Chainlink
去中心化的预言机网络

作者：Steve Ellis、Ari Juels、Sergey Nazarov
2017 年 9 月 4 日(第一版)

摘 要

智能合约将替代传统法律合约和自动执行的中心化数字合约，颠覆众多行业。在传统模式中，合约一方通常需要手动验证并执行合约；抑或依靠程序自动获取并更新信息。然而，由于区块链具有特殊的底层共识协议，链上智能合约无法与外部系统交互。

如今出现了一种新的解决方案，即“预言机”。预言机可将链上智能合约连接至链下世界。现存的预言机都采用中心化的服务模式，使用该服务的智能合约可建立单点失效机制，使其相较传统的中心化数字合约具有更高的安全性。

本白皮书介绍了去中心化的预言机网络 Chainlink，并详细阐述了 Chainlink 为智能合约连接链下数据提供的链上模块及其节点网络的底层软件；Chainlink 简单的链上智能合约聚合系统以及更高效的链下共识机制；Chainlink 的声誉系统和安全监控服务，帮助用户理性选择服务提供商，即使在不利环境中也能获得稳健可靠的服务。除此之外，本白皮书还定义了预言机的最高标准，并将其作为 Chainlink 安全战略的指导原则；畅想了未来技术发展和完善空间，包括丰富预言机编程功能、改善数据源基础架构以及提升智能合约的保密性。

目录

一. 背景介绍

智能合约是在区块链等去中心化的基础架构中执行的应用，其具有防篡改性，即包括合约创建者在内的任何一方都无法篡改代码或干预代码执行。一直以来，传统合约的代码都是在中心化的系统中运行的，因此有权限的一方有可能将合约篡改、终止或甚至删除。相比之下，智能合约可保障合约顺利执行，约束合约各方履行各自义务。这就建立了一种全新且稳健的信任关系，无须依赖合约各方彼此间的信任。由于智能合约是自动验证和执行的（注：即具有防篡改性，上文已解释过，这里不再赘述），因此可以很好地执行并管理数字合约。

然而，随着智能合约这种新的信任模式的产生，出现了一个新的技术挑战，那就是连接性。大多数有价值的[27][1] 智能合约应用都需要获取来自关键数据源的链下数据，特别是实时数据和 API 数据，这些数据都不保存在区块链上。由于区块链受自身特殊的共识机制限制，因此无法直接获取这些关键的链下数据。

我们针对智能合约的连接性问题提出了一个解决方案，那就是安全的预言机网络 Chainlink。Chainlink 与其他预言机解决方案的不同之处在于其网络是完全去中心化的。这种去中心化的模式极大降低了合约方彼此间的信任需求，将智能合约防篡改的特性扩展到了数据从 API 端到智能合约端的整个传输过程。让智能合约具有外部连接性，意味着其能够与链下资源交互，这是智能合约替代传统数字合约的先决条件。

如今，大部分传统的数字合约都使用外部数据验证履约情况，并将数据输出至外部系统。如果要用智能合约代替这类传统的数字合约，则需要其数据输入和输出都具有极高的确定性。以下是下一代智能合约及其数据需求的潜在案例：

- 债券、利率衍生品等各类证券智能合约可接入 API 获取市场价格和市场参考数据（如：利率）。
- 保险智能合约可利用物联网数据进行调查取证，比如：在违约时仓库的磁铁门是否锁上了？公司防火墙当时是否正常运行？你购买了航班保险的航班是否按时到达？
- 贸易金融智能合约可利用航运物流 GPS 数据、供应链 ERP 数据以及货物清关数据，以确保合约执行。

上述几个案例中还存在一个常见问题，那就是智能合约无法将数据输出至链下系统。这通常需要将支付消息发送至传统的中心化基础架构，即银行、PayPal 等用户平常会使用的链下系统。Chainlink 能够将智能合约数据安全传输至 API 和传统链下系统，因此既可以保障智能合约的防篡改性，又可以实现外部连接性。

---

1. 如今，以太坊上智能合约的主要用途是通证管理，这也是大多数智能合约网络最常见的功能。我们认为这种情况是由于预言机服务水平不足导致的，这也是 Chainlink 着重想要解决的问题。

白皮书大纲

本白皮书的第二章讨论了 Chainlink 的基础架构；第三章明确定义了预言机的安全标准；第四章详细阐述了 Chainlink 在预言机和数据源层面分别的去中心化/分布式模式；第五章讨论了 Chainlink 的四大安全服务以及 LINK 通证的意义；第六章分享了 Chainlink 的长期发展战略，其中包括提升保密性、使用可信硬件、完善基础架构以及预言机的总体可编程性；第七章简要介绍了其他的预言机设计方案；最后第八章则阐述了 Chainlink 的设计原则和理念。

二．架构总览

Chainlink 的核心目标是嫁接链上和链下环境。我们将在下文具体阐述 Chainlink 各个模块的架构。Chainlink 最初将在以太坊进行开发[16][35]，但我们的目标是支持所有主流的智能合约网络实现链下和跨链交互。Chainlink 链上和链下部分都采用了模块化的设计理念，系统中每个模块都可以升级，因此一旦出现了更好的技术和实现方式，便可轻松替换原有模块。

2.1 链上架构

Chainlink 节点作为预言机，针对由智能合约端亲自发起的或第三方委托发起的数据请求返回结果，我们将这类数据请求称为“请求合约”，并用 USER-SC 表示。Chainlink 与请求合约交互的接口本身也是一个链上合约，我们用 CHAINLINK-SC 表示。

Chainlink 在 CHAINLINK-SC 后方还设置了一个链上模块，这个模块由三大智能合约组成，即“声誉合约”、“订单匹配合约”和“聚合合约”。声誉合约记录预言机服务提供商的历史服务水平。订单匹配合约中包含一个服务水平协议，记录协议中的具体参数，并收集来自预言机服务商的竞标，然后再使用声誉合约从多个服务商中选出优胜者，并最终敲定预言机的服务水平协议。聚合合约收集预言机服务商返回的结果，聚合所有数据，计算出一个最终结果。另外，它还将预言机服务商的数据质量返回到声誉合约中。Chainlink 智能合约采用了模块化的设计理念，用户可以随意配置或替换模块。链上工作流程分成三步，即 1）预言机选择；2）数据上报；3）数据聚合。

预言机选择

用户在购买预言机服务时会在服务水平协议中具体阐明服务要求。服务水平协议包括数据参数以及所需预言机数量等各种内容。另外，用户还会确定对声誉合约和聚合合约的要求。

有了链上声誉系统和可靠的历史服务记录，用户能够通过链下分类信息平台手动分类、过滤并选择预言机服务。我们的目标是让 Chainlink 成为这样一个预言机分类信息平台，收集与 Chainlink 相关的所有日志，并验证平台上预言机合约的源代码。本白皮书将在第五章详细阐述 Chainlink 的分类信息服务和声誉系统。用于生成列表的数据来自于区块链，因此也可以建立其他的预言机分类信息服务。预言机用户可将服务水平需求发送至链下预言机，达成协议后再最终敲定链上协议。

手动选择预言机并不适用于所有场景。比如一个智能合约可能需要根据数据需求量动态调整所需的预言机服务，所以就需要通过自动化手段来解决这一问题，并增强可用性。因此，Chainlink 使用订单匹配合约自动匹配预言机。

一旦用户明确了服务水平需求，他们就可以将服务水平协议发送至订单匹配合约，而无须亲自联系预言机。需求发送到订单匹配合约后，就会生成一条日志，预言机服务商可根据自身能力和服务范围对日志进行查看和过滤。之后，Chainlink 节点将决定是否对任务进行竞标，智能合约只接受满足服务水平协议要求的节点竞标。预言机服务商竞标后，它们就会按服务水平协议的内容提供服务，如出现任何违约行为，则会被没收保证金。

竞标将持续一段时间，一旦达到一定数量且竞标时间结束，则会从所有候选者中按要求选出一定数量的预言机。未被选中的预言机将被退回它们在竞标过程中缴纳的保证金，最终将敲定服务水平协议。最终版的服务水平协议敲定后，便会生成一条日志，对被选中的预言机发送通知。然后，预言机会根据服务水平协议的具体内容交付服务。

数据上报

一旦生成了新的预言机记录，链下预言机就会开始执行合约并将数据发送至链上。第二章第 2 节和第四章将详细阐述链下交互模式。

数据聚合

一旦预言机将返回结果上传至预言机合约，这些数据将被传输至聚合合约。聚合合约收集所有预言机返回的结果，并计算出一个加权数值。每台预言机返回的数据质量都将被发送到声誉合约中。最后，加权数值将被发送至 USER-SC 具体的合约函数中。

并非所有类型的数据和应用都需要侦测返回的异常或错误值。举个例子，如果返回的值是数字，取平均值之前侦测并拒绝异常值是非常必要的，但对于布尔逻辑来说则并非如此。因此，不存在一个万能的聚合合约，取而代之的是一个由用户定制的且可配置的合约地址。Chainlink 会推出标准化的聚合合约套件，但与此同时，在与标准化计算方法兼容的前提下也可以开发定制化合约。

2.2 链下架构

Chainlink 的链下架构最初包括以太坊上的预言机节点网络，我们最终希望能支持所有领先的智能合约网络。这些独立节点获取链下数据，下文中我们将详细阐述如何在几个可行的共识机制中选择一个将所有节点的返回结果聚合成单一数据，并返回至请求合约 USER-SC。Chainlink 节点实现了标准开源的核心功能，其中包括标准化的区块链交互、调度以及连接共同的链下资源。节点运营商可以选择添加软件扩展包，即外部适配器，以提供专业的链下服务。Chainlink 节点已被部署到了公链和企业联盟链。让节点实现去中心化运营是 Chainlink 网络不断前进的动力。

Chainlink 核心软件

节点的核心软件负责与区块链交互、调度任务并平衡各个外部服务的工作量。

Chainlink 节点完成的工作被称为“任务”(assignment)。每个任务都可分成更小的单位，即“子任务”（subtask），子任务以流水线的方式进行。每个子任务都有具体的职责，前一个子任务完成工作后会将结果传递给下一个子任务，并得出最终结果。Chainlink 节点软件内置了几个子任务，其中包括 HTTP 请求、JSON 解析以及转换成不同的区块链格式。

外部适配器

除了内置子任务外，还可以通过创建适配器定制子任务。适配器是配置了最小化 REST API 的外部服务。配置了服务型适配器后，任何编程语言开发的程序都可以通过加上一个中间 API 而轻松实现。同样地，与复杂的分步 API 交互也可以得到简化，通过多个参数化的子任务来实现。

子任务模式

我们预期会出现许多开源的适配器，因此服务可以由各个社区成员负责运营和审计。由于开发者开发出了许多不同的适配器，因此保障适配器之间互相兼容至关重要。

Chainlink 目前基于 JSON 格式[36]运行，规范每个适配器需要什么数据输入，以及应该如何格式化。同样地，适配器也会明确数据输出格式以及每个子任务的数据输出格式。

三．预言机的安全性

要理解 Chainlink 的安全架构，我们必须先讨论安全性的重要性，以及安全意味着什么。

预言机为什么一定要保证安全？

回到我们在第一章举过的那个简单的例子：如果证券智能合约收到错误的数据，那么它就会付款给错误的一方；如果保险智能合约的数据被受保人篡改，那么就可能出现骗保行为；如果贸易金融智能合约的 GPS 数据在传输过程中被篡改，那么在货物还没送到之前就可能冒然付款。

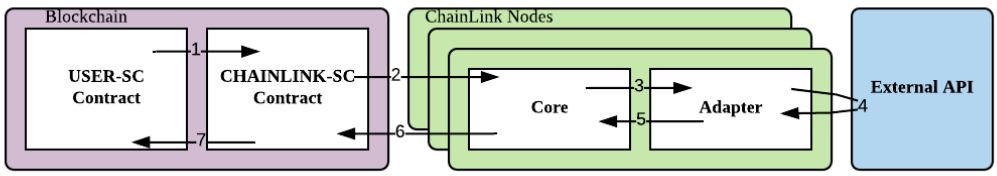

图 1：Chainlink 工作流程示意图：

1）USER-SC 发起链上请求；2）CHAINLINK-SC 为预言机记录事件；3）Chainlink 核心软件收到事件记录并向适配器发送任务；4）Chainlink 适配器执行任务，向外部 API 请求数据；5）Chainlink 适配器处理返回数据并返回至核心软件；6）Chainlink 核心软件将数据传回 CHAINLINK-SC；7）CHAINLINK-SC 将数据聚合成单一数据，并返回至 USER-SC。

通常来说，一个功能正常的区块链，其账本和公告板功能可以在很大程度上保障安全性。用户在区块链上可以可靠地验证交易并避免数据被篡改。对用户来说，区块链就是一个可信的第三方（注：我们会在下文详细讨论这个概念）。对区块链提供支持的预言机服务也必须达到与区块链同等水平的安全性。也就是说，预言机也必须为用户提供有效且可信的第三方服务，以及非常准确及时地返回结果。任何一个系统中安全性最弱的一环就决定了系统的整体安全，因此为了维持区块链的可信水平，就必须要配备同样可信的预言机。

理想型预言机的定义

要论证预言机的安全性，我们首先要给它下个定义。关于预言机安全性的指导性论证最初源于以下思想实验。想象现在有一个可信的第三方（TTP），即理想的实体或功能，总是能忠实执行所有指令，现在由这么一个第三方负责运行一台预言机。我们将这台预言机称作“ORACLE”（注：oracle 是预言机的英文名，我们将这个词全部大写，特指完全被用户信任的完美预言机）。假设这个可信的第三方从一个完全可信的数据源 Src 获取数据。如果是通过这台完美的 ORACLE 预言机提供服务，那么我们会给它下达什么指令呢？

为了保证完整性，即“真实性”[24]，我们就让 ORACLE 预言机执行以下任务：

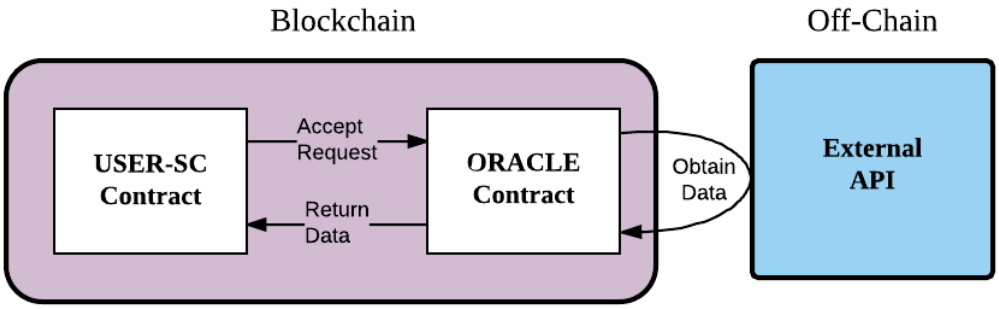

图 2：理想型预言机 ORACLE 的任务分成以下几个步骤：

1）接受数据请求；2）获取数据；3）返回结果。除此之外，为了保护数据请求的隐私，在解密数据时 ORACLE 除了数据源 Src 以外绝对不会使用或向任何一方披露数据内容。

1. 接受请求： 从智能合约 USER-SC 获得数据请求 Req=(Src, t, q)，明确目标数据源 Src、时间或时间范围 t、以及请求 q；

2. 获取数据：将请求 q 在时间 t 发送至 Src；

3. 返回结果：收到返回的数据 a，将 a 返回至智能合约。

这几个简单的指令快速、有效且有意义地勾勒出了安全的概念。从根本上来说，这些指令决定了 ORACLE 预言机是 Src 和 USER-SC 之间可信的桥梁。[2] 举个例子，如果 Src=http://www.FountOfKnowledge.com、t=下午四点、q="ticker INTC 的价格"，那么 ORACLE 预言机就会保证为 USER-SC 提供下午 4 点时来自数据源 http://www.FountOfKnowledge.com 完全准确的 INTC 价格。

保密性是预言机另一个关键功能。USER-SC 在区块链上将数据请求发送至 ORACLE 时，数据请求是公开的。很多情况下数据请求本身比较敏感，因此公开会产生负面影响。假如 USER-SC 是一份航班保险智能合约，向 ORACLE 发送了一个数据请求 Req，请求某个用户的航班信息（q="以太航空航班 338"），那么结果就是全世界都会看到这个用户的航班计划。如果 USER-SC 是一份金融交易智能合约，数据请求可能会泄露用户的交易记录和资产组合。这样的例子不胜枚举。

为了保护数据请求的隐私，我们需要对其进行加密，公钥由 ORACLE 保存。为了继续发挥 ORACLE 可信第三方的作用，我们可以给 ORACLE 设定一个信息流限制：

解密数据请求 Req 后，除了 Src 以外，不能对任何一方披露或使用 Req 内容。

预言机还有其他的重要功能，比如可用性，这也是经典的 CIA（即保密性-完整性-可用性）三角中最后一个角。当然，一个理想型的 ORACLE 预言机是永远不会宕机的。可用性还包括抗审查性等更加隐晦的功能，即一个诚实的 ORACLE 预言机不会拒绝为任何一个智能合约服务。

可信第三方的概念等同于某些模型中证明加密协议安全性的最佳模式[7]。我们也可以用同样的逻辑打造区块链，用可信第三方的概念完美地维持公告板机制。可信第三方得到的指令包括接受、验证并序列化交易，将它们永久保存在公告板上，公告板是一种 append-only 的数据结构。

理想型的预言机（ORACLE）为什么难以实现？

当然，世界上并不存在完全值得信任的数据源 Src。数据会由于网站漏洞、服务提供商作弊或无心的差错等各种原因被善意或恶意篡改。

如果 Src 不值得信任，即使 ORACLE 预言机如上文中的可信第三方一样完美运行，仍无法完全符合我们的安全要求。如果数据源 Src 出现问题，则预言机返回的结果 a 也会出错。如果 Intel 的真实价格是 40 美元，而 http://www.FountOfKnowledge.com 网站上把价格误报成 50 美元，那么 ORACLE 预言机就会将错误值 a=50 美元发送至 USER-SC。如果使用单一数据源 Src，这个问题则无法避免。ORACLE 预言机不可能判断数据源 Src 返回的数据是否正确。

---

2. 当然这里省略了许多细节。ORACLE 预言机通过安全通道（即不可篡改的通道）与 USER-SC 和数据源 Src 交互。（如果 Src 是 web 服务器，就需要 TLS。）ORACLE 预言机要和 USER-SC 交互就必须正确连接到区块链并对 a 进行数字签名。

当然，还有一个问题是我们这里设定的可信第三方预言机 ORACLE 只是一个抽象概念。没有一个服务提供商是百分之百可信的。就算是本身诚实的服务提供商也有可能被黑客攻击或出现风险漏洞。因此，用户或智能合约无法完全相信 ORACLE 预言机会忠实执行指令。

上文通过理想型预言机 ORACLE 的例子讨论了安全的概念。我们希望 Chainlink 能够尽量接近例子中的理想型预言机，在真实的场景中与现实世界的系统交互。下面，我们将详细解释 Chainlink 的工作原理。

为简化叙述，我们将所有 Chainlink 智能合约都称作 CHAINLINK-SC，即 Chainlink 完整的链上功能，而不仅是与请求合约的交互界面。因此，我们将系统架构中多个不同的智能合约抽象集合成一个合约。

四．Chainlink 去中心化方案详解

Chainlink 提出三种互补的方案来避免问题节点的出现，即：1）分布式数据源；2）分布式预言机；3）可信硬件。本章我们先来讨论前两种方案，其中包括去中心化。在第六章中，我们将讨论可信硬件的长期战略，可信硬件是一个与众不同的补充方案。

4.1 分布式数据源

应对单一数据源 Src 风险漏洞的一个简单方法就是改为从多个数据源获取数据，即采用分布式的数据源模式。可信预言机 ORACLE 可以向 Src_1,Src_2,.....,Src_k 数据源集合请求数据，获得 a_1, a_2,.....a_k 结果集合，并聚合成单一数据 $A=agg(a_1,a_2,.....,a_k)$。预言机可以以各种方式来完成任务，比如多数投票。如果大多数数据源返回同一个值 a，那么 agg 函数就返回 a；否则则返回错误。这样的话，如果大多数（即大于 k/2）的数据源都能正常运行，那么 ORACLE 预言机将永远都能返回正确的值 a。

还有许多其他的 agg 函数可以侦测错误数据或处理一段时间内的数据波动(比如：股票价格），并取剩余数据的平均值。

当然，还有一种情况是不同数据源之间的错误是互相关联的，这就会削弱聚合数据的确定性。如果网站 Src_1=EchoEcho.com 从 Src_2=TheHorseMouth.com 获取数据，那么 Src_2 的错误就会导致 Src_1 也出现错误。此外，数据源之间还会出现更难以被察觉到的关联性错误。Chainlink 致力于解决这类问题，对数据源的独立性进行映射和报告，让预言机和用户可以轻松规避此类关联性错误。

4.2 分布式预言机

刚才说了分布式数据源，其实理想型预言机 ORACLE 本身也可以是一个分布式系统。也就是说，网络中并非只有一个预言机节点 O，而是有 n 个不同的预言机节点 $\{O_1,O_2,.....,O_n\}$。每个预言机 O_i 都会有自己的数据源集合，不同预言机的数据源可能会重叠也可能不会。O_i 从各个数据源获取并聚合数据，并将自己的聚合数据 A_i 发送至请求 Req。

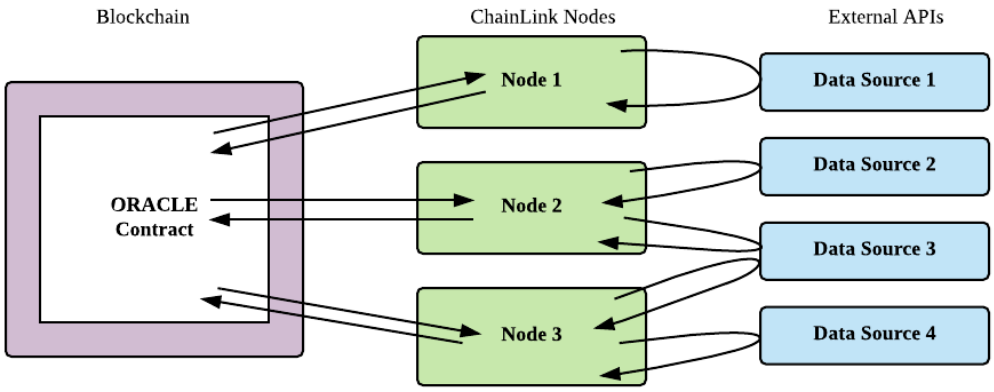

图 3：数据请求在预言机和数据源层面去中心化，分布模式如图所示。

其中一些预言机可能出现问题，因此所有预言机返回的数据 A1,A2,.....An 需要以可信的方式被聚合成一个单一且权威的值 A。但由于可能出现问题预言机，Chainlink 怎么样聚合数据呢？

初步解决方案：合约内聚合

Chainlink 最初提出的解决方案是简单的“合约内聚合”模式。CHAINLINK-SC（注：仍然指代 Chainlink 所有链上合约）自行聚合预言机返回的数据（注：事实上是 CHAINLINK-SC 调用聚合合约，但为了简化概念，我们假设一个完整的 Chainlink 合约包含这两个模块）。也就是说，CHAINLINK-SC 将调用 Agg 函数计算 $A=Agg(A_1,A_2,.....A_n)$（与上文提到的 agg 函数类似），并将结果 A 返回至 USER-SC。

这个方法在 n 值较小的情况下很实用，而且有几大优势：

- 概念简单：虽然采用分布式预言机模型，但 CHAINLINK-SC 仍统一执行 Agg 函数运算聚合数据。
- 可信度高：由于 CHAINLINK-SC 的代码是开源的，因此其正确行为可以得到验证（注：CHAINLINK-SC 代码相对较少也比较简单）。另外，CHAINLINK-SC 的执行在链上完全公开透明，因此用户（即 USER-SC 的创建者）可以对 CHAINLINK-SC 高度信任。
- 灵活性高：CHAINLINK-SC 可以实现最优聚合函数 Agg，这个函数可以是多数投票制，也可以是取平均值等。

这个方案虽然很简单，但是却引出了一个值得深思的技术性问题，即“吃空饷”（freeloading）。预言机 O_z 通过作弊看到另一台预言机 O_i 反馈的结果 A_i，然后决定抄袭它的答案。这样一来，预言机 O_z 就不用花钱向数据源请求数据，而数据源是按照请求次数收费的。吃空饷现象会削弱数据源的多样性，也会打击预言机快速响应的积极性（注：因为响应速度慢的预言机可以免费抄袭别人的答案），因此会影响整体安全性。

我们针对这个问题提出了一个广为人知的解决方案，那就是建立先提交后解密（commit/reveal）的机制。在第一轮，预言机将结果以加密形式返回 CHAINLINK-SC，CHAINLINK-SC 收到合法数量的结果后，就会开启第二轮，预言机将在这一轮解密数据。

算法 1 是一个简单的顺序协议，用 3f+1 个节点保证可用性。它使用先提交后解密的机制避免吃空饷行为。预言机的提交被取消，只有在所有结果都提交后才会被公开，因此无法抄袭。

链上协议可以用区块时间来实现同步协议。而 Chainlink 上的预言机节点由于从不同数据源获取数据，响应时间可能会出现较大差异，而且由于节点在以太坊上使用的 gas 价格不同，取消提交时间也会有所不同。因此为了尽可能提高协议的响应速度，算法 1 被设计成异步协议。

其中，$Commit_r(A)$指代提交结果 A，目击者是 r，SID 指代一组合法会话 ID（session id）。协议假设各方之间存在认证通道。

那么显而易见，算法 1 将成功完成运算。假设一共有 3f+1 个节点，最多 f 个节点有问题，那么至少 2f+1 个节点将在第四步提交结果。在这些提交的结果中，最多有 f 个数据会来自出问题的节点，因此至少 f+1 个数据会来自诚实的节点。所有被提交的结果最终都会被取消。

另外，A 是算法 1 的正确答案。针对单一值 A 的 f+1 个提交取消中，至少有一个肯定来自诚实节点。

调用算法 1 在合约中聚合数据将是 Chainlink 短期内主要采取的方案。初步的实现方案将在这个算法基础上进一步优化并实现并发。我们的长期策略是开发一个更加复杂的 OCA（即链下聚合）协议，具体算法是附件 A 中的算法 2 和算法 3。OCA 是一种链下聚合协议，可将链上交易成本降到最低。这个协议还包括向预言机节点付款的功能，并确保抄袭者拿不到一分钱。

算法 1　InChainAgg({Oi}ni=1)　（CHAINLINK-SC 代码）

---

1. 等待从 USER-SC 收到 Req
2. sid ←$ SID
3. 广播(request,sid)
4. 等待从 O_i 收到 2f+1 条信息的集合 C(commit,ci = Commitri (Ai),sid)
5. 广播(committed,sid)
6. 等待收到 f+1 个合法取消提交的集合 D; 集合中所有元素的值 A_i 都等于某个值 A
7. 将(Answer, A, sid)发送至 USER-SC

中长期战略：链下聚合

合约内聚合存在一个关键问题，那就是成本。预言机 $O_{(n)}$提交的结果在链上处理（注：指 $A_1,A_2,.....A_n$ 的提交和解密）会产生费用。在联盟链中，这项开支是可以承受的。但

在以太坊这样的公链上会收取链上交易费用，如果 n 值很大，那么费用将难以承受。一种成本效益更高的方式是在链下聚合预言机提交的结果，并将最后聚合成的单一数据发送至 CHAINLINK-SC。我们将这个方案称为链下聚合，这是我们的中长期战略。

在节点可能出现问题的情况下达成共识得出结果值 A，这其实很接近区块链自身达成共识的机制。假设有一组预先设定的预言机，一般人可能会考虑使用经典的拜占庭容错（CFT）共识算法来计算结果 A。然而，经典的拜占庭容错协议的目标是最终所有诚实节点都能储存同样的值，也就是说在区块链中，所有节点都储存同一个新产生的区块。而 Chainlink 的预言机目标则稍有不同。我们想要确保 CHAINLINK-SC（以及 USER-SC）能够获得聚合结果 $A=Agg(A_1,A_2,.....A_n)$，无须参与共识协议也无须从多台预言机收到结果。另外，“吃空饷”的问题仍然需要解决。

Chainlink 提议采用基于门限签名的简单协议。现存许多签名机制都可以实现这一功能，但使用 Schnorr 签名机制是其中最简单的方案[4]。用 Schnorr 签名机制，预言机拥有一个公钥集合 pk 和对应的私钥集合 sk，以(t,n)门限的方式分配给 $O_1,O_2.....,O_n$[3]。这里分配是指每个节点 O_i 都有一对专属的私钥和公钥（sk_i,pk_i）。O_i 可以生成部分签名 σi = Sigski [Ai]，可以用 pk_i 进行验证。

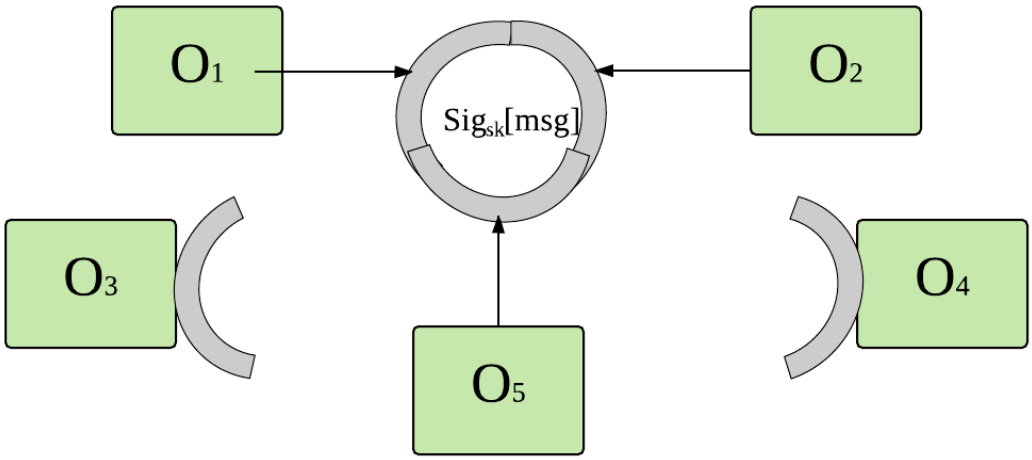

图 4：任意 n/2+1 台预言机可以生成 Sigsk[A]

这个方案最大的特点是，t 台预言机的集合针对同一个值 A 分别生成部分签名，这些签名被聚合成一个完整的合法签名 Σ = Sigsk[A] 。然而，t-1 台预言机的集合就无法针对任何值生成合法签名。因此这个完整签名 Σ 至少由 t 台预言机的部分签名组成。

门限签名只需让 Σ 包含 t 个来自不同节点的独立合法签名即可。门限签名在安全性上与这个方案无异，但它却能大幅提升链上性能，因为它可以将验证 Σ 的工作量和成本降低 t 倍。

也就是说，预言机只需要生成并广播部分签名，直到达到 t 个数量的部分签名，才会开始计算 Σ。然而这个方案同样存在“吃空饷”的问题。因此我们必须保证预言机真实地从数据源获取数据，而非从其他预言机抄袭 A_i。Chainlink 建立了一个经济机制来解决这个问题，即通过 PROVIDER（智能合约）付款给真实获取了数据产生部分门限签名的预言机。

在分布式结构中，很难决定应该付款给哪台预言机。预言机可能在链下互相沟通，CHAINLINK-SC 可能不再是唯一能收到反馈结果的权威性实体，因此无法直接在众多预言机中判断出应该付款给谁。也就是说，PROVIDER 必须从预言机获得证据证明他们是否违规操作，当然其中一些证据可能是不可信的。Chainlink 提出采用类似共识机制的方案，避免 PROVIDER 错误地付款给抄袭的预言机。

附件 A 中详细阐述了 Chainlink 的链下聚合机制以及安全证明。它使用基于门限签名的分布式协议，可以抵御 $f < n/3$ 台预言机"吃空饷"的风险。我们相信这会是一个有潜力的技术发展方向。

五．Chainlink 安全服务

Chainlink 基于上一章提到的多个协议，在网络中出现 f 个问题预言机的情况下仍保障可用性和准确性。另外，我们正在积极研究使用可信硬件（注：第六章中将提到）有效避免问题节点向智能合约提供错误数据。然而，光靠可信硬件可能无法提供充分的安全保障，理由有三。首先，Chainlink 最初开发的网络中不会部署可信硬件。其次，一些用户不相信可信硬件（注：请参见附件 B）。第三，可信硬件只能规避节点违规操作风险，而无法规避节点宕机风险。因此，用户希望能够选择最可靠的预言机，并将 USER-SC 连接到>f 个问题节点的概率降到最低。

为了实现这个目标，Chainlink 开发了四大安全服务，即：验证系统、声誉系统、认证服务和合约升级服务。所有这些服务一开始都由有意愿发布 Chainlink 网络的公司或组织运行，但须严格遵照 Chainlink 去中心化的设计理念。Chainlink 的安全服务无法阻止预言机节点参与或更改预言机反馈的结果。前三项服务仅为用户提供评分或指导意见，而最后一项合约升级服务则完全由用户自行选择。另外，这些服务还可以支持独立的服务提供商，Chainlink 鼓励独立服务提供商的参与，以便为用户提供更加丰富的安全服务。

5.1 验证系统

Chainlink 的验证系统负责监控链上预言机行为，提供客观的性能指标，为用户选择预言机提供指导意见。它会对预言机在以下两个方面进行监控：

- 可用性：验证系统应记录预言机失效，以及时响应数据请求。它将持续汇总运行时间内的统计数据。

- 准确性：验证系统应衡量预言机反馈结果与网络中其他预言机之间的偏差程度，并记录明显错误的反馈结果。[3]

Chainlink 初期的链上聚合系统可以直接监控预言机，因为 CHAINLINK-SC 可以看到所有预言机的活动。

---

3 对于不同数据类型来说偏差的定义有所不同。如果是简单的布尔逻辑，比如航班是否准点，那么"偏差"指某一反馈结果与大多数结果相反。如果是某一城市的气温，正常情况下不同传感器和不同数据源的数据可能有所不同，那么"偏差"指重大数值差异。

然而，在我们前文中设想的 Chainlink 链下聚合系统中，数据会在预言机端聚合。也就是说，CHAINLINK-SC 无法直接看到预言机反馈的结果，也无法直接监控到预言机的可用性和准确性。

所幸，预言机提交的结果都附有电子签名，因此便生成了不可否认的证据。Chainlink 提出以智能合约的形式实现验证服务，针对偏差结果提交证据的预言机将得到奖励。换句话说，预言机将受到激励上报明显有问题的行为。

可用性相对来说比较难监控，因为预言机不会在响应失败的记录上签名。因此 Chainlink 提出预言机在它们从其他预言机收到的反馈结果集合上附上电子签名认证。验证合约会接受预言机提交的认证集合，并对提交认证的预言机给予奖励。认证中会一致显示某一问题节点不响应数据请求。

无论在链上还是链下，预言机的可用性和准确性数据都会在链上公开。这样一来，用户/开发者就可以在客户端实时查看这些数据，其中包括以太坊中的 Dapp 或联盟链上的同类应用。

5.2 声誉系统

Chainlink 的声誉系统将记录预言机服务商以及节点的用户评分，让用户可以评估预言机的历史服务水平。验证系统的报告可能成为决定预言机声誉的主要因素，并确保声誉数据真实可信。而链上服务记录之外的其他因素则可揭示预言机节点本身的安全水平。这些因素包括用户对预言机品牌的熟悉度、节点操作团队背景信息以及基础架构。我们将 Chainlink 的声誉系统设计成链上和链下两个模块。在链上模块中，用户评分可以供其他智能合约参考；另外，声誉衡量指标还能轻松在链下获取，因为在链下可以高效处理大量数据并且加权方式可以更加灵活。

对节点运营商来说，声誉系统一开始包括以下几个衡量指标，既有按不同任务类型划分的指标（详情请见第二章），也有综合所有任务类型的指标：

- 被分配的数据请求总数：节点过去被分配的（包括完成和未完成的）数据请求总数
- 完成的数据请求总数：节点过去完成的数据请求总数。这个数字可以除以被分配的请求总数得出任务完成率。
- 被接受的反馈结果总数：计算合约在比较各台预言机的响应后判定可以被接受的反馈结果总数。这个数除以被分配的数据请求总数或完成的数据请求总数，可以得出准确率。
- 平均响应时间：所有预言机都需要一段时间来确认，但预言机的响应速度将有助于定义未来预言机的响应速度。平均响应时间一般按完成数据请求的时间来计算。
- 罚款金额：预言机支付保证金可保障其服务质量，用经济处罚的手段避免预言机服务商收了钱不办事。这个指标既有现实意义也有经济意义。

声誉较高的服务提供商将受到激励认真完成任务，保证较高的可用性和数据质量。用 户差评将严重损害品牌价值，另外，违规操作还会被罚款。因此，我们希望建立一个良性循环，让表现好的预言机能够不断积累声誉，而较高的声誉也能反过来激励节点维持高水平的服务。

5.3 认证服务

Chainlink 的验证和声誉系统旨在解决一系列预言机违规操作的行为，在多数情况下保障系统完整性。另外，我们在此基础上还添加了一层认证服务，目的是避免重大问题的发生，并在问题发生时进行补救，特别是由于女巫攻击和镜像攻击导致整个区块沦陷的情况。接下来我们将具体阐述这两种攻击。

女巫攻击和镜像攻击

无论是简单版协议还是合约内聚合协议，其目标都是避免节点违规操作，抄袭诚实节点的反馈结果，从而不劳而获。但这两种协议都无法抵御女巫攻击[9]。这类攻击中，有一方能操控多个表面上看起来独立的节点。攻击者试图在预言机池里占据主导地位，操控 f 台以上的预言机参与聚合协议，并在特定时间提供错误数据，这样做可能是为了操纵高价值合约中的大额交易。有时候一个攻击者可以操纵达到合法数量的预言机共同作弊，而有时候则是多个攻击者共同操纵。f 台以上的预言机被攻击者操控的危险性特别高，因为无法单从链上行为判断出来。

另外，女巫攻击者为了降低操作成本，会采取镜像攻击法，操控预言机从同一个数据源获取数据并发送至链上。也就是说，违规操作的节点偷偷从同一个链下数据源获取数据，并假装是从不同来源获取的。无论攻击者是否选择发送错误数据，镜像攻击法都能使他们受益。它所带来的安全威胁比伪造数据要小很多，但还是会对安全造成一定影响，因为多个请求者之间无法进行对比甄别出错误数据。比如，正常情况下就算数据源 http://www.datasource.com 由于偶发的 bug 而发出了错误的数据，最后综合大多数请求者的数据还是能得出正确的结果。

我们的长期战略是使用可信硬件消除女巫攻击及其连带的错误数据、镜像攻击和预言机等风险（详情请见第六章）。

认证服务设计理念

Chainlink 的认证服务旨在保障完整性和可用性，在短期到中期监测并防止镜像攻击和节点窜通行为。认证服务会为优质节点提供背书。我们在这里再次强调，正如上文中提到的那样，我们为服务提供商打分的唯一目的是为用户提供便利。我们这样做不是为了控制节点是否参与到系统当中。

认证服务基于节点部署和操作的几个维度为其背书，监控节点在验证系统中的数据并对其提交到链上的结果开展事后抽查，特别是针对高值交易，将结果与直接来自权威数据源的数据进行对比。市场对预言机数据需求巨大，因此我们相信会有足够的经济激励支持对节点开展链下审计并确认是否符合相关安全标准，比如云安全联盟（CSA）的云控制矩阵相关规定[26]，并提供有用的安全信息，以证明对预言机的数据源和智能合约代码按要求开展了审计工作。

除了声誉衡量指标以及链上和链下反欺诈自动化系统以外，认证系统还能甄别女巫攻击和其他链上系统无法甄别到的违规行为。举个例子，如果所有节点都认为“太阳从西边升起”，那么最终 USER-SC 就会接受这个明显错误的数据。然而在事后审核时，“太阳从西边升起”这条记录很明显就能被甄别出来是错误数据。

5.4 合约升级服务

从近期智能合约被黑客攻击的事件可以看出，光靠代码几乎无法让智能合约在安全性上做到滴水不漏[1],[20],[22]。即使智能合约的代码完全没有问题，它所部署的环境一旦出现变化或错误，仍然会对合约安全造成严重影响。[2]

因此，我们推出了合约升级服务，并强调这项服务并非强制性，而是完全由用户自行选择的。

短期内如果发现了风险漏洞，合约升级服务将在 Chainlink 预言机中创建一套新的预言机合约。新的数据请求合约将迁移到新的预言机合约集合中。

然而，这样一来，新旧版本的合约会同在一台预言机内，而旧版本合约仍存在潜在风险。因此，我们的长期战略是 CHAINLINK-SC 在请求合约向预言机发起的调用中加上一个标识（MIGFLAG），表明这次调用是否应该被转发到新的 CHAINLINK-SC 中。MIGFLAG 的默认设置是 false，可以让请求合约被自动转发并迁移到新版本的 CHAINLINK-SC 中。用户要激活转发，需要用请求合约向 Chainlink 预言机发送带有 MIGFLAG=true 参数的请求。（注：用户可以修改智能合约，其在链上收到有权限的合约管理员发送的更改指令后便可以对标识进行更改。）

用户将新的预言机合约作为“紧急逃生出口”，这个概念一直以来都受到区块链研究者的追捧[23]。这个机制可以弥补错误和漏洞，同时规避了白帽编程[1]或硬分叉等复杂繁琐的方案。迁移到新版合约的过程在链上可见，用户在升级前也可以查看审计结果。

然而，我们发现一些用户不放心任何一个组织控制迁移或转发的“紧急出口”。强制迁移可能会让有权限控制合约的人拥有更多权力，或是让黑客有机会开展恶意攻击，比如修改预言机的反馈结果。因此，请求合约拥有对转发功能的所有控制权，可以选择不激活更新。另外，由于 Chainlink 本质上是去中心化的模式，因此我们希望服务商能够支持社区开发的多个版本的 CHAINLINK-SC。

5.5 LINK 通证使用

Chainlink 网络使用 LINK 通证**向 Chainlink 节点运营商就链下数据获取、重新格式化、链下计算以及服务水平保障支付费用。以太坊等网络中的智能合约如果要使用 Chainlink 节点，就需要使用 LINK 付款给为它们提供服务的 Chainlink 节点，节点根据用户对它们服务的需求量以及其他类似数据源的供应量来定价。LINK 是 ERC20 通证，附带 ERC223“转账加调用”功能（address，uint256，bytes），合约可以在单笔交易内收到并处理通证。

六．长期技术战略

本白皮书中将从三个方向讨论 Chainlink 的长期技术战略，即：预言机的保密性、基础架构变更以及链下计算。

6.1 保密性

分布式预言机网络旨在建立高水平的安全保障，防止问题预言机危害整个网络的安全。在多数情景中，分布式预言机网络都希望在出现拜占庭错误（即前文的简单版聚合协议中 f<n/2）的情况下也能获得正确结果。可信硬件可以产生巨大价值，而且可以更好地保障 Chainlink 网络的安全。可信硬件是 Town Crier（TC）预言机的核心模块[24]，目前 Town Crier 预言机在以太坊主网运行[33]，其创建者与 SmartContract 共同发布了 TC 预言机。

在众多可信硬件中最出名的非英特尔的 Software Guard eXtensions（SGX）莫属，这是一套指令集架构扩展[12]-[15],[18]，旨在为分布式信任模式提供强大的附件。简而言之，SGX 让应用可以在一个叫“enclave”的环境中执行，这个环境具有两个关键的安全属性。其一，enclave 可以保护应用的完整性（即：数据、代码以及控制流）不受其他流程影响。其二，enclave 可以保护应用的隐私，其他流程无法查看其数据、代码和执行状态。SGX 还能保护 enclave 中的应用不受恶意操作系统攻击，因此即使是系统管理员也无法看到应用运行情况。

虽然 AMR TrustZone 等各种其他类型的可信硬件也出现有一段时间了，但 SGX 拥有其他可信硬件不具备的关键功能。它可让平台生成一个证书，每个证书都有一个独特的哈希值，证明某一应用的执行情况。这个证书可以远程进行认证，让具体应用实例与公钥绑定，并与外部建立畅通的保密通道。

在 enclave 中运行预言机并发布证书就可以确保预言机执行某一应用，特别是由 Chainlink 生态系统开发者创建或背书的应用。另外，如果 enclave 可以通过 HTTPS 连接至数据源，那么其中的预言机就可以确保其接收的数据是未经篡改的（详细内容参见[24],[33]）。这些功能都可以有效规避预言机篡改数据或受到女巫攻击等各种风险。

然而，可信硬件还有更大的潜力，那就是其保密功能。保密性是区块链部署最大的难题之一，而具有保密性的预言机是解决这个问题的关键。

为什么分布式预言机无法做到保密？

任何预言机系统要做到保密都是个大难题。如果预言机连接智能合约等链上前端，那么向预言机发出的所有请求都会是公开透明的。数据请求可以在链上进行加密，再在预言机端解密，但即使是这样预言机自己也能看到这些数据。甚至安全的多方计算（即：基于加密数据进行计算）等各种复杂工具也无法解决这一问题（参见[11]中面向应用的角度）。有时，服务器需要向目标数据源服务器发送请求，因此它必须查看请求内容，那么之前所建立的隐私保护也将形同虚设。除此之外，它还能看到发回的数据。

使用 SGX 实现预言机的保密性

预言机使用 SGX 在 enclave 中接收并处理数据，本质上就是一个保障完整性和保密性的可信第三方。首先，这样的预言机能够在 enclave 中解密数据请求。然后，对数据进行处理，不向任何流程或人员透露数据隐私。Enclave 也可以从数据源秘密处理数据，并安全地管理用户身份验证信息等敏感信息，这是一个非常强大的功能，下文中我们将详细阐述。

Town Crier（TC）系统可以保护航班数据请求隐私。航班信息用 TC 公钥加密，发送到 TC 智能合约客户端。TC 将请求解密，然后通过 HTTPS 联系数据源（比如 flightaware.com）。请求智能合约向 TC 发送一个请求："本航班是否延误？"，然后 TC 向其返回一个简单的是/否结果，整个过程中不会在区块链上披露任何其他信息。

TC 还有一个更有趣的功能，那就是它可以在 Steam 游戏平台上开展交易。TC 可以安全地获取用户身份验证信息（即密码）查看游戏所有权是否从卖家转让到了买家。这样可以创建一个安全的市场机制，保障虚拟货币和电子商品的公平交易。相比之下，普通的分布式预言机无法安全地帮用户管理密码。

TC 还能以可信的方式在链下聚合来自多个数据源的数据，并对来自多个数据源和联动数据源的数据进行可信运算（比如：对多个数据源的数据取平均值，或从一个数据源搜索数据以响应另一个数据源的结果。）

可信硬件能够以全新的方式提升区块链的吞吐量[24],[29]，包括智能合约在内的大部分区块链基础架构都在 enclave 中执行。这个架构既保留了区块链的透明性又拥有了链下执行和可信硬件的保密性。虽然也有人提出过使用其他技术（比如 zk-SNARKs）来解决这个问题[21]，但可信硬件相比之下实用性远超过其他方案，而且复杂性也相对较低。我们目前正在研究这个方案，而预言机则发挥了催化剂的作用。

我们将在附件 B 中简要讨论英特尔 SGX 作为信任来源的问题。

用 SGX 定义安全性

一旦有了可信硬件，就可以在形式上定义预言机的正确性，首先是[32]中提出的英特尔 SGX 的形式化描述。其中，SGX 被视作一个全局通用可组合的（UC）[6]函数 $F_{sgx}(\Sigma_{sgx})[prog_{encl}, R]$。Σ 指代包含签名函数（Σ.Sign）和认证函数（Σ.Verify）的签名机制。Σsgx 生成的结果向 $F_{sgx}(\Sigma_{sgx})[progencl, R]$ i 实例传递参数。参数 $prog_{encl}$ 指代在 enclave 中运行的程序（注：enclave 指受到硬件保护的环境）。R 指代在 SGX 上运行的不可信代码（注：这里指调用 enclave 中应用的软件）。

图五[24]展示了 F_{sgx} 函数的运行过程。在初始化阶段中，它运行了 outp := $prog_{encl}$.Initialize()，对 $prog_{encl}$ 和 outp 的代码生成了一个证书。证书 σ_{att} 是 $prog_{encl}$ 在 enclave 中运行的平台电子签名的声明，并生成结果 outp。在典型应用场景中，$prog_{encl}$.Initialize()生成了一个具体实例的公钥，可以向应用实例创建一个安全通道。使用 (id,params) 调用 $prog_{encl}$.Resume 方法，F_{sgx} 继续执行并输出 $prog_{encl}$.Resume(id,params)的结果。其中 id 指代会话标识符，params 指代传入 $prog_{encl}$ 的参数。

Fsgx[progencl, R]: **SGX 的抽象化**

硬编码: sksgx (Σsgx 的私钥)

假设前提: $prog_{encl}$ 有权限调用 Initialize 和 Resume 两个函数

Initialize:

从 R 接收 (init) :

令 outp := $prog_{encl}$.Initalize()

σatt := Σsgx.Sign(sksgx,($prog_{encl}$, outp))

输出(outp, σ_{att})

Resume:

从 R 接收(resume, id, params):

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

输出 outp

图五：描述了 SGX 的运行过程，其中包含了 SGX 功能的一个子集。

通过图五的形式化描述，就可以准确定义预言机的完整性。定义一是[24]中给出的比较普适化的定义，我们称之为“预言机的真实性”。

定义一（预言机的真实性）：预言机 O 调用 F_{sgx} 函数运行程序 $prog_{encl}$，并产生实例密钥 pk_O。如果(pk_O, σ_{att}, params := (url, T), data, σ)中的 data 不等于持有公钥的 url 在 T 时的内容，那么多项式时间攻击者 A 即使可以与 F_{sgx} 随意交互，也无法说服诚实的验证者接受(pk_O, σ_{att}, params := (url, , T), data, σ), 这就是预言机真实性的定义。(注：我们模型中的 $prog_{encl}$.Resume(id, params)）。

$$\Pr\left[\begin{array}{l}(\mathsf{pk}_O, \sigma_{att}, \mathsf{id}, \mathsf{params}, \mathsf{data}, \sigma) \leftarrow \mathcal{A}^{\mathcal{F}_{sgx}}(1^\lambda): \\ \quad \left(\Sigma_{sgx}.\mathsf{Verify}(\mathsf{pk}_{\mathrm{sgx}}, \sigma_{att}, (\mathsf{prog}_{encl}, \mathsf{pk}_O)) = 1\right) \wedge \\ \quad \left(\Sigma.\mathsf{Verify}(\mathsf{pk}_O, \sigma, (\mathsf{id}, \mathsf{params}, \mathsf{data})) = 1\right) \wedge \\ \quad \mathsf{data} \neq \mathsf{prog}_{encl}.\mathbf{Resume}(\mathsf{id}, \mathsf{params})\end{array}\right] \leq \mathsf{negl}(\lambda),$$

A 指代任何基于概率的多项式时间攻击者，λ 指代安全参数。

6.2 基础架构变更

目前保障预言机安全所面临的最大挑战是现有数据源不会在它们提供的数据中附上电子签名。如果有签名的话，用户就不需要信任预言机不会篡改数据。保护网页通信安全的协议 HTTPS 无法对数据签名。然而，该协议有一个底层公钥基础架构（PKI），服务器必须通过认证，原则上也可以支持对数据签名。

基于这个概念开发出了 TLS 的扩展包 TLS-N，允许 HTTPS 服务器对它们与客户端的会话进行签名。这种选择性的签名模式还有其他一些优势，比如客户端无法访问签名的内容，因此用户密码等连接到服务器的身份认证信息可以得到保护。

我们认为 TLS-N 这类基础架构变更可以很好地提升预言机的安全性。然而，由于基础架构变更本身存在以下限制，因此需要与 SGX 等其他技术结合使用：

1. 基础架构变更：可惜的是，除非 TLS-N 成为一个标准，否则数据源必须明确部署 TLS-N 才能为客户端带来价值。近期内数据源几乎不太可能做到这一点。

2. 聚合和计算：TLS-N 无法对来自数据源的数据进行聚合或其他形式的可信计算。因此需要建立可信机制来完成这些任务。

3. 成本：TLS-N 签名的数据链上验证成本相对于普通的签名验证更高。

4. 保密性：TLS-N 无法支持带外用户身份验证信息或请求的保密管理，用户需自己对数据源发起请求。比如，保密的航班信息无法储存在智能合约中，之后无法自动向网站发起保密请求。

6.3 链下计算

在一些有趣的预言机应用情景中，比如使用身份验证信息的 API，预言机需要完成的工作远远不止数据传输。它可能还需要管理身份验证信息或登录账户删除数据等。实际上，在 Town Crier 的 SGX 系统和零知识证明[21]等技术的帮助下，可以实现完全可信和保密的预言机，而这将模糊预言机和智能合约的边界。

Chainlink 现已支持用正则表达式请求数据，用户可以灵活地设置链下数据处理方式。然而，我们的长期愿景是让预言机成为大多数智能合约使用的关键链下计算资源。我们将在预言机网络中开发完全通用且具有隐私保障的链下计算能力，并为智能合约传输计算结果。我们还将为预言机链下计算提供强大的安全保障，将成本高昂且隐私性强的计算逻辑放到预言机端运行，这样做能极大提升保密性、降低合约执行成本并使基础架构变得更加灵活。

七．现有预言机解决方案

如今智能合约领域对新型预言机技术的需求日益高涨，而 Chainlink 的出现正是为了迎合这一需求。虽然市场对兼具可靠性和灵活性的预言机呼声很高，但现如今符合这一条件的预言机系统却非常罕见。我们认为缺乏可信的预言机将严重阻碍智能合约的发展。

目前最常用的预言机模式是中心化预言机。这个模式本身存在诸多问题，因为它创造了一个中心化的控制点，无法达到智能合约对防篡改的高标准。类似[31]这样的系统也试图采用公证模式解决这一问题，来“证明”正确的行为。这种模式存在许多问题[37]，出具的证明在链上也无法得到验证，导致需要进一步验证，或甚至出现递归式验证。

另一个保障预言机数据可信的方法是依靠人工输入非结构性数据。“人工输入的预言机”通常在预测市场比较受欢迎[17],[25],[28]。此类预言机建立了适当的保证金制度，用罚金机制消除理性参与者违规操作的可能性，保障了所有预言机提交结果集合的准确性。这个方案具有去中心化的特性而且方便灵活。一些人工输入数据的预言机从人工处

获取数据，这个方法可以应对无法获取可靠的结构性数据的问题，比如需要对新闻事件进行自然语言处理。然而，人类认知成本高、速度慢，人工输入的预言机需要耗费大量资源，而且无法做到实时处理，一次只能处理少量问题。我们认为可以使用 Chainlink 快速自动地向预测市场合约传输结构性数据。

最后一个方法是从源头改变数据格式。如果数据源对其数据附加数字签名，传递数据的服务器就无需被信任。USER-SC 只需检查数据上的签名即可。TLS-N 在这种情况下非常好用，我们上文中也提到过了，这里不再赘述。然而，正如前文所说，TLS-N 需要改变现有基础架构。

八．总结

本白皮书介绍了去中心化的预言机网络 Chainlink，它能安全地将智能合约连接至链下数据。另外，我们还具体阐述了 Chainlink 的架构及其链上和链下模块。在定义了预言机安全性后，我们讨论了 Chainlink 多层级的去中心化方案。我们介绍了一个功能新颖的协议，防止预言机“吃空饷”，另外附件中还会介绍其他协议和安全证明草图。我们还具体阐述了 Chainlink 的技术和基础架构发展路线图，其中包括可信硬件和数据源对数据签名。最后，我们仔细研究了现有的预言机解决方案及其缺陷，挖掘了市场对 Chainlink 潜在的巨大需求。

设计原则

在我们继续推进 Chainlink 的开发工作之前，需要先梳理我们的核心原则：

- **安全开放的去中心化系统**

去中心化不仅是区块链防篡改属性的基础，而且还决定了区块链无需许可的本质。我们在持续打造去中心化系统的过程中，希望能进一步提升生态系统无需许可的属性。我们相信去中心化是打造长期繁荣的全球生态系统中至关重要的一环。

- **用模块化理念设计简单灵活的系统**

我们对于“从小处着手，深耕某一领域”的理念深信不疑。简单的模块灵活性高，可以安全地集成至大规模系统中。我们相信模块化不仅有利于系统升级，还能推动去中心化发展。我们的目标是设计出一个生态系统，允许各种不同的实现方式互相竞争，避免 Chainlink 核心部件的管理过于集中。

- **开发安全、开源、可扩展的系统**

Chainlink 的成功要归功于之前的许多开源项目。我们非常重视社区，并将持续以开源的方式开发 Chainlink 项目。我们将继续与开发者、学者以及安全专家保持合作，相互评估。我们欢迎来自各方的安全审计、测试和形式证明，与我们共同推动平台的稳健性和安全性，并为未来创新奠定基础。

我们秉持着上述原则，希望使预言机成为生态系统的安全基石，为区块链和智能合约的发展添砖加瓦。

参考资料：

[1] Parity. The Multi-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 ACM 22.11 (1979), pp. 612–613.
[4] Claus-Peter Schnorr. “Effiffifficient 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 Crypto graphic Protocols”. In: FOCS. 2001.
[7] Ran Canetti. “Universally composable security: A new paradigm for crypto graphic 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 signa
tures and a (t, n) threshold scheme for implicit certifificates”. 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: Acous tics, Speech and Signal Processing (ICASSP), 2011 IEEE International Con ference 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 (vis
ited 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 instruc tions 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 Hard
ware Extensions for Strong Software Isolation.” In: USENIX Security Sympo sium. 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 ACM 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 Markup 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 ACM SIGSAC Conference on Computer and Communications Security. ACM. 2016, pp. 270–282.
[25] Augur project page. https://augur.net. 2017.
[26] CSA Cloud Controls Matrix. URL:
https://cloudsecurityalliance.org/group/cloud-controls-matrix. 2017.
[27] Mark Flood and Oliver Goodenough. Contract as Automaton: The Computa tional 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 In centive 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.

披露声明

Ari Juels 是雅各布以色列理工学院-康奈尔大学研究所的教职人员。他兼任 SmartContract Chainlink Ltd.的顾问，并享有财务权益，他参与了本白皮书的撰写工作。

*本白皮书由 SmartContract Chainlink Ltd.（“SCCL”）提供，SCCL 是一家在英属维京群岛注册的企业，是 Chainlink 平台的母公司。Secure Asset Exchange,Inc.（“SAE”）dba SmartContract.com 为 SCCL 提供行政管理、技术和其他支持，其中包括为 SCCL 展开 LINK 通证销售工作，SCCL 向其支付佣金。区块链技术正在渗透至科技、社会和商业系统的各个角落，并将不断向前发展。因此，本白皮书中阐述的计划、战略和实现方案也可能会随着这个大趋势不断变化，有些可能最后完全不会落实到应用。SCCL 和 SAE 保留对 Chainlink 平台计划、战略或实现方案的开发、推进或修改的权利。

**SCCL 依据《通证销售条款和条件》销售 LINK 通证，完整版条款请访问 http://link.smartcontract.com/terms。LINK 通证不是证券、投资或货币，不允许作为上述三类资产进行销售或营销。另外，销售工作可能面临较高的技术性和系统性风险；通证不向美国或加拿大公民或居住在美国或加拿大的个人销售。正如通证销售条款中所述，销售期限、时间、定价以及其他条款可能会变更。LINK 通证销售工作涉及已知和未知的风险、不确定性以及可能导致 LINK 通证的实际功能、用途和使用水平与 SCCL 在协议中预测的结果、使用、功能或用途产生严重偏差的其他因素。

附件

A 链下聚合

为了获得附带签名的合法结果并防止“吃空饷”，Chainlink 提出开发链下聚合协议[4.2 中详述过]，这是基于门限签名技术[8]的简单分布式协议。这个方案的优势是，n 个预言机节点集合可以在链下生成某一请求的完整签名。因此，最后只需将一条经过验证的信息传输至链上进行处理即可，无须将 n 条预言机信息都发到链上。这个相比算法 1 可极大降低成本。这个概念还可以进一步得到扩展（注：具体参考[30]），在同一个门限签名中聚合多个数据请求的结果，这个概念我们在此不会具体展开，但之后的架构中可能会进行探索。

假设有 f<n/3 台预言机出问题并且 t=f+1。问题节点可能会出现“吃空饷”的行为和/或其他不诚实的操作，比如为非法结果签名。

我们完整的链下聚合协议包含一对算法 OCA=(DistOracle, RewardOracles)，通过函数 $Sig_{sk}[A]$计算出一个签名，其中 $A=Agg(A_1,A_2,.....,A_n)$，Agg 是多数函数。算法 2 和算法 3 中展示了这些协议只运行一轮的简化版本。第一个是由参与的预言机节点负责执行，第二个则是由 PROVIDER 负责执行，正如上文所述，这个 PROVIDER 也可能是智能合约。

在讨论 OCA 之前，我们先简单介绍一下 Schnorr 签名以及计算签名的门限机制[8]。

Schnorr 签名

Schnorr 签名机制利用了一组（G）含有基点（generator）的素数阶（p），因此其离散对数问题非常难解。用户的密钥对函数是(sk, pk) = (x, y = gx)，其中 $x \leftarrow\$ Z\times p$，而 $Z\times p=Zp-0$。为不失一般性，我们采用乘法作为群的运算方法。Schnorr 签名可以用椭圆曲线群计算出来，这是一种典型的现代加密实现方法。图 6 展示了 Schnorr 签名机制。

Schnorr 门限签名机制

我们使用了[8]中的门限签名机制。这个机制会生成全局私钥/公钥（sk/pk），并针对信息 m 生成完整的签名 Sigsk[m]。

在初始密钥生成协议中，每个参与者都会被分配到一个 key-share $x_i=sk_i$。这是一次性的操作，可以使用分布式密钥生成协议生成（例如[5]中所示），或针对异步设置生成[10]。

要生成签名，参与者（即预言机）首先要执行分布式密钥生成协议，在设置时生成 key-share。该协议输出一个全局临时密钥 e。每个参与者（O_i）都会接收到 e 的密钥副本 e_i。

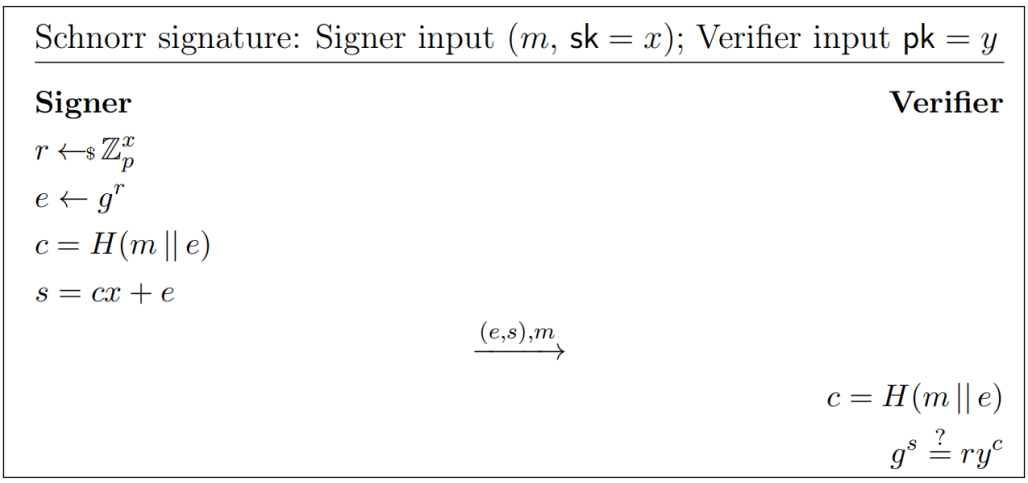

图 6：Schnorr 签名机制

获得临时密钥 e 的预言机 O_i 所产生的部分签名公式为 $\sigma(e)_i = cx_i + e_i$，，其中 c = H(m || e)，正如完整签名所示。每个预言机 O_i 都有一个函数 $valid_i(\sigma_i ; (pk, e))$ ，用来验证预言机生成的部分签名。如果部分签名在 $valid_i$ 函数中被验证是正确的，那么我们则认为 O_i 的签名是合法的。

我们修改了该签名机制的一些符号，并对其做了大量精简。具体细节请参照[8]。

A.1 OCA 协议

我们现在来介绍 DistOracle 和 RewardOracle 算法，它们一起构成了 OCA 协议。下文算法 2 和算法 3 中将详细描述其过程。我们简化了过程，因此相比算法 1 省略了见证者相关的逻辑，我们直接用 Commit 来指代提交函数。

值得一提的是，所有参与者都可以看到 CHAINLINK-SC 收到的信息，因为这些信息都在链上。规定 Σ*是 Chainlink-SC 收到的第一个合法签名 Σ。在算法 3 中，我们用 PS*指代 PROVIDER 收到的一系列取消提交的操作，PROVIDER 的部分签名生成 Σ*。（PS*可以来自发送 Σ*的预言机，但不是必要条件。任何持有 PS*的部分签名的预言机都可以向 PS*发送结果并获得奖励。）

算法 2 DistOracle(f,n,i,sk_i=x_i,pk,Src)(O_i 代码实现）

共同生成临时密钥：

1. 执行分布式密钥生成协议，并返回(e_i,e)。

获取数据：

2. 从 Src 获取 A。

生成部分签名：

3. 计算 σ(e) i (= cx_i + e_i , c = H(m || e)，其中 m = A_i)

结果提交轮：

4. 广播提交的结果 $comm_i$= (Commit($\sigma_i^{(e)}$,A_i);i)
5. 等待收到 n-f 个来自不同的预言机的合法提交集合 C_i
6. 将 C_i 发送至 PROVIDER

准备轮:

7. 广播 prepared
8. 等待收到 n-f 条来自不同预言机的 prepared 信息

解密/取消提交轮：

9. 广播取消提交 $comm_i$($\sigma_i^{(e)}$,A_i)
10. 等待收到一组≥ t 的合法取消提交的操作

计算完整签名：

11. 如果 CHAINLINK-SC 还没有收到合法的 Σ，那么
12. 将 PS 中的部分签名聚合至 Σ = Sig_{sk}[A].
13. 将 Σ 发送至 CHAINLINK-SC
14. 将 PS 发送至 PROVIDER
15. 结束 if 判断

算法 3 RewardOracle （PROVIDER 代码实现）

1. 等待收到从不同的预言机和 PS*收到的 n-f 个提交集合（C_i）的集合 C
2. 对所有的预言机 Oj 执行
3. 如果 σ j 属于 P S∗ 并且 C 中有大于 2f 个集合从 Oj 提交了 σ j
4. 那么就向 Oj 发送奖励
5. 结束 if 判断
6. 结束 for 循环

A.2 证明草图

我们针对 OCA 的关键属性提供了证明草图。假设最多有 f 个问题节点，协议总是能针对正确的结果生成合法签名，绝对不会奖励“吃空饷”的预言机。

证明 2：OCA 协议绝对不会奖励“吃空饷”的预言机。

证明（草图）：假设 O_z 在“吃空饷”。则只能在 t 时间之后才能广播合法结果，t 时间指首个诚实节点 O_i 在算法 2 的第九步取消提交的时间。O_i 收到了 n-f 条 prepared 信息，其中至少 n-2f 条信息来自诚实节点。O_j 指代上述 n-2f 个诚实节点的其中一个节点。O_j 在发出 prepared 信息后不会再接收提交结果，因此任何此类诚实节点 O_j 的集合 C_j

在 t 时间后都不会再改变。所以,算法 3 中的 C 中最多有 n-(n-2f)=2f 个集合将包含 O_z。所以 O_z 不会收到奖励。

可惜的是,OCA 无法保障没有吃空饷的预言机一定会得到酬劳。违规操作的攻击者可以在收到取消提交后生成自己的部分签名并在 Σ 中只放进一个诚实节点的部分签名,因此在算法 2 的第 13 步中挤兑诚实节点。诚实节点提交的结果可能不包含在任何节点采集的 n-f 条信息中,而是可能在之后出现。

证明 3:OCA 每次都会生成合法签名 $\Sigma = Sig_{sk}[A]$,最终被发送至 CHAINLINK-SC。

证明(草图):有 n-f 个诚实节点,且 f<n/3。所以有>2f≥ f+1 个诚实节点,所以至少有 t 个诚实节点。所以算法 2 的第一步将成功完成。

同样地,由于存在 n-f 个诚实节点,每个预言机最终都会完成算法 2 的第 7 步,发送一条 prepared 信息。诚实节点最终会收到至少 n-f 条 prepared 信息,并将取消提交,因此诚实节点会完成第 13 步。

证明 4:OCA 中的 CHAINLINK-SC 收到的任意合法签名 $Sig_{sk}[A]$ 都有合法值 A。

证明(草图):显而易见,合法签名 $Sig_{sk}[A]$ 包含一个合法值 A。因为需要 t 个合法签名来计算 $Sig_{sk}[A]$,且最多 f<t 个节点是问题节点,所以至少 A 的一个部分签名是由诚实节点提供的,因此肯定是正确的。

A.3 讨论

OCA 协议存在一些设计上的问题,我们将在下文简要讨论。

向诚实节点付款

OCA 虽然会拒绝付款给问题节点,但它无法保障诚实节点一定能得到酬劳。实际上,即使在不存在问题节点的理想情景中,消息排序出错也可能导致向 Σ 发送了部分签名的诚实节点收不到钱。

将算法 2 设置成同步可以部分解决这个问题。具体来说,“等待”的几个步骤中可以要求节点等待一段时间Δ,以便保证从诚实节点收到消息。在这种情况下,所有向 Σ 发送了部分签名的节点都一定能收到付款。然而,这个方案的副作用是执行速度较慢,而且正确设置Δ的难度也较大。

异步协议保证付款的问题目前还没有得到解决,这也是我们正在致力于解决的问题。

重复消息

OCA 的目标是将链上沟通需求降至最低,最好只有一条链上消息,即由一组预言机签名的反馈结果 Σ。然而,实际操作中,由于签名 Σ 不会立即发布到区块链上,多个预

言机可以独自将签名结果发送到区块链。避免重复发送消息最好的方式就是让预言机不仅监控区块链，还监控内存池，即等待发送的消息。

密钥管理

这类协议通常会在密钥管理方面比较令人头疼。Sk 备份可以进行分布式分发[5]，可以针对新节点进行更新，移除不参与的节点，并提供积极的安全保护，即持续监测节点密钥被盗情况。另外，节点可以分成不同的小组，以确定 n 的范围。我们建议 Chainlink 使用这些技术建立灵活、快速和安全的分布式预言机。

B SGX 信任假设

英特尔的可信硬件 SGX 可以为预言机提供更强大的安全保障，然而，却无法完全取代 Chainlink 其他的安全保护方案。换而言之，SGX 对 Chainlink 预言机系统起到了锦上添花的作用。

信任 SGX 需要首先信任英特尔，然而这种信任并非是无条件的。假设英特尔的 CPU 没有后门，enclave 中的数据无法泄露，那么它则不可能监测到 enclave 内部的状态。（这类后门是可以实现的，但是需要在每个用户的设备上都安装一次后门，而且一旦暴露声誉会受到严重损害。）

理论上，英特尔或操纵英特尔生产流程的外部攻击者可以伪造证明密钥（即平台 EPID 密钥）。攻击者可以生成与 SGX 服务器不符的密钥，并允许在非 SGX 平台生成证明。实际上，攻击者可以建立虚假服务器，伪造 SGX 证明，从而不为 enclave 中的代码提供保护。如果这样的节点超过 f 个，那么攻击者就能操纵预言机的结果。更麻烦的是，这些节点会向攻击者公开敏感数据。然而，能够伪造 EPID 密钥不一定意味着能够操纵现有的合法 SGX 实例。

另外还有一点很重要，那就是我们目前只能无条件信任英特尔，除此之外别无他法。希望之后会开发出新的同类技术。 无论是你现在使用的计算机 CPU，还是储存本白皮书的服务器 CPU，目前都无法被替代，我们只能选择信任它们。

当然，最好的办法是使用来自不同供应商的可信硬件，我们只能寄希望于其他厂商也能生产出同样强大的可信硬件。目前许多人都在研发开放的可信硬件基础架构，并致力于解决此类硬件的信任问题，比如[19],[34]。然而，丰富技术提供商和基础架构的选择也不能完全保障数据的保密性。

同时，我们也对真假流量混用（cover traffic）的方案很感兴趣，以实现分布式网络的保密性。