是表示 y1 和 y2 是从 S 均匀独立地选择的元素。这个符号，y←A（x）的实验是指随机均匀地选择一个随机带 r（具有适当长度），然后 y 被赋予值 A（x；r）；因此，如果 A 在长度为|x|的输入上使用长度为（最多）l 的随机带，则符号“y←A（x）”是以下简称：

$$r\leftarrow\{0,1\}^{l};\ y:=A(x;r)。$$

当 A 是确定性时，y:=A(x)相当于 y←A(x)，但是当我们想强调 A 是确定性时，我们将使用前者的表示法。

执行某个实验 expt，某个特定事件 E 的概率写成 Pr[expt:E]。位于冒号左侧的内容表示实验本身（其组成部分按从左到右的顺序执行），感兴趣的事件被写入到冒号的右侧。事件可以用谓词表示，如果谓词为真，则称事件发生。通常，∧是“逻辑与”运算，∨表示“逻辑或”，$\{0,1\}^k$ 表示长度为 k 的二进制字符串集，如果 E 是事件，则$\bar{E}$表示 E 的对立事件（即 E 没有发生的事件）。

1.3 签名方案的定义

如果我们要理解任何特定结构所保证的安全性，那么精确的定义是至关重要的，而且在我们希望为我们将要开发的方案提供严格的安全性证明之前也是必不可少的。为了完整性，我们从一个纯粹的功能定义开始，该定义描述任何签名方案都应该实现的基本功能；接下来是许多不同的安全定义，详细描述了签名方案可能提供的各种弹性（安全）级别。 在本章的其

余部分，我们重点介绍用于增强给定签名方案的安全性技术。即，我们展示了如何采用一种实现相对较弱的安全性概念的方案并对其进行调整，以便获得一种实现更强的安全性概念的新方案。 这些技术在安全签名方案的设计和开发中非常有用，我们将在后面的章节中介绍的构造中看到这些技术的许多示例。

定义 1.2 签名方案由三种概率多项式时间算法（Gen，Sign，Vrfy）以及相关的消息空间 $M = \{M_k\}$组成，使得：

- 随机密钥生成算法 Gen 将安全参数 k（一元）作为输入。它输出一对密钥（pk，sk），其中 pk 称为公钥或验证密钥，而 sk 称为私钥，秘密密钥或签名密钥。我们假设安全参数 k 在 pk 和 sk 中都是隐含的。
- 对于安全参数 k，（可能是随机的）签名算法 Sign 将密钥 sk 和消息 $m \in M_k$ 作为输入。它输出一个签名 σ。我们将其写为 $\sigma \leftarrow Sign_{sk}$（m）。我们假设如果 $m \notin M_k$，则签名算法输出一个专有符号⊥。
- 对于安全参数 k，确定性验证算法 Vrfy 将公钥 pk，消息 $m \in M_k$ 和（声称的）签名 σ 作为输入。它输出一个比特 b，b = 1 表示“接受”，b = 0 表示“拒绝”。我们将其写为 $b := Vrfy_{pk}(m, \sigma)$。我们假设如果 $m \notin M_k$，则验证算法拒绝。

当从上下文中了解给定的公钥时，我们说如果 $Vrfy_{pk}(m, \sigma) = 1$，则消息/签名对(m，σ)是有效的。我们要求对于所有 k，通过 $Gen(1^k)$算法输出所有(pk，sk)，所有 $m \in M_k$，$Sign_{sk}$（m）输出的所有 σ，则消息/签名对(m，σ)有效。

应该清楚的是，以上定义恰好使前面描述的直观概念形式化。具体地，签名方案的使用方式如下。充当签名者的一方 S 运行 Gen（1^n）以获得密钥(pk，sk)。当 S 要发送消息 m 时，它计算签名 $\sigma \leftarrow Sign_{sk}(m)$并发送(m，σ)。收到(m，σ)后，知道 pk 的接收者通过检查 Vrfypk(m，σ)=？1 来验证 m 的真实性。这既确定了 S 发送了 m，又确定了 m 在传输中没有被修改。值得强调的是，成功的验证并不能证明发送 m 的时间，因此所谓的重放攻击是可能的。我们稍后再回到这一点。

一些宽松定义

我们简要讨论 1.2 的一些宽松定义。

该定义中对正确性的要求可以放宽到允许极小的出错。在我们的介绍中，我们在很大程度上忽略了这个问题，尽管我们提醒读者，我们提出的一些方案确实有可以忽略不计的错误概率。

上述定义的另一个版本是允许随机验证算法的可能性，当正确生成的签名被验证时，允许的错误概率可以忽略不计。文献中很少有方案使用随机验证，我们也不会在本书中看到任何例子。

与其要求预先固定消息空间（对于 k 的每个值），有时允许合法消息集依赖 Gen 生成的公钥是很方便的。当本书中描述的方案有时会出现这种情况时，从上下文中可以明显看出这一点，因此我们不会赘述。请注意，对于任何实际应用程序，消息空间应该由所有比特字符串组成（可能是有界长度）。

签名算法有可能是有状态的（而不是上面定义的无状态）：形式上，在这种情况下，某些状态 s 将由 Gen 初始化为 NULL；除了消息和密钥之外，签名算法会将当前状态 s 作为额外输入。签名算法除了输出签名之外，还将输出状态的更新值 s'。（我们强调，不需要状态来验证签名。此外，除非另有说明，否则假定敌手无法查看签名人的状态。）在大多数情况下，签名方案的状态在实际中被认为是不可取的；然而，一些签名方案的安全性在很大程度上取决于签名算法保持这种状态的能力。（我们将在第 3 章中看到一个这样的示例。）除非另有说明，否则我们始终假定无状态签名方案。

1.4 安全性的定义动机

粗略地说，签名方案提供的基本安全保证是，即使敌手与该用户“交互”并因此获得了对多个消息的合法签名(使用与 pk 关联的私钥 sk 生成)，不存在高效的敌手有能力针对诚实用户生成的公钥 pk 伪造有效的消息/签名对。如前所述，我们将“高效”等同于（概率）多项式时间的概念，并允许该敌手以很小的可忽略的概率成功地输出有效的伪造（回想一下，对手的运行时间及其伪造签名的可能性都是根据安全性参数 k 来衡量的）。要正式规定所需的安全性概念，则需要更精确地定义两件事：首先，“伪造”有效的消息/签名对意味着什么；第二，与合法用户“交互”的确切含义。这两个组成部分可以独立指定，从而导致许多可能的定义。我们在这里探索其中一些可能性。

首先，让我们以非正式的方式讨论“伪造”有效消息/签名对的含义的一些合理解释。一个直接的观察结果是，如果合法的签名人本人分别在消息 m_1，...，m_l 上生成签名 σ_1，...，σ_l，那么就不能阻止敌手重放有效的消息/签名对(m_i, σ_i)。那么，显然，此类“重放”攻击不应视为伪造，我们也不会在我们的定义中尝试防止此类攻击。鉴于此，一种自然的方法是说，如果敌手生成了有效的消息/签名对(m, σ)，且 m 不是先前由合法签名者签名的消息（即$m \notin \{m_1, ..., m_l\}$），那么它就成功地伪造了签名。我们将这种伪造称为存在性伪造，并称其为不可行的存在性伪造的方案。我们强调，不以任何方式要求 m 是“有意义的”。实际上，“有意义”的概念取决于应用程序，因此，在签名方案的安全性定义中并入任何语义（或等效地，任何“有意义”概念）都没有多大意义。这有一个额外的好处：任何存在性不可伪造的签名方案都可以被用于需要访问签名功能的任何应用程序中，而不必担心应用程序的语义是否与签名方案本身假设的任何语义兼容。

最近引入了一种更强的概念，称为强存在不可伪造性，并且对于数字签名方案的某些应用是必需的。在这里，即使敌手在先前签名的消息上输出了新的（有效的）签名，也可以说是伪造的（这是对上面定义的存在不可伪造的概念的补充）；形式上，只要敌手输出有效的消息/签名对$(m, \sigma) \notin \{(m_1, \sigma_1), ..., (m_l, \sigma_l)\}$（和前面一样，其中，$\sigma_i$ 表示从合法签名者那里获得的 m_i 上的签名）。请注意，这给出了存在不可伪造性的严格推广。

在讨论了伪造的概念之后，我们接下来考虑敌手可能与合法签名人进行交互的可能方式。我们特别关注此交互的两个特征：（1）合法签名者生成多少签名，以及（2）敌手对签名消息具有什么样的控制权。关于第一个问题，我们将只区分生成单个签名的情况和可能生成无数个签名的情况。旨在仅生成单个签名场景的签名方案称为一次性签名。除了考虑该定义相对较弱的历史原因（特别是第一个可证明安全的签名方案是一次性签名方案）外，一次性签名还可以用作标准签名方案的有用组成部分（例如我们将在后面看到）。一次性签名方案本身对某些应用程序也很有用。

第二个问题（即，敌手对签名消息的控制量）更有趣。在这里，我们非正式地描述了我们将要考虑的三种不同场景，并提供了一些理由，说明为什么每种场景在某些设置下都代表自然的敌对行为。但是，我们强调，我们的最终目标是在最后（最强大）的场景中实现安全；然而，在其他两个模型的背景下安全的方案将成为实现这一目标的有用“垫脚石”（尤其参见第 1.7.1 和 1.7.2 节）。

随机消息攻击：我们将要考虑的第一个场景可以被视为模拟这样一种情况，即敌手对签名的消息没有任何控制权。举个例子，在这里可以想象敌手仅观察到其他（诚实）各方提供的消息上产生的签名。我们通过考虑给敌手一个随机消息序列上的签名的形式来对此进行形式化。尽管这似乎与我们刚才讨论的场景不太相近，稍加思考就可以看出，如果不引入一些（先验的不合理的）关于已签名消息分布的假设，就没有其他合理的选择。此外，在某些情况下，签名的消息实际上是随机的，然后提供了敌手攻击的精确模型；一个很好的例子是，

通过验证者发送的随机挑战计算签名来对用户进行认证。

已知消息攻击：现在假定敌手对签名的消息具有有限的控制权，或者在可能签名的消息类型中可能存在某些确定的模式（因此，随机消息攻击不再是最合适的模型）。我们将通过授予敌手对已签名的消息的控制权来形式化此攻击场景，但必须限制敌手事先指定这些消息，特别是，独立于签名者的公钥以及敌手观察到的任何后续签名。我们可以将其视为对要签名的消息的“最坏情况”选择进行建模，只要这些消息不直接受敌手的控制（因此一旦确定了公钥就无法更改）。

（自适应）选择消息攻击：此模型是最强大的模型，因为它可以使敌手完全控制已签名的消息。具体地说，对手不仅可以在看到公钥后选择这些消息，还可以根据敌手观察到的先前签名（在先前选择的消息上）来选择消息。因此，术语“自适应”。 （但是，由于我们不考虑选择消息攻击的任何非自适应变体，因此我们通常会省略该术语。）

有人可能会反对现实的敌手永远不会完全控制已签名的内容，因为合法的签名人肯定会拒绝对某些消息进行签名（例如“我是骗子”）。但是，如前面在伪造中所讨论的，在安全性定义中将任何语义强加到消息空间上几乎没有任何意义。此外，很可能在某些情况下（例如，签名人充当公证人），敌手可能会对签名内容拥有大量控制权。一个被证明可以抵御这种强大攻击的方案肯定对任何所需的应用程序都是安全的（无需考虑应用程序的需求是否“符合”安全定义所提供的保证）。

鉴于上述情况，我们将在以下部分中介绍许多形式上的定义，对应于伪造概念和攻击模型的各种组合。具体来说，我们将定义（1）一次性签名方案存在性不可伪造的概念；（2）随机消息攻击下的存在不可伪造性；（3）已知消息攻击下的存在不可伪造性；（4）（强）自适应选择消息攻击下的存在不可伪造性。正如我们已经提到的，我们将介绍第一个定义，因为它的历史意义以及它在构造满足更严格的安全性定义的方案中的用处，我们将在下一章中看到。同样，第二和第三定义将在第 1.7.1 和 1.7.2 节中使用，以构造满足第四个（最强）概念的方案。我们认为的最后一个定义是最强的定义（至少就上述讨论的可能性而言），已成为任何“好的”签名方案都应满足的安全性的事实上的概念。有人可能会说，相对于实际中的需要，这种定义是“过大的”，尽管我们在上面提出了相反的意见。但是，即使这是正确的，该定义对于实际中的签名方案的任何应用也肯定是足够的，因此，构造满足此强安全性概念的方案将有很大的好处。

1.5 安全性的形式化定义

我们给出的定义顺序与上一段中讨论的顺序略有不同。

1.5.1 抵抗随机消息攻击的安全性

我们首先定义针对随机消息攻击的安全性。在这里，回想一下，攻击者无法控制签名的消息。而是从消息空间中随机选择这些消息。为了完整性，我们定义了存在不可伪造性和强存在不可伪造性。

定义 1.3 如果对于所有多项式 $l(\cdot)$和所有概率多项式时间的敌手 A，在下面的实验中 A 的成功概率为可以忽略不计（作为 k 的函数），则签名方案（Gen，Sign，Vrfy）在随机消息攻击下是存在性不可伪造的：

1. 从消息空间 M_k 中随机地均匀选择 $l = l(k)$个消息 m_1，...，m_l 的序列。
2. 运行密钥生成算法 $Gen(1^k)$以获得一对密钥(pk，sk)。
3. 计算签名 $\sigma_1 \leftarrow Sign_{sk}(m_1)$，...，$\sigma_l \leftarrow Sign_{sk}(m_l)$。

4. 给敌手 pk 和$\{(m_i，\sigma_i)\}_{i=1}^{l}$，并输出(m，σ)。
5. 如果（1）$Vrfy_{pk}(m，\sigma)=1$ 且（2）$m\notin\{m_1，...，m_l\}$，则敌手 A 成功。
如果对于所有多项式 $l(\cdot)$和所有概率多项式时间的敌手 A，在如上定义的实验中将最后一个条件 5 更改为 5′，A 的成功概率可忽略不计（作为 k 的函数），则该方案在随机消息攻击下强不可伪造。上述实验最后一个条件，现在更改为：
5′. 如果（1）$Vrfy_{pk}(m，\sigma)=1$ 且（2）$(m，\sigma)\notin\{(m_1，\sigma_1)，...，(m_l，\sigma_l)\}$，则 A 成功。

作为技术要点，如果消息空间依赖于公钥，则更改上述实验的顺序，首先生成公钥和私钥，然后从定义的消息空间中随机选择消息。（实验的其余部分保持不变。）

只是为了适应这种表示法，请注意，我们可以按照以下紧凑的方式来表示上面的定义。如果对于所有多项式 $l(\cdot)$和所有概率多项式时间的敌手 A，在实验中 $\varepsilon_A(k)$可忽略，签名方案（Gen，Sign，Vrfy）在随机消息攻击下是不可伪造的：

$$\varepsilon_A(k)\stackrel{\text{def}}{=}\Pr\left[\begin{array}{c}\{m_i\}_{i=1}^{\ell}\leftarrow M_k;\ (pk,sk)\leftarrow \mathsf{Gen}(1^k);\\ \forall i\in[\ell]:\sigma_i\leftarrow \mathsf{Sign}_{sk}(m_i);\\ (m,\sigma)\leftarrow A\left(pk,\{(m_i,\sigma_i)\}_{i=1}^{\ell}\right)\end{array}:\begin{array}{c}\mathsf{Vrfy}_{pk}(m,\sigma)=1\wedge\\ m\notin\{m_1,\ldots,m_\ell\}\end{array}\right].$$

当给出定义时，我们将继续完整地写出实验，但是当给出安全性证明时，将转换到更紧凑的表示法。

有时考虑一个非常弱小的只允许获得一条消息的签名的敌手也很有用。

定义 1.4 如果对于所有概率多项式时间的敌手 A，在以下实验中 A 的成功概率可以忽略不计（作为 k 的函数），则在一次随机消息攻击下，签名方案（Gen，Sign，Vrfy）是存在不可伪造的：
1 从消息空间 M_k 中随机地均匀地选择消息 m_1。
2 运行密钥生成算法 $Gen(1^k)$以获得一对密钥(pk，sk)。
3 计算签名 $\sigma_1\leftarrow Sign_{sk}(m_1)$。
4 给敌手 pk 和$(m_1，\sigma_1)$，并输出(m，σ)。
5 如果（1）$Vrfy_{pk}(m，\sigma)=1$ 且（2）$m\neq m_1$，则敌手 A 成功。
如果对所有概率多项式时间的敌手 A 来说，在上述定义的实验中将最后一个条件 5 更改为 5′，A 的成功概率可忽略不计（作为 k 的函数），该方案在一次性随机消息攻击下是绝对不可伪造的，现在我们将最后一个条件更改为：
5′ 如果（1）$Vrfy_{pk}(m，\sigma)=1$ 且（2）$(m，\sigma)\neq(m_1，\sigma_1)$，则 A 成功。

引申一下，我们还可以考虑任何多项式 l 的 l 次攻击。但是，请注意，由此产生的安全性概念与定义 1.3 之间的区别：在前一种情况下，l 是先验固定的，并且签名方案允许依赖于 l，而在定义 1.3 的情况下，方案是固定的，然后需要对任何（多项式）l 都必须具备安全性。

1.5.2 抵抗已知消息攻击的安全性

现在，我们定义已知消息攻击的安全性。回想一下，敌手可以选择签名的消息，但是它必须在得到公钥之前做出选择。同样，出于完整性考虑，我们还给出了强不可伪造的定义。

定义 1.5 如果对于所有多项式 $l(\cdot)$和所有概率多项式时间的敌手 A，在下面的实验中 A 的成功概率为可以忽略不计（作为 k 的函数），则签名方案（Gen，Sign，Vrfy）在已知消息攻击下是存在性不可伪造的：
1 $A(1^k)$输出 $l=l(k)$个消息 m_1，...，$m_l\in M_k$ 序列。

2 运行密钥生成算法 $Gen(1^k)$获得一对密钥(pk，sk)。
3 计算签名 $\sigma_1 \leftarrow Sign_{sk}(m_1)$，...，$\sigma_l \leftarrow Sign_{sk}(m_l)$。
4 给敌手 pk 和 $\{\sigma_i\}_{i=1}^{l}$，并输出(m，σ)。
5 如果（1）$Vrfy_{pk}(m，\sigma)= 1$ 且（2）$m \notin \{m_1，...，m_l\}$，则敌手 A 成功。
（我们假设 A 是一种有状态算法，尤其是允许其在步骤 1 和 4 之间保持状态。）
如果对于所有多项式 $l(\cdot)$和所有概率多项式时间的敌手 A，在上面的实验中将最后一个条件 5 更改为 5′，A 的成功概率为可以忽略不计（作为 k 的函数），则该方案在已知消息攻击下强不可伪造。对于最后一个条件，现在更改为：
5′。如果（1）$Vrfy_{pk}(m，\sigma)= 1$ 且（2）$(m，\sigma) \notin \{(m_1，\sigma_1)，...，(m_l，\sigma_l)\}$，则 A 成功。

如果消息空间依赖于公钥，则会再次产生一个微妙的问题。在这种情况下，很难（或不可能）正式定义针对已知消息攻击的安全性概念。但是，在几乎任何“自然”签名方案中，当然也可以在实际中使用的任何方案中，无论公钥如何，消息空间中总是包含一些消息集 Mk'。（例如，Mk'可能由某个给定长度的所有比特串组成，这些串就可以嵌入到消息空间中，而与公钥无关）然后，我们可以将敌手限制为在 Mk'中输出消息。当公钥可以自然地分为两个部分，只有第一部分确定消息空间时，会发生另一种可能。在这种情况下，可以修改实验，在敌手要生成要签名的消息之前，仅向敌手提供公钥的第一部分。在本书的其余部分中，我们将在应对已知消息攻击的安全性时隐式地进行这种假设。

当然，我们还可以在一次已知消息攻击下定义存在性不可伪造（和强不可伪造性）。此定义与定义 1.4 完全相似，因此这里我们不给出此类定义。

1.5.3 抵抗自适应选择消息攻击的安全性

现在，我们定义最强的攻击模型，在这种模型中，攻击者可以根据公钥以及先前获得的任何签名来自适应地选择要签名的消息。为了正式定义这一点，我们将为敌手提供签名 oracle $Sign_{sk}(\cdot)$的访问。该预言机应该被认为是在固定私钥 sk 下计算签名的“黑匣子”：敌手可以向该预言机提交任何输入 m，获得签名 $\sigma \leftarrow Sign_{sk}(m)$。重要的一点是，oracle 使用的签名密钥 sk 正好与实验中的公钥 pk 一起生成；也就是说，敌手正在获得对其“攻击”的公钥有效的签名。要强调的是，签名预言机不代表敌手在现实世界中可以访问的任何物理设备。相反，此预言机只是在敌手可以说服签名者签署任何由敌手选择的消息的假设下（如第 1.4 节所述），用来可以轻松地模拟敌手与签名者的交互的模型。

定义 1.6 如果对于所有概率多项式时间的敌手 A，在以下实验中 A 的成功概率都可以忽略不计（作为 k 的函数），则签名方案(Gen，Sign，Vrfy)在选择消息攻击下是不可伪造的：
1 运行密钥生成算法 $Gen(1^k)$，获得一对密钥(pk，sk)。
2 给敌手 A 一个 pk，并允许它与签名 oracle $Sign_{sk}(\cdot)$进行交互，并根据需要在许多消息上请求签名。（我们用 $ASign_{sk}(\cdot)(pk)$表示这一点。）让 M 表示 A 询问到签名 oracle 的消息集合。
3 最终，敌手 A 输出(m，σ)。
4 如果（1）$Vrfy_{pk}(m，\sigma)= 1$ 且（2）$m \notin M$，则 A 成功。

如果针对所有概率多项式时间的敌手 A，在以下实验中 A 的成功概率可以忽略不计（作为 k 的函数）该方案在选择消息攻击下是强不可伪造的：
1 运行密钥生成算法 $Gen(1^k)$，获得一对密钥(pk，sk)。
2 给 A 一个 pk，并允许它与签名 oracle $Sign_{sk}(\cdot)$进行交互，并根据需要在许多消息上请求签名。令 $Q = \{(m_i，\sigma_i)\}$，其中 m_i 表示 A 对签名 oracle 进行的第 i 个查询，而 σ_i 是第 i 个响应。

3 最终，A 输出(m，σ)。
4 如果（1）$Vrfy_{pk}$(m，σ)= 1 且（2）(m，σ)∉Q，则 A 成功。
（在这两个定义中，我们都假定 A 是有状态算法。）

上面定义中考虑的攻击在文献中通常被称为自适应选择消息攻击，但是由于我们不考虑任何“非自适应”变体，因此我们通常会丢弃多余的限定词。

由于我们经常提到它，因此我们还为一次性攻击定义了相应的安全性概念。

定义 1.7 如果对于所有概率多项式时间的敌手 A，以下实验中 A 的成功概率都可以忽略不计(作为 k 的函数)，则签名方案（Gen，Sign，Vrfy）在一次选择消息攻击下是存在不可伪造的:
1 运行密钥生成算法 Gen(1^k)以获得一对密钥(pk，sk)。
2 给敌手 A 一个 pk，并允许它在单个消息 m_1 上请求签名。返回 $\sigma_1 \leftarrow Sign_{sk}(m_1)$。
3 敌手 A 输出(m，σ)。
4 如果（1）$Vrfy_{pk}$(m，σ)= 1 且（2）$m \neq m_1$，则 A 成功。
如果对于所有概率的多项式时间的敌手 A，在以下实验中 A 的成功概率可以忽略不计（作为 k 的函数），则该方案在一次选择消息攻击下是强不可伪造的：
1 运行密钥生成算法 Gen(1^k)，获得一对密钥(pk，sk)。
2 给敌手 A 一个 pk，并允许它在单个消息 m_1 上请求签名。返回 $\sigma_1 \leftarrow Sign_{sk}(m_1)$。
3 敌手 A 输出(m，σ)。
4 如果（1）$Vrfy_{pk}$(m，σ) = 1 且（2）(m，σ)≠(m_1，σ_1)，则 A 成功。
（通常，两个定义都假定 A 是有状态算法。）

1.5.3.1 术语

默认情况下，当我们谈论数字签名方案的“安全性”时，是指存在不可伪造性（对于适当的攻击模型）。我们也经常用“不可伪造”代替“存在性不可伪造”。为简便起见，我们有时将针对随机消息攻击，已知消息攻击和选择消息攻击的安全性分别称为 RMA 安全性，KMA 安全性和 CMA 安全性。

用术语的方式来说，我们说每当 A 输出消息/签名对(m，σ)使得 $Vrfy_{pk}$(m，σ)= 1，且 A 先前未在 m 上获得任何签名，敌手 A “在新消息上伪造签名”或“输出伪造”。我们说，每当 A 输出消息/签名对(m，σ)使得 $Vrfy_{pk}$(m，σ)= 1 且 A 先前未在消息 m 上获得签名 σ 时，A 就会“输出强伪造”。注意，每当 A 输出伪造签名时，它也会输出强伪造签名。

1.6 概念之间的关系

上一节中的定义导致了严格的安全层次结构（假设首先存在安全的签名方案）。即：

- 存在签名方案在一次选择消息攻击下根本无法伪造的，在选择的消息攻击下是可伪造的（实际上，不难发现我们在第 3.1 节中展示的方案具有此属性），以及类似的随机消息攻击和已知消息攻击。
- 存在一些签名方案，这些签名方案在随机消息攻击下是无法伪造的，但在已知消息攻击下却不是不可伪造的。类似地，有些签名方案在已知消息攻击下是不可伪造的，而在选择消息攻击下是可伪造的。在这两种情况下，构造复杂方案来证明这些观察是相对简单的，但是我们不知道具有这些性质的任何“自然”方案。（但是，我们强调，就较弱的安全性概念而言，可以证明许多自然方案是安全的，而尚无证据表明它们满足更强的安全性概念。）

- 有些签名方案在选择消息攻击下是无法伪造的，但并不是强不可伪造的（即使对手仅在一条随机消息上获得了签名）。

上述观察结果说明了为当前应用程序选择满足适当安全定义的方案的重要性；我们不能简单地“假设”一个满足弱定义的方案也会自动满足强定义。另一方面，如果一个特定的应用程序需要一个只满足较弱的安全性概念的签名方案，那么人们可以希望通过使用完全满足该安全性的方案（而不是更强大的方案）来提高效率。

1.7 通过更弱的原语实现 CMA-Security

如前一节结尾所讨论的，我们定义的安全性较弱的概念（针对随机消息攻击的安全性和针对已知消息攻击的安全性）可能在某些受限场景中提供有意义的保证。更有趣的是，它们的用途是构建满足我们最强定义的方案——选择消息攻击的安全性。在这一节中，我们探讨了这种可能性，我们将探讨这种可能性，并展示如何将相对于较弱概念的安全方案转换为在最强攻击模型中安全的方案。我们展示的构造不仅从理论角度来看是有趣的，而且具有实际意义：许多方案依赖于以下构造中使用的思想，并且转换本身相对有效（将原始较弱方案的效率降低大约两倍）。

1.7.1 来自 RMA-Security 的 CMA-Security

我们首先显示如何基于任意的 RMA 安全方案构造 CMA 安全方案。基本思想是将每个消息 m“拆分”为两个随机片段 m_L，m_R，但要受 $m_L \oplus m_R = m$ 的约束。然后，使用 RMA 安全方案的两个独立实例对 m_L 和 m_R 进行签名。（为防止敌手将来自两个不同签名消息的片段混合匹配，随机选择一个 nonce，并与每个片段一起签名。）细节如下。

构造 1.1：来自 RMA-security 的 CMA-安全性

令 Π=(Gen，Sign，Vrfy)是消息长度为 k + q(k)的签名方案。对于长度为 q = q(k)的消息，构造签名方案 $\Pi^* = (Gen^*, Sign^*, Vrfy^*)$如下：

密钥生成：算法 $Gen^*(1^k)$定义如下：

- 运行两次（独立）$Gen(1^k)$，获得密钥(pk_L, sk_L)，(pk_R, sk_R)。（我们用“L”和“R”代表“左”和“右”。）
- 公钥为 $pk^* = (pk_L, pk_R)$，私钥为 $sk^* = (sk_L, sk_R)$。

签名生成：算法 $Sign^*_{sk^*}(m)$定义如下：

- 将 sk^*分为 sk_L，sk_R。
- 选择 $r \leftarrow \{0,1\}^k$ 和 $m_L \leftarrow \{0,1\}^q$。设置 $m_R := m \oplus m_L$。
- 计算 $\sigma_L \leftarrow Sign_{skL}(r \| m_L)$和 $\sigma_R \leftarrow Sign_{skR}(r \| m_R)$，其中“‖”表示级联。
- 输出签名 $\sigma^* = (r, m_L, m_R, \sigma_L, \sigma_R)$。

签名验证：算法 $Vrfy^*_{pk}(m, \sigma^*)$定义如下：将 σ^*解析为$(r, m_L, m_R, \sigma_L, \sigma_R)$和 $pk=(pk_L, pk_R)$。然后，验证算法输出 1，当且仅当 $m =? m_L \oplus m_R$，同时

$$Vrfy_{pkL}(r \| m_L, \sigma_L) =? 1$$
$$Vrfy_{pkR}(r \| m_R, \sigma_R) =? 1。$$

不难看出上述方案是正确的。（并不是必须要在 $Sign^*$输出的签名中包含 m_R，但这会使该方案的描述更加透明。）以下定理表明，上述构造达到了所需的安全级别：

定理 1.1 如果 Π=(Gen，Sign，Vrfy)在随机消息攻击下是存在性不可伪造（强不可伪造），

则构造 1.1 给出的 Π^* =(Gen*，Sign*，Vrfy*) 在自适应选择消息攻击下存在性不可伪造（强不可伪造）。

证明。我们将重点放在强不可伪性的情况下，但是有关存在性不可伪的声明是通过对证明检查得出的。直觉很简单：在方案 Π^*中，由底层签名方案 Π 所签名的"消息"（即 $r\|m_L$ 和 $r\|m_R$）始终是均匀独立的，而不考虑敌手对 m 的选择。当然，$r\|m_L$ 和 $r\|m_R$ 的联合分布取决于 m，但这里我们感兴趣的它们的的单独分布，因为 $r\|m_L$ 和 $r\|m_R$ 各自使用不同的密钥签名。这表明 Π 抵抗随机消息攻击的安全性足以证明 Π^*抵抗选择消息攻击的安全性。唯一的问题是，如果随机数值 r 被使用过两次，那么伪造就变得微不足道了。例如，如果敌手获得消息 m 上的签名（r，m_L，m_R，σ_L，σ_R）和消息 m'上的签名（r，m'_L，m'_R，σ'_L，σ'_R），则这意味着 σ_L 是关于 pk_L 在 $r\|m_L$ 上的有效签名，而 σ'_R 是关于 pk_R 在 $r\|m'_R$ 上的有效签名。因此，敌手可以输出

（r，m_L，m'_R，σ_L，σ'_R）

关于消息 $m_L \oplus m'_R$ 上的有效签名（并且此消息可能不等于 m 或 m'之一）。幸运的是，不难证明使用两次随机数的可能性很小。

转向形式化证明，令 A*为攻击 Π^*的 PPT 敌手，并用$(m, \sigma^*) \leftarrow \text{Expt}_{A^*,\Pi^*}(1^k)$表示实验

$$(pk^*, sk^*) \leftarrow \text{Gen}^*(1^k); (m, \sigma^*) \leftarrow (A^*)^{\text{Sign}^*_{sk}(\cdot)}(pk^*).$$

设 Forge 为 $\text{Vrfy}^*_{pk^*}(m,\sigma^*) = 1$ 和$(m,\sigma^*) \notin Q$ 的事件，其中 Q 如定义 1.6 所示。定义

$$\text{Succ}_{A^*,\Pi^*}(k) = \Pr[(m,\sigma) \leftarrow \text{Expt}_{A^*,\Pi^*}(1^k) : \text{Forge}]$$

并注意，这正是定义 1.6 中定义的 A*成功的概率。因此，我们的目标是证明 $\text{Succ}_{A^*,\Pi^*}(k)$可忽略不计。

令 $l = l(k)$表示 A*对安全参数 k 的签名 oracle 进行查询的最多次数，不失一般性，假设 A*总是精确地进行这么多次查询；请注意，因为 A*在多项式时间内运行，所以 l 必须是多项式。在给定的实验 $\text{Expt}_{A^*,\Pi^*}(1^k)$中，让 m_i 表示 A*提交给其签名 oracle 的第 i 条消息，$\sigma_i^* = (r_i, m_{L,i}, m_{R,i}, \sigma_{L,i}, \sigma_{R,i})$ 表示收到的第 i 个签名。$\sigma^* = (r^*, m^*_L, m^*_R, \sigma_L^*, \sigma_R^*)$表示由 A*输出的签名。现在定义以下事件：

- 让 Repeat 表示这样的事件：A*从其签名 oracle 中获得的两个签名，其随机数 r 使用相同的值，即，对于某些 $i \neq j$，$r_i = r_j$。
- 令 $Q_L = \{(r_i \| m_{L,i}, \sigma_{L,i})\}$表示"左"消息/签名对的集合。设 Forge_L 表示事件 $\text{Vrfy}_{pkL}(r^*\|m^*_L, \sigma^*_L) = 1$ 和$(r^*\|m^*_L, \sigma^*_L) \notin Q_L$。
- 同样，令 $Q_R = \{(r_i \| m_{R,i}, \sigma_{R,i})\}$表示为"右"消息/签名对的集合，设 Forge_R 为事件 $\text{Vrfy}_{pkR}(r^*\|m^*_R, \sigma^*_R) = 1$ 和$(r^*\| m^*_R, \sigma^*_R) \notin Q_R$。

我们推断，每当发生 Forge 时，都会发生 Repeat，Forge_L 或 Forge_R 中的至少一种。要看到这一点，我们假设发生了伪造而没有发生 Repeat。由于发生伪造，我们知道 $\text{Vrfy}_{pkL}(r^*\|m^*_L, \sigma^*_L) = 1$ 且 $\text{Vrfy}_{pkR}(r^*\|m^*_R, \sigma^*_R) = 1$。由于不发生 Repeat，因此我们知道 $r^* = r_i$，最多为 i 的一个值。有两种情况需要考虑：

情况 1:对于任何 i 值，$r^* \neq r_i$。在这种情况下，显然，对于所有 i 都有 $r^* \| m^*_L \neq r_i \| m_{L,i}$ ，而且此，尤其是$(r^* \| m^*_L, \sigma^*_L) \notin Q_L$ 。这意味着出现了 $Forge_L$。（通过对称参数，在这种情况下也会发生 $Forge_R$。）

情况 2: $r^* = r_i$ 对于某些（唯一）i。如果$(m^*_L, \sigma^*_L)=(m_{L,i}, \sigma_{L,i})$和$(m^*_R, \sigma^*_R) = (m_{R,i}, \sigma_{R,i})$两者，则我们有 $m = m_i$ 和 $\sigma^* = \sigma^*_i$ 与$(m, \sigma^*) \notin Q$（自发生伪造以来）相矛盾。因此，必须是$(m^*_L, \sigma^*_L) =(m_{L,i}, \sigma_{L,i})$（在这种情况下出现 $Forge_L$）或$(m^*_R, \sigma^*_R) \neq (m_{R,i}, \sigma_{R,i})$（在这种情况下，会发生 $Forge_R$）。

我们得出结论

$$\Pr[\mathrm{Expt}_{A^*,\Pi^*}(1^k) : \mathrm{Forge}] \leq \tag{1.1}$$

$$\Pr[\mathrm{Expt}_{A^*,\Pi^*}(1^k) : \mathrm{Forge}_L] + \Pr[\mathrm{Expt}_{A^*,\Pi^*}(1^k) : \mathrm{Forge}_R]$$

$$+\Pr[\mathrm{Expt}_{A^*,\Pi^*}(1^k):\mathrm{Repeat}].$$

我们表明，右侧的每个项都可以忽略不计，从而完成了定理的证明。我们优先处理最容易分析的项。

Claim. $\Pr[\mathrm{Expt}_{A^*,\Pi^*}(1^k):\mathrm{Repeat}]$是可忽略的。

证明: 该 **Claim** 很容易根据“生日问题”计算得出。（有关更多信息，请参见[72，附录 A.4]。）具体地说，我们从大小为 2^k 的集合$\{0,1\}^k$ 中均匀选择了 l 个随机数 r_1，...，r_l。 这些值中的两个相等的概率最多为 $l^2 / 2^{k+1}$，在 k 中可以忽略不计。

Claim. $\Pr[\mathrm{Expt}_{A^*,\Pi^*}(1^k):\mathrm{Forge}_L]$是可忽略的。

证明: 为了证明这一 claim，我们将其归约为方案 Π 的强不可伪性。考虑以下 PPT 敌手 A，将 A^*作为子程序，并在随机消息攻击中攻击方案 Π：

Algorithm *A*:
给定该算法一个公钥 pk，使用 $Gen(1^k)$算法生成的随机消息 m'_1，...，$m'_l \in \{0,1\}^{k+q}$ 上的 l 个签名 σ_1，...，σ_l。

- 设置 pk_L：= pk。
- 运行 $Gen(1^k)$以获取密钥(pk_R, sk_R)。
- 设置 pk^*：=(pk_L, pk_R)并运行 $A^*(pk^*)$。
- 当 A^*在第 i 条消息 m_i 上请求签名时，请执行以下操作：

1. 让 r_i 是 m'_i 的前 k 比特，让 $m_{L,i}$ 是 m'_i 的后 q 个比特。
2. 设置 $\sigma_{L,i} := \sigma_i$。
3. 设置 $m_{R,i}$:＝$m_i \oplus m_{L,i}$。
4. 计算 $\sigma_{R,i} \leftarrow \mathrm{Sign}_{skR}(r_i \| m_R)$。
5. 将签名$(r_i, m_{L,i}, m_{R,i}, \sigma_{L,i}, \sigma_{R,i})$返回给 A^*。

• 当 A^*输出(m，$\sigma^* = (r^*, m^*_L, m^*_R, \sigma^*_L, \sigma^*_R)$)时，则敌手 A 输出($r^* \| m^*_L, \sigma^*_L$)。

首先观察到敌手 A 为 A^*提供了完美的模拟；也就是说，当敌手 A 运行时（以及由 Gen(1^k)生成 pk 时）的 A^*视图与 $\text{Expt}^{A^*,\Pi^*}(1^k)$ 中的 A^*视图相同。不难发现，在两个实验中，公钥 pk^*是同一分布的。至于 A^*的签名查询的回答，请注意，在两种情况下，它们也是同分布的；在此，我们依赖于这样的事实，即 m'_i是随机地均匀地（独立地）选择的。

总结证明，我们仅观察到无论何时发生 Forge_L，敌手 A 都会输出强伪造。因此，在定义 1.3 的意义上（即针对随机消息攻击），A 攻击方案 Π 的成功概率完全等于 $\Pr[\text{Expt}^{A^*,\Pi^*}(1^k) : \text{Forge}_L]$ 。由于假定 Π 在随机消息攻击下是强不可伪造的，因此 A 攻击方案 Π 的成功概率忽略不计。

一个完全相似的论点表明 $\Pr[\text{Expt}^{A^*,\Pi^*}(1^k) : \text{Forge}_R]$ 是可忽略的。由于方程（1.1）右侧的每个项均可以忽略不计，因此我们得出结论，在选择消息攻击下，A^*攻击 Π^*的成功概率可以忽略不计。这样就完成了定理的证明。

1.7.2 来自 KMA-Security 的 CMA-Security

在本节中，我们说明如何构造一个签名方案，该方案可以抵抗在已知消息攻击下的任何安全方案的选择消息攻击。请注意，由于任何 KMA 安全的方案也是 RMA 安全的，因此我们也可以使用上一节中的构造。尽管如此，这里描述的构造提供了一种替代方法，当应用于特定方案时，有时这种方法会更有效。

这里的关键思想是使用间接级别。令 Π 为 KMA 安全的方案，令 Π'为 KMA 安全的一次性签名方案。我们构造如下的 CMA 安全的方案 Π^*：使用 Π 生成公钥 pk。为了签名消息 m，签名者使用 Π'生成新的公钥 pk'；签名公钥 pk'对应 pk；签名消息 m 对应 pk'。注意，此“间接”消除了任何“消息”与对其进行签名的公钥之间的任何依赖：m 独立于 pk'，而 pk'独立于 pk。详细信息如下。我们强调每次签名新消息时都会生成一个新的密钥对(pk'，sk')。不难看出上述方案是正确的。现在，我们证明它可以实现的期望的安全级别。

构造 1.2: 来自 KMA-Security 的 CMA-Security

设 Π'=(Gen'，Sign'，Vrfy')是签名方案，其中公钥长度为 q = q(k)，设 Π=(Gen，Sign，Vrfy)是签名方案，其中消息空间 M_k 包括长度为 q(k)的所有比特串。构造签名方案 Π^*=(Gen^*，Sign^*，Vrfy^*)，其消息空间正是Π'的消息空间，如下所示：

密钥生成：$\text{Gen}^*(1^k)$只需运行 Gen(1^k)即可生成密钥(pk，sk)。它们分别是公钥和私钥。

签名生成：算法 $\text{Sign}^*_{sk}(m)$定义如下：

1.运行 Gen'(1^k)生成密钥(pk'，sk')。

2.使用 sk 对 pk′进行签名：计算 $\sigma \leftarrow \text{Sign}_{sk}(pk')$。

3.使用 sk'对 m 进行签名；计算 $\sigma' \leftarrow \text{Sign}'_{sk'}(m)$。

4.输出签名 $\sigma^* = (pk', \sigma, \sigma')$。

签名验证：算法 $\text{Vrfy}^*_{pk}(m, \sigma^*)$定义如下：将 σ^*解析为(pk′，σ，σ′)。然后，当且仅当 σ 是 pk'（相对于 pk）上的有效签名并且 σ'是 m（相对于 pk'）上的有效签名时，才输出 1；当且仅

$\text{Vrfy}_{pk}(pk', \sigma)$=？1 和 $\text{Vrfy}'_{pk'}(m, \sigma')$=？1。

定理 1.2 如果 Π=（Gen，Sign，Vrfy）在已知消息攻击下是存在性不可伪造的（强不可伪），而 Π'=（Gen'，Sign'，Vrfy'）在一次已知消息攻击下是不可伪造的（强不可伪的），则在自适应选择消息攻击下，Π^*=(Gen*，Sign*，Vrfy*)是存在性不可伪造（强不可伪造）。

证明：我们证明了定理的存在不可伪性，但是可以通过进行适当的修改来推导强不可伪造下的证明。这里的直觉上还是相当简单的。假设敌手自适应地选择消息 m_1，...并获得了一系列签名(pk_1'，σ_1，σ'_1)，...。一个直接的观察结果是，如果敌手在新消息 m 上伪造签名(pk'，σ，σ')，那么对于某些 *i*，要么是 pk'= pk'_i，要么不是。（为简单起见，我们对所有 i≠j 假设 $pk'_i \neq pk'_j$；正如我们将看到的，在正式证明中不需要此假设。）如果 pk'=pk'_i，则敌手有效地伪造了关于方案 Π'和公钥 pk'_i 在新消息 m 上的签名 σ'；否则，敌手就方案 Π 和公钥 pk 在新的“消息”pk'上有效地伪造了一个签名 σ。此外，我们可能会注意到，所有签名的“消息”都是独立于相应的公钥选择的：特别是，使用 pk 签名的密钥{pk'_i}由签名者选择（签名者也彼此独立地生成它们，与密钥 pk 无关），并且每个消息 m_i（尽管是由敌手选择的）都是在敌手知道将用于签名 m_i 的密钥 pk'_i 的值之前选择的。因此，已知消息攻击下的 Π，Π'的安全性就足够了。此外，由于每个密钥 pk'_i 仅用于签名一个消息 m_i，因此在一次已知消息攻击下 Π'满足不可伪造的。

现在，我们进行正式证明。给定一个 PPT 敌手 A*，攻击签名方案 Π*，用$(m, pk', \sigma, \sigma') \leftarrow \mathrm{Expt}_{A^*,\Pi^*}(1^k)$ 表示实验：

$$(pk, sk) \leftarrow \mathrm{Gen}^*(1^k); (m, pk', \sigma, \sigma') \leftarrow (A^*)^{\mathrm{Sign}^*_{sk}(\cdot)}(pk)$$

令 m_i 表示 A*提交给其签名 oracle 的第 *i* 个消息，{(pk'$_i$，σ_i，σ'_i)}表示作为收到的第 *i* 个签名。设 Forge 为 Vrfy^*_{pk}(m，pk'，σ，σ')= 1 且 m∉{m_i}的事件，并定义

$$\mathrm{Succ}^*_{A^*,\Pi^*}(k) = \Pr[(m, pk', \sigma, \sigma') \leftarrow \mathrm{Expt}_{A^*,\Pi^*}(1^k) : \mathrm{Forge}];$$

在定义 1.6 中，这恰好是 A*在攻击方案 Π*中的成功概率。我们的目标是证明 $\mathrm{Succ}^*_{A^*,\Pi^*}(k)$是可忽略的。

现在我们将以 A*在其伪造尝试中是否重用某个密钥 pk'_i 为条件。即，令 Reuse 是对于某些 *i* 的 pk′= pk'_i 的事件，然后定义

$$Succ^{\overline{Reuse}}_{A*,\pi*}(k) \stackrel{\text{def}}{=} \Pr[(m, pk', \sigma, \sigma') \leftarrow Expt_{(A*,\pi*)}(1^k) : Forge \wedge \overline{Reuse}]$$

$$Succ^{Reuse}_{A*,\pi*}(k) \stackrel{\text{def}}{=} \Pr[(m, pk', \sigma, \sigma') \leftarrow Expt_{(A*,\pi*)}(1^k) : Forge \wedge Reuse]$$

当然，我们有：

$$Succ^{*}_{A*,\pi*}(k) = Succ^{\overline{Reuse}}_{A*,\pi*}(k) + Succ^{Reuse}_{A*,\pi*}(k)$$

我们证明，右侧的每个项都是可以忽略的。在不失一般性的前提下，我们假设 A*总是针对某个多项式 *l* 在恰好 *l* = *l*(k)个消息上请求签名。

Claim. $Succ^{\overline{Reuse}}_{A*,\pi*}(k)$ 是可忽略的。

*证明：*我们构造一个 PPT 的敌手 A，将 A^*作为子程序，在已知消息攻击下攻击方案 Π，且成功概率正好为$Succ_{A*,\pi*}^{\overline{Reuse}}(k)$，因此通过方案 Π 的安全性得到了 ***Claim***。算法 A 定义如下：

算法 A：

- 总共运行 Gen'(1^k) l 次，获的密钥{(pk'$_i$，sk'$_i$)}$^{l}_{i=1}$。
- 输出 pk'$_1$，…，pk'$_l$。接收公钥 pk 和签名{σ_i}$^{l}_{i=1}$。（注意：对于 pk，每个 σ_i 是的相对应 pk'$_i$ 的有效签名）
- 运行 $A^*(pk)$。当 A^*对第 i 个消息 m$_i$ 请求签名时：
  1.计算 $\sigma'_i \leftarrow Sign'_{sk'i}(m_i)$。
  2.将签名(pk'$_i$，σ_i，σ'_i)返回给 A^*。
- 当 A^*输出(m，σ^* =(pk′，σ，σ′))时，输出(pk′，σ)。

我们可以看出，上述实验中 A^*的视图与 $Expt_{A^*,\Pi^*}(1^k)$中的视图相同。因此，在上述实验中发生了 Forge∧$\overline{Reuse}$事件，敌手 A 的概率恰好为$Succ_{A*,\pi*}^{\overline{Reuse}}(k)$。由于没有发生重用，因此我们有 pk′∉{pk′$_i$}；因为发生了 Forge，所以 σ 一定是 pk'上的有效签名（相对于公钥 pk）。因此，我们得出结论，A 在新消息上输出伪造的概率为 $Succ_{A^*,\Pi^*(k)}$。方案 Π 在已知消息攻击下的假设的安全意味着实验 $Succ_{A^*,\Pi^*(k)}$ 成功的概率可以忽略不计。

Claim. $Succ_{A*,\pi*}^{Reuse}(k)$是可忽略的。

*证明：*现在，我们构造 PPT 敌手 A'，在一次已知消息的攻击下攻击方案 Π'，成功概率至少为$Succ_{A*,\pi*}^{Resue}(k)/l$。由于 l 是多项式，并且假设在一次已知消息攻击时，Π'是存在不可伪造的，因此得到 ***Claim***。

算法 A′：

- 计算(pk，sk)←Gen(1^k)，并选择一个随机索引 i^*←{1，...，l}。
- 运行 A^*(pk)，对于第 i 个消息 m$_i$ 的签名查询，如下：

情况 1：$i \neq i^*$。

1.运行 Gen'(1^k)生成密钥(pk'$_i$，sk'$_i$)。
2.计算 $\sigma_i \leftarrow Sign_{sk}(pk'_i)$和 $\sigma'_i \leftarrow Sign_{sk'i}(m_i)$。
3.将签名(pk'$_i$，σ_i，σ'_i)返回给 A^*。

情况 2：$i = i^*$。

1.输出 m$^*_i$ 并接收公钥 pk'$_{i*}$和签名 σ'$_{i*}$。
2.计算 $\sigma_{i*} \leftarrow Sign_{sk}(pk'_{i*})$。
3.将签名(pk′$_{i*}$，σ_{i*}，σ'$_{i*}$)返回给 A。

- 当 A^*输出(m，σ^* =(pk′，σ，σ′))时，如果 pk′= pk′$_{i*}$输出(m，σ')（否则不输出）。

也就是说，A'选择一个随机索引 i^*并正常回答 A^*的所有签名查询，除第 i^*个查询外，A'可以这样做，因为它知道对应于 pk 的“主”私钥 sk。对于消息 m$^*_i$ 的第 i^*个签名查询，敌手 A'将 m$^*_i$ 输出到其自己的“oracle”，并接收生成的公钥 pk$_{i*}$（使用 Gen'（1^k）生成）以及在消息上 m$_{i*}$的签名 σ$_{i*}$。然后它自己计算 pk′$_{i*}$上的签名 σ$_{i*}$（使用 sk），并返回（pk'$_{iI}$，σ$_{i*}$，σ'$_{i*}$）给 A^*。很容易看出上述实验中 A^*的视图与 $Expt_{A^*,\Pi*}(1^k)$中的视图相同，因此，在上述实验中发生了 Forge∧Reuse 事件，敌手 A 的概率恰好为$Succ_{A*,\pi*}^{Reuse}(k)$。当发生重用时，至少有一个索引 i，其中 pk'= pk'$_i$；因为给定 A^*的视图，i*的分布是均匀的，所以 pk'= pk'$_{i*}$的概率至少为 1 / l。假设发生伪造，必然是 m≠m$^*_i$ 且 Vrfy'$_{pki'*}$(m，σ')= 1 的情况，因此 A'在这种情况下输出有

效的伪造。总而言之，A' 输出有效伪造，其概率为至少 $Succ_{A^*, \Pi^*}(k)/l$。因此得到 **claim**。前面两个 **claim** 完成证明。

使用预计算可提高效率。除了可用于从 KMA 安全的方案构造 CMA 安全的方案之外，构造 1.2 还可以用于使用不依赖于所签名消息的预计算来提高签名生成的效率。（可以应用预计算来提高效率的签名方案有时被称为“在线/离线”方案。）具体来说，签名生成的步骤 1 和步骤 2 与要签名的消息无关；请参见图 5。因此，可以提前执行它们，并缓存结果。当要签名的消息 m 已知时，所需要做的就是使用底层的一次性签名方案在 m 上计算签名。正如我们将在 8.2.3 节中看到的那样，可以基于多种数论假设构造具有高效签名的一次性签名方案。

1.8 从不可伪造到强不可伪造

在前两节中，我们根据签名方案可以承受的攻击来详述了签名方案的安全性。在这里，我们详述了不可伪造的概念，示范如何将不可伪造（在选择消息攻击下）的方案转换为强不可伪造的方案。我们将签名方案用作构建单元，这种签名方案在一次选择消息攻击下是强不可伪造的。 我们将在本书的后面部分看到，这样的方案相对容易构造。

构造 1.3: 来自不可伪造的强不可伪造

设 Π=(Gen，Sign，Vrfy)和 Π'=(Gen'，Sign'，Vrfy')是签名方案；为简单起见，假设它们每个都可以签名无限长度的消息(请参阅下一节的构造)。考虑以下签名方案 $\Pi^*=(Gen^*, Sign^*, Vrfy^*)$的构造：

密钥生成：$Gen^*(1^k)$只需运行 $Gen(1^k)$即可生成密钥(pk，sk)。它们分别是公钥和私钥。
签名生成：算法 $Sign^*_{sk}(m)$定义如下：

- 运行 $Gen'(1^k)$生成密钥(pk'，sk')。
- 使用 sk 对 pk'‖m 进行签名：即，计算 $\sigma \leftarrow Sign_{sk}(pk'\|m)$。
- 使用 sk'对 σ 进行签名；即，计算 $\sigma' \leftarrow Sign'_{sk'}(\sigma)$。
- 输出签名 $\sigma^* = (pk', \sigma, \sigma')$。

签名验证：算法 $Vrfy^*_{pk}(m, \sigma^*)$定义如下：将 σ^*解析为(pk', σ, σ')。然后，当且仅当 σ 是 pk'‖m 上的有效签名（关于 pk）且 σ'是 σ 上的有效签名（关于 pk'）时，才输出 1；即当且仅当

$$Vrfy_{pk}(pk'\|m, \sigma) =? 1 \text{ 和 } Vrfy'_{pk'}(\sigma, \sigma') =? 1。$$

定理 1.3 如果在自适应选择消息攻击下 Π=(Gen，Sign，Vrfy)是存在性不可伪造，而 Π'=(Gen'，Sign'，Vrfy')在一次选择消息攻击下是强不可伪造，则 $\Pi^*=(Gen^*, Sign^*, Vrfy^*)$在自适应选择消息攻击下是强不可伪造的。

证明：对于强不可伪造的直觉很简单。考虑分别在消息 m_1，...，m_l 上获得的签名序列$(pk'_1, \sigma_1, \sigma'_1)$，...，$(pk'_l, \sigma_l, \sigma'_l)$。以压倒性的概率，{pk'$_i$}中的每一个都是不同的。令(m，(pk'，σ，σ'))为敌手的强伪造。我们可以分几种情况：

情况 1：pk'不属于{ pk′$_i$ }。显然，这导致了关于方案 Π 的伪造。
情况 2：对于某些唯一的 i，pk′= pk′$_i$。注意我们必须有 m = m$_i$，否则我们有一个关于 Π 的伪

造。然后有两个情况：
• $\sigma \neq \sigma_i$。 这给出了关于方案 Π'（和公钥 pk'_i）的伪造。
• $\sigma = \sigma_i$。 这意味着 $\sigma' \neq \sigma'_i$（否则就不存在关于 Π^*的强伪造），但这会给方案 Π'（和公钥 pk'_i）带来强伪造。
对于正式证明，让 PPT 敌手 A^*攻击 Π^*，并用$(m, pk', \sigma, \sigma') \leftarrow Exp_{tA^*,\Pi^*}(1^k)$实验：

$(pk, sk) \leftarrow Gen^*(1^k)$;$(m, pk', \sigma, \sigma') \leftarrow (A^*)Sign^*_{sk(\cdot)}(pk)$。

令 m_i 表示 A^*提交给其签名 oracle 的第 i 个消息，而让$\{(pk'_i, \sigma_i, \sigma'_i)\}$表示收到的第 i 个签名。设 Forge 为 $Vrfy^*_{pk}(m, pk', \sigma, \sigma') = 1$ 且$(m, \sigma^* = (pk', \sigma, \sigma')) \notin Q$ 的事件，其中 Q 如定义 1.6。定义

$$Succ_{A^*,\Pi^*}(k) = \Pr[(m, pk, \sigma, \sigma) \leftarrow Expt_{A^*,\Pi^*}(1) : Forge];$$

像通常一样，根据定义 1.6 中定义的强不可伪造性，这恰恰是 A^*在攻击方案 Π^*的成功概率。我们的目标是证明 $Succ_{A^*,\Pi^*}(k)$可忽略不计。

就像定理 1.2 的证明一样，我们将以 A^*是否在其伪造中重用密钥 pk'_i 中的一个为条件。但是，在这里，我们对该事件的定义稍有不同：对于某些 i，Reuse 现在是$(m, pk') = (m_i, pk'_i)$的事件。定义

$$Succ^{\overline{Reuse}}_{A^*,\pi^*}(k) \stackrel{\text{def}}{=} \Pr[(m, pk', \sigma, \sigma') \leftarrow Expt_{(A^*,\pi^*)}(1^k) : Forge \wedge \overline{Reuse}]$$

$$Succ^{Reuse}_{A^*,\pi^*}(k) \stackrel{\text{def}}{=} \Pr[(m, pk', \sigma, \sigma') \leftarrow Expt_{(A^*,\pi^*)}(1^k) : Forge \wedge Reuse]$$

当然，

$$Succ^{*}_{A^*,\pi^*}(k) = Succ^{\overline{Reuse}}_{A^*,\pi^*}(k) + Succ^{Reuse}_{A^*,\pi^*}(k)$$

我们证明右边的每个项都是可以忽略的，从而证明了该定理。

Claim. $Succ^{\overline{Reuse}}_{A^*,\pi^*}(k)$ 是可忽略的。

*证明：*该 **claim** 的证明与定理 1.2 的证明中的第一个 **claim** 的证明几乎相同，因此将其省略。

Claim. $Succ^{Reuse}_{A^*,\pi^*}(k)$是可忽略的。

*证明：*该定理的证明与定理 1.2 的证明中的第二个 **Claim** 的证明非常相似；我们在这里给出了强不可伪造性的详细信息。
在不失一般性的前提下，假设 A^*恰好做了 $l = l(k)$的签名查询。我们在一次选择消息攻击下构造了一个攻击 Π'的 PPT 敌手 A'，成功概率（就强不可伪造性而言）至少 $Succ_{A^*,\Pi^*}(k)/l$。由于 l 是多项式和 Π 在一次已知消息攻击下是不可伪造的，因此 **Claim** 得证。
算法 A'：
给该算法一个公钥 PK'（使用 $Gen'(1^k)$生成），并获得对该签名 Oracle $Sign_{SK'}(\cdot)$的访问权，它只能查询一次。
• 计算$(pk, sk) \leftarrow Gen(1^k)$，选择一个随机索引 $i^* \leftarrow \{1, ..., l\}$。设置 $pk'_{i^*} := PK'$。
• 运行 $A^*(pk)$，回答消息 m_i 的第 i 个签名查询如下：
情况 1：$i \neq i^*$。
1. 运行 $Gen'(1^k)$生成密钥(pk'_i, sk'_i)。

2. 计算 $\sigma_i \leftarrow \text{Sign}_{sk}(pk'_i \| m_i)$和 $\sigma'_i \leftarrow \text{Signsk}'_i(\sigma_i)$。
3. 将签名$(pk'_i, \sigma_i, \sigma'_i)$返回到 A^*。
情况 2：$i = i^*$。
1. 计算 $\sigma^*_i \leftarrow \text{Sign}_{sk}(pk'_{i^*} \| m_{i^*})$。
2. 查询 σ^*_i 的签名 oracle 并返回签名 σ'_{i^*}。
3. 将签名$(pk^*_i, \sigma_i, \sigma^*_i)$返回给 A。
•当 A^*输出$(m, \sigma^* = (pk', \sigma, \sigma'))$时，如果 $pk' = pk'_{i^*}$ 输出(m, σ')（否则不输出）。
如先前定理的证明，上述实验中 A^*的视图与其在 $\text{Expt}_{A^*, \Pi^*}(1^k)$中的视图相同，因此，Forge∧Reuse 的概率在上面的实验中发生的重用恰好是$Succ^*_{A^*,\pi^*}(k)$。当发生重用时，至少有一个索引 i，其$(m, pk') = (m_i, pk'_i)$；因为给定 A^*的视图，i^*的分布是均匀的，所以$(m, pk') = (m_{i^*}, pk'_{i^*})$的概率至少为 $1 / l$。假设情况如下。
由于 Forge 事件发生了，我们知道 $\text{Vrfy}'_{pk' i^*}(\sigma, \sigma') = 1$。此外，我们必须具有$(m, pk, \sigma, \sigma) \neq (m_{i^*}, pk'_{i^*}, \sigma_{i^*}, \sigma'_{i^*})$。（回想一下，在这里，Forge 表示 A^*输出了强伪造。）由于我们假设$(m, pk') = (m_{i^*}, pk'_{i^*})$，所以这意味着$(\sigma, \sigma') \neq (\sigma_{i^*}, \sigma'_{i^*})$。但是，这意味着 A'输出了强伪造。
总之，A' 输出一个强伪造，其概率至少为$Succ^{Reuse}_{A^*,\pi^*}(k)/l$。Claim 和定理得证。

1.9 延长消息长度

在本章结束时，我们将演示如何扩展 k 位消息的签名方案以提供任意长度和可变长度消息的签名方案。（从技术上讲，我们只处理短于 $2^{k/4}$ 比特的消息，由于此界限是指数的，因此这不是一个严重的限制。）虽然我们将在下一章中看到实现同一目标的其他方法，但是本转换的优点是它不需要任何其他原语。

构造 1.4：从“短”消息到任意长消息
令 Π=(Gen，Sign，Vrfy)是 k 位消息的签名方案。对于长度小于 $2^{k/4}$ 的消息，构造签名方案 Π^*=(Gen*，Sign*，Vrfy*)，如下所示：
密钥生成：$\text{Gen}^*(1^k)$只需运行 $\text{Gen}(1^k)$，即可生成密钥(pk，sk)。它们分别是公钥和私钥。
签名生成：算法 $\text{Sign}^*_{sk}(m)$定义如下：
• 令 $L < 2^{k/4}$ 为 m 的长度，并将 m 解析为 l 个块 m_1，...，m_l，每个块的长度为 k / 4。 （如果需要，最后一块用 0 填充，尽管在确定长度 L 时不计入任何此类填充。）
• 选择一个随机标识符 $r \leftarrow \{0,1\}^{k/4}$。
• 对于 $i = 1$，...，l，计算 $\sigma_i \leftarrow \text{Sign}_{sk}(r \| L \| i \| m_i)$，其中 L 和 i 是唯一的编码为长度为 k / 4 的字符串。
• 输出签名 $\sigma := \langle r, \sigma_1, ..., \sigma_l \rangle$。
签名验证：算法 $\text{Vrfy}^*_{pk}(m, \sigma)$定义如下：
• 令 $L < 2^{k/4}$ 为 m 的长度，并将 m 解析为 l 个块 m_1，...，m_l，每个块的长度为 k / 4。（必要时用 0 填充，尽管在确定长度时也不再计算该填充）。
• $\sigma := (r, \sigma_1, ..., \sigma_{l'})$。
• 对于 $1 \le i \le l$，如果 $l' = l$ 和 $\text{Vrfy}_{pk}(r \| L \| i \| m, \sigma_i) = 1$，输出 1。
定理 1.4 如果在自适应选择消息攻击下 Π 是存在性不可伪造的（强不可伪造的），那么在自适应选择消息攻击下 Π^*是存在性不可伪造的（强不可伪造的）。
证明：我们证明（除极小概率外）关于 Π^*的伪造意味着对 Π 的伪造。我们将其转化为正式的证明将留待练习。我们讨论存在不可伪造的情况，但是强不可伪的情况基本上是相同的。令 pk 为签名者的公钥。假设总共签署了 $q = q(k)$条消息，并令 $r^{(i)}$是签署第 i 条消息时签署者选择的标识符。观察到$\{r^{(i)}\}_{i=1}^{q}$都是不同的。对于证明的其余部分，我们假设情况是这样，

并证明（在此假设下）关于 Π^*的伪造意味着关于Π的伪造。

考虑关于Π^*的一些伪造(m，σ)。令 L <2k / 4 为 m 的长度，并将 m 解析为 *l* 个块 m_1，...，m_l，每个块的长度为 k / 4（通常填充为 0）。将 σ 解析为（r，σ_1，...，σ_l）。有两种情况：

情况 1：$r \notin \{ r^{(i)} \}_{i=1}^{q}$。由于(m，σ)是有效的伪造，因此特别假设

$$Vrfy_{pk}(r \| L \| 1 \| m_1, \sigma_1) = 1,$$

由于 Π 尚未签名任何带有前缀 r 的消息（k 位）。因此$(r \| L \| 1 \| m_1, \sigma_1)$是对 Π 伪造的。

情况 2：对于某些唯一 *i*，$r = r^{(i)}$。（唯一性来自所有$\{ r^{(i)} \}$都不同的假设。）令 $m^{(i)}$表示已签名的第 *i* 个消息，令 L(*i*)表示其长度。如果 $L \neq L^{(i)}$，则$(r \| L \| 1 \| m_1, \sigma_1)$显然是 Π 的伪造，因此假设 $L = L^{(i)}$。将 m(*i*)解析为 $m^{(i)}_1$，...，$m^{(i)}_l$，令 j 为第一索引，其中 $m^{(i)}_j \neq m_j$（必须有一些这样的索引，因为 $m^{(i)} \neq m$ 但它们的长度是相同的）。那么（$r \| L \| j \| m_j$，σ_j）对 Π 的伪造。（所有使用 Π 签名的块的第一部分或第三部分都不相同；以前的有符号块 $r \| L \| j \| m^{(i)}_j$ 与 $r \| L \| j \| m_j$ 不同。）这样就完成了证明。

类似的证明也适用于 KMA 安全情况：

定理 1.5 如果 Π 在已知消息攻击下是不可伪造的（强不可伪造的），则 Π^*在已知消息攻击下是不可伪造的（强不可伪造的）。

1.10 拓展阅读

严格定义安全性的计算概念，通过将安全性归约到一个更基本的假设来证明构造的安全性的想法，是由 Goldwasser 和 Micali[59]在公钥加密中所做的开创性工作。对于“可证明安全性”概念的一个很好的概述，以及关于具体安全性 vs.渐近安全性的讨论，请参见[72]。

数字签名方案的基本思想是由 Diffie 和 Hellman[40]在一篇极具影响力的文章中首次提出的，这篇文章开启了公钥密码学的研究。在他们的工作中，Diffie 和 Hellman 提出了概念和通用实例;尽管他们构建签名方案的方法被证明是不安全的(在签名方案的安全性定义被正式确定之前，这一点并不完全是显而易见的)，但他们的想法成为了未来工作的动力。在介绍 RSA 密码体制(见第 2 章)时，Rivest、Shamir 和 Adleman[99]提出了用它构造一个基本签名方案。在数字签名方面的其他值得关注的早期工作包括[97,80,110,78,44,79]。

Lamport [76]首先考虑了一次签名，而 Goldwasser，Micali 和 Yao [62]研究了已知消息攻击（尽管在两种情况下都没有完全正式的定义）。这些文件还包含符合所提定义的安全构造。Goldwasser，Micali 和 Rivest [61]的开创性论文中介绍了自适应选择消息攻击下的存在不可伪造的定义，该定义已成为数字签名安全性的“默认”概念。他们的论文还对签名方案安全性的各种可能定义进行了精彩的讨论，其中包括一些此处未讨论的内容。 Goldwasser，Micali 和 Rivest 还提供了满足其最严格定义的构造。(有趣的是，[61]的作者当时似乎还没有意识到可以使用 1.7.2 节的结构将 KMA 安全方案从[62]转换为 CMA 安全方案。) [7]（在对称密钥场景中定义了类似的概念）的工作和[2]的工作推广了强不可伪造的概念。关于数字签名方案的安全性概念（包括此处未描述的一些其他概念）的进一步讨论可以在 Goldreich [57]的教科书中找到。

实际上，Goldreich 和 Micali [45]证明了定理 1.2，本质上使用此处所示的相同构造，从那时起，基本思想就隐含在许多数字签名方案的构造中。实际上，Goldreich 和 Micali 也是第一个发现构造 1.2 通过预先计算提高了签名的效率的人，正如在第 1.7.2 节末尾讨论的那样。本质上，用于证明定理 1.1 的相同构造是由 Cramer 和 Pedersen [35，32]提出并证明是安全的，后来在特定签名方案的背景下由[86]进行了重新研究。（Even 等人[45]也给出了从 RMA 安全方案构造 CMA 安全签名方案的方法，但其构造效率不如本章所述构造。）1.8 节中的构造应归功于 Bellare 和 Shoup [12]。[68，106]中出现了类似结果的其他证明。