ChainLink

一个去中心化的预言机网络

史蒂夫·埃利斯，阿里·朱尔斯，谢尔盖·纳扎罗夫

2017 年 9 月 4 日（卷一）

摘要

通过取代传统的法律合同和中央自动化的数字合同，智能合同将使许多行业发生革命性的变化。不管是运行认证还是执行都需要缔约某方手动操作或一个自动检索并更新相关变更的系统手动操作。可惜的是，由于它们背后的基本共识机制，支撑智能合同运行的区块链无法运用外部系统来支持本地通信。

今天，解决这个问题的方法是引入一种新的功能，称作“预言机”（oracle），它连接外部世界。现有的预言机是中心化服务。任何使用这种服务的智能合同都面临单点故障的问题，使得它并不比传统的中央运行数字合同更安全。

本文介绍了一种叫作 ChainLink 的去中心化的预言机网络。我们会介绍 ChainLink 为合同提供的可用于外部链接的链路组件，以及为网络节点供能的软件。我们提出了一个简单的链上合同数据聚合系统，以及更高效的链下合同机制。本文也将介绍 ChainLink 的信誉和安全监控服务，这些服务能帮助用户挑选供应商并在不利条件下保障稳定的服务。最后，我们将界定一个理想的预言机应具备的特性，来作为我们安全策略的指导，并阐述可能的未来改进方案，包括更为丰富的预言机编程、数据源基础设施的修改和机密智能合同的执行。

目录

1 引言

智能合同是在去中心化的基础设施上执行的应用程序，例如：区块链。智能合同具备防篡改属性，没有任何一方（即便是合同的制定者）可以改变它们的代码或者干扰它们的执行。历史上，代码中的合同以中心化方式运行，特权方可以任意授权、终止甚至删除它们。相反，智能合同的执行保证将所有各方都绑定到合同中，创建了一种新的、强大的信用体系，其存在并不依赖于任何一方的信用。由于它们是自我验证和自我执行的（即如上所述的防篡改），智能合同为实现和管理数字合同提供了先进的工具。

不过，智能合同所体现的强大的新信任模型引入了新的技术挑战——连通性。大多数有趣的[27][1]智能合同应用程序都依赖于现实世界中关键来源的数据，特别是数据馈送和在区块链外部的应用程序编程接口（API）。由于区块链背后的共识机制，一个区块链不能直接获取这些关键数据。

我们提出了以 ChainLink——一种安全的预言机网络的形式解决智能合同连通性问题的方案。与其他预言机解决方案不同的是，ChainLink 能够作为一个完全去中心化的网络进行操作。这种去中心化的方法限制了任何单方的权力，使得智能合同中的防篡改性能渗入到智能合同和它们所依赖的 API 之间的端对端操作。如果要取代当今使用的数字合同，就必须使智能合同具备外部意识，也就是说能够与链外资源交互连通。

今天，传统的合同中的大部分已经被自动化了，这些合同自动使用外部数据来保证合同运行，并要求数据输出被推送到外部系统。当智能合同取代这些旧的合同机制时，它们将需要相同类型的数据输入和输出的高可信度版本。潜在下一代智能合同及其数据要求的实例包括：

- 证券智能合同（如债券、利率衍生工具等）将要求访问 API，获取市场价格和市场参考数据，例如利率。
- 保险智能合同将需要与可保事件有关的物联网数据的数据输入，例如：仓库被入侵时磁门是否上了锁，公司的防火墙是否在运行，或者是入了保险的航班是否按时到达目的地。
- 贸易融资智能合同将需要运输的 GPS 数据、来自供应链 ERP 系统的数据以及关于被装运货物的海关数据，来以确认合同义务的履行。

[1]今天以太坊智能合同的主要应用是代币管理，在大多数智能合同网络中，这些代币是一个共同的功能。我们认为，当前过于专注代币而排除了许多其他可能的应用程序，原因就在于缺乏足够的预言机服务，ChainLink 就是要专门弥补此点不足。

其中常见的另一个问题是，智能合同不能将数据输出到链下系统中。这种输出通常采取经由传统的集中式系统的支付消息的形式，在集中式基础设施中，用户已经有了账户，用于例如银行支付、PayPal 和其他支付网络等的账户。ChainLink 能够为智能合同将数据安全地推送到 API 以及各种遗留系统，从而允许创建能够感知外部的防篡改合同。

白皮书路线图

在白皮书*中，我们回顾了 ChainLink 架构（第 2 章）。接着我们解释如何定义预言机的安全性（第 3 章）。我们描述了对预言机和数据源的去中心化/分布式的 ChainLink 方法（第 4 章），然后讨论了 ChainLink 所建议的四种安全服务以及 LINK 代币（LINK tokens）所扮演的角色（第 5 章）。然后，我们提出了一个长期发展策略建议，包括更好的保密防护、可信硬件的使用、基础设施的变更和通用预言机可编程性（第 6 章）。我们简要回顾了可替代的预言机设计（第 7 章），最后简要讨论了指导 ChainLink 开发的设计原则和理念（第 8 章）。

2 架构综述

ChainLink 的核心功能目标是连接两个环境：链上和链下。下面我们将描述每个 ChainLink 组件的架构。ChainLink 最初将建立在以太坊之上[16]，[35]，但是我们想让它支持所有的包括链下和跨链交互的领先的智能合同网络。在它的链上和链下的版本中，ChainLink 的设计思想是模块化的。ChainLink 系统的每一块都是可升级的，因此可以随着更好的技术和竞争实现不同的组件的替换。

2.1 链上架构

作为一项预言机服务，ChainLink 节点返回对由用户合同或用户合同作出的数据请求或查询的回复，我们称之为请求合同，并用 USER-SC 表示。ChainLink 的链上界面请求合同本身是一个链上的合同，我们用 ChainLink-SC 表示。

ChainLink-SC 的背后，ChainLink 具有由三个主要合同组成的链上组件：信誉契约、订单匹配契约和聚合契约。信誉契约跟踪预言机服务提供商的性能度量。订单匹配智能合同采取建议的服务水平合同，记录 SLA 参数，并从预言机供应商收集投标书。然后，它使用信誉契约来筛选竞标，并终结预言机 SLA。聚合契约收集预言机供应商的回复并计算 ChainLink 查询的最终汇总结果。它还将预言机供应商的度量数据反馈回信誉契约。ChainLink 合同是以模块化方式设计的，允许用户根据需要配置或替换。链上的工作流程有三个步骤：1）预言机筛选，2）数据报告，3）结果汇总。

预言机筛选

预言机服务购买者指定构成服务级别合同(SLA)方案的要求。SLA 方案包括诸如查询参数和购买者所需的预言机的数量等细节。此外，买方规定了本合同其余部分使用的信誉和聚合合同。

使用链上记载的信誉，以及从过去合同记录中收集的更完备的数据，购买者可以通过链下列表服务手动排序、筛选和选择预言机服务。我们的意图是让 ChainLink 也具备这样一个的列表服务，收集所有与 ChainLink 相关的记录，并验证列出的预言机合同的二进制文件。我们在第 5 节中进一步详细介绍了列表服务和信誉系统。用于生成列表的数据将从区块链中提取，从而允许构建供替代的预言机列表服务。买方将向预言机提交 SLA 提案，并在最后在敲定链上 SLA 之前达成合同。

在所有情况下手动匹配都是不可能的。例如，合同可能需要不时请求预言机服务以对接收到的信息进行动态回应。自动化解决了这个问题并提高了可用性。由于这些原因，通过使用订单匹配合同，ChainLink 提出了自动化的预言机匹配。

一旦买方指定了 SLA 方案，而不是直接联系预言机，他们将提交 SLA 到订单匹配合同。向订单匹配合同提交提案就会启动一项日志文件（log），预言机提供商可以基于其能力和服务目标来监视和过滤这项日志文件。接着 ChainLink 节点将选择是否对该方案进行投标，而合同只接受来自满足 SLA 要求的节点的投标。当预言机服务提供商在合同上投标时，他们会对其负责，特别是采用附加惩罚金的方式，根据 SLA 中的条款，如提供商有不当行为，这笔惩罚金将会被上缴。

在整个投标期内都可以投标。一旦 SLA 已经接收到足够的合格投标并且投标期已经结束，则从投标池中选择一定数量的预言机。在投标过程中，惩罚金将返回给未被选中的预言机，一份最终的 SLA 记录将会创建出来。当 SLA 记录完成时，将触发一个日志文件通知入选的预言机。然后，预言机将执行 SLA 中指定的任务。

数据报告

一旦新的预言机记录被创建，链下的预言机就执行该合同并向链路报告。关于链下交互的更多细节，参见第 2.2 节和第 4 章。

结果汇总

一旦预言机向预言机合同公布他们的结果，这些结果将被馈送到聚合合同。聚合合同收集汇总结果，并计算了一个加权值。然后将每个预言机回应的有效性报告给信誉合同。最后，加权值返回到 USER-SC 中指定的合同函数。

检测异常值或错误值是特定于每种类型的数据馈送和应用的问题。例如，在平均计

算之前检测和拒绝异常答案对于数值数据是必要的，对于布尔值却并非如此。因此，不存在具体的聚合合同，只存在一个买方指定的可设置合同地址。ChainLink 将包括一组标准的聚合合同，但是定制的合同也可以指定，只要它们符合标准计算接口。

2.2 链下架构

在链下，ChainLink 最初由一个连接到以太坊网络的预言机节点网络组成，我们希望它支持所有领先的智能合同网络。这些节点独立地获取对链下请求的回复。如我们解释的，他们的个体响应经由若干可能的共识机制中的一个，被聚集成全局响应，该全局响应被返回到请求的 USER-SC。ChainLink 节点由标准的开源核心实现提供动力，该开源核心的执行会处理标准的块链交互、调度和与公共外部资源的连接。节点操作员可以选择添加软件扩展，称为外部适配器，允许运营商提供额外的专门的链下服务。链接节点已经在企业设置中沿着公共块链和私有网络同时部署；使得节点以分布式的方式运行是 ChainLink 的出发点。

ChainLink 核心

节点软件负责与区块链、调度和平衡其各种外部服务。ChainLink 节点完成的工作被格式化为任务分组(assignments)。每个任务分组都是一组较小的指定作业，称为子任务，被处理为管道。每个子任务在将其结果传递到下一个子任务之前执行它的特定操作，并最终达到最后的结果。ChainLink 的节点软件附带一些内置的子任务，包括 HTTP 请求、JSON 解析和各种块链格式转换。

外部适配器

除了内置的子任务类型外，自定义子任务可以通过创建适配器来定义。适配器是具有最小 REST API 的外部服务。通过以面向服务的方式建模适配器，任何程序语言中的程序都可以简单地通过在程序前面添加一个小的中间的 API 来实现。类似地，与复杂的多步骤 API 交互可以简化为具有参数的单个子任务。

子任务模式

我们预期许多适配器将是开源的，因此可以由各种社区成员进行审计和运行服务。因为许多不同的开发人员开发了许多不同类型的适配器，确保适配器之间的兼容性是很重要的，

ChainLink 当前使用基于 JSON 模式[36]的模式系统来操作，以指定每个适配器需要什么输入以及如何对它们进行格式化。类似地，适配器指定一个输出模式来描述每个子任务的输出。

3 预言机的安全性

为了解释 ChainLink 的安全架构，我们必须首先解释为什么安全性很重要，以及安全性意味着什么。

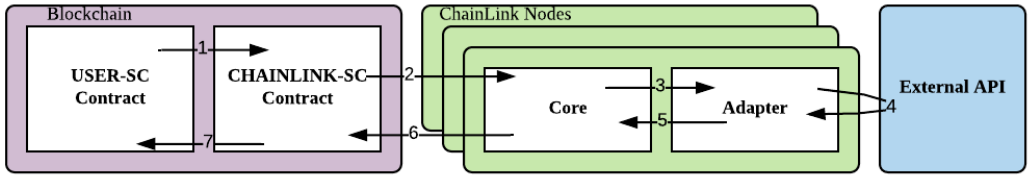

图 1：ChainLink 的工作流程：1）USER-SC 发出链上请求；2）CHAINLINK-SC 为预言机记录事件；3）ChainLink 核心接收事件并将其任务送至适配器；4）ChainLink 适配器想外部 API 执行请求；5）ChainLink 适配器处理请求并将其传递回核心；6）ChainLink 核心将数据报至 CHAINLINK-SC；7）CHAINLINK-SC 聚集响应并将其以单一响应送回至 USER-SC。

为什么要确保预言机的安全性？回到第一节中所举的简单例子，如果智能合约得到的数据馈送是错的，那么它的支付对象就会发生错误。如果智能合约的保险数据馈送可以任由投保人篡改，那么就有可能出现保险欺诈。如果送至贸易融资合约的 GPS 数据可以在其离开数据提供者后被修改，那么支付将在货物尚未到达的情况下完成。

更普遍的是，一个功能良好且包含分类帐或公告栏的区块链，拥有非常强大的安全性能。用户依赖于区块链的一种功能，即它可以正确地验证交易并防止数据被修改，实际上如同对待可信第三方（这个概念我们会在下文中详细探讨）。支持的预言机服务必须提供与它所支持的区块链相当的安全级别。因此，预言机也必须将用户作为有效的可信第三方，尽可能提供正确且及时的响应。任何系统的安全性都是它最薄弱的环节，因此想要维护精心设计的区块链的可信度，就必须要有一个可信度极高的预言机。

定义预言机的安全性：一个理想的观点

为了解释预言机的安全性，我们首先需要给它下个定义。一种有指导意义的、有原则的解释方式源于以下的思维实验。假设一个可信第三方（TTP）——一个理想的实体或功能，总是忠实地执行指令——任务是运行预言机。我们将预言机表示为 ORACLE（通常通过将所有字母大写来表示用户完全信任的实体），并假设可信第三方 TTP 从一个完全可信的数据源 Src 中获取数据。那么，我们会要求 ORACLE 执行什么指令呢?

为了保证属性真实完整，亦被称为真实性属性[24]，我们只需要 ORACLE 执行以下几个步骤：

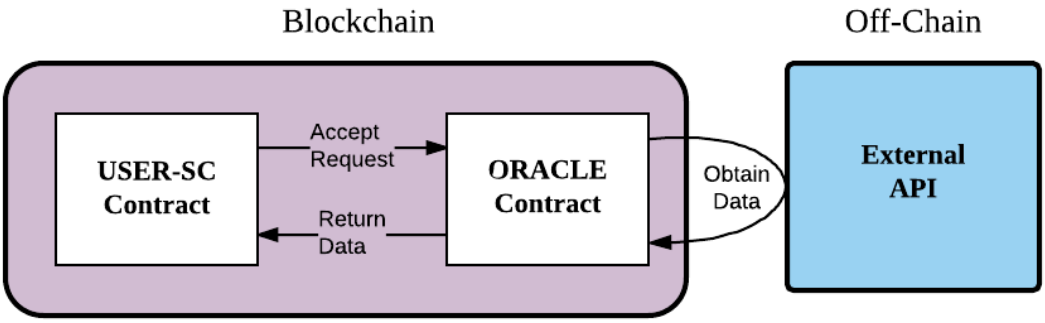

图 2：理想的预言机 ORACLE 的行为可以通过以下几步来定义：1）接收请求；2）获取数据；3）返回数据。此外，为了保护请求的保密性，一旦开始解密，ORACLE 就不再使用或显示其中包含的数据，查询 Src 例外。

1. *接受请求*：从智能合约 USER-SC 中提取请求 Req=(Src,τ,q)，其中 Src 为指定的目标数据源、τ 为一个时间点或不同时间段，*q* 为查询；
2. *获取数据*：时间 τ 将查询 *q* 发送至目标数据源 Src；
3. *返回数据*：接收回答 a 时，即将 a 送回智能合约。

正确地执行上述这些简单的指令，即定义了一个强大、深刻，但又非常简单的安全概念。人们直观地要求 ORACLE 充当 Src 和 USER-SC 之间可信的桥梁。[2] 举例说明，如果 Src 是 https://www.FountOfKnowledge.com，τ 是下午 4 点，q = “英特尔公司的股票价格”，那么 ORACLE 的完整性就保证了它会提供给 USER-SC https://www.FountOfKnowledge.com 中在下午 4 点整时英特尔公司的股票价格。

保密性是预言机的另一个理想性能。当 USER-SC 将 Req 用明码通过区块链送至 ORACLE 时，Req 是公开可见的。但在许多情况下，Req 包含敏感信息，不宜公开。比如说，如果 USER-SC 是一个飞行保险合约，并向 ORACLE 发送一个有关特定用户的航班信息的查询请求 Req (q = "以太航班 338 ")，那么结果就会是该用户的航班信息在全球范围内均可见。再比如，如果 USER-SC 是一份金融交易合约，Req 就有可能泄露用户交易和投资组合的信息。当然了，类似的例子还有很多。

为了保护 Req 的保密性，我们可以要求在 Req 中使用属于 ORACLE（公钥）的加密

[2] 当然，此处省略了许多细节。ORACLE 应与 USER-SC 和 Src 均有安全通信，也就是说，要有防篡改渠道（如果 Src 是一个 web 服务器，那么就必须要有 TLS。为了与 USER-SC 进行通信，ORACLE 必须确保刮掉正确的区块链，并适宜地签上 A）。

数据。接着利用 ORACLE 的可信第三方 TTP 本质，我们可以简单地为 ORACLE 提供信息流约束：

在解密 *Req* 时，除了查询 *Src* 外，不得在显示或使用 *Req* 中的数据。

预言机还有一些其他的重要属性，比如可用性，即经典 CIA（保密性—完整性—可用性）三组合中的最后一个。当然了，真正理想的 ORACLE 永远不会出现故障。可用性还包括一些更微妙的性能，譬如审查阻力：诚实的 ORACLE 不会挑出特定的智能合约并拒绝其要求。

可信第三方的概念类似于理想功能的概念[7]，用于证明某些模型中加密协议的安全性。我们也可以用类似的术语来模拟区块链，按照维持理想公告栏的可信第三方将其概念化。其指令为接受交易，验证交易，将交易序列化，并在公告栏上永久地维护交易，是一种只可追加的数据结构。

为什么理想的预言机（ORACLE）很难实现。因为完全可信的数据源 Src 是肯定不存在的。由于网站错误、服务提供商欺诈或者一些人为的无心之过等原因，或无意或恶意地造成了数据损坏。

如果 Src 不能保证可信，那么即使 ORACLE 的操作完全像上述的可信第三方 TTP 一样，也仍然不能完全满足我们想要的安全性。如果 Src 是错的，那么上文所定义的完整性属性意味着预言机的回答 a 不再是正确的。举例说明，如果英特尔公司的真实股票价格是 40 美元，然而 https://www.FountOfKnowledge.com 上面误报成了 50 美元，那么 ORACLE 将把错误值 a = 50 美元发送给 USER-SC。在使用单一数据源 Src 时，此类问题将不可避免。ORACLE 根本无法知道 Src 为其查询提供的答案是否正确。当然了，更大的问题是，我们对 ORACLE 提出的可信第三方 TTP 只是一个抽象概念。可以无条件绝对信任的服务提供商是不可能存在的。即便是最善意的服务提供商亦可能会存在故障或被黑客攻击。因此，对于用户或智能合约而言，是无法保证 ORACLE 一定会忠实地执行指令的。ChainLink 就理想功能 ORACLE 推出其安全协议。ChainLink 就是为了实现我们的目标——打造一个真实的世界系统，在现实的信任假设下，属性尽可能地接近 ORACLE。我们现在来给出解释。

简单起见，我们现在用 CHAINLINK-SC 来表示 ChainLink 的完整集合，即其全部的链上功能（不仅仅是其请求合约的接口）。因此，我们将系统架构中实际使用的多个单独合约抽象出来。

4 ChainLink 去中心化方法

为避免节点出现错误，我们提出了三种基本的互补方法：

(1)分散数据来源；(2)分散预言机；(3)使用可信硬件。在本节中，我们主要讨论前两种方法，其中涉及*去中心化*。在第六节中，我们将讨论可信硬件的长期策略，一个不同的互补方法。

4.1 分散数据来源

处理单个错误数据源 Src 的简单方法是获取来自多个源的数据，即分散数据源。可信的 ORACLE 可以查询一系列的数据源 Src1, Src2，…，Srck，从而获得一系列响应 a1, a2，…，ak，并将以上响应集合为一个单一的答案 A = agg (a1,a2，…，ak)。ORACLE 完成此过程的方法可以有很多种。比如说，多数投票制。倘若大多数的数据源返回了相同值 a，那么 agg 最终也将返回 a；否则将会返回错误指示。在此情况下，如果大多数(> k/2)数据源正确运行，那么 ORACLE 将始终返回正确值 a。

许多替代性 agg 可以确保其对抗错误数据的鲁棒性，或处理数据值随时间变化而产生的波动（比如股票价格）。举例说明，agg 可能会弃掉异常值（即最大值和最小值 a_i）然后输出剩余值的平均数。

当然了，错误亦可能和不同数据源在某种程度上相互关联，从而削弱集合的可信度。如果网站 Src_1 = EchoEcho.com 从 Src_2 = TheHorsesMouth.com 中获取数据，Src_2 中有错将意味着 Src_1 始终有错。不同数据源之间可能会有更加微妙的相关性。Chainlink 还提出通过一种易于理解的方式来研究数据来源的独立性，从而预言机和用户可以避免不必要的相关性。

4.2 分散预言机

正如数据源可以被分散一样，理想的 ORACLE 本身也可以近似为一个分布式系统。这就是说，我们可以有 n 个不同的预言机节点$\{O_1,O_2,...,O_n\}$，来代替单一的预言机节点 O。

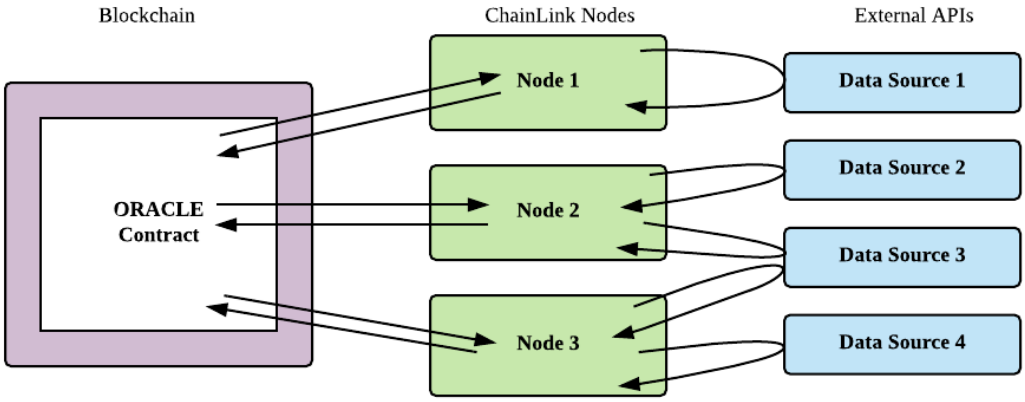

图 3：请求分布在预言机和数据源中，此图展示了此类二级分布的示例。

每个预言机 O_i 都有其特定的数据源集，这些数据源也许会但也许不会与其他预言机的数据源相重叠。O_i 从其数据源中聚集响应，并将其特定答案 A_i 输出到查询 Req 中。

有些预言机可能是错误的。显然易见的是，所有预言机这一系列回答 $A_1, A_2,..., A_n$ 需要通过一种可信的方法聚集为一个单一且具有权威性的值 A。但考虑到预言机可能会有错误，那么在 ChainLink 中，这种聚集会在何处、如何发生呢?

初始解决方案：合约中聚集。我们在 ChainLink 中最初提出的解决方案可称之为合约中聚集。再次用 CHAINLINK-SC 表示 ChainLink 的链上部分，它将聚集预言机的响应（或者，CHAINLINK-SC 也许会调用另一个聚集合约，但为了概念上尽可能简单，我们假设这两个组件将形成一个合约）。也就是说，CHAINLINK-SC 将为一些功能 Agg（和上文所述的 agg 相似）计算出 A=Agg $(A_1,A_2,...,A_n)$，并将结果 A 发送至 USER-SC。

此方法适用于 n 值较小的情况，并有以下几点益处：

- *概念简单：*尽管预言机是分散的，但单个实体 CHAINLINK-SC 可通过执行 Agg 将数值聚集。

- *可信度高：*由于 CHAINLINK-SC 的代码可以公开可见，那么其正确行为将能够得到验证（CHAINLINK-SC 的代码相对较少且简单）。除此之外，CHAINLINK-SC 的执行在链上是完全可见的。因此，用户即 USER-SC 的创建者，可以在 CHAINLINK-SC 中获得高度的信任。

- *非常灵活：* CHAIN LINK-SC 可以实现最受欢迎的聚集功能 Agg——多数决定功能，平均值等等。

正因其简单，这种方法亦造成了一个新颖有趣的技术挑战，即“不劳而获（freeloading）”的问题。一个作弊的预言机 O_z 可以观察到另一个预言机 O_i 的响应 A_i 并将其复制过来。通过这种方式，预言机 O_z 省去了查询数据源的费用花销，因为查询数据源可能会对每单个查询收取费用。这种“不劳而获”破坏了数据源查询的多样性，从而降低了安全性，同时也抑制了预言机快速响应的能力：因为慢速响应和“不劳而获”的策略成本更低。

对此问题，我们提出了一个众所周知的解决方法，即使用提交/显示方案。在第一轮中，预言机将 CHAINLINK-SC 的响应进行加密提交。在 CHAINLINK-SC 收到了有效数量的响应后，随即启动第二轮，并在本轮中，预言机显示其响应。

算法[1]显示了一个简单的序列协议，保证了给定的 3f + 1 节点的可用性。它使用提交/显示方案来防止“不劳而获”的情况发生。只有在所有提交均已完成之后，预言机的响应才会暴露给潜在的“不劳而获者（freeloader）”，从而排除了“不劳而获者”复制其他预言机响应的可能。

链上协议可以利用区块时间来支持同步协议设计。然而，在 ChainLink 中，预言机节点获取数据的数据源可能具有极易变化的响应时间，而节点提交的时间也可能会因为譬如以太坊中使用不同气体价格，从而有所不同。因此，为了确保协议能够最快得响应，算法[1]被设计为异步协议。

这里的 $\text{Commit}_r(A)$ 表示提交值 A 和证人 r，而 SID 表示一套有效的会话识别码（session id）。该协议假定所有参与者之间的信道已被认证。

不难看到的是算法[1]将会成功终止。如果总共有 $3f+1$ 个节点，至多 f 都是错误的，所以至少 $2f+1$ 将在第 4 步中提交。而在所提交的范围内，至多 f 是来自于错误节点，所以至少 $f+1$ 是来自于诚实节点。最终所有提交的这些都将被取消。

除此之外，不难看出算法[1]中的 A 是正确的。对单值 A 的 $f+1$ 提交取消中，至少有一个一定是来自于诚实节点。

算法 1　　链上聚集($\{O_i\}_1^n$) (CHAINLINK-SC 代码)

1: 等待从 USER-SC 接收 Req
2: sid←$ SID
3: 广播（request, sid）
4: 等待接收来自特定 O_i 的 $2f+1$ 信息中的集合 C（commit，c_i=$\text{Commit}_{ri}(A_i)$，sid）
5: 广播（committed, sid）
6: 等待接收 $f+1$ 特定有效提交取消中的集合 D(Answer, A, sid)，其中所有 $A_i = A$
7: 将(Answer, A, sid)发送至 USER-SC

经由算法[1]的合约中聚集将是 ChainLink 在短期内所支持的主要方法。所提出的初始方法将会涉及一种更加复杂且并发的算法变体。我们的长期建议反映在附录 A 中算法[2]和算法[3]所明确的更为复杂的链下聚集（OCA）协议。OCA 作为链下

聚集协议，将链上交易成本最小化。该协议还包括对预言机节点的支付，并确保不向“不劳而获者”支付。

中期战略：链下聚集 合约中聚集有一个主要的弊端：成本高。主要的成本来自传输和处理链上 $O(n)$预言机信息（提交并显示 $A_1,A_2,...,A_n$）。对于私链来说，此项费用可以接受。但公链已经有了链上交易费用，比如以太坊，如果 n 数值很大，那么成本将会令人望而却步。性价比更高的方法是在链下将预言机的响应聚集成一条单一的信息传输给 CHAINLINK-SC A。我们建议在中长期部署这种称为链下聚集的方法。

面对可能出现错误的节点难以取得一致的值 A，这一点和支撑区块链本身的一致性问题非常相似。给定一组预先确定的预言机，我们可以考虑使用经典的拜占庭容错（BFT）一致性算法来计算 A 。然而，经典的 BFT 协议为了确保在协议调用结束时，所有诚实节点都存储相同的值，比如在区块链中，所有节点都存储相同的新块。在我们的预言机设置中，目标略有不同。我们想要确保 CHAINLINK-SC（然后是 USER-SC）聚合得到回答 A = Agg(A1,A2，…)，*是不参与协商一致协议，也不需要从多个预言机中获得回答的*。除此之外，还需要解决“不劳而获”的问题。

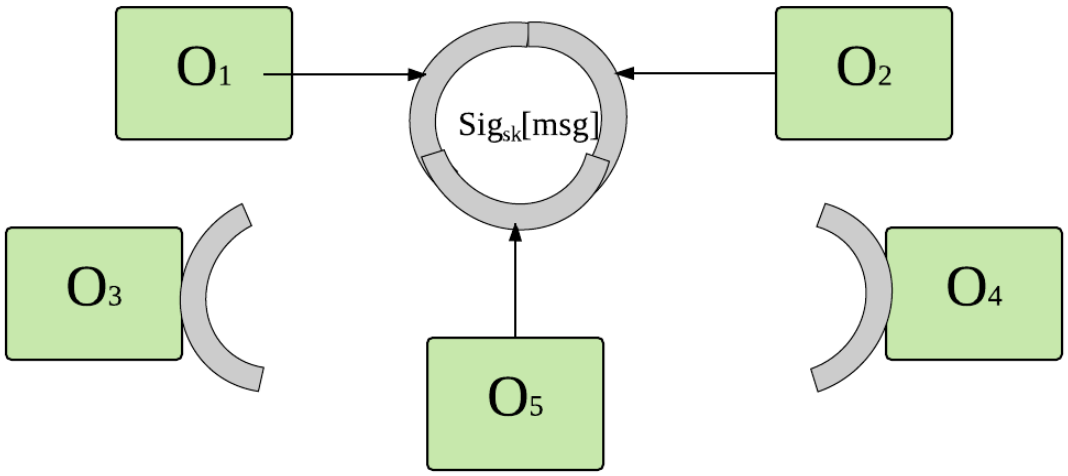

图 4： $Sig_{sk}[A]$ 可以通过预言机的任何 n/2+1 实现

ChainLink 系统提出使用一个包含阈值签名的简单协议。这样的签名可以通过使用任意数量的签名方案实现，但如果使用 Schnorr 签名实施起来会格外简单[4]。在此方法中，不同的预言机都会有一个集体公钥 pk 和一个 $O_1,O_2,...,O_n$ 之间共享的相应的私钥 sk，以(t,n)阈值的形式出现[3]。这样的共享意味着每个节点 O_i 都拥有一个特定的私钥/公钥对(sk_i, pk_i)。O_i 可以生成部分签名 $\sigma_i = Sig_{sk_i}[A_i]$，从而验证 pk_i。

此项设置的关键特征是，拥有相同值 A 的部分签名可以在任何预言机组 t 中聚集，从而得到回答 A 的一个有效*集体*签名 $\Sigma = Sig_{sk}[A]$。然而，并没有语言机组 $t-1$ 可以生成对于任意值的一个有效签名。因此，此项签名 Σ 暗含了至少是预言机 t 的部分签名。

阈值签名可以单纯地通过使 Σ 明确包含一组 t 从各个节点中取得的有效独立签名实现。阈值签名具有与这种朴素方法相类似的安全属性。但亦有一个链上性能提升的重要方法：降低通过 t 验证 Σ 的大小与成本。

凭借此项设置，似乎预言机能够生成并广播部分签名直到 t 那些部分签名启用了 Σ 的计算。然而，“不劳而获”的问题再一次出现了。因此，我们必须确保预言机一定是从其指定的来源获得数据，而不是从另一个预言机中欺骗并复制得来 A_i。我们的解决方案涉及到一种金融机制：实体提供者（可为智能合约）只奖励那些从其原始数据中获得部分签名的预言机。

在分布式设定中，确定哪些预言机是符合支付条件的是一件十分棘手的事情。预言机之间可能会在链下相互联络，而这时我们不再拥有一个权威的实体（CHAINLINK-SC）来接收响应，因此，我们无法在参与的预言机中直接识别出合格的受款人。故而，提供者必须从预言机本身获取错误行为的证据，而其中有些可能还是不可信的。我们提出使用一致性机制解决 ChainLink 的问题，从而确保提供者不会为“不劳而获者”支付。

我们为 ChainLink 提出的链下聚集系统，附带安全证明草图，可以参见附录 A。它利用了基于阈值签名的分布式协议，提供了对 $f < n / 3$ 预言机“不劳而获”的抵抗性。我们认为解决“不劳而获”是一个有趣且新颖的技术问题。

5 ChainLink 安全服务

最初由一个有了前一章中提到的协议，ChainLink 可在故障预言机错误率高达 f 的情况下继续保证其可用性与准确性。此外，第 6 节中谈及的受信任硬件也正作为防止预言机受损及提供错误回应的安全方式被列入积极考虑范围中。然而，基于以下三种原因，受信任的硬件并不一定能够提供最为可靠的保护：首先，在 ChainLink 网络的初始版本中，受信任的硬件并不会被部署；第二，部分用户可能不会选择信任受信硬件（详见附表 B 中的讨论）；最后，受信任的硬件无法在节点故障时提供保护，而仅能防止节点本身的不当行为。因此，用户希望确保自己能够选择最可靠的预言机，并尽可能减小 USER-SC 依靠错误率高于 f 的故障预言机的可能性。

为实现这一目的，我们建议启用四种关键安全服务，即验证系统、信誉系统、证书服务和合约升级服务。以上四种服务在初始阶段可由对启动 ChainLink 网络感兴趣的公司或团体运行，但它们的设计要求是严格依照 ChainLink 的去中心化设计理念操作。ChainLink 提出的安全服务无法阻碍预言机节点参与或改变预言机回应。其中，前三种安全服务仅为用户提供评级或指导信息，最后一种合约升级服务则完全由用户自由选择是否启用。此外，这些服务均支持独立供应商使用，对他们的参与加以鼓励，从而使用户最终能够享有可供挑选的多个安全服务选项。

.

5.1 验证系统

ChainLink 验证系统会监测链上预言机行为，为用户选择预言机提供客观的表现衡量标准。这一系统将会监测预言机行为的以下几个方面：

- *可用性*：验证系统应该记录预言机未能及时回应查询的情况，并累计正在运行的线上事件数据。
- *准确性*：验证系统应该记录预言机输出明显的错误回应的情况，错误回应的测量应以与同类预言机回应之间的偏差为准[3]。

在 ChainLink 初始的链上聚合系统中，这类监测非常直接，因为所有预言机行

[3] 此处的“偏离”必须是在数据层面上进行定义。对于简单布尔值的回应，如航班是否准时到达，偏离仅仅意味着与大多数回应相反的回应。对于其他数据，如可能由于传感器和测量来源不同而产生合理差异的城市气温数据，偏离则意味着重大的数值差距。诚然，由于不同原因，比如传感器失灵等，即使是运转良好的预言机也有可能在某些情况下输出偏离主流的回应

为对 CHAINLINK-SC 而言都是可见的。然而，在设想的 ChainLink 链下聚合系统中，聚合行为是由预言机本身执行的。在这种情况下，CHAINLINK-SC 无法直接获得预言机的回应情况，因此就不能独立监测预言机的可用性与准确性。

幸运的是，预言机会以电子签名的形式标记自己的回应，从而侧面为这些回应的所属提供确凿无疑的证明。因此，我们建议采用的方式是将验证系统以奖励预言机提交偏离回应证据的智能合约形式实现。换言之，预言机会被鼓励报告明显错误的行为。

从某种程度上来说，可用性更加难以监测，因为预言机显然不会在未能及时回应的情况下进行签名标记。因此，我们所提议的协议增强会要求预言机在从其他预言机接收到回应的证明上进行电子签名标记。随后，验证合约即会接受（并同样奖励）与同类预言机相比表现欠佳的个体提交其长期缺乏回应的系列证明的行为。

在链上和链下两种情况中，预言机的可用性和准确性数据均为仅链上可见。因此，用户/开发者能够通过使用合适的前端，如以太坊的分布式应用(Dapp)或其他获许可的区块链上相应的应用，来实时获取这些数据。

5.2 信誉系统

与 ChainLink 配套的信誉系统会记录并公布用户对预言机供应商及预言机节点的评级数据，从而为其他用户提供一个全面评估预言机表现的参考。验证系统报告在决定及有力担保预言机信誉时可能会成为重要的影响因素。不过，链上历史以外的因素也能够提供关于预言机节点安全配置文件的关键信息。这些因素包括用户对预言机品牌、操作实体和建构的熟悉程度。ChainLink 信誉系统将囊括一个基本的链上部分，使信誉系统的用户评级能为其他智能合约提供有效的参考。除此之外，该信誉标准在链下也应能够便捷访问，因为相比而言，链下是能够更高效地处理大量数据并为其更加灵活地赋予权重的环境。

对于某一特定预言机而言，信誉系统首先是为支持下列衡量标准而提出的，既关注具体任务类型的颗粒度（详见第 2 章），又涉及总体上节点支持的全部任务类型：

- *分配请求总数：*预言机过去许可的请求总数，包括完成和未完成的请求。
- *完成请求总数：*预言机过去完成请求的总数。这一数量除以分配请求总数可以计算请求完成率。

- *接受请求总数：*通过计算与同类预言机回应比较的合约后被认定为可接受的请求总数。这一数量除以分配请求总数或完成请求总数可以得知回应的准确率。
- *平均回应时长：*尽管为预言机回应预留一段确认时间的必要性不言而喻，与此同时，预言机回应的及时程度也能帮助确定其未来回应的及时程度。平均回应时长会基于已完成请求的数据进行计算。
- *惩戒性保证金总量：*如果通过锁定惩戒性保证金以确保节点算符的表现，就会得到衡量某一预言机供应商不参与欺诈性行为承诺的金融性标准，其中欺诈性行为具体是指供应商收取用户费用却不向用户提供相应服务的行为。这一标准会同时涉及暂时性和金融性两个维度。

高信誉服务在任何市场中都会受到高度激励以保证其正常运行，并确保服务的可用性与效能。用户的负面反馈会对品牌价值造成重大影响，针对不当行为的惩罚措施也会产生相同的效果。因此，我们希望促成某种良性循环，其中运转良好的预言机将获得更好的信誉，而更好的信誉反过来又会激励预言机持续提供高效的服务与表现。

5.3 证书服务

我们的验证系统与信誉系统的设计意图在于帮助解决预言机一系列的错误行为，并被建议作为大多数情况下保证系统完整性的方式。尽管如此，ChainLink 也可以包括另一种补充性机制，即证书服务。该机制的目标在于预防和/或修复罕见但却是灾难性的事件，尤其是以女巫攻击和镜像攻击形式进行的总体欺诈行为，我们将在接下来的章节中对此予以解释。

女巫攻击与镜像攻击。我们的简单聚合协议与合约内部聚合协议都致力于阻止恶意节点直接抄袭其他节点回应这种不劳而获的行为。然而，二者却都无法防范所谓的女巫攻击[9]。这类攻击通常涉及一个控制着多个看似独立的预言机的恶意设备。该恶意设备会企图控制整个预言机池，使错误率超过 f 的故障预言机混入聚合协议并在关键时刻提供虚假数据，比如说，为了影响高价值合约中的大额交易。欺诈性预言机的大量出现也可以在非单一恶意设备控制的情况下产生，但同样是由多个恶意设备合谋导致。涉及错误率高于 f 的预言机攻击或故障尤为恶性，因为仅从链上行为无法对其进行监测。

除此之外，为减少运行成本，女巫攻击者会采取一种叫做“镜像”的行为，这会使预言机基于从单一数据源查询所取得的数据直接发送单一回应。换言之，行为不当的预言机可能会在链下共享数据，同时仍伪装成独立获取数据。无论恶意设备是否选择发送错误数据，镜像行为本身都会使其获利。这种行为带来的安全威

胁远小于数据伪造，但仍会使安全级别小幅降低，因为它会阻止在针对某一特定数据源进行多样化查询时进行的错误纠正过程。例如，如果网站 https://www.datasource.com 由于偶然触发的程序错误而传输出错误的数据，许多查询者仍可获得正确的主流结果。

导致数据错误的女巫攻击、镜像行为和广义的恶意设备合谋问题均可通过使用我们的长期战略中提到的受信任硬件来解决（详见第 6 章）

证书服务设计。ChainLink 证书服务力求提供总体上的完整性和可用性保证，监测并协助阻止短中期内的镜像行为和合谋预言机数目。证书服务会为高质量的预言机供应商进行背书。我们希望再次强调，正如上述提到的，这项服务仅会出于使用户获益的目的而对供应商进行评级，而不是为指定某一预言机节点是否有资格参与系统而进行的评级。

证书服务系统支持基于预言机部署和行为的多个特征的背书。它会监测预言机验证系统数据并执行链上回应的事后抽查，尤其是针对高额交易，并将检查结果与从可信数据来源直接获得的回应进行比较。有了对预言机供应商数据的大量要求，我们预期会有足够的经济激励来证明对预言机供应商进行线下审计、确认其符合相关安全标准的合理性，如符合云安全联盟（CSA）中的云控制矩阵相关控制要求，同时要求预言机供应商提供有效的安全信息以证明他们为智能合约对预言机来源和字节代码进行了充分的审计。

除了预言机信誉衡量标准、自动化链上和链下系统欺诈行为监测以外，证书服务还是识别女巫攻击和其他不当行为的主要计划方式，因为自动化链上系统无法完成上述任务。比如说，如果所有节点都同意“月亮是由新鲜干酪组成的”这一陈述，它们就能够使 USER-SC 接受这一错误的事实。不过，月亮组成部分={新鲜干酪}这一信息则会被记录在区块链上，并在后期审查中及时发现。

5.4 合约升级服务

近期智能合约被非法侵入的众多事件表明，写出万无一失的智能合约代码是一项极具挑战性的任务[1]，[20]，[22]。即使智能合约的编程无懈可击，环境变化或程序错误仍会导致安全漏洞产生，如[2]。

介于这一原因，我们建议运行合约升级服务。我们需要强调这一服务的使用完全是由用户自由控制、自愿选择的。

短期来看，安全漏洞一旦被发现，合约升级服务只需基于 ChainLink 网络中的可用资源重新制作一组支持性预言机合约即可。随后，新创建的智能合约就能直

接迁移至最新一组预言机合约中。

然而，现有的智能合约仍会与存在安全漏洞的旧预言机合约组绑定。因此，从长期来看，CHAINLINK 应该支持预言机调用中的标志（MIGFLAG），一旦新的 CHAINLINK-SC 变得可用，请求合约将直接指示是否将调用迁移至新的 CHAINLINK-SC。将初始设置（如标志缺失）设定为错误状态，MIGFLAG 会使请求合约从自动转发行为中受益，从而迁移至 CHAINLINK-SC 的新版本。为了激活转发行为，用户需使其请求合约用 MIGFLAG = true 指令发布 ChainLink 请求（用户可以巧妙设计自己的智能合约，使其在收到指令时直接通过授权的合约管理员在链上改变标志）。

用户迁移至新预言机合约的可能起到了类似“安全舱”的作用，这是区块链研究人员（见[23]）一直以来所推崇的方式。因为这样可以直接修复程序错误以及阻止系统入侵，而无须使用更为繁琐的白帽反入侵[1]或硬分叉的方法。向升级合约的迁移在区块链上是可见的，并且可供用户在升级前进行审计和审查。

即便如此，我们认识到仍有部分用户并不希望任何团体以迁移/转发的方式控制安全舱。强迫进行的迁移能够使迁移合约的控制者或损害相关证明的入侵者进行恶意活动，如改变预言机回应等。正因如此，请求合约对于转发特征有全权控制，可以自主选择是否退出安全舱激活过程。除此之外，与 ChainLink 对去中心化的关注一致，我们期待更多供应商能够支持来自开发者社区的不同版本的 CHAINLINK-SC。

5.5 LINK 代币使用

ChainLink 网络使用 LINK 代币**奖励 ChainLink 节点算符从链下数据馈送提取数据、将数据转变为区块链可读格式、运行链下计算以及作为算符提供正常运行保证的行为。以太坊等网络上的智能合约使用 ChainLink 节点时，需要使用 LINK 代币支付给他们所选择的 ChainLink 节点算符，具体价格则是由各个算符基于其 ChainLink 节点提供的链下资源需求量与其他相似资源的供应量而决定的。LINK 代币是 ERC20 标准的代币，并具有 ERC223 标准“转移和调用”的额外功能，使其能够在单次交易中被合约接收和处理。

6 长期技术战略

ChainLink 在本白皮书中提议的长期技术战略包括三个重要方向，分别是预言机保密性、基础设施变更以及链下计算。

6.1 保密性

分布式预言机网络力求提供防范故障预言机的高规格保护。在大多数使用场景中，分布式预言机网络力求在故障预言机错误率高于 f (因为我们的简单聚合协议中 $f < n/2$)的情况下仍能保证正确回应的输出。受信任的硬件能够提供的则远不止如此，并因此被建议作为保护 ChainLink 网络的更佳方式。受信任的硬件是 TC 预言机[24]的基石，后者正在以太坊主网上运行，其开发者在 TC 启动时便与智能合约公司建立了合作关系。

一些特殊形式的受信任硬件，尤其是英特尔最新开发的软件守护扩展（SGX）系列指示设定的建构扩展[12]-[15],[18]，力求为分布式信任提供强大的辅助。简要来说，SGX 允许应用在有着两种安全特性、被称为飞地的环境中执行。首先，飞地能保护应用的完整性，包括保护应用数据、代码和控制流不受其他流程的破坏。其次，飞地可以维护应用的保密性，这意味着该应用的数据、代码和执行状态对其他流程而言是完全不可见的。SGX 致力于保护飞地环境下的应用免受恶意操作系统的破坏，即使是应用运行主机的管理员也无法破坏其完整性。

尽管其他形式的受信任硬件，如 ARM 的 TrustZone, 早已存在，SGX 却能够这些技术所缺乏的额外的关键功能。它能使平台为某一个特殊应用的执行（由其散列状态的构造识别）出具证明。这一证明可以被远程核实，使该应用受限于公共密钥，由此与他方建立有授权的机密渠道。

在飞地环境中运行预言机和分布证明能够为预言机执行某类特殊应用提供强有力的保证，在执行 ChainLink 生态中的开发者创造或背书的应用时尤其如此。除此之外，通过超文本传输协议安全连接到数据来源且在飞地环境中运行的预言机能够有力确保所取得的数据未被篡改（详见[24],[33]）。这些特性在防止预言机行为不当（尤其是数据损坏、女巫攻击等）方面能够提供有效的保护。

然而，更大的机遇在于受信任硬件提供高度保密性的能力。总体来说，保密性需求是区块链使用的主要障碍之一。维持保密性的预言机对于解决这一问题可以起到至关重要的作用。

分布式预言机为何不能保证保密性。从根本上而言，保密性在预言机系统中极难实现。如果预言机有区块链前端，比如智能合约，那么对该预言机的任何查询都将是公开可见的。查询可在链上加密并由预言机服务解密，然而这样预言机服务本身就能获取查询的信息。即便是安全多方计算这样重量级的工具，能够许可机密数据的计算，也无法在现有的基础设施下解决这一问题。（基于应用视角的案例见[11]）。总会有一些情况需要服务器向目标数据源服务器发起查询。因此，它必须能够直接获取查询，不管该查询之前享有怎样的保密级别。此外，服务器也能够获取对该查询的回应。

通过 SGX 维持保密性的预言机。使用 SGX 的预言机能够在飞地环境下接收并处理数据，本质上就像一个完整性与保密性受信任的定时令牌协议。首先，这样的预言机能够在其飞地环境内解密查询。其次，它能在不将查询暴露给其他流程（或人类）的情况下对其进行处理。飞地环境同样能够从源头以保密方式处理数据并能安全地管理敏感信息，如用户证书等。我们会在接下来的章节进行演示。

TC 系统支持保密航班数据的查询。航班数据可以被传递至 TC 服务公共密钥加密下的 TC 智能合约前端。TC 会解密查询，继而通过超文本传输协议确认数据来源（比如 flightaware.com 网站）。它能够就进行查询的智能合约所提出的“航班是否延误”这一问题给出简单的肯定或否定的回答，而且不会在链上泄露其他数据。

TC 另一项更为有趣的能力是能够支持 Steam 游戏平台上的交易。TC 能够安全地接收用户证书（即用户的登录密码）以确保游戏所有权在买卖双方之间完成传递。由此，它能够创造一个其他形式无法创造的安全交易市场，并有效保证电子货物以加密货币的形式进行公平交易（与之对比的是，简单的分布式预言机无法安全地代表用户管理他们的登录密码）。

TC 也能够从多个数据来源执行受信任的链下数据聚合，以及进行受信任的数据计算（如计算平均值）和对数据来源进行交互性查询（如搜索某一数据来源的数据库以回应另一数据库的结果）。

受信任的硬件为区块链的高扩展性使用提供了激动人心的新方案[24]，[29]，其中大部分区块链基础设施，包括智能合约，都在飞地环境下执行。这一建构将区块链的透明性与链下执行和受信任硬件的保密性有效地结合起来。尽管有使用其他技术完成这一目的的想法存在，如 zk-SNARKs [21]，受信任的硬件其实远为实际（以及更不复杂）。我们当前的研究目标就包括这一极具前景的可能性，预言机 z 则是其中催化剂式的服务。

我们在附表 B 中简要地谈及了将英特尔作为可信根的问题。

鉴于 SGX 定义安全。在使用受信任硬件的情况下，更正式地定义预言机准确性成为可能，英特尔 SGX 在[32]中提出的体系就是其一。这一体系使 SGX 被看做全球性的通用可组合（UC）[6] 函数 Fsgx(Σsgx)[progencl, R]. 在这里，Σ 代表着签名方案，其中签名由 Σ.Sign 表示，核实由 Σ.Verify 表示。Fsgx(Σsgx)[progencl, R]由团体签名方案 Σsgx 进行参数化。增强 progencl 代指飞地环境中运行的程序，即受硬件保护的环境。R 代指在 SGX 主机上运行的不受信任的代码，即调用应用在飞地环境中运行的软件。

图 5（截取自[24]）展示了函数 Fsgx 的运作。在初始化后，它运行 outp := progencl.Initialize()公式，产生并证明 progencl 和 outp 代码。证明 σatt 是平台电子签名的声明，证明 progencl 在飞地环境中运行并产生了输出 outp。在典型的使用中，progencl.Initalize()输出可以被用来创建应用程序实例安全渠道的公共密钥。在使用(id, params)调用 resume 后，Fsgx 继续执行并输出结果 progencl.Resume(id, params)，其中 id 指代会话标识符，params 指代对 progencl 的参数输入。

$F_{sgx}[prog_{encl}, R]$: **abstraction for SGX**

Hardcoded: sk_{sgx} (private key for Σ_{sgx})

Assume: $prog_{encl}$ has entry points **Initialize** and **Resume**

Initialize:

On receive (init) from R:

Let outp := $prog_{encl}$.**Initalize**()

σ_{att} := Σ_{sgx}.Sign(sk_{sgx}, ($prog_{encl}$, outp))

Output (outp, σ_{att})

Resume:

On receive (resume, id, params) from R:

Let outp := $prog_{encl}$.**Resume**(id, params)

Output outp

图表 5： 展示 SGX 子集功能的 SGX 执行的正式演算

在图表 5 中的形式下，准确定义预言机的完整性成为可能。定义 1 是被我们称为预言机真实性定义的简要概括。

定义 1（预言机真实性）。我们称预言机 O 运行程序 progencl 使用 Fsgx 并输出实例密钥 pkO 满足预言机真实性。如果，对任何多项式定时的设备 A 能够与 Fsgx 随意交互，A 不能使诚实的校对机接受(pkO, σatt, params := (url, , T), data, σ)，因为这里的数据不是我们的模型中时间 T(progencl.Resume(id, params) 下公共密钥的 url 内容。形式上，对于任何可能的多项式定时的设备 A，：

$$\Pr\left[\begin{array}{l}(pk_O, \sigma_{att}, id, params, data, \sigma) \leftarrow A^{F_{sgx}}(1^{\lambda}) : \\ \dot{\Sigma}_{sgx}.Verify(pk_{sgx}, \sigma_{att}, (prog_{encl}, pk_O)) = \dot{1} \wedge \\ \dot{\Sigma}.Verify(pk_O, \sigma, (id, params, data)) = \dot{1} \wedge \\ data \neq prog_{encl}.\mathbf{Resume}(id, params)\end{array}\right] \leq negl(\lambda),$$

其中安全参数为 λ.

6.2 基础设施变更

在构建安全预言机的过程中，许多挑战都来源于一个事实，即现有的数据来源不会在其提供的数据上进行电子签名标记。如果它们能够做到这一点，那么预言机就不需要受到用户信任才能不篡改数据。超文本传输安全协议不能进行数据签名，

但有一个要求服务器拥有原则上支持数据签名证书的潜在公共密钥基础设施（PKI）。

这一观察是 TLS 扩展 TLS-N 的基础。TLS-N 允许 HTTPS 服务器在他们与用户的部分会话上进行签名标记。这一签名行为的选择性提供了其他有用的功能，如使客户能够排除签署记录，并由此维持他们用来连接服务器的用户证书的保密性（如用户的登录密码）。

我们相信 TLS-N 这类的基础设施变更是支持预言机安全的潜在方式。然而，由于下列限制，他们可能需要与其他技术，如 SGX 共同使用：

1. 基础设施修改：令人遗憾地是，除非 TLS-N 成为标准，数据来源必须专门使用它才能使用户获益。然而，极少有数据来源能在短期内做到这一点。
2. 聚合与计算：TLS-N 无法支持对来自数据源的数据进行聚合或其他形式的受信计算，因此仍需一些受信机制才能完成这些任务。
3. 成本：相比简单的签名核验而言，核验 TLS-N 签名的数据会产生相对高昂的链上成本。
4. 保密性：TLS-N 无法支持用户证书或查询的带外保密性管理，而是要求用户自行查询数据来源。比如说，机密的航班信息无法储存在智能合约上以供网页的自动化保密查询。

6.3 链下计算

预言机的某些应用，如使用依赖证书的应用程序编程接口，需要预言机做的远远不止传输数据这么简单。它还需要预言机管理证书、登入账户获取数据等等。确实，在按 TC 方式由 SGX 支持的系统和零知识证明[21]的帮助下产生了真正可信和保密的预言机后，预言机和智能合约之间的界限就会变得流动。

ChainLink 已经支持基于正则表达式语言使用户自由指定处理链下数据的查询。然而， 我们的长期战略是创造一个使预言机成为大多数智能合约使用的关键链下计算资源的世界。我们坚信通过在智能合约消化其结果的预言机中建立完全通用的私链计算，我们能做到这一点。如果能在高度安全的条件下实现这一点，正如我们所坚信的那样，那么推动昂贵而敏感的计算逻辑进入预言机则会带来更优化的保密习惯、更低的合约执行成本以及更加灵活的建构。

7 现有的预言机解决方案

ChainLink 是用来填补智能合约系统中新预言机技术的普遍需求的。遗憾的是直到今天，高度安全和灵活的预言机系统仍然非常有限。我们坚信缺少可信的预言机是智能合约发展的重大阻碍。

当今预言机服务最常用的选项是中心化预言机供应商。这一方式的问题在于它创建了一个中心化的控制点，因此无法达到去信任的智能合约所要求的数据篡改阻止标准。有些中心化系统，如[31]，试图通过用公证来解决这一问题以“证明”正确的行为。这种对公证服务的使用令人担忧，因为这些服务存在众多记录在案的问题[37]，而且他们提供的证明并不能在链上被核实，从而导致（潜在循环性的）进一步核实的需要。

另一个提供可信预言机数据的方式是依赖非结构化数据的人工输入。这些“人工输入的预言机”通常被建议在预测市场中使用[17]，[25]，[28]。 通过适当的金融利益并假设理性的经济参与者缺乏参与欺诈行为的动机，这类预言机能够在很大程度上确保提供正确的众包答案。这一方法既去中心化又灵活。由于人工输入的预言机从人类获取回应，它们能够对结构化数据难以发现或很难可靠提取的问题作出回应，比如需要自然语言处理的新闻事件。

然而遗憾的是，由于人类认知行为不仅价格高昂，而且速度缓慢，导致人工输入的预言机非常耗费资源，无法做到实时回应，而只能在有限的时间内处理相当少的问题。我们相信 ChainLink 在结构化数据能够解决的高速、自动化预测市场合约中也会大有可为。

最后一个方案是从源头改变数据的形式。如果数据来源对其提供的数据进行电子签名标记，那么就无需对后续的服务器授权信任。USER-SC 只需检查所收到数据上的签名即可。正如上述提到的，TLS-N 就是这种方法的一个出色代表。遗憾地是，上面也已经提到，TLS-N 的使用需要改变现有的基础设施。

8 结论

我们介绍了 ChainLink，一个能够与区块链外部资源安全互动、为智能合约而生的去中心化预言机网络。我们列出了 ChainLink 的架构，描述了其链上和链下的组成部分。在定义了预言机语境下的安全后，我们对 ChainLink 去中心化的多层次方式进行了详细的描述。我们提出了能够阻止恶意预言机不劳而获行为的全新协议（具体补充协议和安全性证明更改请见附录）。我们也为 ChainLink 如何利用技术和基础设施进步指明了道路，如使用受信任的硬件以及让数据来源对所提供的数据进行电子签名标记。最后，在回顾了现有的预言机解决方案及其各自的短板后，我们指出了现在的形势正需要 ChainLink 这样的系统。

设计原理：在继续完善 ChainLink 时，我们会力求突出以下核心价值观：

- 安全和开放系统的去中心化。去中心化不仅仅是区块链防篡改特性的基石，也是区块链无需许可属性的基础。通过继续构造去中心化的系统，我们的目标是进一步促进区块链生态中无需许可的发展。我们相信去中心化是发展全球繁荣、长期可持续的区块链生态系统必不可缺的组成部分。
- 简单、灵活的系统设计的模块性。我们欣赏那种专注于解决一个问题的小工具。简单的组成部分更易掌握，也能更加安全地融入更大的系统。我们相信模块性不仅能够使可升级的系统成为可能，也能够进一步促进去中心化的发展。当 ChainLink 的关键部分过于依赖或仅由少数参与方管理时，我们会致力于设计鼓励竞争的生态系统。
- 安全、可扩展系统的开源。ChainLink 的存在离不开众多开源项目提供的基础。我们重视开发者社区，并将继续通过开源的方法完善 ChainLink，从而为社区做出更多贡献。我们计划继续与开发者、学者和安全专家保持交流并进行同行评审。我们鼓励安全测试、审计和正式证明，这一切都是为了创造一个有活力而安全的平台，以为未来创新奠定基础。

在这些关键性原则的指导下，我们期待通过将预言机变为区块链生态的安全基石，继续扩大区块链和智能合约在全世界的使用范围与影响力。

参考文献

[1]Parity. *The ftulti-sig hack: A postmortem*. https://blog.ethcore.io/the-multi-sig-hack-a-postmortem/. 20 July 2017.

[2]Gun Sirer. *Cross-Chain Replay Attacks*. Hacking, Distributed blog. 17 July 2016.

[3]Adi Shamir. "How to share a secret". In: *Communications of the ACft* 22.11 (1979), pp. 612–613.

[4]Claus-Peter Schnorr. "Efficient signature generation by smart cards". In: *Journal of cryptology* 4.3 (1991), pp. 161–174.

[5]Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, et al. "Secure distributed key generation for discrete-log based cryptosystems". In: *Eurocrypt*. Vol. 99. Springer. 1999, pp. 295–310.

[6]R. Canetti. " Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *FOCS*. 2001.

[7]Ran Canetti. " Universally composable security: A new paradigm for cryptographic protocols". In: *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*. IEEE. 2001, pp. 136–145.

[8]Douglas R Stinson and Reto Strobl. "Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates". In: *ACISP*. Vol. 1. Springer. 2001, pp. 417–434.

[9]John R Douceur. " The sybil attack". In: *International Workshop on Peer-to-Peer Systems*. Springer. 2002, pp. 251–260.

[10]Aniket Kate and Ian Goldberg. " Distributed key generation for the internet". In: *Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE International Conference on*. IEEE. 2009, pp. 119–128.

[11]Claudio Orlandi. " Is multiparty computation any good in practice?" In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE. 2011, pp. 5848–5851.

[12]Ittai Anati, Shay Gueron, Simon Johnson, et al. "Innovative technology for CPU based attestation and sealing". In: *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*. Vol. 13. 2013. url: https://software.intel.com/en-us/articles/innovative-technology-for-cpu-based-attestation-and-sealing (visited on 05/23/2016).

[13]Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, et al. " Using

Innovative Instructions to Create Trustworthy Software Solutions". In: *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Se- curity and Privacy*. HASP '13. Tel-Aviv, Israel: ACM, 2013, 11:1–11:1. isbn: 978-1-4503-2118-1. doi: 10.1145/2487726.2488370. url: http://doi.acm.org/10.1145/2487726.2488370.

[14]Frank McKeen, Ilya Alexandrovich, Alex Berenzon, et al. " Innovative instructions and software model for isolated execution." In: *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*. 2013, p. 10. url: http://css.csail.mit.edu/6.858/2015/readings/intel-sgx.pdf (visited on 05/23/2016).

[15]Intel. *Intel Software Guard Extensions Programming Reference*. 2014. (Visited on 05/23/2016).

[16]Gavin Wood. " Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum Project Yellow Paper* (2014).

[17]Jack Peterson and Joseph Krug. " Augur: a decentralized, open-source platform for prediction markets". In: *arXiv preprint arXiv:1501.01042* (2015).

[18]Victor Costan and Srinivas Devadas. " Intel SGX Explained". In: *Cryptology ePrint Archive* (2016). url: https://eprint.iacr.org/2016/086.pdf (visited on 05/24/2016).

[19]Victor Costan, Ilia A Lebedev, and Srinivas Devadas. "Sanctum: Minimal Hardware Extensions for Strong Software Isolation." In: *USENIX Security Symposium*. 2016, pp. 857–874.

[20]Kevin Delmolino, Mitchell Arnett, Ahmed Kosba, et al. " Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2016, pp. 79–94.

[21]Ahmed Kosba, Andrew Miller, Elaine Shi, et al. "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts". In: *S&P'16*. IEEE. 2016.

[22]Loi Luu, Duc-Hiep Chu, Hrishi Olickel, et al. " Making smart contracts smarter".
In: *Proceedings of the 2016 ACft SIGSAC Conference on Computer and Com- munications Security*. ACM. 2016, pp. 254–269.

[23]Bill Marino and Ari Juels. " Setting standards for altering and undoing smart contracts". In: *International Symposium on Rules and Rule ftarkup Languages for the Semantic Web*. Springer. 2016, pp. 151–166.

[24]Fan Zhang, Ethan Cecchetti, Kyle Croman, et al. " Town Crier: An

authenti- cated data feed for smart contracts". In: *Proceedings of the 2016 ACft SIGSAC Conference on Computer and Communications Security*. ACM. 2016, pp. 270– 282.

[25] *Augur project page*. https://augur.net. 2017.

[26] *CSA Cloud Controls ftatrix*. URL: https://cloudsecurityalliance.org/group/cloud- controls-matrix. 2017.

[27]Mark Flood and Oliver Goodenough. *Contract as Automaton: The Computational Representation of Financial Agreements*. https://www.financialresearch. gov/ working- papers/ files/ OFRwp - 2015 - 04 _ Contract- as- Automaton- The-Computational-Representation-of-Financial-Agreements.pdf. Office of Financial Research, 2017.

[28] *Gnosis project page*. https://gnosis.pm. 2017.

[29] *Hyperledger Sawtooth*. https://intelledger.github.io/introduction.html.

2017. [30]Abhiram Kothapalli, Andrew Miller, and Nikita Borisov.

"SmartCast: An Incentive Compatible Consensus Protocol Using Smart Contracts". In: *Financial Cryptography and Data Security (FC)*. 2017.

[31] *Oraclize project page*. http://www.oraclize.it. 2017.

[32]Rafael Pass, Elaine Shi, and Florian Tramer. "Formal abstractions for attested execution secure processors". In: *Eurocrypt*. Springer. 2017, pp. 260–289.

[33] *Town Crier Ethereum service*. http://www.town-crier.org/. 2017.

[34]Florian Tramer, Fan Zhang, Huang Lin, et al. "Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge". In: *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*. IEEE. 2017, pp. 19–34.

[35]Vitalik Buterin et al. *Ethereum white paper*. https://github.com/ethereum/wiki/wiki/White-Paper.

[36] *JSON Schema*. http://json-schema.org/.

[37]Hubert Ritzdorf, Karl Wüst, Arthur Gervais, et al. *TLS-N: Non-repudiation over TLS Enabling Ubiquitous Content Signing for Disintermediation*. IACR ePrint report 2017/578. URL: https://eprint.iacr.org/2017/578.

免责声明

† 艾利•约尔斯系康奈尔大学纽约理工校区雅克布学院的教授。他以智能合约 ChainLink 公司顾问的身份参与编撰了此白皮书，并在该公司中有相关金融利益。

* 此白皮书由注册于英属维尔京群岛的智能合约 ChainLink 公司（以下简称为 SCCL）为支持 ChainLink 平台而提供。安全资产交易所公司（以下简称为 SAE）通过 SmartContract.com 为 SCCL 提供管理、技术及其他方面的支持，包括支持 SCCL 进行的 LINK 代币销售，并为此接受 SCCL 提供的相应报酬。使用区块链技术的技术、社会与商业结构均在不断发展，并将在可预见的未来继续升级。相应地，此白皮书中所提及的规划、战略与实施细节亦会随之更新，且存在永不付诸执行的可能。SCCL 和 SAE 保留一切开发和制定与 ChainLink 平台相关的补充规划、战略和实施细节的权利，保留一切开发和制定与 ChainLink 平台相关的新的规划、战略和实施细节的权利。

** LINK 代币由 SCCL 依照 https://link.smartcontract.com/terms 中所公布的代币销售条款进行销售。具体细节请参照条款原文。LINK 代币不属于证券、投资或货币，且从未以上述形式进行销售或营销。此外：参与购买代币可能产生重大的技术和系统风险；该销售不对居住或国籍隶属于美国及加拿大两国的个人开放；代币销售期、销售时长、销售定价及其他销售相关规定均有可能变更，这一点已在代币销售条款中予以说明。LINK 代币销售涉及各类已知和未知的风险、不确定性及可能导致 LINK 代币的实际功能、用途和可用性与 SCCL 发布的条款中所声明或暗示的未来预期结果、用法、功能性或用途显著不同的其他因素，特此声明。