随机谕言机是实用的：设计有效协议的范式

Random Oracles are Practical:A Paradigm for Designing Efficient Protocols

Mihir Bellare* Phillip Rogaway†

翻译：李晓峰 (cy_lxf@163.com)‡

V1.1

2023 年 12 月 6 日

摘要

我们认为，随机谕言模型 (random oracle model)[1]——所有各方都可以访问公共随机 Oracle——提供了密码学理论和密码学实践之间的桥梁。在我们建议的范式中，首先为随机 Oracle 模型设计并证明正确的协议 P^R，然后通过“适当选择”函数 h，用 h 的计算取代 Oracle 访问，从而产生实用的协议 P。这种范式产生的协议比标准协议更有效，同时保留了可证明安全性的许多优点。我们在包括加密、签名和零知识证明在内的问题上说明了这些优点。

1 引言

密码学理论为密码学实践提供了一个潜在的宝贵概念: 可证明安全性的思想。不幸的是，理论工作似乎往往只能以效率为代价来获得可证明的安全

*Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093. E-mail: mihir@cs.ucsd.edu

†Department of Computer Science, University of California at Davis, Davis, CA 95616, USA. E-mail: rogaway@cs.davis.edu

‡译文来自于经典文献翻译项目 https://gitee.com/uisu/InfSecClaT, 欢迎大家加入经典翻译项目，为更多的人能够获取这些经典文献所传递信息做一点贡献。

[1]译者注：中文也有翻译为“随机预言机模型”，本翻译中会和“随机谕言机模型”混用。

性。这部分是由于以下原因，理论家将某些原语 (如单向函数) 视为“基本”，并以低效的方式构建更强大的原语 (如伪随机函数); 但在实践中，功能强大的原语很容易获得，而所谓的“基本”原语似乎并不容易实现。事实上，理论家们否定了实践中的原语能力，这些原语不仅满足了他们喜欢做出的最强的假设，而且还具有尚未定义或形式化的优势。

为了将可证明安全性的一些好处带到实践中，在我们的模型中合并一些对象是有意义的，这些对象捕获了实际原语真正似乎拥有的属性，并将这些对象视为基本对象，即使从理论的角度来看，关于它们的假设非常强。本文着重介绍了其中一种方法的有效性和潜力。这个想法很简单: 也就是说，为所有各方（不论是敌是友）提供访问（公共）随机预言机的权限；证明该模型中的协议是正确的；然后用一个类似散列函数的对象替换随机预言机。我们强调证明是在随机预言模型中，最后一步本质上是启发式的。这是本文的一个论点，尽管如此，这个方法的好处依然是显而易见的。

这种范式的想法建立在 Goldreich, Goldwasser 和 Micali[20,211] 和 Fiat-Shamir[14] 的工作基础上。它是由以前许多“不合理的”哈希函数使用引起的。最后，它包含了同行共同分享和口头表达的观点，应该被视为民间传说。有鉴于此，我们认为我们的贡献如下。首先，我们将使用随机谕言 (random oracle) 背后的隐含哲学提升为明确表达的范式，我们认为这给实践带来了显著的好处。其次，我们系统地将该范式应用于各种密码问题，以获得有效的解决方案。第三，我们提供了定义和证明，以表明一些先前“不合理”的哈希函数使用可以在随机谕言模型中找到合理的理由。最后，我们提出了我们认为适合实例化随机谕言的哈希函数构造。我们将进一步详细地描述该范式。背景及相关工作请参见 1.3 节。

1.1 随机谕言范式 (The Random Oracle Paradigm)

下面的例子说明了理论家和实践者对原语的看法之间的上述差异。理论家将单向函数视为基本对象，并从中构建伪随机函数。但在实践中，正如 Luhy 和 Rackoff[30，31] 所指出的，DES 提供了 64 位到 64 位的伪随机函数。具有讽刺意味的是，如果一个单向函数需要一个实用的协议，那么很可能将“简单”原语变化为“复杂”原语，例如从 DES 来构建原语。

如果试图设计有效的协议，开始对将使用的原语做出强有力的、现实的假设更有意义。基于上述段落，64 位字符串上的伪随机函数是一个很好的

起点。如下所述，采用更宽泛的假设似乎是合理的。[2]

强大的原语。 让我们看一下第二个可有效计算的原语: 由 MD5 算法 [35] 定义的映射 h_2，它被限制为长度 < 400 的输入。人们对这个函数有这样的期望: 很难找到一个使 $h_2(x) = x$ 的 x;[3] 很难找到一个使 $h_2(x)$ 的汉明权值 (Haming weight) 超过 120 的 x;$f_a(x) = h_2(xa)$ 实际上是一个伪随机函数族; 等等。它到底是什么? 到目前为止，还没有令人满意的答案。尽管如此，还没有一个正式的定义能够捕捉到这个函数似乎拥有的大部分良好性质——而且目前还不清楚是否能找到这样的定义。

范式 (The Paradigm)。 我们对“像 h_2 这样的函数可以完成什么?”的回答是，它可以被认为是一个随机函数，因为它可以在接下来的设计方法中作为 h 的角色使用。假设有一个协议问题 Π(该问题“独立于”原语 h)。为了给 Π 设计一个好的协议 P:

- 在所有各方 (包括对手) 共享随机 Oracle R 的计算模型中，找到 Π 的正式定义。[4]。
- 在这个随机 Oracle 模型中，为 Π 设计一个有效的协议 P。
- 证明 P 满足 Π 的定义。
- 用 h 的计算来代替对 R 的谕言访问。(Replace oracle accesses to R by computation of h.)

我们的论点是，这种方法，适当地使用，可以产生安全有效的协议。事实上，在这种范式下构建的协议到目前为止在实践中被证明是“安全的”。但我们强调，所有可证明安全性的声明都是在随机预言机模型中做出的声明，用 h 实例化预言机只是一种探索性方法，我们从经验中相信它的成功。

注意，h 不能真的像一个随机函数，因为它有一个简短的描述。在许多方面，h 与随机谕言非常不同。但这并没有改变这种方法的成功。

我们强调协议问题 Π 和协议 P 必须“独立”于我们要使用的哈希函数。很容易构建非自然问题或协议, 这个非自然问题或协议的描述和目标显式依赖于 h，使得协议在随机 Oracle 模型中是安全的，但在使用哈希函数实例化随机 Oracle 时失败。“独立性”的概念将不会在本文中形式化。

[2]译者注：原文 more generous，开始翻译为:“更慷慨”，但是不太通顺，后来觉得作者本意应该是“更宽泛”。

[3]译者注：这里应该是一个印刷错误，应为：$h_x(x) = x'$，就是找一个哈希值同为 x' 的碰撞。

[4]译者注：formal definition，通常译为“形式化定义”, 其实就是“正式定义”。

实例化 (Instantiation). 对于本文的主体，我们假设一个从 $\{0,1\}^*$ 到 $\{0,1\}^\infty$ 的随机 Oracle R。我们使用这样一个 Oracle，而无需进一步解释，以方便地描述给定协议的随机映射。

当通过具体函数 h 实例化随机 Oracle 时，首先，必须注意确保 h 在其设计中足够谨慎，以免屈服于密码分析攻击，[5] 其次，要确保 h 不暴露任何低级原语定义的相关“结构”属性。第 6 节给出了这两种陷阱的例子，本节给出解释，这是因为，MD5 和 SHA 等标准哈希函数本身并不能很好地替代随机预言机; 但我们不需要看得更远。候选实例包括输出被截断的哈希函数; 输入长度受限制的哈希函数; 以及以非标准方式使用的哈希函数，例如 $h_3(x) = MD5(xx)$。参见第 6 节。

1.2 结论

本文的研究结果可分为三个。首先是针对各种密码学问题给出新的高效解决方案。第二个是对已知探索性的论证。第三个是随机谕言模型中的一些“理论”结果，我们的研究证明了这些结果。在每种情况下，我们都提供适合于随机谕言场合的协议、定理和新定义。

有效加密 (Efficient Encryption) 在标准场景中可能但不切实际的目标在随机 Oracle 设置中变得可行。我们用一个例子来说明: 公钥加密。$G : \{0,1\}^*$ 到 $\{0,1\}^\infty$ 是一个随机生成器;k 为安全参数;$H : \{0,1\}^*$ 到 $\{0,1\}^\infty$ 是一个随机哈希函数;f 是一个逆为 f^{-1} 的陷门排列;$G(r) \oplus x$ 表示 x 与 $G(r)$ 输出的前 $|x|$ 位的逐位异或; “||” 表示串联操作。对于一个具体的实现 (a concrete implantation)，f 可能是平方 [42,3] 或 RSA [38]。

我们提出了两种在随机 Oracle 模型中有效加密的方案:

- 对于一个在域 f 中的随机数 r，设 $E^G(x) = f(r)||G(r) \oplus x$.
- 对于一个在域 f 中的随机数 r，设 $E^{G,H}(x) = f(r)||G(r) \oplus x||H(rx)$.

这里 x 是要加密的消息，f 是接收者的公钥，f^{-1} 是他的私钥。关于背景、定义、结果的精确陈述以及与已知方案的效率比较，请参见第 3 节，但是，简要地说，讨论的内容如下: 第一种方案实现了 [24] 定义的多项式/语义安全性; 第二种的安全是，可以抵御 [36] 意义上的选择密文攻击 (chosen-ciphertext attack)，以及 [13] 意义上的不可延展性 (non-malleable); 对于相同的目标，两者都比以前的可证明安全方案 [24,4,34,36,11,13] 要有效得多。

[5] 译者注：这里的意思是这个实例化函数不能被密码分析者攻破，至少目前是这样的。

已知探索的正当性 (Justification of known heuristics) 各种众所周知的“技巧”通过移动到随机 Oracle 场景中来找到正式证明 (formal justification)。(这并不意味着现有协议通常可以通过采用随机 Oracle 模型来证明; 相反，这似乎是一个例外，而不是普遍现象。) 我们用下面两个例子来说明。

RSA 等流行的签名方案是以下实例: 对于陷门排列 f 和哈希函数 H，消息 x 的签名是 $f^{-1}(H(x))$。人们普遍认为，没有哈希函数的自然属性使这种方法成为安全的签名方案。然而，对于 H 随机哈希函数，我们证明了该方案对自适应选择消息攻击是安全的。参见第 4 节。

在零知识交互式证明中，消除交互的一种探索性方法 (归于 M. Blum[6]) 是让证明者本质上是问自己这个查询，验证者会把双方之间已经交换的消息的哈希值计算为这个查询。我们证明了这种构造在随机 Oracle 模型中是安全的。为了证明这一点，必须对随机 Oracle 模型中的零知识给出形式化的定义。参见第 5 节。

理论结果 (Theoretical Results) 推广刚才描述的结果，我们表明任何具有交互式证明的语言都可以有效地将其证明转换为非交互式零知识证明。计算模型是，所有各方, 包括作弊的证明者, 只可向随机预言提供多项式次的查询。我们还证明了在随机 Oracle 模型中，常数轮 (constant round)、信息理论安全函数的评估 (information theoretically secure function evaluation) 是可能的[7]，由于篇幅所限，省略了这些结果的定义和证明。

1.3 背景和相关工作

在一个模型中证明协议是正确的，其中各方有一个随机的 Oracle，然后用适当的加密原语实例化该 Oracle 的基本思想起源于 [20,21]。[20] 为此目的提出并构造的密码原语是伪随机函数 (PRF)。然而，为了使 PRF 保持其属性，指定它的种子 (种子使能计算) 必须对对手保持未知。因此，此范式适用性仅限于拒绝对手访问随机 Oracle 的协议。[8] 因此，在许多应用中 (尤其是本文中的应用)，这些 PRF 是不够的。但是，请注意，当设置允许通过

[6] Personal communication, via S. Micali and S. Rudich.

[7] In this application it does not suffice to replace the pseudorandom generator used in [1] by a random generator. 在这个应用中，用一个随机生成器替换 [1] 中使用的伪随机生成器是不足够的。

[8] That is, the adversary is denied direct access to the oracle. A particular problem might permit the adversary indirect access to the oracle via her interaction with the good parties. 也就是说，对手被拒绝直接访问谕言机。一个特定的问题可能允许敌手通过与好的一方的互动间接访问谕言机。

这些 PRF 实例化 Oracle 时，通常可以在标准复杂性理论假设下的标准计算模型中证明生成的协议是正确的，而我们建议的通过哈希函数实例化无法实现。

第一个明确采用公共随机谕言模型 (包括对手在内的所有各方都可以访问 Oracle) 的工作是 Fiat 和 Shamir[14]。作者使用该模型将身份方案转换为数字签名方案 (在此转换过程中“完全”不牺牲严谨性)。

M. Blum 前面提到的使交互式证明非交互式的想法可以被认为是 Fiat-Shamir 思想的延伸。Micali[32] 最近在计算界检查 (computationally bounded checking) 方面的一个令人兴奋的结果，部分利用了相同的技术。

Impagliazzo 和 Rudich[27] 将单向函数建模为随机谕言。他们这样做是为了表明，在给定黑盒单向函数的情况下，证明密钥交换协议的存在性就像将 P 从 NP 中分离出来一样困难。他们也会使用随机谕言来获得积极的结果; 其中，他们形式化并证明了随机谕言模型中私钥密码系统的存在性。

Leighton 和 Micali[28] 将哈希函数视为公共随机谕言，以证明新的高效签名方案的安全性。他们使用随机谕言模型来定义和证明精确的、非渐近的安全性。在另一篇论文 [29] 中，同样的作者使用哈希函数作为随机谕言来给出新的密钥交换方案。

由于本文主题的广度，我们对于历史的总结限定在一个特定目标，并给出这个目标的描述。

1.4 未来方向

在目前的工作中，方式有限，在 [28] 中充分提出了这样一个事实，即随机谕言模型有助于在避免复杂性理论和渐近性的意义上给出精确的定义和结果。在这个意义上使我们的结果精确是可行和可取的。一个典型的定理是表达敌手根据她所做的谕言查询的数量所获得的优势。

我们知道没有复杂性理论假设可以很好地捕获公共随机谕言的所有良好属性。是否有一种方法可以将 [20] 伪随机函数族的概念扩展到一个同样有用和引人注目的概念，其中涉及非隐藏随机性?

2 预备 (preliminaries)

概念 (notation). $\{0,1\}^*$ 表示有限二进制串空间，$\{0,1\}^\infty$ 表示无限二进制串空间。除非我们特别说明，否则串都是有限的。串 a 和 b 拼接表示

为 $a||b$，或 ab。空串表示为 Λ。多项式时间算法是在其第一个参数中以时间多项式运行的算法。“PPT”代表“概率，多项式时间”。一个函数 $\epsilon(k)$ 是可忽略的 (negligible)，如果对于每一个 c，都存在一个 k_c，对于每一个 $k \geq k_c$，都有 $\epsilon(k) \leq k^{-c}$。如果一个函数不是可忽略的，就说这个函数不可忽略的 (non-negligible)。[9] 我们将用“$k^{-\omega(1)}$”来表示这类可忽略函数或该类中特定的匿名函数。对于指定概率实验和空间，我们使用起源于 [26] 的概念。特别回想一下，如果 A 是输入 $x, y, \ldots$ 的概率算法, 然后 $a \leftarrow A(x, y, \ldots)$ 表示运行 $A(x, y, \ldots)$ 选择 a 的实验，$[A(x, y, \ldots)]$ 表示 $A(x, y, \ldots)$ 以正概率可以输出的所有元素的集合。

********** 译者注 ********** 如果 S 是一个概率空间,$x \leftarrow S$ 表示从 S 中随机选择一个元素赋予 x，$Pr[x \leftarrow S; y \leftarrow T; \ldots : p(x, y, \ldots)]$ 表示顺序执行算法 $x \leftarrow S; y \leftarrow T; \ldots$ 后，谓词 $p(x, y, \ldots)$ 为真的概率。**********

谕言 (Oracles). 为了方便起见，随机谕言 R 是从 $\{0,1\}^*$ 到 $\{0,1\}^\infty$ 的映射，对每个 x，均匀且独立地选择 $R(x)$ 的每个比特。当然，实际的协议没有使用无限长的输出，这只是让我们不必说多长是“足够长”。我们用 2^∞ 表示所有随机谕言的集合。

字母“R”将表示“通用”随机谕言，而 $G : \{0,1\}^* \to \{0,1\}^\infty$ 将表示随机生成器，$H : \{0,1\}^* \to \{0,1\}^k$ 表示一个随机哈希函数。当提到多个 Oracle 时，所有这些都是独立选择的。通过各种自然编码，单个随机谕言 R 可以用来提供任意多的独立随机谕言。

通常，提供给算法的谕言用上标表示。有时谕言是明确的，那么就省略谕言符号。[10]

陷门排列 (Trapdoor permutations). 继 [26] 之后，一个陷门排列发生器是一个 PPT 算法 $\mathcal{G}_*$，它在输入 1^k 时输出 (编码) 算法三元组 (f, f^{-1}, d)。前两种算法是确定性的，最后一种是概率的。我们要求 $[d(1^k)]$ 是 $\{0,1\}^k$ 的一个子集，并且 f, f^{-1} 是 $[d(1^k)]$ 上互为逆的排列。我们要求存在一个多项式 p 使得 f, f^{-1} 和 d 在时间 $p(k)$ 中是可计算的，并且对于所

[9] 译者注：算法时间复杂度 $O(n) = n^c$，c 为常数，是多项式算法复杂度上界，可忽略函数用倒数 k^{-c} 表示其上界，也就是说, 但安全参数 k 大于某个值后，这个函数值变的越来越小，也就是可以忽略的概念。如果我们把安全参数 k 同时也看成 PPT 算法输入，那么 PPT 时间表示是 $k^{c'}$，我们总可以取一个常数是 $maxc, c'$, 所以可以将忽略函数定义写为 $\epsilon(k)k^c \leq 1$, 所以随着 k 越大，算法所需时间也越长，但是可以说明 $\epsilon(k)$ 是递增的吗？已经 $\epsilon(k)$ 比 k^c 变化快吗？好像不行，因为不能保证有一些坏点突然时间变少了，函数值增加了，这种坏点是不是就是可能攻击？

[10] 译者注：就是下文表示的 E^R、D^R 之类的，如果上下文明确就省去上标 R，直接写为 E、D。

有非均匀多项式时间敌手 M，

$$\epsilon(k) = Pr[(f, f^{-1}, d) \leftarrow \mathcal{G}_*(1^k); \\ x \leftarrow d(1^k); y \leftarrow f(x) : M(f, d, y) = x]$$

$\epsilon(k)$ 是可忽略的。如前所述，对合适的合数进行平方模 [42,3]，它的变体 [26] 或 RSA[38] 都是陷门排列的好例子。如果对于所有 k 和所有 $(f, f^{-1}, d) \in [G(1^k)]$，$d$ 是 $\{0,1\}^k$ 上的均匀分布, 我们称陷门排列生成器 $\mathcal{G}_*$ 均匀 (uniform)。

3 加密

我们依赖于 [24,33,19,18,34,13] 中阐述的工作。为简单起见，我们考虑非均匀 (多项式时间) 算法的对手，可能是概率的; 扩展到均匀的情况，可以参照 [18]。

加密 (encryption). 我们将公钥加密的概念 [12] 扩展到随机谕言模型。该方案由一个 PPT 生成器 $\mathcal{G}$ 指定，该生成器接受一个安全参数 1^k，并输出一对概率算法 (E, D)，分别称为加密算法和解密算法，它们在 $\mathcal{G}$ 的时间复杂度限定的时间内运行。用户 U 运行 $\mathcal{G}$ 获取 (E, D)，将前者公开，将后者保密。为了加密消息 x，任何人都可以计算 $y \leftarrow E^R(x)$, 并将其发送给 U; 要解密密文，用户 U 计算 $x \leftarrow D^R(y)$。对于所有 x，我们要求 $D^R(E^R(x)) = x$，并且为了简单起见，我们假设如果 y 不是任何字符串 x 在 E^R 下的加密，则 $D^R(y) = 0$，。

3.1 多项式安全 (Polynomial Security)

背景. 公钥加密的“基本”安全目标在 Goldwasser 和 Micali 的多项式和语义安全的等价概念中得到形式化 [24]。如果 B_f 表示 f 的核心谓词 (hard core predicate)(参见 [5,43,23])，则设置 $E(x) = f(r_1)||\ldots||f(r_{|x|})$ 即可实现 [24] 意义上的安全性, 其中每个 r_i 都是从 f 的定义域中随机选择的, 并且 $B_f(r_i) = x_i$。这就产生了一个长度为 $O(k|x|)$ 的加密，这需要对 f 进行 $O(|x|)$ 次求值来加密，对 f^{-1} 进行 $O(|x|)$ 次求值来解密，这是不实际的。Blum 和 Goldwasser[4] 的一种更有效的构造产生了大小为 $O(|x| + k)$ 的加密，需要 $O(|x|)$ 个模平方运算来加密，需要 $O(1)$ 个模幂加上 $O(|x|)$ 个模平方运算来解密，这个代价仍然太昂贵了。从业者通常将消息 x 嵌入到一

个随机值 r_x 中，然后设置 $E(x) = f(r_x)$。(例如，这正是 [39] 所规定的。) 通常使用的嵌入并不能保证 x 像 r_x 一样难以找到 (更不用说 x 的所有属性都是隐藏的)。

定义 (definition). 我们将多项式安全性的概念 [24] 应用于随机谕言模型。(语义安全的类似扩展概念仍然是等效的。) 一个 CP 敌手 (选择明文敌手)A 是一对非均匀多项式时间算法 (F, A_1)，每个算法都可以访问一个 Oracle。为了使加密方案 $\mathcal{G}$ 在随机谕言模型中是安全的，我们要求对于任意 CP 敌手 $A = (F, A_1)$, 有

$$\begin{aligned} Pr[R \leftarrow 2^{\infty}; (E, D) \leftarrow \mathcal{G}(1^k); (m_0, m_1) \leftarrow F^R(E); \\ b \leftarrow \{0,1\}; a \leftarrow E^R(m_b) : \\ A_1^R(E, m_0, m_1, a) = b] \leq \frac{1}{2} + k^{-\omega(1)}. \end{aligned}$$

请注意，用于加密和解密的 Oracle 是提供给试图区分字符串 m_0 和 m_l 加密后的攻击者的，因此，例如，从 R 派生出 H 的散列 $H(x)$ 几乎肯定不会出现在字符串 x 的安全加密中。

通过 $E(x) = f(r)||G(r) \oplus x$ 加密 为了说明我们的加密方案，设 $\mathcal{G}_*$ 是一个陷门置换生成器，设 $G : \{0,1\}^* \to \{0,1\}^{\infty}$ 是随机生成器。在输入 1^k 上，我们的生成器 $\mathcal{G}$ 运行 $\mathcal{G}_*$, 得到 (f, f^{-1}, d)。E 是输入 x 选择 $r \leftarrow d(1^k)$ 并输出 $E^G(x) = f(r)||G(x) \oplus x$ 的算法，其中 $G(r) \oplus x$ 表示 $G(r)$ 的开始的第 $|x|$ 个比特与 x 的 XOR。当然，解密函数是 $D^G(ys) = x \oplus G(f^{-1}(y))$。

定理. 在附录 A 中，我们表明上述方案在随机谕言机模型中是多项式安全的。

比较. 我们实现了加密大小为 $O(|x| + k)$。除了可以忽略不计的哈希开销外，加密需要应用一次 f，解密需要应用一次 f^{-1}。设置 f 为平方意味着用一个模平方来加密，用一个模幂来解密。这比上面讨论的方案 [4] 要高效得多。

3.2 选择密文安全 (Chosen Ciphertext Security)

Naor 和 Yung[34] 给出了选择密文安全性的定义和第一个可证明实现该安全性的方案。Rackoff 和 Simon[36] 提出了更强的概念和相应的解决方案;De Santis 和 Persiano[11] 给出了另一种解决方案。后两种方法利用了知识证明，如前面 [17,6] 所建议的那样。在标准假设下，所有已知的可证明安全的方案都依赖于非交互式零知识证明 [7,16]，并且效率非常低。Damgård[10]

提出了一种实现 [34] 中定义的有效方案，但该方案并没有被证明能够实现 [34] 的定义，也没有达到我们感兴趣的 [36] 中的定义。Zheng 和 Seberry[44] 的方案与我们的方案密切相关，我们将在后面讨论。

定义. 我们将 [36] 的定义适应于随机 Oracle 场景。RS-敌手 (Rackoff-Simon adversary) A 是一对非均匀多项式时间算法 $A = (F, A_1)$, 每个访问 Oracle 的 R 和 D^R 的黑盒实现。F 的工作是拿出一对 (长度相等) 消息 m_0 和 m_1, 将其中之一加密为 a 给 A_1, 只要不允许 A_1 拿 a 问一个解密 Oracle,A_1 无法猜测 a 是哪一个消息的密文。从形式上讲，A_1 被禁止询问等于其最终参数的 Oracle 查询。如果对于每个 RS-对手 $A = (F, A1)$，加密方案 $\mathcal{G}$ 对 RS 攻击是安全的，则

$$\begin{aligned} Pr[&R \leftarrow 2^{\infty}; (E, D) \leftarrow \mathcal{G}(1^k); (m_0, m_1) \leftarrow F^{R, D^R}(E); \\ &b \leftarrow \{0,1\}; a \leftarrow E^R(m_b) : \\ &A_1^{R, D^R}(E, m_0, m_1, a) = b] \leq \frac{1}{2} + k^{-\omega(1)}. \end{aligned}$$

用 $E(x) = f(x)||G(r) \oplus x||H(rx)$ 加密. 很容易看出，前一节的方案对 RS-攻击是不安全的。我们现在给出一个有效的方案。设 $\mathcal{G}_*$ 是一个陷门置换发生器。设 $G : \{0,1\}^* \to \{0,1\}^{\infty}$ 是一个随机生成器，设 $H : \{0,1\}^* \to \{0,1\}^k$ 是一个随机哈希函数，独立于随机 Oracle 导出。我们方案的生成器 $\mathcal{G}$ 运行 $\mathcal{G}_*$ 得到 (f, f^{-1}, d)。E 是在输入 x 上选择 $r \leftarrow d(1^k)$ 并输出 $E^{G,H}(x) \leftarrow f(r)||x \oplus G(r)||H(rx)$ 的算法。要解密字符串 y，当 $|a| = |b| = k$ 时，y 解析为 $a||\omega||b$，如果 $H(f^{-1}(a)||\omega \oplus G(f^{-1}(a))) = b$, 则将 $D^{G,H}(y)$ 定义为 $\omega \oplus G(f^{-1}(a))$，否则定义为 0。

定理. 在附录 A 中，我们证明了上述方案对选择密文攻击是安全的。

比较. 将 Zheng 和 Seberry[44] 的方案用我们的符号翻译成随机 Oracle 模型，是 $E^*(x) = f(r)||I(G(r) \oplus (xH(x)))$。该方案与我们的方案一样高效，并且具有相同的安全特性。因此，随机 Oracle 模型用于证明 [44] 的构造是合理的。

3.3 非延展性 (non-malleability)

背景. 非延展性的概念是由 Dolev, Dwork 和 Naor[13] 提出的。非正式地说，如果您不能通过见证字符串 x 的加密来产生相关字符串 x' 的加密，那么加密方案是不可延展性的。例如，给定 x 的加密，您不应该能够生成 $\bar{x}$ 的加密。该概念扩展了多项式安全性，特别是前者隐含了后者。非延展性方

案的构造在文献 [13] 中给出。然而，这种构造是完全不切实际的，涉及到巨大的公钥、多个签名的计算和许多非交互式零知识证明。

定义. 我们采用随机 Oracle 设置 [13] 的定义。一个有趣的关系 $\rho^R_{E,\phi}(x,x) : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ 必须满足 $\rho^R_{E,\phi}(x,x) = \rho^R_{E,\phi}(x,0^i) = 0$ 对于每个 $x \in \{0,1\}^*; i \in N; R \in 2^\infty; E, \phi \in \{0,1\}^*$; 更进一步，$\rho$ 必须可由多项式时间图灵机 $M^R(x,y,E,\phi)$ 计算。一个 M-敌手 (“可延展敌手”“malleability adversary”) $\mathcal{A}$ 是一对非均匀概率多项式时间算法 (F,A)，每个算法都访问一个 Oracle R。当 F 运行时，它输出一个算法 ϕ 的描述，该算法也使用一个 Oracle，并且运行的时间复杂度不大于 F。对于一个不可延展的加密方案，我们要求对于每个有趣的关系 ρ 和每个 M-敌手 (F,A) 存在一个 (非均匀) 多项式时间 A_*，使得 $|\epsilon(k) - \epsilon_*(k)|$ 可以忽略不计，其中

$$\begin{aligned}\epsilon(k) = Pr[&R \leftarrow 2^\infty; (E,D) \leftarrow \mathcal{G}(1^k); \phi \leftarrow F^R(E);\\ &x \leftarrow \phi^R(1^k); a \leftarrow E^R(x); a' \leftarrow A^R(E,\phi,a) :\\ &\rho^R_{E,\phi}(x, D^R(a')) = 1]\end{aligned}$$

$$\begin{aligned}\epsilon_*(k) = Pr[&R \leftarrow 2^\infty; (E,D) \leftarrow \mathcal{G}(1^k); \phi \leftarrow F^R(E);\\ &x \leftarrow \phi^R(1^k); a'_* \leftarrow A^R_*(E,\phi,a) :\\ &\rho^R_{E,\phi}(x, D^R(a'_*)) = 1]\end{aligned}$$

参见 [13] 对这个定义的直观解释，包括对关系 ρ 的限制。

通过 $E(x) = f(r)||G(r) \oplus x||H(rx)$ 加密. 加密方案与上一节相同。

定理. 在附录 A 中，我们证明了上述方案是不可延展性的。

4 签名

定义. 我们将 [26] 的定义扩展到随机 Oracle 设置。数字签名方案是一个三元组 $(\mathcal{G}, Sign, Verify)$ 的多项式时间算法，分别称为生成器、签名算法和验证算法。前两个是概率的，后两个可以访问随机 Oracle。在输入 1^k 时，生成一对匹配的公钥和密钥 (PK, SK)。要签名消息 m，计算 $\sigma \leftarrow Sign^R(SK,m)$; 验证 (m,σ), 计算 $Verify^R(PK,m,\sigma) \in \{0,1\}$。对于所有 $\sigma \in [Sign^R(SK,m)]$，必须满足 $Verify^R(PK,m,\sigma) = 1$。S-敌手 (“签名敌手”,“signing adversary”) 是一个 (非均匀的) 多项式时间算法 F，可以

访问 R 和签名 Oracle。F 的输出是一对 (m, σ)，且 m 没有在签名 Oracle 中查询。我们称签名方案是安全的，如果对于每个 S-敌手 F 定义的函数 $\varepsilon(k)$ 为，

$$\begin{aligned} Pr[R \leftarrow 2^{\infty}; (PK, SK) \leftarrow \mathcal{G}(1^k); \\ (m, \sigma) \leftarrow F^{R, Sign^R(SK, \cdot)}(PK) : \\ Verify^R(PK, m, \sigma) = 1] \end{aligned}$$

而 $\varepsilon(k)$ 是可忽略的。如果 F 的输出 (m, σ) 满足 $Verify^R(PK, m, \sigma) = 1$，我们就说 F 成功了。

协议. 修补一个陷门排列发生器 $\mathcal{G}_*$. 为简单起见，假定它是均匀的; 请参阅下面如何为标准程序打补丁。设 $H : \{0,1\}^* \to \{0,1\}^k$ 表示随机哈希函数。签名方案为 $(\mathcal{G}, Sign^H, Verify^H)$，其中输入 1^k 的 $\mathcal{G}$ 输出 $PK = f$, $SK = f^{-1}$;$Sign^H(f^{-1}, m)$ 等于 $f^{-1}(H(m))$; 且当且仅当 $f(\sigma) = H(m)$ 时，$Verify^H(f, m, \sigma)$ 为 1。换句话说，就是借助哈希函数进行签名的“经典”方法。

统一: 技术细节 标准的陷门排列 (基于平方或 RSA) 不是统一的，必须修补方案来处理它们。修补的方法有很多。RSA 和 [26] 中定义的平方具有稠密域，[11] 可以有效地测试成员和集合的隶属关系。为了给 m 签名我们可以修改方案来计算 $H(1||m), H(2||m), \ldots$，直到找到域的成员 $y = H(i||m)$，然后返回 $(i, f^{-1}(y))$。验证的定义是显而易见的。这些函数的另一种替代方法是通过构造 [2,Section 4.2] 来使它们统一。在 [42,3] 中定义的平方函数没有有效的可测试域，但仍然可以制作各种补丁。事实上，函数甚至不必是置换 (参见 [35])。

安全性. 假设 F 是一个 S-敌手，以不可忽略的概率 $\lambda(k)$ 成功。我们构造算法 $M(f, d, y)$，它课如下所示不可忽略地计算 $f^{-1}(y)$。M 让 $PK = f$。它为 F 抛硬币并开始运行 F，我们假设 F 对 H 进行了 $n(k)$ 次查询，都是不同的，如果 F 进行了签名查询 m，那么它已经查询了 $H(m)$; 这显然不失一般性[12]。现在 M 随机选择 $t \in \{1, \ldots, n(k)\}$。然后，它会如下回复查询:

[11] 译者注：稠密的意思很简单，就是在任意两个元素之间存在第三个元素。比如：任意两个有理数之间有无数个有理数和无理数，任意两个无理数之间也有无数个有理数和无理数。所以有理数和无理数都是稠密的。实数包括有理数和无理数，所以实数也是稠密的。连续是函数的属性，设函数 $f(x)$ 在某个区间 $[a, b]$ 上有定义。如果对于这个区间内的任意一个数 c，当 x 趋近于 c 时 $f(x)$ 也趋近于 $f(c)$，那么我们称函数 $f(x)$ 在区间 $[a, b]$ 上是连续的。

[12] 译者注：原文为“this is easily seen to be wlog.”, 此处 wlong 应该为“Without Loss Of Generality”的缩写，也就是“不失一般性”.

- 设 m_i 表示 F 执行的第 i 个 H 查询。如果 $i = t$，则 M 返回 y 做为应答，否则它选择 $r_i \leftarrow \{0,1\}^k$ 并返回 $y_i = f(r_i)$。
- 假设 F 进行 m 签名查询，如果 $m = m_t$，则 M 停止，承认失败。否则 M 回答 r_i，其中 $i \neq t$ 满足 $m = m_i$。

设 (m, σ) 是 F 的输出。如果 $m \neq m_t$，则 M 停止，承认失败。否则输出 σ 并停止。$M(f, d, y)$ 成功计算 $f^{-1}(y)$ 的概率至少为

$$(1 - \frac{1}{n(k)}) \cdot \frac{\lambda(k)}{n(k)} - 2^{-k}$$

并且这个概率依然不可忽略。

5 零知识

我们给出了零知识 (ZK) 证明的定义，然后展示了在这个模型中 ZK 的交互证明如何转换为模型中的非交互。该变换是有效的，因此我们得到了非交互 ZK 复杂度证明等于交互 ZK 复杂度证明的结论。

5.1 定义

Oracle 模型中零知识的定义不仅仅是简单地将标准定义“相对化”。下面扩展了通常交互设置下的公式 [25]，以及通用随机串模型 (common random string model) 下的公式 [6,7]。

设置 (Setting). 为简单起见，我们讨论一种语言 $L \in NP$ 的证明。给定一个定义 L 的 NP 关系 ρ;x 是 L 中的成员表示一个字符串 w 满足 $\rho(x, w) = 1$。见证选择器 (witness selector) 是一个函数 W，它对任意输入 $x \in L$ 返回一个见证，证明 x 在 L 中的隶属关系。

验证者是多项式时间函数 V，给定公共输入 x，会话 $\kappa \in \{0,1\}^*$，一个 (私有) 随机磁带 $r \in \{0,1\}^\infty$，返回 $V(x, \kappa, r)$，它要么是给证明者的下一条消息，要么是表明他决定接受或拒绝的比特值[13]。证明者是一个 PTT 函数[14]，给定公共输入 x，对话 κ，辅助输入 a，证明者将下一个消息 $P_a(x, \kappa)$

[13]译者注：就是这个比特值表示他的判定结果，比如 0 表示拒绝，1 表示接受。

[14]原则上，Oracle 模型中的结果要求我们只限制 Oracle 调用的次数，而不限制验证程序的运行时间。但是在使用哈希函数实例化时，运行时间应该受到限制，因此我们立即做出假设。因此，我们处于在 [9] 中“论证”的模型。

返回给验证者。(当 $x \in L$ 辅助输入是这个事实的见证，否则它是空字符串)。在随机 Oracle 模型中，证明者和验证者也使用该 Oracle。

对于任意 Oracle R，用 $conv(V^R, p_a^R, x, r)$ 表示, 当公共输入为 x, V 的随机磁带为 $r \in \{0,1\}^\infty$ 时，所有对话 (文本) 的空间在 p_a^R 与 V^R 之间。用 $ACC_V(\kappa, r) \in \{0,1\}$ 表示验证者是否接受的决定。设

$$ACC_V(P_a, V, x) = Pr[R \leftarrow 2^\infty; r \leftarrow \{0,1\}^\infty; \\ \kappa \leftarrow conv(V^R, P_a^R, x, r) : ACC_V(\kappa, r) = 1]$$

表示 V 与 P_a 在公共输入为 x 时，交互中同意的概率。在证明和协议中，我们经常会滥用符号，只使用与无限字符串 r 相关的前缀。

证明系统. 我们说 (P, V) 是 L 的一个交互证明，在随机预言机模型中，误差为 $\epsilon(x)$，如果 $\epsilon(x) \leq 1/2$，并且以下两个条件成立。完整性条件 (completeness condition) 要求，如果 $x \in L$，那么对于所有证据 w 到 x 是 L 中的成员，这是 $ACC(V, P_w, x) = 1$ 的情况。健全性条件 (soundness condition) 要求对于所有 PPT $\hat{P}$ 和足够长的 x,这是 $ACC(\hat{P}_\Lambda, V, x) \leq \epsilon(|x|)$ 的情况。

视图 (views). 为了定义零知识，首先更新验证器的视图以包括随机谕言机；我们定义

$$rview(V, P_a, x) = \{R \leftarrow 2^\infty; r \leftarrow \{0,1\}^\infty; \\ \kappa \leftarrow conv(V^R, P_a^R, x, r) : (\kappa, r; R)\}.$$

模拟器 (Simulators). 由于随机谕言机是视图的一部分，它也必须是模拟器输出的一部分；即允许模拟器构建 Oracle 的“模拟”。这类似于非交互式零知识 [6，7]，其中允许模拟器构造并输出公共随机串的“模拟”。然而，随机谕言机是一个无限的对象，因此我们不能要求模拟器输出它。相反，我们允许模拟器指定预言机的一小部分（多项式大小），并“神奇地”随机填充其余部分。形式上，模拟器是一个 PPT 算法，它在任意输入 x 上输出一个三元组 (κ, r', T)，其中 $T = (x_1, y_1), \ldots, (x_t, y_t)$ 是字符串对序列, 且 $x_1, \ldots, x_t$ 不同的。随机 Oracle 补全操作 (random oracle completion)ROC 将输入 T 作为输入，并返回一个随机的 Oracle R，该 Oracle R 受约束，即对于所有 $i = 1, \ldots, t$, [15] $R(x_i)$ 以 y_i 为前缀, 类似地定义随机字符串补全操

[15]可以理解，该操作指的是使用中的预言机，而不是底层的“通用”预言机，因此，如果我们使用的是随机哈希函数 H，则返回的是满足该约束的 H，等等。

作 (random string completion) RSC 也很方便，它接受字符串 $r' \in \{0,1\}^*$ 并附加一个无限的随机位序列。我们将 $s(x)$ 的补全定义为概率空间

$$S^c(x) = \{(\kappa, r', T) \leftarrow S(x); R \leftarrow ROC(T); \\ r \leftarrow RSC(r') : (\kappa, r; R)\}.$$

区分器 (distinguisher). 区分符器是一个多项式大小的 Oracle 电路族 $D = \{D_x\}_{x \in L}$。当给定 Oracle R 和输入 κ, r 时，写出电路 D_x 的输出 $D_x^R(\kappa, r)$, [16] 然后定义

$$diff_D(S^c(x), rview(V, P_a, x)) = \\ |P_r[(\kappa, r; R) \leftarrow S^c(x) : D_x^R(\kappa, r) = 1] - \\ Pr[(\kappa, r : R) \leftarrow rview(V, P_a, x) : D_x^r(\kappa, r) = 1]|$$

零知识. 我们说模拟器 S 是验证者 $\hat{V}$ 在语言 L 的 P-模拟器, 如果对于每一个区分器 D，每一个证据选择器 (witness selector)W，每一个常数 d 和所有足够长的 $x \in L$, 有这样的关系

$$diff_D(S^c(x), rview(V, P_W(x), x)) < |x|^{-d}.$$

我们说 P 在随机 Oracle 模型中定义了 L 上的 (计算的)ZK 协议，如果对于 L 上的每个验证者 $\hat{V}$ 存在一个 L 上的 $\hat{V}$ 的 P 模拟器，统计 ZK 可以类似地定义。(P, V) 是 L 的 ZK 证明，在随机 Oracle 模型中错误为 ϵ，如果它是 L 的一个错误为 ϵ 的证明系统，并且 P 定义了 L 上的 ZK 协议。

多定理证明 (multi-theorem proofs). 在应用中，重要的是，我们能够多项式地证明零知识中许多自适应选择的定理，就像普通随机串模型中的零知识一样。为了简单起见，我们坚持以上一个定理的情况；在最后的论文中，我们将给出一般的定义。

知识证明 (proofs of knowledge). 在最后的论文中，我们还将定义随机预言机模型中的知识证明，并展示如何构建高效、无交互的零知识证明。

6 协议

问题. 设 (P', V') 是 $L \in NP$ 的 ZK 证明, 标准（例如随机谕言机）模型中，错误概率达到 1/2。设 $k(n) = w(logn)$, 我们希望在随机预言机模型

[16] 这里 r 将是一个无限字符串，将 κ, r 作为 D_x 的输入意味着后者将只看到一个有限前缀。

中有一个非交互式的 ZK 证明 (P,V)，它可以实现误差 $\epsilon(n)=2^{-k(n)}$，同时将计算时间和通信比特增加至多 $O(k(n))$ 的因子。

简化假设 (simplifying assumptions). 像大多数这样的 ZK 证明一样，假设 (P',V') 是三个移动：$P'_w \to V' : \alpha$, 后跟 $V' \to P'_\omega : b$, 后跟 $P'_w \to V' : \beta$。这里 b 是一个随机比特（在 V' 的随机带上的第一个，我们现在认为它就是这个比特），ω 是 P' 的辅助输入。消息 α 由一组信封组成，大小为 $n^{\Theta(1)}$。根据挑战 b 打开这些信封的某个子集，并且对于任何字符串 α，恰好存在一个值 $b\in\{0,1\}$，对于该值存在 β，使得 $ACC_{V'}(\alpha b\beta,b)=1$。零知识由算法 S' 捕获，该算法给定 x、b 输出 $\alpha b\beta$，使得 $ACC_{V'}(\alpha b\beta,b)=1$，并且对于任何证据选择器 W，以下集合在计算上是不可区分的：$b\leftarrow\{0,1\};\alpha b\beta\leftarrow S'(x,b) : (\alpha b\beta,b)_{x\in L}$ 和 $\{\alpha\leftarrow P'_{W(x)}(x,\Lambda);b\leftarrow\{0,1\};\beta\leftarrow P'_{W(x)}(x,\alpha b) : (\alpha b\beta)\}_{x\in L}$。

变换 (The transformation). 设 $H:\{0\,1\}^*\to\{0,1\}^{2k}$ 是一个随机散列函数。新的证明器 P^H_ω 计算 $\alpha_1\leftarrow P'_\omega(x,\Lambda);\ldots;\alpha_{2k}\leftarrow P'_\omega(x,\Lambda)$；将 b'_i 设置为 $H(\alpha_1\ldots\alpha_{2k})$ 的第 i 位；计算 $\beta_1\leftarrow P'_\omega(x,\alpha_1 b'_1);\ldots;\beta_{2k}\leftarrow P'_\omega(x,\alpha_{2k}b'_{2k})$；向 V^H 发送 $(\alpha_1,\ldots,\alpha_{2k},\beta_1,\ldots,\beta_{2k})$。$V^H$ 将 b_i 设置为 $H(\alpha_1,\ldots,\alpha_{2k})$ 的第 i 位，并接受, 对所有 i, 当且仅当 $ACC_{V'}(\alpha_i b_i\beta_i,b_i)=1$。事实上，新协议是非交互式的，并如所声称的那样有效。

(P,V) 是一个误差为 $2^{-k(n)}$ 的 ZK 证明系统. 完整性是显而易见的。我们可以证明，如果 $\hat{P}^H$ 做了 $T(n)$ 次 Oracle 查询，则 $ACC(\hat{P}_\Lambda,V,x)\le T(n)\cdot 2^{-2k(n)}$，对于足够长的 n，它最多为 $2^{-k(n)}$。对于 ZK，交互的策略意味着我们只需要模拟诚实验证者 V 的视图，相应的模拟器 S 如下所示。给定 $x\in L$ 算法，S 选择 $b_1\leftarrow\{0,1\};\ldots;b_{2k}\leftarrow\{0,1\}$。现在对于每个 $i=1,\ldots,2k$, 它让 $\alpha_i b_i\beta_i\leftarrow S'(x,b_i)$. 它使 $T=(\alpha_1\ldots\alpha_{2k},b_1\ldots b_{2k})$，输出 (c,Λ,T)。对 T 进行随机 Oracle 补全操作，得到一个映射 $H:\{0,1\}^*\to\{0,1\}^{2k}$，该映射随机服从于 $H(\alpha_1\ldots\alpha_{2k})=b_1\ldots b_k$ 的约束。现在基于我们对 S' 的假设，我们可以通过定义来检验 S 是 L 上的 V 的 P 模拟器。由于篇幅有限，我们省略了细节。

7 实例化

在 1.1 节讨论的基础上，这里我们提供了使用哈希函数等原语实例化随机 Oracle 的进一步指导。

首先也是最重要的一点是，没有必要 (或不希望) 关注正在实例化其随机预言机的目标协议的细节。重要的是使用了几个 Oracle，以及它们对长度的要求是多少。我们的论点是，随机 Oracle 的适当实例化应该适用于任何协议，这些协议不会故意通过预测实例化 Oracle 的确切机制来挫败我们的方法。

在选择一个具体的函数 h 来实例化 Oracle 时，必须非常小心。让我们从一些行不通的例子开始。

首先看看 MD5。这个函数不适合替代随机 Oracle，因为 [41] 已经观察到，对于任何 x 都有一个 y，使得对于任何 z, 在给定 $|x|, MD5(x)$ 和 z 时，$MD5(xyz)$ 可以很容易地计算出来。特别是，[41] 指出，这意味着 $MD5(ax)$ 不能用作密钥 a 下字符串 x 的消息验证码。

试图通过避免像 MD5 这样的“结构化”操作来克服困难，人们可能更喜欢“低级”原语，例如它的压缩函数 $\mu : \{0,1\}^{640} \to \{0,1\}^{128}$。这也不能成为随机预言的合适替代品，因为 [8] 已经证明了碰撞可以在这个映射中有效地找到。

标准哈希函数过于结构化，无法生成良好的随机预言机 (如上所示)，无需进一步研究; 自然候选结构包括以下结构，或它们的组合:

- 以某种方式截断或折叠输出的哈希函数; 例如:$h_1(x) = MD5(x)$的前 64 位。
- 输入长度受到适当限制的哈希函数; 例如，$h_2(x) = MD5(x)$，其中，$|x| < 400$。
- 以非标准方式使用的哈希函数; 例如，$h_3(x) = MD5(xx)$。
- 加密哈希函数的“第一个块压缩函数”，例如，$h_4 : \{0,1\}^{512} \to \{0,1\}^{128}$ 是计算 $MD5(x)$ 时 512 位 x 的压缩。

作为一个例子，假设一个人决定 (纯粹的启发式) 选择一个映射 $h' : \{0,1\}^{256} \to \{0,1\}^{64}$，由 $h'(x) = h_4((xx) \oplus C)$的前 64 位 定义，对于一个随机选择的 512 位常数 C。[17] 要根据给定应用程序的需要扩展域和范围，可以首先定义 $h''(x) = h'(x < 0 >)||h'(x < 1 >)||h'(x < 2 >)||\ldots$，其中 $|x| = 224$，$< i >$ 是将 i 编码为 64 位。接下来，通过 x' 编码每个输入 x 来扩展 h''，x' 由 x、比特“1”和足够多的 0 组成，使 $|x'|$ 成为 128 位的倍

[17]在实例化时选择 C 可以确保算法目标“独立”于 Oracle 的选择; 它“分离”了不同应用使用的随机预言机的实例化; 它提供了一种创建多个“独立”随机预言机的简单方法。

数。现在，设 $x'x'_1 \ldots x'_n$，此处 $|x'_i| = 128$，并且定义 $h(x) = h''(x'_0 < 0 >) \oplus h''(x'_1 < 1 >) \oplus \ldots \oplus h''(x'_n < n >)$ 生成一个映射，为了所有实际目的，它需要 $h : \{0,1\}^* \to \{0,1\}^\infty$。当然，还有许多其他同样简单的方法来实例化随机 Oracle; 这只是一个例子。

8 总结

实践中使用的协议几乎总是通过一个反复的过程来设计的: 假设一个具体的协议，寻找一个成功的攻击，找到一个，并试图关闭它。这种方法并不奏效。通过坚持定义我们的目标并可证明地实现它们，现代密码学提供了比任何特定结果集更多的实践; 它是一种超越迭代设计过程的方法，用于解决不明确的任务。尽管在使用我们的范例时，“仅”得到的结果是“该协议在实例化随机 Oracle 的程度上是安全的”，但仍然取得了比声明协议可靠更多的成果，因为还没有人提出成功的攻击。

致谢

早期与 Bob Blakley 就许可服务器问题进行的讨论 [40] 有助于明确我们的想法。我们从 Oded Goldreich、Birgit Pfitzmann, 和 Steven Rudich 那里得到了有用的建议和参考。最后，感谢 ACM 项目委员会成员的所有评论。

参考文献

[1] D. Beaver, S. Micali and P. Rogaway, “The round complexity of secure protocols,” Proceedings of the 22nd Annual Symposium on Theory of Computing, ACM, 1990.

[2] M. Bellare and S. Micali, “How to sign given any trapdoor permutation,” JACM Vol. 39, No. 1, 214-233, January 1992.

[3] L. Blum, M. Blum and M. Shub, “A simple unpredictable pseudo-random number gener- ator,” SIAM Journal on Computing Vol. 15, No. 2, 364-383, May 1986.

[4] M. Blum and S. Goldwasser, "An efficient probabilistic public-key encryption scheme which hides all partial information," Advances in Cryptology – Crypto 84 Proceedings, Lecture Notes in Computer Science Vol. 196, R. Blakely ed., Springer-Verlag, 1984.

[5] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo- random bits," SIAM Journal on Computing, Vol. 13, No. 4, 850-864, November 1984.

[6] M. Blum, P. Feldman and S. Micali, "Non-interactive zero knowledge and its applica- tions," Proceedings of the 20th Annual Symposium on Theory of Computing, ACM, 1988.

[7] M. Blum, A. De Santis, S. Micali and G. Persiano, "Non-interactive zero-knowledge proof systems," SIAM Journal on Computing, 20(4), 1084-1118 (December 1991).

[8] B. den Boer and A. Bosselaers, "Collisions for the compression function of MD5," Ad- vances in Cryptology – Eurocrypt 93 Proceedings, Lecture Notes in Computer Science Vol. 765, T. Helleseth ed., Springer-Verlag, 1993.

[9] G. Brassard, D. Chaum and C. Crépeau, "Minimum disclosure proofs of knowledge," JCSS Vol. 37, No. 2, 156–189, October 1988.

[10] I. Damg ̊ ard, "Towards practical public key cryptosystems secure against chosen ciphertext attacks," Advances in Cryptology – Crypto 91 Proceedings, Lecture Notes in Computer Sci- ence Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.

[11] A. De Santis and G. Persiano, "Zero-knowledge proofs of knowledge without interaction" Proceedings of the 33rd Symposium on Foundations of Computer Science, IEEE, 1992.

[12] W. Diffie and M. E. Hellman, "New directions in cryptography," IEEE Trans. Info. Theory IT-22, 644-654 (November 1976).

[13] D. Dolev, C. Dwork and M. Naor, "Non-malleable cryptography," Proceedings of the 23rd Annual Symposium on Theory of Computing, ACM, 1991.

[14] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," Advances in Cryptology –Crypto

86 Proceedings, Lecture Notes in Computer Science Vol. 263, A. Odlyzko ed., Springer-Verlag, 1986.

[15] U. Feige, A. Fiat and A. Shamir, "Zero knowledge proofs of identity," Journal of Cryp- tology, Vol. 1, pp. 77–94 (1987).

[16] U. Feige, D. Lapidot, and A. Shamir, "Multiple non-interactive zero-knowledge proofs based on a single random string," Proceedings of the 31st Symposium on Foundations of Computer Science, IEEE, 1990.

[17] Z. Galil, S. Haber and M. Yung, "Symmetric public key cryptosystems," manuscript, July 1989.

[18] O. Goldreich, "A uniform complexity treatment of encryption and zero-knowledge," Journal of Cryptology, Vol. 6, pp. 21-53 (1993).

[19] O. Goldreich, "Foundations of cryptography," Class notes, Spring 1989, Technion Univer- sity.

[20] O. Goldreich, S. Goldwasser and S. Micali, "How to construct random functions," Journal of the ACM, Vol. 33, No. 4, 792–807, (1986).

[21] O. Goldreich, S. Goldwasser and S. Micali, "On the cryptographic applications of ran- dom functions," Advances in Cryptology – Crypto 84 Proceedings, Lecture Notes in Computer Science Vol. 196, R. Blakely ed., Springer-Verlag, 1984.

[22] O. Goldreich and H. Krawczyk, "On the composition of zero knowledge proof systems," ICALP 90 Proceedings, Lecture Notes in Computer Science Vol. 443 , M. Paterson ed., Springer-Verlag, 1990.

[23] O. Goldreich and L. Levin, "A hard predicate for all one-way functions,"Proceedings of the 21st Annual Symposium on Theory of Computing, ACM, 1989.

[24] S. Goldwasser and S. Micali, "Probabilistic encryption," J. of Computer and System Sciences 28, 270–299, April 1984.

[25] S. Goldwasser, S. Micali and C. Rackoff, "The knowledge complexity of interactive proof systems," SIAM J. of Comp., Vol. 18, No. 1, pp. 186–208, February 1989.

[26] S. Goldwasser, S. Micali and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," SIAM Journal of Computing, 17(2):281–308, April 1988.

[27] R. Impagliazzo and S. Rudich, "Limits on the provable consequences of one-way per- mutations," Proceedings of the 21st Annual Symposium on Theory of Computing, ACM, 1989.

[28] T. Leighton and S. Micali, "Provably fast and secure digital signature algorithms based on secure hash functions," Manuscript, March 1993.

[29] T. Leighton and S. Micali, "New approaches to secret key exchange," Advances in Cryp- tology –Crypto 93 Proceedings, Lecture Notes in Computer Science Vol. 773, D. Stinson ed., Springer-Verlag, 1993.

[30] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudo- random functions,"SIAM J. Computation, Vol. 17, No. 2, April 1988.

[31] M. Luby and C. Rackoff, "A study of password security,"manuscript.

[32] S. Micali, "CS proofs," Manuscript.

[33] S. Micali, C. Rackoff and B. Sloan, "The notion of security for probabilistic cryptosys- tems," SIAM J. of Computing, April 1988.

[34] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen cipher- text attacks," Proceedings of the 22nd Annual Symposium on Theory of Computing, ACM, 1990.

[35] M. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," MIT Laboratory for Computer Science TR-212, January 1979.

[36] C. Rackoff and D. Simon, "Non-interactive zero-knowledge proof of knowledge and cho- sen ciphertext attack," Advances in Cryptology – Crypto 91 Proceedings, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.

[37] R. Rivest, "The MD5 message-digest algorithm," IETF Network Working Group, RFC 1321, April 1992.

[38] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," CACM 21 (1978).

[39] RSA Data Security, Inc., "PKCS #1: RSA Encryption Standard," June 1991.

[40] P. Rogaway and B. Blakley, "An asymmetric authentication protocol," IBM Technical Disclosure Bulletin (1993).

[41] G. Tsudik, “Message authentication with one-way hash functions,” IEEE INFOCOM ’ 92.

[42] H. Williams, “A modification of the RSA public key encryption procedure,” IEEE Transac- tions on Information Theory, Vol. IT-26, No. 6, November 1980.

[43] A. Yao , “Theory and applications of trapdoor functions,” Proceedings of the 23rd Sympo- sium on Foundations of Computer Science, IEEE, 1982.

[44] Y. Zheng and J. Seberry, “Practical approaches to attaining security against adaptively chosen ciphertext attacks,” Advances in Cryptology –Crypto 92 Proceedings, Lecture Notes in Computer Science Vol. 740, E. Brickell ed., Springer-Verlag, 1992.

A. 加密证明

我们给出了一些加密方案的安全性证明。我们将假设 (wlog) 对于任何算法和该算法的任何 Oracle，对 Oracle 进行的所有查询都是不同的。

$E(x) = f(r)||G(r) \oplus x$ **方案是多项式安全的。**证明是用反证法。设 $A = (F, A_1)$ 是击败协议的敌手; 无限次地，对于某个逆多项式 λ，它获得了 $\lambda(k)$ 的优势。我们构造了一个算法 $M(f, d, y)$，当 $(f, f^{-1}, d) \leftarrow \mathcal{G}(1^k); r \leftarrow d(1^k); y \leftarrow f(r)$，通常用于计算 $f^{-1}(y)$。算法 M 根据我们方案中指定的 f 定义 E。它以自然的方式模拟 Oracle G(通过自己投掷硬币来回答查询) 和样本 $(m_0, m_1) \leftarrow F^G(E)$。如果 G 问的 r 满足 $f(r) = y$，则 M 输出 r 并停止。否则，$F(E)$ 终止，M 为 $s \leftarrow \{0,1\}^{|m_0|}$ 选择 $\alpha \leftarrow y||s$。然后 M 模拟 $A_1^G(E, m_0, m_1, \alpha)$，观察 A_1 所做的 Oracle 查询，看是否有 $f(r) = y$ 的 Oracle 查询 r，如果有，M 输出 r，设 A_k 为 A_1 查询 $r = f^{-1}(y)$ 的事件。A_1 在区分 m_0 和 m_1 方面没有优势, 因为 A_1 没有询问 G 在 r 处的像，所以

$$\begin{aligned}\frac{1}{2} + \lambda(k) =& Pr[A\text{succeeds}|A_k] \cdot Pr[A_k] + \\ & Pr[A\text{succeeds}|\bar{A_k}] \cdot Pr[\bar{A_k}]\end{aligned}$$

上式最多为 $Pr[A_k] + 1/2$。因此，$Pr[A_k] > \lambda(k)$ 必然是不可忽略的，而 M 在 f 的逆运算中往往是不可忽略的。

$E(x) = f(r)||G(r) \oplus x||H(rx)$ **方案是抗选择密文攻击安全.** 设 $A = (F, A_1)$ 是一个 RS-敌手，对于某个不可忽略的函数 $\lambda(k)$，其成功概率为 $1/2 + \lambda(k)$。我们构造一个算法 $M(f, d, y)$，它不可忽略地经常计算 $f^{-1}(y)$，其中 $(f, f^{-1}, d) \leftarrow \mathcal{G}_*(1^k); r \leftarrow d(1^k); y \leftarrow d(r)$。算法 M 从运行 $F(E)$ 开始，其中 E 由我们的方案指定的 f 定义。F 取三个神谕，即 G、H 和 $D^{D,H}$，其查询由 F 回答如下。如果查询 r 到 G 满足 $f(r) = y$，则 M 输出 r 并停止; 否则，它返回一个适当长度的随机字符串。如果查询 rx 到 H 满足 $f(r) = y$，则 M 输出 r 并停止; 否则，它返回一个适当长度的随机字符串。为了回答查询 $a||\omega||b$ 到 $D^{G,H}$ 算法 M 看它是否已经问了某个 G 的查询 r 和 H 的查询 ru，其中 $a = f(r)$ 和 $\omega = G(r) \oplus u$，如果是，则返回 u; 否则返回无效。如果 M 完成 $F(E)$ 的运行，那么它得到一个输出 (m_0, m_1)。现在 M 运行 $A_1(E, m_0, m_1, \alpha)$，此处对于 $\omega \leftarrow \{0,1\}^{|m_0|}$ 和 $b \leftarrow \{0,1\}^k$，有 $\alpha = y||\omega||b$。同样，M 必须模拟对 G、H 和 $D^{G,H}$ 的查询行为。这和之前一样，当 F 被 M 运行时。

要查看此构造是否有效，首先考虑运行其 Oracle 的“真实”环境。让 A_k 表示的事件 $a||\omega||b \leftarrow F(E)$, 对于一些 a、ω、b, 和 A 做某个 Oracle 调用 $G(r)$ 或 $H(ru)$, 此处 $f(r) = a$。让 L_k 表示如果 A_1 问 $D^{G,H}$ 一些查询 $a||\omega||b$，此处 $b = H(f^{-l}||\omega \oplus G(f^{-1}(a)))$, 但 A_1 从未询问 H-Oracle 关于 $f^{-1}(a)||\omega \oplus G(f^{-1}(a))$ 的像。设 $n(k)$ 是 Oracle 查询的总次数，很容易验证 $Pr[L_k] \leq n(k)2^{-k}$。这一点也很容易看出

$$Pr[A \text{ succeeds } |\bar{L_k} \wedge \bar{A_k}] = 1/2.$$

因此，$1/2 + \lambda(k) = Pr[A \text{ succeeds}]$ 以下式为界，

$$\begin{aligned} &Pr[A \text{ succeeds}|L_k]Pr[L_k]+ \\ &\quad Pr[A \text{ succeeds } |\bar{L_k} \wedge A_k]Pr[\bar{L_k} \wedge A_k]+ \\ &\quad Pr[A \text{ succeeds } |\bar{L_k} \wedge \bar{A_k}]Pr[L_k \bar{\wedge} \bar{A_k}] \end{aligned}$$

其最大值为 $n(k)2^{-k} + Pr[A_k] + 1/2$。所以有

$$Pr[A_k] \geq \lambda(k) - n(k)2^{-k}.$$

现在，回到 M 对 A 的模拟，注意 M 不能像 A 一样以 $Pr[L_k]$ 为界，并且

$$Pr[M \text{ inverts } f \text{ at } y] \geq \lambda(k) - n(k)2^{-k+1}$$

依然是不可忽略的。证明完成。

$E(x) = f(r)||G(r) \oplus x||H(rx)$ **方案是非延展的.** 直观地说，加密字符串 α' 中有效标签 $H(r'x')$ 的存在，它不是提供给对手 A 的加密副本充当“知识证明”，即“知道”（可以恢复）zx'。现在假设 A，看到 $\alpha = ||\omega||b$ 加密 $x = G(f^{-1}(a)) \oplus \omega$，设法提出与 x 相关的字符串 x' 的加密。当 r 没有向 G 问时，对手 A 不能将 x 与（已知值）zx' 相关联，因为她不知道 $G(r)$ 的值。因此 A 必须合理询问 G 有关 r 的像。每当她这样做时，她就有效地对陷门排列求逆。上面的论证可以形式化; 我们现在概述一下如何这样做这个形式化。

给定一个 M-敌手，$\mathcal{A} = (F, A)$ 和一个由多项式时间机器 M 计算的有趣关系 ρ，定义多项式时间算法 $A_*(E, \pi)$ 如下:

> $A_*(E, \pi)$ 计算 $x_* \leftarrow \pi(1^k)$; $r_* \leftarrow d(1^k)$;
> $\alpha_* \leftarrow f(r_*)||G(r_*) \oplus x_*||H(r_*x_*)$; $\alpha_*^{'} \leftarrow A(f, \pi, \alpha_*)$; 如果 $\alpha_*^{'} = \alpha_*$，那么 A_* 输出 0 的加密，否则 A_* 输出 $\alpha_*^{'}$。

我们将证明 $|\varepsilon(k) - \varepsilon_*(k)|$ 是可忽略的，其中这些量与不可延展性的定义相同。需要一些案例分析来证明这种说法。它是基于考虑两个相关的实验，第一个定义 $\varepsilon(k)$，第二个定义 $\varepsilon_*(k)$。我们从描述实验 1 开始。这里 $G \leftarrow 2^\infty$;$H \leftarrow 2^\infty$;$(f, f^{-1}, d) \leftarrow \mathcal{G}_*(1^k)$; 则 E 为 f 表达的加密算法，D 为对应的解密; $\pi \leftarrow F^{G,H}(E)$; $x \leftarrow \pi^{G,H}(1^k)$; $r \leftarrow d(1^k)$; $a = f(r)$; $\omega = G(r) \oplus x$; $b = H(rx)$; $\alpha = a||\omega||b$; 和 $\alpha' \leftarrow A(E, \pi, \alpha)$。写 $\alpha' = a'w'b', r' = f^{-1}(a')$ 和 $x' = \omega' \oplus G(r')$，我们感兴趣的是 $M^{G,H}(x, x', E, \pi)$ 的值，它的期望，我们记为 $E_1[\rho(x, x')]$，正好是 $\varepsilon(k)$。在进行实验 1 时，我们区分了以下情况:
Case 1: $a' = a$. 在这种情况下，根据我们对一个有趣关系的定义，$\rho(x, x') = 0$。
Case 2: 假设情形 1 不成立，并且 $\mathcal{A}$ 没有对 $r'x'$ 进行 H-Oracle 查询:
Case 2a: $b' = H(r'x')$。这个事件发生的概率是 2^{-k}。
Case 2b: $b' \neq H(r'x')$。在这种情况下，加密是乱码的，解密是 0，根据我们对一个有趣关系的定义，$\rho(x, x') = 0$。
Case 3: 假设 Case 1 和 Case 2 都不成立。
Case 3a: 对于 $H(r'x') = b'$ 的 H 查询的任何字符串 $r'x'$，要么是 $f(r') \neq a$，要么是 $G(r') \oplus x' \neq \omega'$)。那么 $p(x, x') = 0$。
Case 3b: 这里 α' 是有效的加密，$\mathcal{A}$ 可以提取 r' 和 x' 的区别。设 λ_1 表示这种情况的概率。我们区分:

Case 3b(i)：当 $\mathcal{A}$ 没有对 r 进行 G-Oracle 调用时, 并且
Case 3b(i)'：$M^{G,H}$ 问一个查询 r，设 $\lambda(k)$ 是一个可以忽略的函数限定这种情况的概率。设 λ_2 是这种情况的概率。
Case 3b(i)"：$M^{G,H}$ 不要求查询 r。
Case 3b(ii)'：当 $\mathcal{A}$ 进行 r 的 G-Oracle 调用时，设 $\epsilon(k)$ 是一个可忽略的函数，限定这种情况的概率。

我们可以通过下式得到 $E_1[\rho(x,x')]$ 的上界

$$\begin{aligned} E_1[\rho(x,x')] \leq\ & Pr[\text{Case 2a}]\cdot 2^{-k}+ \\ & Pr[\text{Case 3b}]\cdot E[\rho(x,x')|\text{Case 3b(i)}]+ \\ & Pr[\text{Case 3b(ii)}] \leq 2^{-k}+\lambda(\epsilon(k)+\lambda_2)+\epsilon(k). \end{aligned}$$

我们现在描述实验 2。其定义为 $G \leftarrow 2^{\infty}; H \leftarrow 2^{\infty}; (f, f^{-1}, d) \leftarrow \mathcal{G}_*(1^k)$；则 E 是由 f 指定的加密算法，D 是相应的解密；$\pi \leftarrow F^{G,H}(E)$; $x \leftarrow \pi^{G,H}(1^k)$; $x_* \leftarrow \pi^{G,H}(1^k)$; $r_* \leftarrow d(1^k)$; $a_* = f(r_*)$; $\omega = G(r_*) \oplus x_*$; $b_* = H(r_* x_*)$; $\alpha_* = a_* || \omega_* || b_*$; $\alpha_*^{'} \leftarrow A(E, \pi, \alpha_*)$。如果 $\alpha_* = \alpha_*^{'}$，可写为 $\alpha_*^{'} = a_*^{'} \omega * {}^{'} b_*^{'}$, $r_*^{'} = f^{-1}(a')$ 和 $x_*^{'} = \omega_*^{'} \oplus G(r')$，否则为 0. 我们对 $M^{G,H}(x, x_{*'}, E, \pi)$ 的值感兴趣，我们将其期望表示为 $E_2[\rho(x, x_*^{'})]$，它就是 $\varepsilon_*(k)$。

在分析实验 2 中，我们进行与上述相同的案例分析。一个重要的观察结果是，在实验 1 和 2 中，A 的第三自变量的分布是相同的。因此，$Pr_1[\text{Case 3b}] = Pr_2[\text{Case 3b}]$。此外，很容易看出 $E_1[\rho(x,x')|\text{Case 3b(i)"}] = E_2[\rho(,x')|\text{Case 3b(i)"}]$。我们可以通过下式得到 $E_2[\rho(x, x_*^{'})|\text{Case 3b(i)"}]$ 的下界

$$\begin{aligned} & E_2[\rho(x, x_*^{'})] \geq P_r[\text{Case 3b(i)"}]\cdot \\ & E_2[\rho(x, x_*^{'})|\text{Case 3b(i)'}] \geq (\lambda_1 - 2\epsilon(k))\lambda_2. \end{aligned}$$

因此 $|E[\rho(x,x')] - E[\rho(x, x_*^{'})]| \leq 4\epsilon(k) + 2^{-k}$，证完。

B. 签名方案的安全证明

译者注：这一节的证明在作者个人网页提供的论文拷贝中有。

假设 F 是一个成功的 S-对手，其概率为 $\lambda(k)$，不可忽略。我们构造算法 $M(f,d,y)$，使

$$\varepsilon(k) = Pr[(f, f^{-1}, d) \leftarrow \mathcal{G}_*(1^k); x \leftarrow d(1^k); y \leftarrow f(x) : M(f,d,y) = x]$$

不可忽略，这与 $\mathcal{G}_*$ 是陷门排列生成器的事实相矛盾。$M(f,d,y)$ 的作用如下。它让 $PK = f$, 它为 F 抛硬币并开始运行 $F(PK)$。我们假设 F 对 H 进行了 $n(k)$ 次查询，所有查询都是不同的，如果 F 进行了一次签名查询 m，那么它已经查询了 $H(m)$; 这是显而易见的。M 随机选择 $t \in \{1, \ldots, n(k)\}$。然后，它对查询的回复如下

- 令 m_i 表示 F 执行的第 i 个 H 次查询。如果 $i = t$，则 M 返回 y。否则，它选择 $r_i \leftarrow \{0,1\}^k$ 并返回 $y_i = f(r_i)$。
- 假设 F 进行签名查询 m，如果 $m = m_t$，则 M 停止，承认失败。否则 M 回答 r_i，其中 $i \neq t$ 满足 $M = m_i$。

设 (m, σ) 为 F 的输出。如果 $m \neq m_t$，则 M 停止承认失败。否则，如果 $f(\sigma) = m$，则 M 输出 σ 并停止; 否则，它再次承认失败。为了进行分析，考虑这样一个实验，其中 t 是随机选择的，然后 F 以通常的方式运行它的预言; 那就是:

$$R \leftarrow 2^\infty; t \leftarrow \{1, \ldots, n(k)\}; (PK, SK) \leftarrow \mathcal{G}(1^k); (m, \sigma) \leftarrow F^{R, Sign^R(SK, \cdot)}(PK).$$

设 S_k 表示 F 在实验中成功的事件。注意，如果 F 成功并且它的输出 (m, σ) 满足 $m = m_t$，那么根据定义，m_t 没有被签名 oracle 查询。由此可以得出 $\epsilon(k) = Pr[S_k \bigwedge (m = m_t)]$, 这里的概率大于刚刚定义的实验。后一种可能性估计如下, 若 $m \notin \{m_1, \ldots, m_n(k)\}$, 则 F 成功的概率不超过 2^{-k}。我们有

$$\lambda(k) - 2^{-k} = \sum_{i=1}^{n(k)} Pr[S_k \wedge (m = m_i)].$$

因为 t 是随机选择的，我们有 $Pr[S_k \wedge (m = m_t)] \geq (\lambda(k) - 2^{-k})/n(k)$，正如所期望的那样，$\epsilon(k) \geq (\lambda(k) - 2^{-k})/n(k)$ 依然是不可忽略的。