Orchid：助力实现去中心化网络形成和概率性微支付

David L. Salamon，Gustav Simonsson，Jay Freeman，Brian J. Fox 和 Brian Vohaska 以及 Stephen F. Bell 和 Steven Waterhouse 博士

2018 年 5 月 28 日
版本 0.9.2

摘要

随着审查浏览内容和发现私人浏览信息的方法愈加有效，人们对匿名处理方法的兴趣不断增加。 不幸的是，诸如 I2P 和 Tor 之类的无限制、不受监控的现有 Internet 访问方法尚未广泛采用。 事实上，只有几千名无偿志愿者采用中继和出口节点，这使得老练的攻击者能够监控或以其他方式破坏数量可观的节点。 我们提出了一款基于市场的、完全去中心化的、基于“带宽挖掘”的匿名对等系统，我们相信该系统通过直接激励参与者来解决中继和出口节的不足。

本文旨在说明这一尚在开发中的系统。 因此，该系统无疑还会进行改动并添加新内容，以解决具体实现时出现的任何差异；它在使用库组件和特定加密算法方面非常灵活。 但是，系统的本质、目的和目标将保持不变。

该系统贡献的包括：

- 基于区块链的随机支付机制，交易成本约为一个数据包
- 销售带宽的商品规格
- 对等系统中用于分布式归纳证明的方法，这使得 Eclipse 攻击变得异常困难
- 一种高效的安全强化拍卖机制，适合在攻击者在攻击过程中可能更改出价的情况下销售带宽
- 完全分布式的匿名带宽市场

目录

1. 简介

Orchid 协议将带宽销售商组织成一个称为“Orchid 市场”的结构化对等 (P2P) 网络。 客户接入到“Orchid 市场”并向带宽销售商支付费用，以便形成一个代*理链*，连接到 Internet 上的特定资源。

与从全球 Internet 发送和接收数据的更为常见的方法不同，“Orchid 市场”上的代理链自然而然地将数据源信息与其目的地信息分开；没有一个中继或代理同时拥有这两种信息，也不会知道同时拥有这两种信息的人的身份。 “Orchid 市场”的结构通过提供强大的针对*共谋攻击*（一组带宽销售商破解这种知识分离的能力）的抵御能力，进一步支持了这种信息分离。

代理链参与者的角色是：

- ***源*节点或客户** — 发起交易的参与者。
- ***中继节点*** — 转发网络流量的中间参与者。
- ***代理*或出口节点** — 连接到请求的全球 Internet 站点的参与者。
- **带宽销售商** — 中继或代理。

与从全球 Internet 发送和接收数据的不太常见的方法（数据分隔）不同，“Orchid 市场”**提供固定速率中***继*来防止流量分析，而且会奖励与隐藏或发现信息无关的参与行为： 以代币付款。

在我们介绍系统的细节之前，我们将简要回顾它解决的核心问题，以及我们为系统的基础选择的一般解决方案。

*流量分析*问题

问题情况说明： 想象一下，您在一间坐满了数学家的自助餐厅里，希望在别人不知道的情况下给房间另一头的朋友发一条信息。 您尚未协商消息传递协议，因此必须将所有实现细节公开告知房间中的每个人。 该怎么办？

Chaum 在 1981[56] 年提出了一种特别优雅的解决方案，是让每个人既充当中继又充当接收者。 在这个方案中，参与者准备加密的消息，其数字信息等价于“包含信封的信封”- 将消息发送给 Alice，您将计算

Enc(“ToBob”|| Enc(“ToAlicel”|| Enc(message, Alice), Bob), Carol)

并将该消息发送给 Carol，由 Carol 解密并将其发送给 Bob，然后由 Bob 解密并将其发送给 Alice。 为了防止流量分析，每个人每个周期都发送固定数量的消息。 为了处理返回地址，我们可以让 Bob 和 Carol 记住一个唯一的消息标识符，并沿这条链发送回消息。

对于使用上述方法的系统而言，特别重要的是*共谋*。 如果 Bob 和 Carol 合作，他们就有可能确定是谁发送的特定消息以及要向谁发送该消息。

Sybil 问题

上面的自助餐厅问题情况说明采用现实中的人来防止女巫攻击 – 在这种情况下，一个参与者可能会假装成任意数量的用户。 不幸的是，在数字系统中不能使用这种方法。

问题情况说明： 我们如何才能知道某人在纯数字环境中是“真实”的？

可以在 Hashcash[85] 中找到解决该问题的方法。 如果我们要求那些声称自己是“真实”的人来投入计算资源，我们就可以让女巫攻击者面临这样一种局面：声称是数量惊人的网络参与者需要实际拥有数量惊人的计算资源。

*随机选择*问题

上面的自助餐厅问题情况说明假设了一种简单的方法，可以向系统的每个其他用户发送一条消息（例如，在自助餐厅里大喊大叫）。 要实现最大限度地抵御*共谋攻击*的 Chaumian 混合，我们需要能够从那些“真实”的中继中随机选择。天真地说，这需要在某人加入或离开网络时得到通知。 不幸的是，在现实的 P2P 网络中，让每个用户维护这样的列表会导致网络流量过大（($O(n_2)$ 个通知）。

问题情况说明： 如何维护所有当前“真实”中继的分布式列表，从而最大程度地减少网络开销并支持对等点的高效随机选择？

可以在 Chord[85] 分布式哈希表 (DHT) 中找到这个问题特别优雅的解决方案。 在该方案中，对等点被分配到一个大空间中的唯一地址，然后以特定方式连接，以便将执行查找的次数控制在 $O(log(n))$ 内。 添加或删除用户只需要通知对等点 $O(log(n))$ 次。

系统概述

Orchid 协议的核心是对上述解决方案的综合。 在我们的方法中，对等点需要生成*奖章*来证明它们的“真实性”，然后被组织成一个称为 *Orchid 市场*的分布式 P2P 网络。 为了保证“Orchid 市场”的参与者诚实，每个对等点都会检查邻居行为的正确性。 然后，客户使用“Orchid 市场”随机选择对等点进行 Chaumian 消息转发。 为了激励参与，“Orchid 市场”让客户按转发字节数支付中继和代理费。

这是一个简单的想法，但难点在于细节。 该系统将完全分散，完全自治，完全匿名，且用于处理支付。 因此，本设计文档的大部分内容都集中在防止对客户安全、系统性能和系统经济完好性的攻击上。 尽管攻击分析很重要，而且会占用我们很多时间，但它最终只不过是对市场运行环境的必要考虑；如果您发现自己“迷失在森林中”，我们希望您能利用前面的论述为您指引方向 - 系统的设计细节都旨在为上述三大问题提供现实的解决方案。

2. 替代方法

未受保护的 Internet 访问

在没有保护措施的情况下访问 Internet 的用户会向其 ISP 提供完整的浏览历史和网站使用情况，而 ISP 可以共享或出售这些数据。

虚拟专用网 (VPN) 服务

虚拟专用网 (VPN) 使用加密技术在较大的不安全网络上安全传输 VPN 订户的流量。VPN 收到流量后，便会对流量进行解密，并在另一个大型不安全网络上转发。 转发可以帮助用户绕开网站施加的访问限制，并在较小程度上减少网站对用户网站浏览习惯的跟踪。 加密可防止用户的 ISP 看到其流量的内容，从而避免监控攻击。 这是通过使 VPN 成为用户的新 ISP 来实现的。 VPN 提供商可以轻松执行 ISP 以前可能进行的任何攻击。

VPN 用户不应假定其 VPN 提供商是可信赖的。 尽管 VPN 服务提供商比 ISP 面临的竞争更多，但他们最终会从相同来源吸引人才，并具有类似的“现金换带宽型”业务模式。 VPN 提供商不太可能无视那些导致用户不信任其 ISP 的相同动机。 此外，在 VPN 设置中重复使用 IP 地址来中继流量可以相对容易地阻止商业网站使用这些地址 [13]。

Tor

Tor [60] 是一个免费软件项目，它的知名之处是让更广泛的受众认识了*洋葱路由* 的概念。 在此系统中，用户下载中继和出口节点的全局列表，从该列表中随机选择，然后通过他们选择的节点形成*洋葱路由*。 洋葱路由是中继的有序列表；沿洋葱路由发送的数据包依次对每个对等点加密，以确保每个节点必须接收一个在途数据包，出口节点才能解析该数据包。 这样一来，除非多个节点被同一用户破坏或运行，否则没有两个中继同时知道谁发送了数据包以及数据包的去向。

3. 攻击

许多 Orchid 协议旨在防止攻击，因此我们先介绍一下有关 P2P 网络常见攻击的文献资料。

推断攻击

Orchid 协议必须防御的最大一类攻击是泄漏其用户信息的攻击。 由于 Orchid 作为现有 互联网上的覆盖层实现，因此*不可避免地* 需要与一些对等点共享部分信息。 另外，由于 Orchid 的基础支付系统使用 ERC20 代币，因此一些交易信息也可能用于以太坊网络。 在以下列表中，此类信息标记有“*”。 某些信息在本文档中没有明确列出为*不可避免* 需要共享，但存在能发现这些信息的方法，此类信息称为*信息攻击*，属于 Orchid 的 White Hat Bug Bounty 范围。 要详细了解共享信息，请参阅第 7 节中的协议规范和第 B.1 节中的共谋讨论，以及我们的网络参考实现 [1]。

假定攻击者感兴趣的数据类型（不受时间影响）：

- 真人身份信息。 用户的姓名、社会保障号码 (SSN)、地址等。
- 网站帐户信息。 用户在特定网站上的帐户。 请注意，这可能与真人身份信息不同。
- *IP 信息。 用户访问 Orchid 网络的 IP 地址。 请注意，在某些使用场景下，这可能等同于得知真人身份信息。
- *以太坊信息。 与用户钱包关联的密钥（*公钥或私钥）。 请注意，在某些使用场景下，这可能等同于得知真人身份信息。

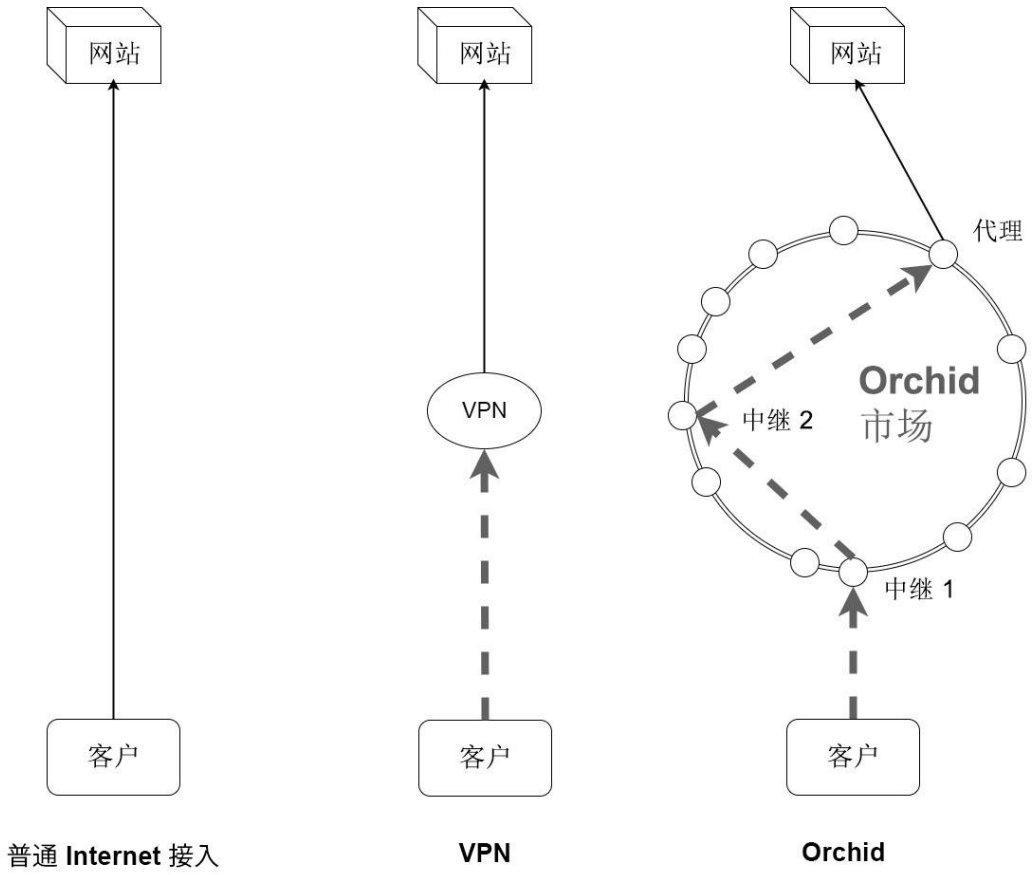

图 1： 直接连接、VPN 连接、Orchid 连接

- *Orchid 网络信息。 与 Orchid 网络上节点的当前交易关联的密钥（*公钥或私钥）。

假定攻击者感兴趣的行为信息类型（与时间和链关联的数据）：

- *客户识别。 攻击者得知客户的 IP 地址。
- *中继识别。 攻击者得知中继的 IP 地址。
- *代理识别。 攻击者得知代理的 IP 地址。
- *链接识别。 攻击者得知链中使用了两个 IP 地址。
- *网站访问。 攻击者得知 Orchid 网络与特定网站建立了出站连接。
- *Web 服务器访问。 攻击者得知 Orchid 网络与特定 Web 服务器（可能托管多个网站）建立了出站连接。
- *以太坊链接。 攻击者得知 Orchid 用户拥有以太坊公钥。
- *购买链接。 攻击者得知两笔交易的付款人是同一个人。
- *购买信息。 攻击者得知通过链发送的带宽的数量和时间。

尽管在正常运行期间，上述所有行为信息均与 Orchid 网络上的其他节点共享（如下所述），但在大多数情况下，将假定用户在攻击者能够一次得知多项信息的情况下，只受到**行为信息收集**的直接影响。 例如，假设用户 X 访问了网站 Y，攻击者需要： 买方识别、网站访问信息和若干链接识别。 因此，除非提供客户购买的服务所必需，遵循参考规范的对等点不存储或共享上述任何信息。

源于系统行为统计建模的去匿名化称为推断攻击或监控攻击。 此类攻击通常结合有“探测动作”（例如精心制作或定时的请求）或其他攻击（例如通过 DoS 攻击使特定对等点脱离网络并观察流量模式的响应情况）。

- 通过 SSL 加密的网络流量推断最终用户的疾病、家庭收入和投资选择 [57]。
- 从全局流量日志中对 Tor、I2P 和 Orchid 流量进行去匿名化处理 [59]。
- 通过时序分析得知 OpenSSL 服务器的私钥 [54]。

经济型攻击

与类似系统不同，Orchid 协议还必须关注对支付机制的攻击。 本白皮书使用的分类法如下：

1. **经济型漏洞利用**。 有利可图的不良行为，例如用户提供“免费试用”带宽，使其他用户能够独自使用免费试用带宽。
2. **经济型拒绝服务攻击 (EDoS)**。 使用大量支付使包含购买的 Orchid 网络上的另一节点不堪重负，从而使节点脱机。

女巫攻击

通过假装为多个用户而执行的恶意操作称为女巫攻击，此名称源于一名患有多重人格障碍的患者。这类攻击的应用包括：

- 向 Yelp、Amazon 等提交多个评论。
- 假装为多个 leacher，从而在 BitTorrent 上实现更快的下载速度 [74]。

日蚀攻击

在日蚀攻击中，攻击者的目标是将系统的一部分隐藏起来。 所采用的方法通常是权限提升攻击的网络等效方法： 控制对网络有更多控制权的网络位置，然后利用这一控制权获得更多控制权。

- 分割比特币挖矿 P2P 网络，当攻击者控制的计算能力远低于 51% 时，允许所谓的“51% 攻击”[65]。
- 通过接管与 BitTorrent DHT 磁力链接相关的地址空间，从中删除文件 [87]。

中间人攻击

只有将自己插入两个交互方之间后才能执行的操作统称为中间人攻击。 可以记录加密信息以进行元数据分析（第 3 节），同时可以另外更改未加密数据以控制行为。 如果未保护密钥交换，则中间人还可能欺骗两方，使他们错误地认为攻击者的密钥是另一方的密钥。

服务质量攻击

某些攻击者可能需要普遍降低 Orchid 网络用户的系统性能，从而可能减少使用。

拒绝服务攻击

为了使特定资源脱机而进行的攻击称为拒绝服务攻击 (DoS)。 “意外”情况下的系统行为通常没有得到充分指明和测试。 DoS 攻击能够有效地对 P2P 网络中的节点去匿名化。 明显示例：

- 有针对性的 DoS 攻击与基于女巫攻击的监控配合使用，可以使 Tor 流量去匿名化 [52]。
- DoS 离线操作仅需 20 个 Sybil 节点，就可以完全控制 I2P 的泛洪数据库，从而将网络上的所有流量去匿名化 [64]。

黑客攻击

通过将历史上可信赖的对等点转换为攻击向量，有动机的攻击者可能会直接破坏网络上的节点。 当使用链部署带宽时，迭代黑客攻击最终可能会允许攻击者“回溯”连接。 此类攻击具有重要的安全隐患，但超出了 Orchid 网络的范围。 如果 Orchid 网络的设计达到其目标，这将是针对系统用户的主要攻击。

4. Orchid 市场

Orchid 市场是构建 Orchid 协议的基础。 从根本上讲，它是一个分布式 P2P 网络，可促进中继、代理和用户之间的带宽购买与销售。 要进入市场并继续参与市场，需要提供工作量证明，我们称之为 Medallion。 Orchid 市场的网络结构类似于分布式哈希表 (DHT)，可以看作是修正 Chord 的扩展 [83, 85]。

4.1. 基本市场操作

总体来看，Orchid 市场提供的操作包括：

- Peddler 加入 Orchid 市场的方法。
- 向 Peddler 询问其待销售服务的方法。
- 用于选择所有对等点的子集的方法，子集由计算资源随机加权，以使*查找属性* 拥有，

$$lookup(random\_address) \Rightarrow random(Peddler)$$

其中，Peddler 可视为 Chord 中的节点，这些节点通过其指针表模拟（称为签名路由表）来跟踪有关其近邻的信息。 在 Orchid 市场中，Peddler 作为带宽的买方和卖方，中继或代理也可能是带宽的买方和卖方。 在 Orchid 市场中，用户无需成为 Peddler，但所有中继和代理需要成为 Peddler。 *查找属性* 之所以重要，是因为它能让客户知道，如果从一组 n 个 Peddler 中随机选择了一个 Peddler，且有 a 个依附者，那么随机 Peddler 不是攻击者的概率为，

$$P(Attacker|random(Peddler)) = 1 - \frac{a}{n}$$

Orchid 白皮书 [90] 在多个章节中表明，此属性可防止日蚀攻击及其他攻击。 为实现这些操作，Orchid 市场采用了没有密钥和值的 DHT 结构。 要执行随机选择，用户只需生成一个随机地址，然后找到最接近该点的 Peddler。 Orchid 市场可以表示为 2^{256} 阶的类似于 Chord 的环，因此，选择的任何随机地址必须作为 $\{1,0\}^{256}$ 中的随机整数。

4.2. 基本 Peddler 操作

Orchid 市场中 Peddler 支持的操作：

- *清单服务*。 向 Peddler 询问其销售的服务清单。
- *获取路由表和 Medallion*。 这将返回 Peddler 的 Medallion、签名路由表，以及将流量中继到路由表成员的成本。
- *中继流量*。 向 Peddler 付费以将流量转发到其路由表中的对等点之一。
- *购买服务*。 使用 Peddler 作为服务提供商。

Medallion 是 Orchid 市场用于工作量证明的代币和市场*参与许可*；如果 Peddler 没有 Medallion，将按照当前以太坊区块摘要的 TTL 顺序将其从市场中删除。 签名路由表将在 Orchid 白皮书 [90] 中进一步讨论。客户使用前两个签名路由表导航到关注的 Peddler，后两个签名路由表用于在找到 Peddler 后协商服务购买。

Orchid 市场中的导航类似于在链中使用的导航。 客户连接到某个已知 Peddler（通过拔靴法找到，请参阅第 4.6 节），检查其路由表，并付费将流量转发到最接近其选定点的 Peddler。 正如我们将在路由表一节中看到的，这使客户可以将其 IP 地址保密，同时仍可以相对高效地随机访问 $O(log^2 n)$ 个数据包的 Peddler。

请注意，Orchid 市场中所有需要带宽的操作需要支付与任何其他实体相同的带宽成本。 这些成本对于每个 Peddler 都是最低的，因为每个 Peddler 由于市场操作而销售带宽的频率不会低于其购买带宽的频率。 Peddler 的这一带宽成本可防止附录 E 中提到的攻击。

Peddler 在 Orchid 协议中通过修正 Chord DHT 使用的相同方案连接。 由于存在比较成熟的文献资料以及机器检查的正确性证明 [83]，我们选择了 Chord 而不是 Kademlia。

Peddler 地址集以 2^{256} 大小的 Chord 环中的整数表示，在此环中，对等点地址 a 和 b 之间的距离 d 按如下定义，

$$a, b, d \in (0, 2^{256})$$

$$a + d \equiv b \pmod{2^{256}}$$

A 是 Orchid 市场中 Peddler 的集合，e 是特定 Peddler。 回想一下，在 Chord 中，任何节点的最大预计对等点数量为 $\log_2(n)$。 e 的强制连接集按如下定义，

$$L = \{f : \min_{log_2(n)}\{dist(e + t, f)\}\}$$

其中 $t \in \{1, 2, 4, ..2^{255}\}$ 是任意 Peddler，min 返回 dist(...) 中最小 $\log_2(n)$ 元素集，以及与 f 之间的最小距离。

我们之所以选择使用此路由结构，是因为它的成熟度、已部署系统中的成功跟踪记录以及正确性证明。 有兴趣深入了解的读者请阅读 [85]。 就我们的目的而言，注意此路由方案提供以下两个属性就足够了：

1. *有限的确定性连接*。 每个 Peddler 预计具有 ≤ 256 个强制连接。
2. *对数遍历距离*。 给定一个随机地址 t、一个随机连接 Peddler e 且连接数为 C，则 $dist(e, t) \approx 2 * \min_{f \in C} dist(f, t)$。 由于距离会通过每个跳跃点减半，因此网络上的预计遍历长度为 $log_2(n)$，其中 n 为网络大小。

4.3. Orchid 市场上的 Medallion

Medallion 是工作量证明的代币，与以太坊区块摘要、持有人的公钥以及附录 D 中讨论的其他数量紧密相关。在 Orchid 市场中，Medallion 有两种使用方式，

- 防止琐碎人员进入市场而导致攻击
- 防止攻击者选择他们在市场中的位置

为了防止攻击者使用超过其在 Orchid 市场总计算能力中份额比例的 Peddler，每个 Peddler 需在每次 Medallion 周期中检查其所有连接的 Medallion 的有效性。 如果未提供有效 Medallion，则将断开它与网络之间的连接。 根据附录 C 中的定义，Peddler 的位置定义为其 Medallion 的密码哈希值，换句话说，

$$\text{Peddler 地址} = H(Medallion, ...)$$

这可使 Orchid 市场的每个成员都能轻松评估和验证 Medallion 持有人在市场中的位置。 此外，通过将 Peddler 的市场地址与 Medallion 关联，进行日蚀攻击会更加困难。

4.4. 签名路由和日蚀攻击

分布式网络存在一个问题，由于没有人（可能攻击者除外）拥有网络的全局视图，因此很难确定 Peddler 是否已经*在日蚀攻击下*进入完全隔离的恶意子网。 例如，设想一下，如果在上述路由方案中，攻击者选择对于其拥有的连接说谎，若买方无法检测这一点，则可能会被引向一个虚假的 Orchid 市场，其中所有"参与者"都由攻击者所有。 为了减轻对这种情况的利用，Peddler 路由表将通过算法进行选择，并通过路由表中包含的对等点进行验证。

当一个节点想要建立强制连接时，该节点必须向其强制连接列表上的每个节点证明该列表上的每个其他节点都是同一 Orchid 市场的成员。 为此，我们先选择一个随机 Peddler G，方法是找到与路由表 $H(C_i)$ 中所有连接的哈希值最接近的地址的 Peddler。 然后我们提供：

1. 证明列表中的所有 Peddler 都可以路由到 G。
2. 证明 G 可以路由到每个 Peddler。
3. 证明列表中的每个 Peddler 确实是强制连接。

这些证明均采用从 C_i 到 G 的签名路由表链形式，对于 (3)，采用从入口 Peddler 到每个 C_i 的签名路由表链形式。 提供此类证明后，新路由表上的所有对等点都对该表进行签名，连接的 Peddler 对自己的路由表进行签名。 对于新 Peddler 为强制连接的 C_i 中的元素，将向其每个连接发送相同证明以进行签名。

这是将 Peddler 添加到 Orchid 市场的唯一方法，因此这些要求形成了对 Orchid 市场完好性的归纳证明。 如果 C_i 中的一个节点尝试提供虚假路由表，它不会路由到与 C_i 中的其他 Peddler 相同的 G。 如果 C_i 中的一个节点不是 Orchid 市场的成员，G 将无法对其进行路由。 如果想要连接的 Peddler 对于离其强制连接点最近的节点 C_i 说谎，则 (3) 会证明它是假 Peddler。

通过这些属性，我们可以看到留给攻击者的攻击手段包括：

- 如果攻击者可以生成一个 Medallion 地址，使所有 C_i 受其控制，上述系统将停止运行。 这种情况的发生概率为 $\left(\frac{a}{n}\right)^{\log(n)}$。 如果发生这种冲突，$(1 - \frac{n-\log(n)}{n})^{\log(n)}$ 百分比的查询将受到影响。 解释一下这些数字，如果攻击者控制网络的 10%，且包含 100 万个节点，则发生此类冲突的几率为 1×10^{-8}%，如果发生此类冲突，则会有约 1×10^{3}% 的系统查询将受到影响。 对于 1 亿个节点，此几率将降至 1×10^{-12}%，这将破坏 1e-5% 个查询。 请注意，这一损害将在再生期间修复（请参阅第 4.5 节）。
- 如果攻击者现已加入网络，但只能使用有效的路由表，则攻击者可以执行的唯一攻击与销售服务有关，而不是在 Orchid 市场中路由流量。 这是其余攻击模式中预期的情况（攻击者控制与计算资源成比例的一些 Peddler），因此我们不将此视为攻击。

4.5. 日蚀攻击与再生

长期使用的 P2P 网络会受到日蚀攻击。 尽管上述签名路由方案会包含不断增加的对等点数量来进行验证，使攻击变得更加困难，但有一种方法可以轻松限制对等点的使用期限。 因此，Orchid 市场上的 Peddler 必须每隔 100 个以太坊区块更换一次密钥。

4.6. 查找入口节点

入口节点的分布是一个困难的主题。 如果专制的政府能够访问此列表，则他们将阻止用户访问列表的能力。 因此，我们找到了在阻止后会中断 互联网 的基本服务，并且想出了向其中包含的数据添加入口节点信息的方法。

4.7. 识别 Orchid 市场

如果没有办法在新的机器上找到"正确的 Orchid 市场"，则以上对安全性的讨论最终将毫无意义。 不能将用于*入口 Peddler* 的任何分布方法视为不会受到攻击者控制的入口 Peddler 的渗透。 为此，我们只需估计给定 Orchid 市场的计算能力，然后选择拥有最大总计算能力的市场。

- 密度估计。 由于 Peddler 的强制连接定义为最接近 2^{256} 地址空间中某个地址集的 Peddler，因此在任何实际情况下，理想连接与实际连接之间都会存在可测量的间隙。 为了估计该空间中的密度，我们可以观察到这些连接是随机二项过程的结果： 理想点和实际点之间的每个点均失败，实际点成功。 因此，对于给定数量的丢失节点 M 和给定数量的已实现连接 C，网络密度的均匀先验 MAP 估计值为，

$$\frac{C}{C+M} * 2^{256}$$

- 遍历距离。 Orchid 市场在 $O\left(log_2(n)\right)$ 个跳跃点中提供地址查找。 我们可以反向使用它来估计网络密度。

有人可能认为密度估计就足够了，但是拥有适度 Sybil 网络规模的狡猾攻击者可以自由选择要提出哪个节点作为虚假网络的入口 Peddler，而来自"真正 Orchid 市场"的入口 Peddler 的密度是网络的随机采样。 更糟糕的是，如果选择遍历距离作为度量标准，设置者可能认为攻击者会预料到这一点，因此创建需要长于 $O(log_2(n))$ 的遍历距离的次优路由表。 幸好次优连接的 Orchid 市场在密度度量方面表现欠佳。 Orchid 系统使用的验证方法是遍历到一个随机地址，在此过程中保存路由表，然后从所有跳跃点（前两个除外）使用路由表执行密度估计。

4.8. 代理白名单

某些希望提供代理服务的用户可能不愿意提供"开放访问"。 例如，允许用户访问 facebook.com 具有类似于充当中继的风险状况，而允许任意连接到 互联网 可能会导致当地执法部门访问。 因此，Orchid 市场上的 Peddler 会设置一个网站白名单，允许用户在将这些网站用作代理时连接这些网站，并在他们的响应中指定其白名单以*获取连接*。

5. Medallion

完全分散、完全匿名的数字系统常常受到一名恶意用户伪装成数千名用户的攻击（女巫攻击）。 为减少女巫攻击的产生以及此类攻击的其他影响，Orchid 协议采用了工作量证明方案。 我们将此方案的代币称为 Medallion。 每个 Medallion 包含的数据以密码方式证明，生成者在给定时间拥有*大量* 计算资源。 由于计算是一种昂贵的资源，因此使用 Medallion 会对给定攻击者模拟多个用户的能力施加预算限制。

5.1. Medallion 工作量证明

Medallion 是我们的核心安全性假设与整个网络之间的桥梁。 由于我们的基本安全目标是限制积极的攻击者获得对 Orchid 网络的控制权，因此我们选择的 Medallion 创建必须满足以下条件，

1. Medallion 必须*便于*非恶意节点创建
2. Medallion 必须能*轻松*验证
3. Medallion 必须*难以*批量创建

满足这些条件后，我们定义*困难度*来表示时间和金钱上的禁止可扩展性。 简言之，我们需要一个*便于*普通节点进入网络，但攻击者很难扩展进入网络的工作量证明系统。 在第 5.2 节中，我们将讨论为什么选择工作量证明而不是其他方法，例如权益证明 [46, 70, 72] 和空间证明 [63, 80]。

目前存在两种能满足上述要求的主要方法： 挑战响应协议和密码难题。 可惜的是，挑战响应协议可能无法在 Orchid 模式中提供足够的安全性，因为攻击者可能能够通过合谋预先计算挑战和响应。 还有密码难题，现在存在许多密码难题 [50, 78]，每个难题都有自己的权衡方案。 同样，为了满足 Orchid 的要求，只有一部分密码难题适用。 也就是说，有些密码难题无法轻松并行化、转为 ASIC 或简单衡量。 研究人员最近发现了可以生成易于验证的结果的算法，这些结果具有可调节的创建难度 [50]。 这些算法集合利用了内存和总硅面积扩展昂贵的趋势 [45, 61]。 此类算法称为非对称性内存硬度函数，我们将其用于创建 Medallion。 这些函数有多种变体 [50, 75, 86]，但我们选择使用 Equihash。 Equihash 基于 k-XOR 生日问题，并通过时空权衡[1] 来提供内存硬度。 Equihash 可调节、简单、基于 NP 问题并且已在加密货币社区获得认可，因此我们认为使用此函数作为工作量证明基础可提供受到认可的安全性与未来适用性水平。

为生成 Medallion，一个对等点使用公钥 K 以及上一个以太坊区块哈希值 E，然后执行一系列计算以定位加密盐 S，使 $F(K, E, S, \ldots) \geq N$，其中 N 是难度比例因子。 当新的以太坊区块添加到链中时，必须计算新的 S 以保持 Medallion 最新。 Medallion 规范将在附录 C 中进一步说明。

5.2. 证明类型选择

熟悉其他基于市场的分布式网络的读者会认识到，我们对 Medallion 的使用前提与其他工作量证明系统（比特币等）类似。 其他读者可能会问： 为什么不使用权益证明、空闲证明或其他方法来获得加入 Orchid 协议（特别是 Orchid 市场）的认可？ 在这一节中，我们将说明为什么选择工作量证明而不是其他方法。

权益证明

权益证明取决于没有攻击者会控制大多数代币的假设。 由于我们的攻击模式包括动机充分、资源充足甚至专制性的政府，无法将权益证明假设视为始终正确。 比特币的惊人市值也远远低于一个中等规模国家的 GDP。 更为复杂的是，在不久的将来，我们打算将这个系统扩展到支持匿名支付，这将使得这种“敌意收购”更加难以发现。 因此，我们不能将 Medallion 建立在权益证明模式之上，因为系统中的足够权益会永久且不可逆转地损害 Orchid 协议的匿名性和安全性。 简而言之， 我们并没有使用权益证明，这是因为我们不想设计一个能将用户的隐私权出售给出价最高者的系统。

[1] 这一时空权衡让人联想到 [67] 首次发现的时空权衡，这绝非偶然。

空间证明

在空间证明中，与工作量证明系统中使用的资源相类似的计算资源交换为存储空间。 简而言之，空间证明是一个 *交互* 协议，其中两个参与者（一个证明者和一个验证者）进行交互，通过执行验证者指导的计算来验证证明者具有一定的存储空间。 假设这些计算仅在证明者存储并调用它们的情况下才可行 [63]。 尽管我们不确定是否会找到合适的方法，但我们正在为即将发布的 Orchid 协议版本探索使用空间证明的可能性。

空闲证明

空闲证明基于额外假设，即周期性同步工作证明足以证明用户在全局计算能力中的份额。 不过，在网络尚处于起步阶段（≤1,000 万 Peddler）的情况下，任何一家控制超级计算中心的公司仅需牺牲 1% 的计算能力就能控制网络。 在我们拥有足够数量的 Peddler 使攻击停止破坏之前，我们不会对此版本使用空闲证明，我们预计这会有很长一段时间。

5.3. Medallion 规范

概括来说，生成 Medallion 包括两个步骤，(1) 生成公钥/私钥对 K 并检索最近的以太坊区块摘要 E，(2)（以迭代或并行方式）定位加密盐 S，以使 $F_N(K, E, S)$ 在取胜条件下取胜，其中 N 是难度比例因子。 回想一下，Medallion 的目的是为特定实体提供工作量证明。 因此，每个 Medallion 都必须以密码方式链接到特定公钥，使 Medallion 不能用于模拟多个对等点。 此外，我们需要限制任何实体可以利用的预计算优势量。 因此，Medallion 以密码方式绑定到以太坊区块摘要，该区块摘要每 10 秒变化一次。 以下是 Medallion 规范的定义，

pk_m 是属于 peer_m 的唯一公钥

sk_m 是与 pk_m 关联的唯一密钥

e_t 是时间为 t 时的以太网区块摘要

$h(y)$ 是输入为 y 的密码哈希函数的摘要

$sig(sk, r)$ 是使用密钥 sk 的 r 的基本签名

$F_{n,k}(x_j)$ 是 Equihash 输出[2]，其开始计数器为 x_j，难度为 (n, k)

$seed$ 为 $h(e_t | sig(sk, e_t))$

$h(y)$ 可以是任何密码哈希函数，但 Orchid 协议使用 Keccak。 可以在附录 D.1 中找到有关此哈希函数选择的说明。 我们将基本签名定义为通过密钥对适当大小的数据求幂。

使用这些定义，我们将 Medallion 定义为以下集，

$$M = \{t, e_t, pk_m, sig(sk, e_t), F_{n,k}(seed)\}$$

用于在全局达成一致的 Equihash 难度参数 (n, k)。 有关这些参数的更多信息，请参阅 [50]。 请注意，使用 $seed$ 作为对 F 的输入会以密码方式将对等点的私钥与 Medallion 关联。 由于 Medallion 将确定对等点在 Orchid 市场中的 Chord 地址，因此拥有 Medallion 的任何实体可以通过与特定对等点关联的 pk_m 进行验证。 此外，实体可以从特定 Chord 地址请求公钥所有权证明。 Medallion 的设计细节在附录 C 中说明。

[2] 请注意，$F(x_j)$ 的输出是一组计数器 j，因此对于 $XOR_j = h(j)$，$\Sigma XOR_j=0$（对于输出中的所有 j）。

6. 支付

6.1. Orchid 支付要求

在大多数支付系统中，目标项的成本远远大于交易成本。 特别地，目标项的成本远大于将资金从一方转移到另一方的关联成本。 大部分网络购买也是这样，网络成本几乎可以忽略不计。 但是在 Orchid 网络中，目标项的成本是带宽。 即，通过网络发送的每个数据包都具有关联成本。 因此，如果用于发送付款的交易成本与单个数据包的成本一样低，则这些成本将相等。 当然，这将破坏 Orchid 协议的经济假设。

我们希望以任意精度出售带宽，并且**要求**交易费用足够低，因此我们需要一种新型支付系统，使用户能够以最小的交易成本来支付任意数量的中继流量。 现在，我们需要一种具有极低交易成本和任意带宽可分性的支付系统。 此外，Orchid 协议的目的是大大减少网络监视和审查。 因此，支付机制的其他要求必须包括：不可审查性、匿名性并且不依赖于可信任的第三方。 这就是说，即使底层网络不受监视和审查，但支付机制受到监视和审查，则系统仍然可以被利用来审查或跟踪用户。 同样，依赖于可信任的第三方也会使 Orchid 网络受到动机充分或强大的实体的影响，这些实体可能会影响支付提供商。

因此，Orchid 支付的要求包括：

1. *经济可行性*，支付应尽可能便宜。
2. *不可伪造性*，只有支付代币的所有者才能将代币用于支付。
3. *可用性*，没有实体可以阻止用户发送 Orchid 付款或收到付款。
4. *不可逆转性*，**任何**实体都不能反转过去的付款。
5. *匿名性*，参与者应与帐户地址、支付金额或时间无关。[3]

我们将在以下各节讨论可能的支付解决方案。 我们将证明 Orchid 支付（第 6.5 节）可满足除匿名性要求之外的所有要求。

6.2. 传统支付

在目前的金融支付系统中，交易是通过两个或两个以上实体（如银行或支付服务提供商 [2]）之间的谈判并使用协议（支付卡采用 ISO/IEC 7816 [3]，银行支付采用 EBICS [4]）来解决。 这些协议在 SWIFT [5] 和 NYCE [6] 等网络上运行，以支持国内和国际交易。 组成这些网络的每个实体都保留自己的分类账，并通过电子支付收据和手动调节对分类账持续更新 [7]。

连接到传统的支付网络通常需要大多数司法管辖区的特殊许可以及连接实体之间的逐案业务协议。由此产生的全球金融网络可以被视为连接企业和一系列协议与网络的许可临时网络。 每个分类账都是一个单点故障，缺乏密码的完整性，并且可由控制业务实体随意修改。

虽然典型的支付协议通常本身不规定交易费，但是运行这些协议的实体会额外收费。 单笔交易费可能从支付卡的几美分[8] 最高到国际电汇的 75 美元[9] 不等。 另外，许多系统会改为按交易金额的一定百分比收费，或者按该比例加收费用，可能达到银行转账交易[10] 金额的 13%，支付卡交易[11] 金额的 3.5%。

[3]理想情况下，不仅应对恶意观察者匿名，还应对恶意发送者或接收者匿名。

由于传统支付依赖于受信任方，因此在不牺牲我们所要求的属性的情况下，几乎无法将其用于 Orchid 网络。 尤其是，可逆性特意以逆向交易[77] 的形式存在。 交易通常难以伪造，但信用卡欺诈十分普遍，并且身份盗用或黑客入侵可能会导致用户账户遭到泄露。 此外，这些支付系统仅提供部分可用性，因为它们往往会在不合时宜的时间发生故障，还会定期停机。 由于管理支付的受信任方通常不仅具有发送者、接收者、付款金额和时间记录，而且往往还具有发送者的身份信息。最后，正如我们将在后续各节中看到的，在 Orchid 网络中，与传统支付类似的交易费将非常昂贵。

6.3. 区块链支付

比特币彻底改变了传统支付系统的现状，并且还在继续冲击着全球支付和国际转账市场。 比特币是一种不考虑地理边界的全球性网络和协议。 通过运用公钥加密，交易可以在用户自己生成的地址之间转移比特币金额，无需任何受信任方。 用户将生成密钥对，其中公钥哈希值可以用做支付地址，同时需要私钥对来自该地址的转账进行签名[12]。 比特币支付不可伪造，也不可逆[79]（在合理的时间内，用于完成区块确认）。 自创立以来，比特币网络发生的停机极少，除去矿机不太可能进行的主动审查（在第 6.6 节中进一步讨论）外，可以将其视为总体可用。 比特币支付是伪匿名的，匿名程度在很大程度上取决于网络的使用方式[68]。

一般来说，去中心化的加密货币有史以来第一次让人类和计算机系统二者在没有受信任方的情况下交易价值，对于像 Orchid 这样的激励型、分布式叠加网络来说，这是一个至关重要的要求。

比特币中的交易费不是由交易金额决定的，而是由交易数据结构的大小与发送者配置的系数的乘积决定的。 截至 2017 年，平均交易费用一直远低于 1 美元，但在 2017 年 2 月，由于比特币网络达到最大交易容量，因此交易费迅速上涨。 平均费用上升[13] 至 8 美元之高，使得依赖低费用的应用在比特币网络上变得不可行。

以太坊网络同样植根于公钥加密技术，也像比特币一样通过工作量证明加以保护，派生出了相同的不可伪造性、可用性和（非经典）不可逆性属性。 以太坊的交易容量更高，而且可动态调整，自 2015 年发布以来，以太坊网络的费用一直较低。然而，由于交易数量增加以及以太坊底层原生代币“以太”的价格上涨，交易费（称为 *gas*）也已增长[14] 至平均 0.20 美元，最高达到 1.00 美元。 执行智能合约代码的交易成本甚至更高，与所执行的计算量成比例。

在热门的公共区块链网络中，交易费的增加抑制了它们直接处理微支付的潜力，迫使微支付进入支付渠道等第 2 层解决方案。

6.4. 基于区块链的概率微支付

为了更易于表达核心理念，让人们了解如何将概率支付应用于区块链协议，我们这里将略去一些细节。 引用的原始论文中给出了 MICROPAY1 方案的正式描述，而第 6.5 节中正式确定了 Orchid 概率支付方案。

根据 Pass 和 Shelat 对 MICROPAY1[81] 的描述，数字签名和承诺方案以相互结合的方式构建放款条件，包括确切概率的随机结果。 首先，发送者通过将比特币转移到新生成密钥的第三方存管地址，进行“存款”。 然后，接收者（MICROPAY1 术语中的商户）选取一个随机数字，通过此数字向发送者发送承诺。 除了承诺，接收者还会提供新的比特币地址。 发送者也选取一个随机数字，并对此数字（以明文形式）、接收者的承诺和其他支付数据（如接收者提供的支付目的地地址）的串联进行签名。

验证所生成的票证时，需要确认接收者的承诺与他们显示的号码相匹配，还要确认发送者提供的签名与比特币存款地址相符。 如果发送者和接收者提供的随机数字的 XOR 后两位为 00，则该票证票证，接收者可以花费它。

直观来说，我们可以认为此方案中的“抛硬币”是无偏见的，除非发送者可以破坏承诺的绑定属性（或伪造签名），或者用户可以破坏承诺的隐藏属性。

请注意，发送者可以通过同时向多个接收者发行票证来“双重支付”他们的存款，或者，当看到接收者的票证索款时，通过支出广播抢先与接收者交易。 MICROPAY1 的作者讨论了如何通过“罚金第三方存管”来解决此问题，这是发送者存入的第二笔款项，可以在以后某个时间重新支出给发送者，在此之前，任何能够提交两张有效的票证来索取相同付款第三方存管的人都会将其“砍减”或“烧毁”。 这可以防止发送者与接收者共谋，或充当自己的接收者。

MICROPAY1 的作者在 MICROPAY2 和 MICROPAY3 中构建了迭代改进，引入了受信任方来对票证执行一些计算验证步骤，如果计算正确，则释放签名。

6.5. Orchid 支付方案

既然我们已经为我们的支付找到了合适的抽象，问题就变成了： 应该如何实施？

除了第 6.1 节中讨论的要求外，我们还希望满足以下要求：

- *可重用性*，用于构建每个新票证的方法不能要求为每个新票证支付新的交易费或进行新的链上交易，否则交易费将再次成为问题。
- 必须防止双重支出，否则将无法盈利。
- 就计算成本而言，系统必须具有足够的性能，以免压倒数据包的成本。

在这些要求中，最后一个要素可能最为棘手。 据我们所知，基于以太坊代币构建票证，且不需要与验证 ECDSA 签名类似的计算这种方法不存在的。 正如本节中所详述的那样，这源于以下要求，即发送者不仅要以加密方式向接收者证明加密票证的金额和中奖概率，还要证明发送者的以太坊账户已锁定足够数量的 Orchid 代币以用于发送票证。

因此，尽管不足以单独使用，但我们不得不采用类似于上述方法的交易余额法。 这又会导致新的要求，即“交易余额必须保持在足够小的水平，以免成为在交易过程中断开连接的诱因”。 这是由实际实施引起的机制设计问题，因此我们暂且假设存在解决方法，从而将重点放在实施上，同时将进一步的讨论推迟到第 6.9 节中进行。

Orchid 支付方案是一种受 MICROPAY1 和相关构想启发的伪匿名、概率微支付方案。 通过利用以太坊智能合约和可砍减的惩罚存款，它可以缓解抢先交易和并行（包括双重）支出攻击，而无需信任方。 Orchid 支付的伪匿名性等同于常规以太坊交易中所能实现的匿名性（但 Orchid 客户端采用了额外的隐私技术来实现有限的匿名性，如一次性地址以及节点身份和支付地址之间的密钥分离）。

MICROPAY2 和 MICROPAY3 中引入的受信方实际上可以被以太坊智能合约代码所替代。 EVM 允许实施用于验证微支付票证的任意逻辑（在计算的经济界限内），并为 ECDSA[71] 恢复操作以及密码哈希函数提供原语[89]。 附录 D 中讨论了此支付方案的详细描述。

6.6. Orchid 代币

为了满足不可伪造性、可用性和不可逆性支付要求，Orchid 网络目前使用基于以太坊的 ERC20 代币。 接下来各节讨论了我们如何能够降低 ERC20 转账的交易费，以便能够发送任意小额的代币。 第 6.10 节中讨论了支付匿名性。

Orchid 代币 (OCT) 用于 Orchid 网络内的付款。 Orchid 代币是一种新的、基于以太坊且兼容 ERC20 的定量供应代币。 代币供应量固定在 1×10^9（10 亿）个，每个代币各有 1×10^{18} 个不可分割的子单位（与以太相同的可分割性）。

乍看之下，以下各节中详细描述的 Orchid 支付系统可以配置为使用以太或任何 ERC20 代币。实际上，使用以太将会简化票证合约，略微降低 gas 成本并提高可用性，因为用户只需要以太，而不必同时获取 Orchid 代币和以太（用于支付交易费）。

不过，以太坊正计划未来进行协议升级，以允许通过任意机制来支付交易费，包括 ERC20 代币 [15] [16]。 这将消除使用新代币所存在的大多数弊端；gas 成本将不会有任何差异，用户只需要获取一种代币即可。 此外，还可以在合约执行中将 gas 价格设置为零，并向矿机添加 ERC20 代币支付（使用 EVM COINBASE[89] 操作码）[17]。 这需要矿机的明确支持，因为他们需要配置其挖矿策略以接受零 gas 价格，并确认交易执行中包括向 Coinbase 地址的 ERC20 代币转账。

然而，做出引入新代币而不是简单地使用以太的决定是出于社会经济而非技术原因。 通过创建新代币并使之成为 Orchid 网络中唯一有效的支付选项，我们精心设计了一些重大的社会经济效应，我们认为它们足以为提高复杂性提供合理性。

6.7. Orchid Gas 成本

通过上述方案的 Solidity 原型实现，我们测得 gas 成本大约为 87,000。 此成本用于完全执行票证索款 API（当使用中奖票证作为输入调用此 API 时）。 票证索款执行包括对 Orchid ERC20 分类帐 *转移* API 的子调用。 经过加密审核和最低限度的外部安全审计，所有 Orchid 智能合约的 Solidity 实现都将会开源。

6.8. 抗审查

与大多数公共区块链网络类似，以太坊交易无法进行审查，除非验证者（以太坊网络中的矿机）选择不将其包括在自己创建的区块中。 由于区块是在所有矿机中随机开采出来的，与哈希算力成比例，因此绝大多数矿机需要主动审查 Orchid 付款，才能显著破坏 Orchid 网络。 例如，即使 90% 的哈希算力选择不包括 Orchid 相关交易，Orchid 网络也将正常运行，但唯一需要注意的是，确认交易所需的时间平均长十倍。 更为严重的审查形式是，一大批矿机（例如 51%）选择拒绝包含 Orchid 相关交易的区块，以此来审查这些交易[73]。 根据以太坊协议的规则，这种做法是有效的，实际上将创造软分叉。 但是，组织大规模的矿机共谋创造这种软分叉伴随着巨大的利润损失风险；如果软分叉达不到足够的哈希算力，则共谋的矿机将错过它们的区块奖励。 除了利润风险外，考虑到以太坊矿机的去中心化特性，以及对区块链挖矿策略的法律和监管限制不足，我们认为上述可能性极小。

6.9. 交易余额

假设两位 Orchid 参与者 Alice 和 Bob 希望以完全匿名的方式进行交易。 Bob 执行某项任务将收费 x，而 Alice 将在 Bob 每完成 y 项任务后向他付款一次。 不幸的是，匿名性的本质是，如果不进行事先交易，Alice 和 Bob 就没有相互信任的机制。 他们可以合作吗？

如果 Alice 和 Bob 的关系存在准备成本 (S_{Alice}, S_{Bob} s.t. $S_{Alice} > xy, S_{Bob} > xy$)，则答案是可以合作：除非 (1) Alice 希望完成的工作量 $\leq xy$ 或 (2) Bob 能够执行的工作量 $\leq xy$，否则资金将会耗尽或工作不再具有经济合理性。 正如我们在讨论 Orchid 市场（第 4 节）时所看到的，Orchid 网络上

存在准备成本，该成本支持超过 1×10^3 个数据包的交易不平衡。 因为 Orchid 市场中的卖家通常比买家支付更高的准备成本，并且不对称的是，客户知道他们需要多少工作，所以 Orchid 网络需要客户预付款。

6.10. 匿名性

前几节中讨论的 Orchid 支付与常规以太坊交易一样是伪匿名的；所有交易都公开，包括金额以及发送者和接收者的账户。 Orchid 客户端旨在通过现代钱包技术（如一次性地址[18] 和使用 HD 钱包[19]）来改进公共区块链交易的默认伪匿名性，提供支付地址的不可链接性（尽管使用单一根密钥）。

随着以太坊“拜占庭”(Byzantium) 的发布，现在可以通过利用新的 EVM 原语进行椭圆曲线操作，实现具有合理 gas 成本的可链接环签名[20]。 通过将以太坊智能合约与 HD 钱包和可链接环签名提供的隐身地址相结合，可以实现诸如莫比乌斯 (Möbius)[76] 混合服务等一类混合技术。 莫比乌斯服务提供强大的匿名性保证，可使用基于博弈的混合服务安全模型以加密方式进行证明。 但是，与以前的混合技术不同，它提供针对恶意观察者和发送者的匿名性，而不针对恶意接收者。 莫比乌斯等服务与 Orchid 概率微支付相结合，可以让我们更接近在支付方面的最终要求 — 匿名性。

为了针对任何恶意行为者（无论支付的观察者、发送者或接收者）实现完全的匿名性保证，我们需要考虑零知识技术。

7. 带宽挖掘

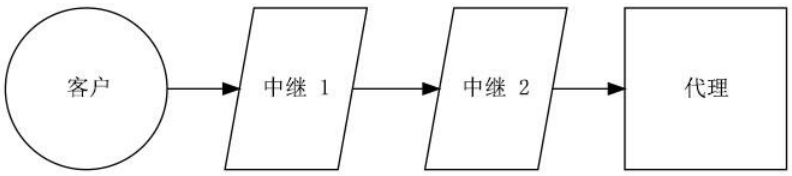

图 2： 用于客户流量路由的三 Peddler 链

在本节中，我们将描述中继和代理行为的规范，并讨论如何将这些节点“链接在一起”以支持不可审查的匿名 Web 浏览。

带宽销售的规范

中继节点实现了一种相对简单的行为模式：

- 维护一个或多个连接，每个连接都有自己的加密密钥。
- 检查收到的所有票证和兑现赢家。
- 监控交易余额，超过预先声明的额度时断开连接。
- 接收来自所有已开启连接的数据，并在消息边界执行解密。
- 按照如下所述处理解密后的消息：
  - 将所有非控制段转发到消息中指定的连接。
  - 处理所有控制段：

* *虚拟数据。* 指示中继丢弃此段。
* *按照一定的速率燃烧。* 指示中继通过连接以固定的速率发送数据，同时对数据包进行排队，在必要时生成数据以维持该速率。
* *Ratchet 票证。* 指示中继将票证传递给从其接收此数据包的对等方。
* *启动连接。* 指示中继建立新的连接。 在设置过程中使用，并处理断连。
* *初始 Web 连接。* （仅限代理。） 指示代理打开与指定主机的 SSL 连接。 为了支持白名单，这不能是原始 IP 地址。

在上述行为中，一个重要的考虑因素是，不会持续要求中继的工作量证明。 加上我们所有连接都是 WebRTC 连接，这为网站通过运行纯 javascript 中继代码潜在地对其访客变现保留了可能性。

有关通过应用特定的控制段可能实现的扩展的讨论，请参阅第 10 节。

保护节点和“带宽燃烧”

客户连接的中继具有非常重要的信息： 客户的 IP 地址。 我们假设客户希望尽可能将此地址保密，因此，默认客户端将倾向于以长期的对等方作为第一跳。

正如我们在讨论源于共谋的信息攻击（第 B.1 节）时所深入探讨的那样，关于第一跳节点的另一个问题是，它们所处的位置非常适合执行时序攻击。

为了防止这些攻击，我们建议注重隐私的用户采用一种称为*带宽燃烧* 的方法，向第二跳付款以便向客户发送固定的带宽量。 由于此方法会导致数据使用完全与网络使用不相关，因此该方法可以防止无法看到中继站 3 的入站流量的敌对者执行定时攻击。

为了向寻求规避的用户提供帮助（第 B.3 节），带宽燃烧还将支持由热门非 Orchid WebRTC 协议的统计属性确定的非固定速率。

链接

有意使用中继匿名访问 互联网 的客户将使用上述规范创建中继“链”。

8. 性能扩展

在本节中，我们将研究系统随着用户数量的增长将如何运行。

算法性能

大致而言，Orchid 协议包括三个部分： 基于以太坊的支付、流形和 Orchid 市场。

基于以太坊的支付随着以太坊正常交易量而扩展。 经过审视以太坊系统的设计，我们相信，即使 Orchid 网络非常成功，在以太坊总交易量中占有相当大的比例，该组件也将在设计公差范围内运行。

流形是带宽销售商（中继和代理）组成的链，它们的性能特征与 Orchid 网络参与者的总数无关。

Orchi 市场的核心运营基于经过充分研究的 Chord DHT。 Peddler 必须维持的连接数量以 $O(log(n))$ 速率增长，最大为 256 个连接。 网络上的查询需要 $O(log(n))$ 跳。 尽管随着网络规模的扩大，这些运营的确会变得更加繁重，但我们认为不会对性能产生任何重大影响。

稀缺资源的分配

Orchid 协议是围绕代币构建的。 通过价格发现，这些代币将允许妥善地处理买卖双方之间的平衡变化。

例如，当中继供应不足时，将不会为所有客户提供缓慢的体验，而是使客户参与竞标，以确定谁可以使用系统，直到供给不足得到纠正。相反，如果中继供应充足，则某些中继可能会离开系统，直到价格上涨为止。

真实表现

由于软件尚未完成，因此我们在此无法提供具体的数字。 在发布时，我们将使用以下图表更新此文档：

1. 随 Orchid 市场规模而变化的链设置时间。
2. 随 Orchid 市场规模而变化的 Orchid 市场加入时间。
3. 价格以多快的速度根据稀缺性和充裕性做出调整。
4. 在这里添加任何有趣的想法！

9. 外部库

Orchid 的功能基于一些重要的原语而构建。 由于某些读者可能不熟悉这些原语，或者可能熟悉 Orchid 网络中使用的特定属性，因此我们在这里简要总结一下。

WebRTC

WebRTC[47] 是一种最初为促进 Web 浏览器之间的实时通信而设计的系统。 它提供了优异的 NAT 实现和防火墙穿越方法，包括 STUN、ICE、TURN 和 RTP-over-TCP。 通过选择 WebRTC 作为我们的网络协议基础，而不是自定义编码的 TCP 和 UDP 网络代码，我们既可以获得这些技术的一流实现，又能够（在一定程度上）将用户的流量掩饰为一般的 Web 流量。

NACL

NaCL[48]（发音为“salt”）是 Daniel J. Bernstein 等人创立的加密库，致力于为构建高级加密工具创建所需的核心操作。 它之所以被选中作为此项目的加密原语源，源于它本身及其作者的卓越声誉。 除了以太坊智能合约加密代码外，下文描述的所有加密操作均使用 NaCL 来实现。

以太坊

以太坊[55] 是一个去中心化的区块链和平台，附带原生货币 (ETH) 和图灵完备的智能合约。 事实证明，智能合约对于 Orchid 的设计极为有用，可以让我们摆脱与跟踪付款余额以及 Orchid 付款票证的验证和公平性相关的设计问题。

10. 未来的工作

本节中的项目分为两类， 一类是值得拥有的功能，还有一类功能，我们内部对于是否向公众发布存在矛盾。 我们认为这种矛盾是普遍存在的，尽管几乎每个人都有喜欢的使用权力进行压迫的例子，但也有无数的例子证明权力可以为善。 诸如 Orchid 之类的协议没有自己的判断力，因此无法分辨它们是在为自由斗士还是恐怖分子、恶棍还是英雄路由流量。

空间证明

如第 5 节所述，我们对探索替代证明类型非常感兴趣。 这是一个重要的问题，原因包括两个方面，一方面因为工作量证明系统对环境造成的影响，另一方面因为我们当前的工作量证明算法要求功能完备的计算机充当网络路由器。

我们很高兴探索使用磁盘空间作为我们安全核心的稀缺资源的可能性，这可能使旧手机或类似硬件能够以有益的方式参与 Orchid 网络。

保护内容主机

许多先前的方法（第 2 节）发现，内容主机寻求与 Web 用户类似的保护。 在这一点上，我们内部存在矛盾，因为我们确实认为，有一些内容为了公共利益而不应自由分发（例如，与核武器制造有关的信息）。 但是，如果发生不可预见的情况，Orchid 可以在必要时进行扩展，以支持“不受限制、不受监视的主机”，如下图所示：

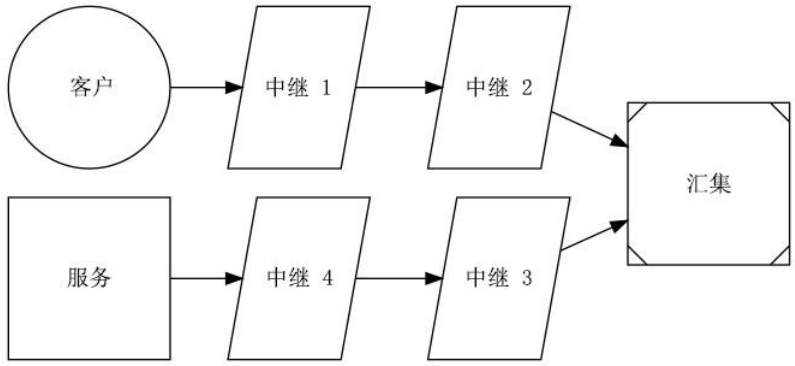

图 3： 在服务与客户之间充当中继的汇集节点

保护以太坊流量

正如我们在防火墙规避部分（第 B.3 节）中所讨论的，客户端的以太坊网络流量可能是薄弱环节。因为所有节点都必须维护此信息，所以 Orchid 协议似乎天生适合用来分发以太坊信息。

遗憾的是，如果依靠那些您向其付款的人来获得付款信息，将会导致一些棘手的问题。 我们希望在不久的将来加入这些信息，但不会在初始版本中加入。

Orchid 即平台

尽管我们预计核心系统的设计将在近期占用我们的大量时间，但我们非常感兴趣的是，添加支持以下用例的功能可能会大大增加通过 Orchid 网络路由的带宽量。

1. 网站使用 API 直接对接网络，并将代币融合到他们的服务中。
2. 网上文件存储和静态网站托管。
3. 文件共享。
4. 电子邮件/消息服务。
5. 仲裁/调解服务。

参考资料

[1] URL: http://www.meshlabs.org。
[2] URL: https://en.wikipedia.org/wiki/Payment_service_provider。
[3] URL: https://en.wikipedia.org/wiki/ISO/IEC_7816。
[4] URL: http://www.ebics.org/home-page。
[5] URL: https://www.swift.com。
[6] URL: http://www.nyce.net/about。
[7] URL: http://www.investopedia.com/terms/r/reconciliation.asp。
[8] URL: https://www.quora.com/What-are-common-credit-card-processing-fees。
[9] URL: https://www.nerdwallet.com/blog/banking/wire-transfers-what-banks-charge。
[10] URL: https://www.economist.com/blogs/dailychart/2010/12/remittances。
[11] URL: https://www.valuepenguin.com/what-credit-card-processing-fees-costs。
[12] URL: https://bitcoin.org/en/developer-guide#transactions。
[13] URL: https://bitinfocharts.com/comparison/bitcoin-transactionfees.html。
[14] URL: https://bitinfocharts.com/comparison/ethereum-transactionfees.html。
[15] URL: https://blog.ethereum.org/2015/07/05/on-abstraction。
[16] URL : https://blog.ethereum.org/2015/12/24/understanding-serenity-part-i-abstraction。
[17] URL: https://github.com/ethereum/EIPs/issues/662#issuecomment-312709604。
[18] URL: https://en.bitcoin.it/wiki/Address_reuse。
[19] URL: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki。
[20] URL : https://ropsten.etherscan.io/address/0x5e10d764314040b04ac7d96610b9851c8bc02815# code。
[21] URL : https://p16.praetorian.com/blog/man- in- the- middle- tls- ssl-protocol- downgrade- attack。
[22] URL: http://ethgasstation.info。
[23] URL: https://etherscan.io/token/OmiseGo。
[24] URL: https://solidity.readthedocs.io/en/develop/assembly.html。
[25] URL: https://hackernoon.com/zksnarks-and-blockchain-scalability-af85e350a93a。
[26] URL: https://hackernoon.com/scaling-tezo-8de241dd91bd。
[27] URL: http://mojonation.net。
[28] URL: https://en.bitcoin.it/wiki/Payment_channels。
[29] URL: https://en.bitcoin.it/wiki/Contract。
[30] URL: https://lightning.network/lightning-network-paper.pdf。
[31] URL: https://en.wikipedia.org/wiki/Mining_pool#Mining_pool_methods。
[32] URL : https://bitcoin.stackexchange.com/questions/1505/what-is-a-share-can-i-find-it- while-mining-solo-or-only-when-pool-mining。
[33] URL: https://blog.coinbase.com/app-coins-and-the-dawn-of-the-decentralized-business- model-8b8c951e734f。
[34] URL: http://onchainfx.com。
[35] URL: https://0xproject.com/。
[36] URL: https://etherdelta.com/#REQ-ETH。
[37] URL: https://augur.net。
[38] URL: https://gnosis.pm。
[39] URL : https://medium.com/@vishakh/a- deeper- look- into- a- financial-derivative- on- the- ethereum-blockchain-47497bd64744。
[40] URL: https://tools.ietf.org/html/draft-goldbe-vrf-00。
[41] URL: https://blog.ethereum.org/2017/10/12/byzantium-hf-announcement。
[42] URL: https://github.com/ethereum/EIPs/pull/213。
[43] URL: https://github.com/ethereum/EIPs/pull/212。
[44] URL: https://theethereum.wiki/w/index.php/ERC20_Token_Standard。

[45] Martin Abadi 等人。“Moderately hard, memory-bound functions”。 出处： *ACM Transactions on Internet Technology (TOIT)* 5.2 (2005)，第 299–327 页。

[46] Iddo Bentov、Rafael Pass 和 Elaine Shi。 “Snow White: Provably Secure Proofs of Stake”。*出处： IACR Cryptology ePrint Archive* 2016 (2016)，第 919 页。

[47] Adam Bergkvist 等人。“Webrtc 1.0: Real-time communication between browsers”。 出处： *W3C 工作草案* 91 (2012)。

[48] Daniel Bernstein、Tanja Lange 和 Peter Schwabe。 “The security impact of a new cryptographic library”。 出处： *Progress in Cryptology–LATINCRYPT 2012* (2012)，第 159–176 页。

[49] Jean-Luc Beuchat 等人。“High-Speed Software Implementation of the Optimal Ate Pairing over Barreto–Naehrig Curves” 。 出处： *4th International Conference on Pairing-Based Cryptography*。 Springer。 2010。 URL： https://eprint.iacr.org/2010/354.pdf。

[50] Alex Biryukov 和 Dmitry Khovratovich。 “Equihash: Asymmetric proof-of-work based on the gener- alized birthday problem”。 出处： *Ledger* 2 (2017)。

[51] N. Bitansky 等人。“Recursive composition and bootstrapping for SNARKs and proof-carrying data”。 出处： *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*。 ACM。 2013，第 111–120 页。URL： https://eprint.iacr.org/2012/095.pdf。

[52] Nikita Borisov 等人。“Denial of service or denial of security?”出处： *Proceedings of the 14th ACM conference on Computer and communications security*。 ACM。 2007，第 92–102 页。

[53] Michael Brown 等人。“Software implementation of the NIST elliptic curves over prime fields”。*出处： Topics in Cryptology—CT-RSA 2001*。 Springer。 2001，第 250–265 页。URL： https://pdfs.semanticscholar.org/ac3c/28ebf9a40319202b3c4f64cc81cdaf193da5.pdf。

[54] David Brumley 和 Dan Boneh。 “Remote timing attacks are practical”。 出处： *Computer Networks* 48.5 (2005)，第 701–716 页。

[55] Vitalik Buterin。 *Ethereum: A next-generation smart contract and decentralized application platform*。https://github.com/ethereum/wiki/wiki/White-Paper。 访问时间： 2016-08-22。 2014。 URL： https://github.com/ethereum/wiki/wiki/White-Paper。

[56] David L Chaum。 “Untraceable electronic mail, return addresses, and digital pseudonyms”。 出处： *Communications of the ACM* 24.2 (1981)，第 84–90 页。

[57] Shuo Chen 等人。“Side-channel leaks in web applications: A reality today, a challenge tomorrow”。 *出处： Security and Privacy (SP), 2010 IEEE Symposium on*。 IEEE。 2010，第 191–206 页。

[58] Alessandro Chiesa 等人。“Decentralized Anonymous Micropayments”。 出处： *Annual International Conference on the Theory and Applications of Cryptographic Techniques*。 Springer。 2017，第 609–642 页。

[59] George Danezis。 “The Traffic Analysis of Continuous-Time Mixes”。 出处： *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*。 第 3424 卷。LNCS。 2004 年 5 月，第 35-50 页。

[60] Roger Dingledine、Nick Mathewson 和 Paul Syverson。 Tor： *The second-generation onion router*。 美国海军研究实验室（华盛顿特区）技术代表，2004。

[61] Cynthia Dwork、Moni Naor 和 Hoeteck Wee。 “Pebbling and proofs of work”。 出处： *CRYPTO*。 第 5 卷。Springer。 2005，第 37–54 页。

[62] Kevin P Dyer 等人。“Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail”。 出处： *Security and Privacy (SP), 2012 IEEE Symposium on*。 IEEE。 2012，第 332–346 页。

[63] Stefan Dziembowski 等人。“Proofs of space”。 出处： *Annual Cryptology Conference*。 Springer。 2015，第 585–605 页。

[64] Christoph Egger 等人。“Practical attacks against the I2P network”。 出处： *International Workshop on Recent Advances in Intrusion Detection*。 Springer。 2013，第 432–451 页。

[65] Ittay Eyal 和 Emin Gu¨n Sirer。 “Majority is not enough: Bitcoin mining is vulnerable”。 出处： *International conference on financial cryptography and data security*。 Springer。 2014，第 436–454 页。

[66] Mike Hearn。 [*Bitcoin-development] Anti DoS for tx replacement*。 比特币开发邮件列表。 2013。 URL： https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002417.html。

[67] Martin Hellman。 “A cryptanalytic time-memory trade-off”。 出处： *IEEE transactions on Information Theory* 26.4 (1980)，第 401–406 页。

[68] J Herrera-Joancomart´ı。 “Research and challenges on bitcoin anonymity”。 出处： *Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance*。 Springer。 2015，第 3–16 页。URL： https://www.researchgate.net/profile/Jordi_Herrera-Joancomarti/publication/281773799_Research_and_Challenges_on_Bitcoin_Anonymity/links/55f7c7d408ae07629dcbc471.pdf。

[69] Douglas R. Hofstadter。 *Metamagical Themas: Questing for the Essence of Mind and Pattern*。 美国纽约州纽约市： Basic Books, Inc.，1985。isbn： 0465045405。

[70] Nicolas Houy。 “It Will Cost You Nothing to’Kill’a Proof-of-Stake Crypto-Currency”。 出处： *浏览器 下载此论文* (2014)。

[71] Don Johnson、Alfred Menezes 和 Scott Vanstone。 “The elliptic curve digital signature algorithm (ECDSA)”。 出处： *International Journal of Information Security*，第 1 期 (2001)。 Springer，第 3–63 页。URL： http://residentrf.ucoz.ru/_ld/0/34_Digital_Signatu.pdf。

[72] Aggelos Kiayias 等人。“Ouroboros： A provably secure proof-of-stake blockchain protocol”。 出处： *Annual International Cryptology Conference*。 Springer。 2017，第 357–388 页。

[73] Joshua A. Kroll、Ian C. Davey 和 Edward W. Felten。 “The economics of Bitcoin mining, or Bitcoin in the presence of adversaries”。 出处： *Proceedings of WEIS (Vol. 2013)*。 WEIS。 2013，第 11 页。URL： http: //www.thebitcoin.fr/wp-content/uploads/2014/01/The-Economics-of-Bitcoin-Mining-or-Bitcoin-in-the-Presence-of-Adversaries.pdf 。

[74] Thomas Locher 等人。“Free riding in BitTorrent is cheap”。 出处： *Proc. Workshop on Hot Topics in Networks (HotNets)*。 2006，第 85–90 页。

[75] Daniel Lorimer。 *Momentum–a memory-hard proof-of-work via finding birthday collisions*，2014。URL： http://www.hashcash.org/papers/momentum.pdf。

[76] Sarah Meiklejohn 和 Rebekah Mercer。 “Möbius: Trustless Tumbling for Transaction Privacy”。 出处： 2017。 URL： https://allquantor.at/blockchainbib/pdf/meiklejohn2017moebius.pdf。

[77] Adnan Noor Mian 等人。 “Enhancing communication adaptability between payment card processing networks”。 出处： *IEEE Communications Magazine, 53(3)*。 IEEE。 2015，第 58–64 页。

[78] Satoshi Nakamoto。 *Bitcoin: A peer-to-peer electronic cash system*。 2008。

[79] Arvind Narayanan 等人。 *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*。 Princeton University Press，2016。

[80] Sunoo Park 等人。*Spacecoin: A cryptocurrency based on proofs of space*。 IACR 技术代表。 Cryptology ePrint Archive 2015, 2015。

[81] Rafael Pass 和 Abhi Shelat。 “Micropayments for Decentralized Currencies”。 出处： *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*。 ACM。 2015。 URL： https://pdfs.semanticscholar.org/bca9/92b35d844160b30edbbf1809e17551d867ea.pdf。

[82] Ronald L Rivest。 “Electronic Lottery Tickets as Micropayments”。 出处： *International Conference on Financial Cryptography*。 Springe 。 1997。 URL： http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.135.2668&rep=rep1&type=pdf。

[83] David L. Salamon 等人。 “How to make Chord correct”。 出处： (2017)。 URL： https://orchidprotocol. com/whitepaper.pdf。

[84] Rabin Silvio 和 Vadhan。“Verifiable Random Functions”。出处：*Foundations of Computer Science，1999。第 40 届年度研讨会。* IEEE。1999。URL：https://dash.harvard.edu/bitstream/handle/1/5028196/Vadhan_VerifRandomFunction.pdf。

[85] Ion Stoica 等。“Chord: A scalable peer-to-peer lookup service for internet applications”。出处：*ACM SIGCOMM Computer Communication Review* 31.4 (2001)，第 149–160 页。

[86] John Tromp。“Cuckoo Cycle: a memory-hard proof-of-work system”。出处：*IACR Cryptology ePrint Archive* 2014 (2014)，第 59 页。

[87] Liang Wang 和 Jussi Kangasharju。“Real-world sybil attacks in BitTorrent mainline DHT”。出处：*Global Communications Conference (GLOBECOM), 2012* IEEE。IEEE。2012，第 826–832 页。

[88] David Wheeler。“Transactions using bets”。出处：*International Workshop on Security Protocols。Springer。* 1996，第 89–92 页。

[89] Gavin Wood。“ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER”。URL：https://ethereum.github.io/yellowpaper/paper.pdf。

[90] Pamela Zave。“Orchid: A Fully Distributed, Anonymous Proxy Network Incentivized Through Band- width Mining”。出处：*arXiv preprint arXiv:1502.06461* (2015)。

A. 拍卖

购买带宽时，对价格敏感的客户可能会被提供超低价格的攻击者利用。 例如，想象一个客户计划以最低出价购买长度为 4 的链。 知道这一点的攻击者可以将其价格设置得尽可能低，从而获得大于链中每个节点的中选机会 $\frac{a}{n}$。

为了解决这个问题，使用 Orchid 市场的客户选择带宽销售人群的一个随机子集，然后从创建自该随机子集的具备所需长度的可负担链集合中随机选择提供商。

在这种情况下，攻击者需要恰好位于所分配的位置，而且他们选择的价格也要恰好比随机选择的另一个卖家设置的价格更优惠。 即使设定了这种限制，常见的市场特性仍然保持不变，例如卖家能通过价格变化影响需求。

A.1. 附录概述

在本附录中，我们将研究注重隐私的带宽买家可使用的策略以及不同策略的效果。

我们推出了适用于分析此问题的 Orchid 市场的简化模型，并研究出了理论上和实例中的几种示例方法。 我们希望本节的读者能够理解我们在选择带宽购买算法时所做出的取舍，并劝止读者对其客户端硬编码以采用其他方法。

A.2. 简化的分析模型

评估拍卖策略时，无需考虑 Orchid 市场的全部复杂性。 为简化我们的分析，我们在此引入有关参与者目标的一组假设：

1. **卖家**。 卖家拥有 r 个“带宽槽位”，想出售这些槽位的访问权限。 卖家的唯一目标是将由此得来的收益最大化。
2. **攻击者**。 攻击者拥有 $a \geq 2$ 个卖家。 他们的唯一目标是让单个买家从他们的卖家那里购买多个槽位，这就叫做一次成功的攻击。
3. **买家**。 买家想要以最低的价格向三个不同的卖家购买三个带宽槽位（从而阻止攻击者）。

我们不关心 Orchid 市场的细节，而是假设所有买家都拥有所有当前 Medallion 持有人及其当前带宽价格的最新列表。 我们也不会关心中继和代理之间的区别，或者白名单和其他功能筛选所带来的复杂性。

现在我们已经掌握了基本结构，并了解了参与者目标，我们再来充实一下游戏结构：

1. **设定**。买家要使用的策略将告知给所有卖家和所有攻击者。 卖家要使用的策略将告知给攻击者，该策略可能因买家策略而改变。
2. **第一阶段**。 所有卖家都选择一个价格。 然后卖家的价格会透露给攻击者，然后攻击者再选择自己的价格。
3. **第二阶段**。 所有价格都会透露给买家。 买家将受到随机顺序的询问，从列表中选择最多三个报价。

4. **第三阶段**。 利润分配。

   (a) 卖家和攻击者收到钱，金额等同于选择他们的任何买家的报价。

   (b) 如果买家购买了三个槽位而没有遭受攻击，买家将获得特定于每个买家的一定利润额（减去这些槽位的付费）。

   (c) 成功攻击买家的任何攻击者都将获得特定于买家的赏金 U_b。

所以，这个市场就是一个 n 个玩家的游戏，我们要寻求三种策略： 买家应该怎么做，卖家应该怎么做以及攻击者应该怎么做。 在上述设计中，我们故意让买家“先行”，因为买家的需求最偏离经济常态。

有些读者可能不支持上述游戏中买家策略透露给攻击者的想法。 我们之所以明确指出这一点，是因为最初攻击失败的主动型攻击者仍可能获取特定买家所支付的大概价格的信息（例如，通过确定所有卖家的 IP 地址和价格，然后监视该买家的 互联网 连接以确定第一跳所使用的带宽销售商。） 随着时间的流逝，泄漏的这种信息可被用来推断买家采用的策略，因此，出于分析目的，我们认为最好假设攻击者拥有该策略。

成功标准

成功的解决方案将取决于以下标准的符合情况：

1. **安全**。 在预算和链长度固定的情况下，最大程度保护客户免受攻击。
2. **稳定性**。 这三个群体中，任何一个成员都不能通过改变策略来提高其个人效用。
3. **经济兼容**。 卖家可以提高和降低价格，以调节他们收到的采购订单的数量，从而可以采用常见方法最大化卖家的利润。

熟悉优化但不熟悉博弈论的读者应该特别注意稳定性，因为提出群体级别的“最佳策略”有着巨大的诱惑力，这样群体成员就能团结起来，保护他们的整体利益。 尽管这种行为表面上看似合理，但完全匿名的分布式市场中存在的激励结构阻碍了它们的稳定性。 例如，当卖家的在售槽位总数超过需求时，卖家可能联合起来，设定最低价格（在经济学中叫*托拉斯*，在博弈论中叫超理性[69]，在马克思主义中叫阶级革命。） 遗憾的是，因为将价格降低到托拉斯价格以下的任何卖家都可以获得额外的利润，所以这种安排是不稳定的。 因此，在此分析中我们不予以考虑。 这并不是说这种策略最终无法在此领域中应用；也许区块链技术的未来发展会支持对此类协议的遵守情况进行分布式验证。

可能令人惊讶的另一点是，我们试图显式确定攻击者最大程度利用买家的行为方式，并将这种策略的稳定性纳入“成功”的界定标准。我们这样做是因为没有其他手段可以界定特定方法的安全性。

A.3. 选择攻击

在我们深入讨论买家策略之前，首先思考攻击者的目标以及构成攻击的操作。 如果我们想象有一个非常偏执的买家，预算无限制，而且只有卖家列表中所含的信息，显然他们没有比随机选择更好的做法。 这使得攻击者的成功攻击率为 $\left(\frac{a}{n}\right)^2$。 由于这是最好的结果，所以我们考虑一种能够稳妥实现这种攻击几率的策略。

于是，该领域的攻击就是攻击者为将成功攻击几率提高到 $\left(\frac{a}{n}\right)^2$ 概率以上所使用的任何方法。

A.4. 候选策略

掌握了这种背景后，我们现在开始评估买家策略。

最低价格

如果买家选择成本最低的提供商，则有能力的攻击者会将其价格设置为允许的最低价格（0 个代币）。 这会产生 $\frac{a}{z}$ 的成功攻击概率，其中 z 是零收费的节点数。
例如，假设一个市场中有三个真正的卖家，每个卖家提供一个槽位，价格点为： {2, 4, 6}，买家的最高总价为 12。一个有能力的攻击者将以 {0, 0} 的价格进入市场，这样将保证交易达成，使得成功攻击的概率达到 1。

价格加权式随机

如果买家选择以带宽成本差异的某种单调函数作为购买概率，则会适度优化以下结果：

$$\frac{af(0)}{\sum_{e\in s} f(e)}$$

回到上面的示例，使用成本增量的的平方反比作为我们的函数，可得出成功攻击的概率为 $\frac{288}{337}\approx 85\%$。 虽然 85% 比 100% 好得多，但仍然不令人满意。

随机选择可负担的中继

如果买家选择随机选取收费低于其最高价格 $\frac{1}{3}$ 的卖家，有能力的攻击者会将其价格设置为等于或低于该最高价格。 拿不准时，攻击者可再次选择价格 0。

回到我们的示例，有能力的攻击者将以 0 和 1 的价格（或等效价格）进入市场，导致两种非攻击组合： $\{(0, 2, 4), (1, 2, 4)\}$，和两种攻击组合： $\{(0, 1, 2), (0, 1, 4)\}$。 此时成功攻击的概率为 $\frac{1}{2}$。

受制于成本的随机选择

如果买家选择随机选取卖家三元组 (S_i, S_j, S_k)，以使总成本小于或等于最大成本，那么攻击者可以选择这样的价格：(1) 让买家可负担的三元组数量最大化 (A_i, A_j, S_k)；(2) 让买家可负担的三元组数量最小化 (A_i, S_j, S_k)。

回到我们的示例，有能力的攻击者将以 1 和 4.1 的价格（或等效价格）进入市场，导致 5 种非攻击组合： $\{(1, 2, 4), (1, 2, 6), (1, 4, 6), (2, 4, 4.1), (2, 4, 6)\}$ 和三种攻击组合： $\{(1, 2, 4.1), (1, 4, 4.1), (1, 4.1, 6)\}$ 因此，成功攻击的概率是 $\frac{3}{8}$。

请注意，攻击者选择价格点 4.1 会实现有效的“挤出效应” — 包含 4.1 的四种组合中只有一种*不*是成功的攻击。 值得注意的是，即使在考虑了攻击者的这种行为之后，受制于成本的随机选择仍然比放弃考虑收费为 6 的卖家更好。

按对等点标准化的随机成本选择

人们很自然会问： 如果我们偏置随机样本以防止对等点被忽视，会怎么样？ 很遗憾，答案是攻击者会利用这种做法让自己获益。

继续上面的示例，有能力的攻击者将选择 1 和 6.1（或等效价格）为价格。 因为我们的买家只能承受 6.1（与 1 搭配时），所以调整几率会导致 $\frac{3}{7}$ 的成功攻击概率。

按对子标准化的随机成本选择

与上述思路类似的另一个问题是： 如果我们偏置随机样本，以使选中卖家*对子*的概率标准化，会怎么样？ 通过控制罕见的对子，攻击者同样能增加成功攻击的概率。

继续上面的示例，有能力的攻击者将再次选择 1 和 6.1（或等效价格）为价格，这次由于包含 6.1 的对子完全被忽视，所以攻击成功率为 $\frac{24}{49}$。

A.5. 稳定性分析

现在我们已经罗列了一些候选方法，那么问题出现了： 购买者会在激励下偏离给定策略吗？如果会，那么当攻击者试图利用混杂的买家人群时，每种策略的安全属性会发生什么变化？

为避免为了分析而分析，我们省略了在价格角度和安全角度上次优/不稳定的策略内容。

最低价格

最低价格相对于经济激励是稳定的，因为选择的价格已经是最低价格。 但是，从安全角度来看，它是次优的，所以不稳定。 具有安全意识的买家将使用其他策略，可能是受制于成本的随机选择。

这会导致一种有趣的情况，其中攻击者将被迫决定如何在这两种类型的买家之间分配他们的资源，以满足两个群体的安全效益，如下一节所述。

受制于成本的随机选择

与前面讨论的策略相反，这种策略从安全的角度来看是稳定的，但相对于经济激励而言却不是稳定的 — 不关注安全的买家将选择“最低成本”策略。

说这种策略从安全角度看是稳定的，可能会让某些读者感到惊讶。 也许 Orchid 市场上的一些买家会利用他们所掌握的攻击者如何利用受制于成本的随机选择的知识，创立自行改进的选择方法。如前所述，Orchid 市场无法阻止推断购买策略，更麻烦的是，没有办法得知攻击者对于其他所选方法的推断程度。 因此，对于足够偏执的买家，此方法是稳定的，因为它在最坏情况的假设下效果最佳。

但是，同样有许多 Orchid 市场买家忽略此建议，或只是采用最低成本选择方法，这对于有安全意识的买家来说是个好消息。 任何二次攻击优化都只会导致攻击者无法最优地利用受制于成本的随机选择。

A.6. 经济兼容性分析

现在，我们将注意力转向卖家策略问题。 我们的目标是说明卖家在多大程度上可采用常见的经济算法。

我们省略了在价格角度和安全角度上次优/不稳定的策略内容。

最低价格

由于这种方法是经济学中的预期情况，因此在经济上完全兼容。

受制于成本的随机选择

尽管此策略最初可能在经济上看似不兼容，但考虑到买家群体不共享最高价格，所以对卖家商品的兴趣频率呈现出熟悉的价格敏感型形状。

由于卖家通常可以提高和降低价格来调整收到的采购订单数量，因此“受制于成本的随机选择”在经济上是兼容的。

A.7. 结语

现在，我们已经分析梳理了 Orchid 市场上适合买家的拍卖方法，从而说明了为什么一般选择“受制于成本的随机选择”策略。

我们选择“纯随机”方法是基于这样的假设：攻击者将充分掌握买家的策略，而且攻击者将在合法的卖家定价之后选择自己的价格。 在这种情况下，偏置样本毫无用处，而且在最坏的情况下，还会使攻击者增加他们被选中的概率。 因此，我们没有偏置样本，而是将可选数量最大化，并从该范围内统一选择。

如果读者担心这种方法相对于较传统的拍卖模式增加了买家的成本，我们建议您将这种溢价视为“安全价格”，因为我们已经发现，为了降低价格而放任自己陷入被攻击的风险是完全不足取的。

B. 攻击与安全

B.1. 链上共谋攻击

在本节中，我们将研究控制或监视多个中继和/或 互联网 服务提供商 (ISP) 的攻击者可能推断或推理的信息类型。 假设中继和代理是随机选择的（因此 ISP 也是随机选择的），我们建立一个概率模型，其中特定的攻击可在不同的链长上执行。

各个中继和代理可用的信息

鉴于基于 IP 的网络的固有结构以及基于以太坊的付款所使用的 Orchid 协议，中继和代理节点*及其 IPS* 可获取以下信息：

- 它们连接的所有计算机的 IP 地址。
- 它们转发的数据包的大小、时间和数量。
- 控制支付给它们的代币的公钥。
- 导向给它们的任何控制段的内容。

此外，代理节点*及其 ISP* 可获取以下信息：

- Web 服务器的主机名，以及 SSL/TLS 会话协商的明文部分。

共谋的潜在各方

以下角色有权访问客户信息，因此可能有意共谋或成为攻击中被监视的对象：

- 客户、中继、代理或 Web 服务器的 互联网 服务提供商 (ISP)。 不可信概率为 s。
- 网站。 代理连接到的 Web 服务器。 不可信概率为 w。
- 中继 $_n$。 链中的第 n 个中继。 不可信概率为 $\frac{r}{n}$。
- 代理。 代理将带宽中继到 Web 服务器。 不可信概率为 $\frac{x}{n}$。

我们已经分离出了上面的 r 和 x，因为虽然攻击者无法控制他们可用于工作量证明计算的计算总量，但他们可以控制如何在中继节点和代理节点之间分配计算。

攻击类型

共谋攻击的中心目标是，将特定的 Orchid 客户与特定的 SSL 连接关联起来。 达到此目标的方法有两种：

- 关系。 在可行的情况下，攻击者可以推理出客户正在与特定的网站对话，因为他们可以观察该路线上足够的点。
- 定时。 在可行的情况下，攻击者可以通过先后控制和观察数据包的时间来推断客户正在与特定的网站对话。
- 不燃烧。 在可行的情况下，尽管客户使用带宽消耗，但攻击者仍可以执行定时攻击。

“普通”访问 互联网： 无中继、无代理

尽管客户直接连接网站时自然不会使用 Orchid 系统，但我们认为有必要检查此设置中存在哪些信息风险，以便为其余的分析打好基础。

ISP	网站	P（关系）	P（定时）	P（不燃烧）
x		s		
	x	w		

在上表中，“X”表示参与共谋，而 P（关系）和 P（定时）中的值表示发生这种情况的概率。 此处省略了不可能发生攻击的行，以及带有无关“X”的行，在可能发生较简单攻击的情况下，也没有提及较复杂的攻击。

VPN： 无中继、无代理

为了给我们的分析奠定基础，我们还说明了 VPN 访问固有的共谋风险。

ISP	VPN	网站	P（关系）	P（定时）	P（不燃烧）
	x		g		
x		x	sw		

其中 g 是 VPN 提供商受到监视或与对手共谋的概率。 请注意，g 可能会随着时间的推移变得难以建模，例如使用 VPN 会导致此情况。

无中继、一个代理

ISP	代理	网站	P（关系）	P（定时）	P（不燃烧）
	x		$\left(\frac{x}{n}\right)$		
x		x	sw		

毫不奇怪，这种情况下的风险与使用 VPN 的风险非常相似。 不使用中继的链等效于 VPN，其中在每个浏览会话之前随机选择一个新的 VPN 提供商，并且该 VPN 提供商不存储任何个人信息。

一个中继、一个代理

ISP	中继 $_1$	代理	网站	P（关系）	P（定时）	P（不燃烧）
	x	x		$\left(\frac{rx}{n^2}\right)$		
	x		x	$w\left(\frac{r}{n}\right)$		
x		x		$s\left(\frac{x}{n}\right)$		
x			x		sw	

如果在此配置中使用带宽燃烧，可以缓解所有定时攻击。 请注意，在定时案例中添加中继 $_1$ 或代理可以产生关系。

两个中继、一个代理

ISP	中继 1	中继 2	代理	网站	P（关系）	P（定时）	P（不燃烧）
	x		x		$\left(\frac{rx}{n^2}\right)$		
x		x	x		$s\left(\frac{rx}{n^2}\right)$		
	x	x		x	$s\left(\frac{r}{n}\right)^2$		
x		x		x	$sw\left(\frac{r}{n}\right)$		
	x			x		$s\left(\frac{r}{n}\right)$	
x				x		sw	

如果在此配置中使用带宽燃烧，可以缓解所有定时攻击。 在中继 1 和网站实施定时攻击的情况下，将中继 2 或代理添加到共谋会导致一种关系。 如果客户的 ISP 与网站共谋，添加中继 2 会导致一种关系。

三个中继、一个代理

ISP	中继 1	中继 2	中继 3	代理	网站	P（关系）	P（定时）	P（不燃烧）
	x	x		x		$\left(\frac{r^2x}{n^3}\right)$		
	x		x	x		$\left(\frac{r^2x}{n^3}\right)$		
x		x		x		$s\left(\frac{rx}{n^2}\right)$		
	x		x		x	$w\left(\frac{r}{n}\right)^2$		
x		x	x		x	$sw\left(\frac{r}{n}\right)^2$		
	x				x		$s\left(\frac{r}{n}\right)$	
x					x		sw	
	x	x			x			$s\left(\frac{r}{n}\right)^2$
x		x			x			$sw\left(\frac{r}{n}\right)$

B.2. SL 和 TLS 漏洞

SSL 和 TLS 是复杂的协议，随着实现缺陷的发现，不断收到安全更新。 遗憾的是，用户有时会延迟升级软件，使用不可信或编写不良的软件，还会错误配置软件。 为了尽可能保护用户，Orchid 协议提供了“健全性检查”功能。

SSL 降级攻击

在所谓的 *SSL 降级攻击* 中，攻击者会导致安全连接使用质量差的加密技术 ([21])。 为执行此攻击，攻击者只需从初始密钥协商数据包中删除客户端支持的较安全的加密方法提示。 为了防止这种攻击，Orchid 客户端尽可能自动反向操作 — 从密钥协商数据包中删除提及的不安全选项的内容（“SSL 升级”攻击）。

旧浏览器和手机应用

Web 浏览器中会定期发现并修补 SSL 和 TLS 安全漏洞。 但是，我们不能假定所有用户都使用最新的浏览器。 手机应用程序也会发生类似情况，开发人员有时会忽略诸如 SSL 证书验证之类的程序。

为了解决这些问题，Orchid 客户端使用最新的“Boring SSL”（Google Chrome 中使用的开源 SSL 库）自动核实证书链。

B.3. 防火墙规避功能

如果只有能免费和开放访问 互联网 的用户可以使用上述系统，那么该系统几乎没有用处。 在本节中，我们将讨论便于由攻击者提供其 互联网 访问权限的那些用户访问的功能。

请注意，如果攻击者有意完全阻止所有 互联网 访问，则无法在此方面进行防御。 因此，本节中的所有防御分析都假定攻击者为地毯式阻止付出了一定的代价，并力图充分利用这一代价，以免攻击成本过高。

自举

我们预计防火墙提供商尝试对 Orchid 网络进行的首次攻击之一是，创建入口 Peddler 列表，并阻止对它们的所有访问。 这是因为，如果客户无法访问入口 Peddler，则他们将无法使用网络。更复杂的是，我们必须假定有能力的攻击者拥有客户可用的任何 IP 地址的列表。

为了初步解决这个问题，我们将提供一项服务，允许用户学习新的中继 IP 地址以换取工作量证明。 为防止自举反复被阻止，我们将通过 Web、电子邮件和流行的即时消息传递平台提供对该自举服务的访问。 用户将客户端选项屏幕上的挑战复制/粘贴到最方便的通信机制中，然后将回复再复制/粘贴回客户端。

DPI、推断和主动探测

较复杂的防火墙采用深度数据包检查（DPI；分析数据包内容，而不只是报头）、时间推断（使用数据包大小、数量和时间的聚合统计度量）以及主动探测（尝试与要连接的用户或服务器建立连接，以尝试识别所提供的服务）等方法。

我们不期望使用深度数据包检查或主动探测方法来提供重要信息。 我们通过使用 WebRTC，对所有通信加密，并且除非发出活动的 WebRTC 报价，否则不开放端口。 由于这与 WebRTC 的所有其他使用行为匹配，所以此行为不能澄清 Orchid 用户身份。

定时推断可能是检测 Orchid 用户的一种有效方法，因为加密流上 Web 请求的时间和大小不可能类似其他类型的 WebRTC 流量 ([62])。 为了解决这个问题，我们鼓励在可能发生推断攻击的情况下访问 Orchid 网络的用户使用“带宽燃烧”（第 7 节）。

披露： 以太坊流量

由于当前客户端使用以太坊客户端跟踪付款状态，并且以太坊具有自己的非强化网络签名，因此与这种以太坊流量相关的检测可能是薄弱环节。 防火墙运营商可能只会问：“计算机是否正在运行以太坊*并*在消耗大量的 WebRTC 流量？”

为了保持项目有的放矢，以太坊的强化和/或通过 Orchid 网络提供以太坊流量不是我们最初版本的功能。 我们计划在将来的版本中解决此问题，请参阅第 10 节。

B.4. 攻击分析和攻击者用户案例

压制性 Web 应用程序

攻击目标： 识别所有 Orchid 中继和代理 IP 地址。

因为 Orchid 市场包含所有中继和代理，所以这种攻击与第 B.3 节中讨论的相反。

Orchid 市场上的强制连接数量以 $O(log(n))$ 的速度增长，其中 n 是网络规模。 如果攻击者持有 m% 的全局算力，他们将在每次完成工作量证明计算时学习 $log(n)$ 个 IP 地址。 因此，在 c 次迭代中，他们将学习 $1 - (1 - \frac{\log(n)}{n})^c$ % 的中继和代理 IP 地址。

熟悉如何阻止 Tor 流量淘金的读者可能会担心，上面描述了系统的一个可怕问题。 幸运的是，事实并非如此。 Tor 大约有 1,000 个出口节点，很容易筛选。 就我们而言，主要是由于我们支持白名单，所以预计有数十万个出口节点。 除了使用上述方法给解除屏蔽造成更大的计算挑战之外，阻止这些 IP 地址还将导致压制性 Web 应用程序阻止其自己的用户。

企业网络和“大”防火墙

攻击目标： 阻止接受您提供的 互联网 访问权限的用户使用 Orchid 网络。

第 B.3 节将更详细地讨论与此相关的功能。 该攻击者的前景黯淡： Orchid 网络用量表现为中继转发固定数量数据的 WebRTC 连接。 既没有可探测的开放端口，也没有可依赖的 IP 地址“淘出”它们。

被动监视和推断，可能伴随 Sybil 攻击

攻击目标： 学习客户 IP 标识和网站标识。

第 B.1 节将更详细地讨论与此类攻击有关的分析。 该攻击者的前景黯淡： 将自己定位在链中多个位置的难点在于，攻击者需要拥有（并专门为此攻击分配）很大比例的全局算力。

短时骚乱和 QoS 攻击

攻击目标： 攻击者希望在尽可能大面积的网络上造成混乱。

我们鼓励有安全意识并乐意参与的读者在这一领域内进行分析。 这里的任务是，在有限的预算（可能约为 10,000 美元）下，尽可能破坏网络。

攻击链 — 攻击者可尝试各种攻击： 随机丢弃数据包，仅提供非常慢的服务，间歇性提供慢速服务或只是断开连接。 在所有情况下，客户都只是替换相关节点，这给所有客户带来了很小的不便。如果丢弃数据包，可能还给其他中继带来不便，因为系统可能无法确定是否 A 没有转发该数据包，或者是否 B 谎称没有接收该数据包。 在这种情况下，客户将更换两个中继。

攻击 Orchid 市场 — 攻击者在这种情况下有很多选择。

- 不当实施加入协议。 我们不认为这是一种攻击，因为在这种情况下，我们的“攻击者”只是向其他 Peddler 支付了数据包转发费。
- 加入 Orchid 市场，拒绝向用户提供路由表信息，或者拒绝转发数据包。 这将给 $\frac{log(n)}{n}$ 的 Orchid 市场查询造成一些额外的路由负担，直到相关的 Peddler 与网络断开连接。
- 在 Orchid 市场中不提供服务。 在这种情况下，客户进行的拍卖效率降低（$\frac{log(n)}{n}$ 的时间损失一个参与者）。

C. Medallion 工程规范

此附录建立在第 5.3 节中的 Medallion 总体规范的基础上，旨在更准确地定义 Medallion 及其生成过程。 注意，此附录建立在 [50] 中使用的符号的基础上。 以下列表按类型排序，

t **(uint)** – unix 时间，精度为 p；精度必须至少在 100ms 左右

skm **(uint)** – 随机选择的 x 密钥

e_t **(uint)** – 时间为 t 时的以太坊区块摘要（又名区块哈希）

$h(y)$ **(uint)** – 输入为 y 的 Keccak 摘要

$seed$ **(uint)** – $h(e_t \mid sig(sk, e_t))$

pk_m **(tuple)** – 公钥 $x * G$，位于椭圆曲线 C 上，基点为 G，阶为 N

$sig(sk, r)$ **(tuple)** – r 的 ECDSA 签名，使用密钥 sk

$F_{n,k}(x)$ **(struct)** – $\{n, k, x, i_0, ..., i_2k\} : n, k, x, i_j \in Z_{|h|}$

M **(struct)** – $\{t, e_t, pk_m, sig(sk, e_t), F_{n,k}(seed)\}$

Medallion 算法

建议使用两种方法生成 Medallion。 第一种是非交互式的，要求 Medallion 的生成具有当前的以太坊区块摘要以及一个公钥对。 下面给出的 Equihash 工作量证明已经简化。

算法 1： 非交互式 Medallion 生成

准备步骤：

$sk \leftarrow \text{random} \in Z_{|h|}$
$pk \leftarrow sk * G$
$sig(sk, e_t) \leftarrow \text{ECDSA}(sk, e_t)$
$seed \leftarrow h(e_t, sig(sk, e_t))$

Equihash 工作量证明：

set difficulty (n, k)
set counter $i_1 = seed$
set $\{i_j\}$ of 2^k items
while$\{h(i1) \oplus h(i_2) \oplus ... \oplus h(i_2{}^k) \neq 0\}${
build bigger list of $\{i_j\}$
find subsets of colliding $\{i_j\}$
sort $\{h(i_j)\}$
}

返回： $t, e_t, pk, sig(sk, e_t), \{i_j\}$

第二种方法建立在第一种方法的基础上，增加了让 Orchid 市场中的 Peddler 参与 Medallion 建设的要求。 通过要求 Orchid 市场的参与，创建一个挑战应答类型的协议，就好比原始的时间量证明。

在这个方案中，Orchid 市场上有 m 个行动者、一个 Medallion 生成者、*Alice* 和一个 Peddler 社区，表示为 $pi \in \{Bob, Chris, Dana, ...\}$。 *Alice* 将与入口 Peddler *Bob* 互动，后者将计算 $sig(skBob, e)$ 并将签名返回给 *Alice*。 如果需要 Orchid 市场的进一步参与，*Bob* 将联系 m 个随机 Peddler，后者将 $sig(sk_{pi}, e)$ 通过 *Bob* 返回给 *Alice*。 然后，*Alice* 将计算 *seed*，期间额外输入 $\{sig(sk_{pi}, e)\}$，并将 M 返回给 *Bob*。

算法 2： 以响应为取向的 Medallion 生成

准备步骤：

$sk \leftarrow \text{random} \in Z_{|h|}$

$pk \leftarrow sk * G$

$sig(sk, e_t) \leftarrow \text{ECDSA}(sk, e_t)$

互动步骤：

执行时间量证明

$seed \leftarrow h(et, \{sig(sk_{pi}, e_t)\})$

Equihash 工作量证明：

set difficulty (n, k)

set counter $i_1 = seed$

set $\{i_j\}$ of 2_k items

while$\{h(i_1) \oplus h(i_2) \oplus \ldots \oplus h(i_2k) \neq 0\}\{$

build bigger list of $\{i_j\}$

find subsets of colliding $\{i_j\}$

sort $\{h(i_j)\}$

}

返回： $t, e_t, pk, sig(sk, e_t), \{i_j\}$

D. 支付协议和定义

D.1. 支付代币密码函数选择

为了降低 Orchid 微支付的成本，我们只选择了某些密码函数，是因为它们与其他任意函数相比，减少了以太坊 gas 成本。

h **(Keccak-256)** – Orchid 协议中可使用的任何安全的密码哈希函数。 但是，我们的系统特别选择了 Keccack 而不是其他哈希，是因为它在 EVM[4] 中的所有可用哈希函数中 gas 成本最低。 选择它是为了进一步充分降低 Orchid 交易成本。

sig(*sk, r*) **(ECDSA)** – 使用 secp256k1 曲线，并将 Keccak-256 作为内部密码哈希函数。 同样，做出这种选择是因为 EVM 内置对 ECDSA 的支持，从而降低了 Orchid gas 成本。 此外，选择的曲线与现有的区块链软件库和工具兼容。

D.2. 支付代币定义

Orchid 支付代币具有以下字段：

h **(function)** – 密码哈希函数

timestamp **(uint32)** – 表示值何时开始下降的时间

recipient **(uint160)** – 代币接收者的 160 位以太坊帐户地址

rand **(uint256)** – 由*接收者*选择的随机整数

nonce **(uint256)** – 代币发送者选择的随机整数

faceValue **(uint256)** – 中奖代币的价值

marketValue **(uint256)** – 基于带宽市场的代币预期价值

acceptedValue **(uint256)** – 根据接收者所接受价值的代币预期价值

winProb **(uint256)** – 特定代币赢得发送者的 *faceValue* 的概率

randHash **(uint256)** – *h*(*rand*) 的摘要

ticketHash **(uint256)** – *h*(*randHash, recipient, faceValue, winProb, nonce*) 的摘要

(*v*1, *r*1, *s*1)（**元组**）– 彩币发送者的 ECDSA 签名元素

(*v*2, *r*2, *s*2)（**元组**）– *接收者*的 ECDSA 签名元素

D.3. 支付彩币生成

假设 *Alice* 为接收者，*Bob* 为发送者，

1. *Alice* 选择随机一个 256 位数字 *rand*，计算 *randHash*，然后将摘要发送给 *Bob*
2. *Bob* 选择 (*nonce, faceValue, winProb, recipient*) 的值[5]
3. *Bob* 计算 *ticketHash*

[4]以太坊成本 36 gas [89]，哈希 32 个字节

[5]使用本规范的信息，例如： *接收者*签名的一般带宽市场数据和公共功能

4. *Bob* 计算 Sig(PrivKey, *ticketHash*)
5. 生成的彩币包括：
   (a) *randHash*
   (b) *recipient*
   (c) *faceV alue*
   (d) *winProb*
   (e) *nonce*
   (f) *ticketHash*
   (g) *creator*（对 *ticketHash* 签名的发送者密钥的地址）
   (h) *creatorSig*（发送者对 *ticketHash* 的签名）

请注意，虽然此彩币在接收者可以完全验证的情况下有效，但接收者需要对其进行签名（请参阅下文），才能在 Orchid 支付以太坊智能合约中进行索款。

D.4. 支付彩币验证

Alice（带宽销售者）随后将执行以下操作，

验证：

(a) $randHash = H(rand)$

(b) $faceV alue \geq marketV alue$

(c) $winProb \geq acceptedV alue$

(d) *recipient* = {接收者发布的以太坊帐户地址}

(e) *creator* = {发送者发布的以太坊帐户地址}

确认：

(a) 确认： *creatorSig* 是私钥签名，私钥的对应公钥为创建者地址

检查：

(a) 确认： *创建者* 将足够的 Orchid 代币锁定在 Orchid 支付智能合约中

断言：

(a) 彩币现已证明有效，并可能是中奖彩币

D.5. 申请彩币兑付

虽然接收者可以在本地充分验证彩币是否有效以及是否为中奖彩币，但中奖彩币中代币的实际支付将通过 Orchid 支付智能合约完成。 Orchid 智能合约公开了一个将以下参数作为输入的 Solidity API，

1. *rand*
2. *nonce*
3. *faceV alue*

4. *winProb*
5. *receipient*
6. *recipientSig*（*接收者* 对 *ticketHash* 的签名）
7. *creatorSig*（发送者对 *ticketHash* 的签名）

D.5.1. 智能合约的执行

假设 *Alice* 是希望购买带宽的用户。 *Alice* 必须拥有以太坊帐户地址 *addressAlice* 和 Orchid 代币。 请注意，此地址将具有关联公钥 *PubKeyAlice*。 Alice 还必须将 Orchid 代币*锁定* 在以太坊智能合约中（如先前章节的定义），并通过 *PubKeyAlice* 锁定。 在上一节中，*Alice* 的地址是以太坊帐户地址，等同于通过 *ticketHash* 从 *creatorSig* 恢复的公钥。

假设 SLASH 是一个临时布尔值，设置为 FALSE，*PubKey* 是通过 *ticketHash* 从 *recipientSig* 恢复的公钥，

计算：

(a) *ticketHash*

验证：

(a) *randHash*：如果不是，中止执行[6]。

(b) *PubKey* = *接收者* 地址；如果不是，中止执行。

(c) *addressAlice* 将 Orchid 代币锁定在罚金存管帐户中。 如果不是，中止执行。

(d) *addressAlice* 已将足够的 Orchid 代币锁定在其彩币帐户中以支付彩币。 如果不是，请将 SLASH 设置为 TRUE 并继续执行。

(e) *H*(*ticketHash, rand*)[7] ≤ *winProb*。 如果不是，中止执行。

确定：

(a) 如果 SLASH = FALSE，则支付彩币： *faceV alue* 从创建者的彩币资金转移到*接收者*。

(b) 如果 SLASH = TRUE，则创建者扣减。

结算：

(a) 将创建者的彩币资金（如果有）发送给*接收者* （这是来自之前保证小于 *faceV alue* 的确认）。

(b) 将创建者的罚金存管帐户设置为零（销毁/扣减这些代币）。

请注意，虽然罚金存管扣减通过对彩币发送者建立抑制措施（发送者的损失将超过他们从可能的双花中获得的利润）以避免双花，但仍存在彩币发送者大规模超额支出的风险。 为解决这一问题，中奖彩币的价值应在 *timestamp* 时呈指数降低，从而积极激励中奖者立即兑现。 接收者可以使用这种即时性来计算发送者的 Orchid 代币余额的"浪费率"。

[6]由于交易被中止，因此不会发生以太坊状态转换，也不会支出任何 gas
[7]解析为 uint256

E. 其他支付详细信息

E.1. 数据包的成本是多少？

为便于讨论，假设数据包的平均长度为 1 KB。 我们希望计算数据包平均总成本的合理上限。 我们观察到，Amazon Web 服务的新加坡 CloudFront 是最昂贵的带宽提供商之一，每 1×10^9 个字节收取 0.14 美元。 因此，每个数据包的成本为 1.4×10^{-5} 美分（0.00000014 美元）。

由于带宽对大多数用户来说是浪费性商品（任何未售出的带宽将永远丢失），因此，Orchid 市场的实际带宽价格可能会大大低于此上限。

E.2. 以太坊交易成本

以太坊智能合约允许创建复杂的支付机制，利用以太坊虚拟机 [89] (EVM) 的功能和灵活性，在经济范围内提供图灵完整的执行环境。 以太坊智能合约执行的每条指令都会增加原始交易的交易费用。

每条 EVM 指令都会花费一定数量的 gas，以太坊交易费用定义为交易花费的 gas 总量乘以发送者设置的 gas 价格。 矿工选择任何有效的交易纳入其挖掘的区块，可以包括任何 gas 价格（包括零）的交易。 选择 gas 价格较高的交易可能会带来更多利润，因为每个区块都具有可以包含的交易数限制。 同样，接受较低的 gas 价格也可能带来更多利润，因为如果网络没有以最大容量运行，这可使矿工填满他们的区块。 这种机制创造了一个不断变化但稳定的博弈理论平衡，这一平衡被以太坊加油站这样的站点跟踪 [22]。

截至 2017 年 10 月，在几个区块内获得高概率交易的成本为 0.026 美元。 要在 15 分钟内确认，0.006 美元就足够了。 这些估计数字是交易的基本成本，无需执行任何智能合约代码的普通以太币转账成本为 21,000 gas。 如果交易执行智能合约代码，则每条 EVM 指令都会另外增加 gas 成本。 例如，将新的 256 位值永久存储在智能合约存储中需花费 20,000 gas，更新现有值将花费 5,000 gas。

以太坊 ERC20 分类账只是帐户地址到余额的映射，因此 ERC20 代币转账成本约为：新帐户需要 21,000 + 20,000 gas，后续转账需要 21,000 + 5,000 gas（因为接收者帐户已在代币分类账中具有条目）。 观察实时 [23] ERC20 交易，我们发现转账到新帐户和现有帐户的 gas 成本略高，分别约为 52,000 和 37,000 gas。 不同之处在于智能合约代码执行不变量验证，例如发送者是否具有足够余额以及其他实现细节（支付收据的记录）。 50,000 gas 需要 0.014 美元到 0.062 美元的交易费，具体取决于我们希望交易确认的速度。

E.3. 性能

尽管 Orchid 支付智能合约不可变，但可以通过部署新合约并升级 Orchid 客户端软件以指向新合约来有效升级智能合约（同时在需要时仍与旧合约后向兼容）。 以太坊智能合约支持多种优化措施以降低 gas 成本，我们预计 Orchid 支付智能合约的将来版本将使用内联 Solidity 汇编 [24] 等方法来优化 gas 成本，这类似于常规软件系统通常使用内联汇编来代替昂贵的子例程。

不过，密码操作是验证 Orchid 支付彩币的瓶颈，例如 ECDSA 恢复以及 Orchid 代币分类账中发送者和接收者条目的状态更新。 在这里，一次改进可能需要使用两次具有智能合约 API 调用负载的以太坊交易的接收者签名，因为签名包含彩币数据结构。 目前，Orchid 方案定义了两个签名以实现 Orchid 客户端的灵活性并更轻松地指定和推断出支付方案，而不依赖以太坊细节。 更简单的优化包括紧密封装彩币字段，并将单个 256 位字中的多个内部变量进行编码，以与 EVM 堆栈字和永久合约存储槽（均为 256 位）对齐。

另一方面，为了实现更大的匿名性，可选甚至强制使用混合技术可能会大大增加 Orchid 支付的 gas 成本。 使用基于可链接环签名的混合服务很容易导致交易费用高出约一个数量级 [20]。 但是，如果这能提供强有力的匿名保证，用户可能会认为这是值得的。 由于我们可以轻松调整 Orchid 支付的概率变量，包括彩币频率、中奖概率和中奖金额，因此我们可以调整彩币索款之间的平均时间来降低交易费用（特别是对于长期运行的节点，可以每隔几天按平均值结算一次）。

最后，零知识技术（如 zk-SNARKs）有一个非常有趣的属性，那就是能够大大降低任意计算的计算开销，如以太坊智能合约执行 [25]。 尽管生成 zk-SNARK 证明比较昂贵，但验证更便宜，甚至低于原始代码。 由于只有验证需要在链上执行，因此 Orchid 彩币索款的零知识证明在验证起来可能比原始验证代码更便宜。

进一步来说，递归 SNARKs [51] 有可能将一组 SNARK 证明聚集为一个证明。 虽然它们可能更适用于区块链共识协议 [26]，但它们也可能对 Orchid 协议有用，例如将多个彩币索款批量化为单个智能合约交易，同时避免线性 gas 成本复杂化。

E.4. 在宏支付上建立微支付

刚才讨论了交易成本以及支付代币的选择，现在我们看一下可行的支付方式。 以区块链为基础的微支付面临一个基本挑战，那就是如何避免交易费用。 假设我们需要多次发送一美分，如果我们作为普通的以太坊 ERC20 交易发送每一美分，我们需支付 1.4 美分，每笔支付的交易费为 140%！有效的微支付需要将交易费用降低几个数量级。

MojoNation [27] 采用了一个有趣的方法，即在每对节点之间建立一个“贸易平衡”。 当带宽在节点之间流动时，若余额远高于零，则会定期结算带宽。 不过，正如我们所看到的，使用普通以太坊交易进行结算的交易成本至少会产生 0.014 美元的交易费用。 根据之前讨论的上限，我们可以看到这一价格约等于 140 兆字节的带宽。 这种方法的第二个问题是，接近调节阈值的对等点会知道这一事实，并且会尝试断开连接并创建一个新的身份，而不是支付费用。

E.5. 支付渠道

在比特币网络上首次出现的区块链应用程序流行技术是支付渠道 [28]。 支付渠道这一技术最初由中本聪 (Satoshi Nakamoto) 阐述了一部分 [66]，后来由 Hearn 和 Spilman [29] 定义与实现，再之后由 Poon 和 Dryja [30] 研究用于比特币闪电网络。 支付渠道允许发送者和接收者在彼此之间发送任意数量的交易，并且仅支付两笔交易的交易费用，一个用于设置支付渠道，另一个用于关闭支付渠道。 为此，发送者需要先发布一笔交易来锁定一定数量的代币，这些代币只能发送给接收者或发回给发送者。 通常，代币只能在将来某个时间 T 发回给发送者。 同时，可以将代币（递增或完整地）发送给接收者。 发送者持续签署交易，将越来越多数量的代币发送给接收者，并直接发送给接收者而无需在区块链上发布。 接收者可以在 T 之前，随时发布其最近接收的交易以索取发送给他们的总金额。

支付渠道为发送者提供了一种有效方式，可以为接收者提供连续支付的密码证明。 由于中间支付不产生任何交易费用，因此可以任意支付小额费用并任意多次发送。 在实践中，瓶颈成为了验证交易的计算开销以及发送交易的带宽要求。

尽管支付渠道有效地为任意数量的中间支付提供了不变的交易费用复杂性，但它并非在所有使用情形下都如此高效。 特别是在有大量发送者和接收者的系统中，他们经常改变交互对象，不断创建新的支付渠道可能非常昂贵。 同样，如果提供的服务很小或很短暂（例如单个 HTTP 请求或 10 秒的视频流），则所需链上交易的交易费用可能很高。

E.6. 概率支付

如果我们不能质疑支付结算必须在具有交易费用的区块链上进行这一假设，那么理论上的最低成本为单笔交易的成本，因为区块链至少需要一笔交易才能执行状态转换。 为了对一定金额的（微）支付进行结算，我们至少需要一笔交易。

如果我们可以取消支付渠道所需的设置交易，并且仍能向接收者证明他们获得支付，会怎样？

幸运的是，在区块链行业有一个类似问题已得到解决：矿池共享 [31]。 随着像比特币这样的网络中工作量证明难度的增加，矿工开始将他们的算力集中在一起，以避免高方差，在高方差下，单个矿工需要花费数年才能找到区块解决方案。 矿池按哈希能力比例给予奖励，单个矿工通过不断为同一底层区块哈希发送解决方案来证明他们的哈希能力 [32]，但会选择较低难度。 这项技术使矿池能够以密码方式验证每个池成员的哈希能力，而无论该池成员是否已找到满足实际工作量证明目标的解决方案。

如果我们将相同思维用于支付渠道，我们可以构建概率支付方案，发送者不断向接收者证明他们是*平均*支付，而无论是否实际发生支付。 这样，我们能够创建概率微支付，不需要设置交易，接收者只需在“兑现”时支付交易费用。

在我们研究如何使用以太坊智能合约来构建此类概率微支付之前，我们退一步观察一下，概率支付的最初概念早于区块链技术，于 1996 年由 David Wheeler [88] 首次发布。Wheeler 描述了概率支付的核心概念以及如何将其应用于使用随机数字承诺的电子协议，使发送者和接收者（论文术语中的买方和卖方）都不能操纵概率事件的结果，同时仍向彼此提供概率和兑现金额。

有几篇论文跟进了 Wheeler 的观点，Ronald Rivest [82] 于 1997 年发表了一篇论文，描述了如何将概率支付应用于电子微支付。 2015 年，Pass 和 Shelat [81] 描述了如何将概率微支付应用于比特币等去中心化货币，并指出先前方案均依赖于可信赖的第三方。 次年，Chiesa、Green、Liu、Miao、Miers 和 Mishra [58] 将这项研究加以扩展以用于零知识证明，提供适用于加密货币协议的去中心化和匿名微支付。

鉴于最近以太坊系统中支付渠道受到的关注及其普遍性，从支付渠道的角度来看待概率支付会很有价值。 为了省略第一次设置交易，我们无法保证能发送确切金额，而只能实现一个概率保证。不过，我们将通过调整概率、中奖金额和支付频率来表明确切金额，我们可以尽可能细化概率微支付，以便其能够取代几类区块链应用程序的支付渠道，而没有明显缺陷。

从本质上讲，我们可以避免初始设置交易，我们能够从同一发送者帐户中为任意数量的接收者支付任意小的服务会话，同时仍能向每个接收者提供支付金额的确切概率。 假设服务提供商（Orchid 网络中的中继或代理节点）提供足够的服务量，那么概率支付的方差很快就能均等。

E.7. 更多 Orchid 代币详细信息

激励

激励是通过赋予人们对网络的部分所有权来引导新的协议和网络的一种方式 [33]。 Orchid 这种新型去中心化网络存在“先有鸡还是先有蛋”的问题。 代理和中继节点越多，网络为用户提供的实用性就越高。 用户越多，运行代理或中继节点的价值就越高。 通过部署新的网络代币，可以加速网络效应，因为所有潜在用户都受到激励来尽早使用网络。

解耦

在基于其他去中心化系统的中心化系统中，新的代币将新系统的市场价值从底层系统中解耦出来。例如，截至 2017 年 10 月，以太币的市值约为 300 亿美元，日均全球交易量约为 5 亿美元 [34]。以太币的价格受多种因素影响，例如加密货币的整体猜测、以太坊矿工的哈希能力以及以太坊上建立的数百个项目的成败。 不过，单个项目的成败不会对以太币的价格产生重大影响，但会显著影响所涉及项目的特定代币。 使用新代币解耦市场价值，能够为所涉及项目和系统的规模与运行状况建立更好的指标，并针对系统未来有效建立预测市场。

流动市场

系统特定代币的流动市场可以使严重依赖系统的用户通过做空头寸来对冲潜在的系统失败。 如果这看起来很牵强，我们应该注意到，金融衍生工具的初衷是让企业能够对冲掉不利的将来事件。随着 0x [35] 和 etherdelta [36] 等去中心化交易以及 Augur [37] 和 Gnosis [38] 等预测市场的出现，以太坊代币和系统的衍生工具的出现也不会太远。 实际上，此类衍生工具甚至比传统金融衍生工具更有效 [39]，因为前者没有受信方、无需许可甚至可能匿名。

新代币

通过新代币，还可以更轻松地为利益相关者设计具体激励措施；因为代币价值完全来自于新系统，所以它们对于致力于系统成功的任何人来说都是强有力的激励措施。 以太坊智能合约可以实现代币的自主锁定，以确保代币持有者只能根据定义的时间表访问其代币。 这可使激励措施随时间调整，并将代币持有者的重点放在*系统* 的长期成功上，而不是特定团队或相关公司这种社会结构。如果 Orchid 网络仅使用了以太币，并且利益相关者的以太币被锁定，那么他们实际上会更愿意努力实现以太坊的整体成功，而不是使用以太坊的任何特定系统。 可以说，这样的结果对于 Orchid 网络和项目而言并不是最佳的激励调整措施。

E.8. 可验证的随机函数

通过将接收者的随机数承诺替换为可验证的随机函数 (VRF)，可以减少上一节中描述的支付彩币交互性。 可验证随机函数最初由 Micali、Rabin 和 Vadhan [84] 于 1999 年发布，Goldberg 和 Papadopoulos [40] 在最近提出了 VRF 的 IETF 草案。 该草案指定了两种 VRF 结构，一种使用 RSA，另一种使用椭圆曲线 (EC-VRF)。

使用 VRF，Orchid 支付彩币的发送者将能创建彩币，而无需获得接收者对每个彩币（或遇到中奖彩币之前的每个彩币）的承诺。 发送者只需要知道接收者的公钥。 发送者将使用该公钥替换先前描述的彩币方案中的随机数哈希。 为了提高效率，这可以是用于接收彩币中已存在资金的接收者公钥，但要坚持密钥分离的密码原则，可能需要第二个密钥。

不过，验证 Orchid 支付智能合约中的 EC-VRF 需要明确的 EVM 加速椭圆曲线操作，因为若直接以 solidity 或 EVM 汇编来实现，gas 成本将非常高昂。

幸运的是，在以太坊拜占庭 [41] 版本中，以太坊网络增加了对椭圆曲线标量加法和乘法的 EVM 支持 [42] 以及 alt bn128 曲线的配对检查 [43] [49]。 EC-VRF 结构是针对任何椭圆曲线定义的，IETF 草案特别将 EC-VRF-P256-SHA256 定义为 EC-VRF 密码套件（其中 P256 是 NIST-P256 曲线 [53]）。 不过，似乎没有理由在达到足够安全级别的情况下，不去使用 alt bn128 曲线。 而且，SHA256 可以替换为 Keccak-256。 这将允许在以太坊智能合约中进行 VRF 验证，从而与 Orchid 智能合约相整合。

然而，尽管在 zcash 中使用了 alt bn128 曲线，但是与 P256 相比，这一曲线要年轻得多，而且没有得到充分研究。 更重要的是，EC-VRF 结构是待审核的初期草案，EVM 拜占庭升级发生在撰写本文之时，并且没有在处理重大价值的实际系统中得到证明。 因此，在 Orchid 概率微支付中使用 EC-VRF 并非立即可行，Orchid 项目的目标是指导进一步研究，以确定可由 EVM 验证的 EC-VRF-ALTBN128-KECCAK256 结构等在使用上的可行性。

E.9. 非交互式支付方案

在第 E.8 节中，我们说明了通过将 Orchid 支付方案中的随机数承诺替换为 VRF，可以消除与随机数承诺相关的传达步骤，进而增强方案的非交互性。 接收者不必将承诺传达给发送者以使发送者构建彩币，发送者仅通过公开的接收者信息即可立即构建彩币。

每个接收者将专门为 VRF 生成一个新的密钥对，并将公钥与第 4.1 节中详细介绍的其他公共接收者信息一起发布。 发送者只需在彩币中配置此公钥，接收者将使用相应私钥对接收的彩币进行签名。 第 D.4 节中定义的彩币验证逻辑会将接收方 VRF 签名解析为与中奖概率阈值进行比较的值。

如第 E.8 节所述，这是对支付方案的相对简单的修改，EVM 中 VRF 验证的可行性仍需进一步研究。

F. 相关工作

Orchid 项目在点对点网络 (P2P)、区块链、密码学和覆盖网络领域进行了大量工作。 然后，Orchid 将这些早期工作所带来的见解与区块链技术（尤其是以太坊 [11] 和 Zcash [12]）最新的 P2P 研究相结合。

以下各节说明了先前工作在 Orchid 项目中的作用。

F.1. 虚拟专用网

虚拟专用网 (VPN) 使用加密功能在较大的不安全网络上安全传输 VPN 订户的流量。 这种加密方式可以阻止跟踪浏览习惯和唯一在线标识符（如用户 IP 地址），并绕开访问限制。

VPN 用户不应假定其 VPN 连接是真正安全或匿名的。 一些 VPN 服务提供商跟踪其客户的网络活动，然后在 VPN 订户未予批准甚至不知情的情况下将数据出售给第三方商业实体。 VPN 提供商的网络节点的 IP 地址也可能是可识别的。 这样，政府或商业实体（如 Netflix）就能阻止进入和来自 VPN 提供商服务器的流量 [13]。

VPN 的这些劣势促使了去中心化覆盖网络的发展。 去中心化覆盖网络通过持续变化的一组出口节点来提供 VPN 服务。 如果站点阻止来自 VPN 出口节点的流量，则一个或多个替代出口节点将动态投入使用。

F.2. 点对点协议

点对点协议始于 Napster 文件共享网络 [42]。 Napster 使用激励措施来鼓励订户托管音乐文件，以换取从其他对等点下载文件的权利。

Napster 网络

Napster 使用集中式目录服务来为文件和对等点位置建立索引。 Napster 的方法所导致的知识集中化使他们很容易受到 MPAA（美国电影协会）的法律诉讼。 这最终导致了 Napster 的破产。

Napster 集中式目录的漏洞启发了 Napster 后继网络 Gnutella 的设计者，他们将文件和节点地址的索引分布在网络的每个对等点上 [43]。

Gnutella 网络的分布式索引响应

Gnutella 网络的设计者通过实施分布式索引方法，弥补了 Napster 集中式目录的不足。 与 Napster 相比，这种方法增强了弹性和可扩展性。 这些改进也激发了在 P2P 网络间分布索引的其他框架的开发。 一个著名的例子是采用分布式哈希表 (DHT) 来实现 P2P 网络中节点和资源的有效发现。

Tor 网络

Tor 是由美国海军在 20 世纪 90 年代中期开发的。 从那以后，开源社区的开发和对 Tor 的使用一直保持到现在。 现在，全球约有 7,000 个节点、3,000 个出口节点和大约 200 万用户。

Tor 使用集中式节点选择，并依靠志愿者提供节点服务（包括出口节点服务），这对 Tor 的吞吐量产生了负面影响。 其原因是，Tor 无法使用 BitTorrent 或其他 P2P 文件共享系统，以及出口节点能够检查出口流量的内容。 此外，Tor 没有防止出口节点被迫代表其他用户访问非法或危险信息的机制。

尽管存在这些问题，Tor 相对较小的开发社区仍在不断研究如何使 Tor 网络更快速、更可靠、更安全 [25]。 该讨论的关键部分是如何将较低延迟/较高带宽的节点引入网络 [20, 21, 22, 23, 24]。

要实现这些目标，Tor 网络必须找到一种提供用户激励措施的方法。 通过从用户那里获得捐款的方式来改造 Tor 会遇到很多障碍。 Tor 无法将支付与路由紧密结合在一起，因此很难有效地管理匿名数字支付。 Tor 社区坚持要求某些节点继续提供免费路由，而有一些节点在成为“金星”会员后可以获得付款，这又增加了一层复杂性。

限制 Tor 增长的另一个非技术原因是，它通常被视为主要方便技术精湛的用户（“技术人员”）访问非法服务或暗网网站的工具。 *Silk Road* 网站就是此类隐藏服务的一个例子，该网站提供各种非法商品和服务 [18, 19]。

与之相反，Orchid 网络不会提供隐藏服务，而只专注于对 互联网 的开放、安全、匿名访问。

洋葱路由

此处描述的洋葱路由技术（以及第 F.2 节中说明的大蒜路由）与加密相结合，可在 P2P 网络上提供更高级别的匿名路由。

洋葱路由是一种“分层”的数据加密方法，可通过 P2P 网络创建路径。 消息通过源节点重复加密，然后由消息传输经过的每个节点依次解密。 中间节点仅接收路由消息所需的路由指令。 只有最后（出口）节点会接收到路由指令和消息。

Tor 网络（第 F.2 节）就是常被提到的洋葱路由示例。

大蒜路由

隐形互联网项目 (*I2P*) 是一个去中心化匿名网络，它基于类似 Tor （第 F.2 节） 的原理，但从一开始就设计为一个独立暗网。 I2P 的关键设计特征是使用大蒜路由 [26]。

大蒜路由将多个消息捆绑到单个数据包中，这个数据包称为*蒜头*。 蒜头中的每个消息又以洋葱路由的分层加密形式进行加密。 对于隐藏服务，消息捆绑意味着访问 I2P 的速度大大快于 Tor。 I2P 仅部分支持路由到更广泛的 互联网，因此难以全面比较性能改进。 消息捆绑还使流量分析变得更加困难。

I2P 用户使用点对点加密隧道相互连接，但没有 Tor 使用的集中式目录。 I2P 完全分隔传入和传出的流量。 然后，使用分组交换而不是电路交换，提供跨多个对等点的消息透明负载平衡。 这些设计特征结合在一起，可以提高安全性和匿名性。

I2P 对节点的分布式数据库的管理是它需要进行重大改进的一方面。 I2P 最初使用 Kademlia（最初于 2002 年设计）[27]。 Kademlia 的初始版本持续占用大量 CPU 和网络带宽，无法扩展。 随后，I2P 过渡到他们称为*泛洪* 的算法。 不过，泛洪机制也存在设计缺陷，这些缺陷会被人利用来破坏和操纵 I2P 分布式数据库中的信息 [28]。

F.3. 区块链平台

区块链协议可以就全局状态达成无许可、去中心化的共识，并使用加密代币来激励运行节点。

如比特币、以太坊、Zcash 等区块链设计就是使用状态转换功能在其全局状态中添加或更改条目的区块链协议示例。 这些协议还会奖励节点验证交易，以及使用工作量证明等技术对其顺序达成共识 [44]。

以太坊

以太坊 [55] 与比特币 [78] 共同开创了新形式的应用程序特定密码代币 [33]。 通过支持基于任意选择的计算方法的智能合约，区块链系统可用于创建自定义分类帐，这些分类帐提供应用程序特定功能，例如投票、管理功能和费用支付。

以太坊是去中心化区块链平台，能够执行和部署 *智能合约*。 以太坊的智能合约代码不变，并且在执行中具有完全确定性（除非明确添加了非确定性行为）。 这允许任何节点验证智能合约的执行，并审核对应用程序状态产生的更改。 这样，以太坊智能合约能够完全按编程方式运行。

以太坊应用程序在强大的共享全球基础架构上运行。 应用程序可以快速转移价值并代表资产所有权，从而使软件开发人员可以创建市场、存储债务或承诺的注册表、根据过去创建的规则转移资金等。 所有这些都没有第三方提供商，也不存在交易对手风险。

一般来说，以太坊的功能很有用，在必须应对服务器停机、腐败和欺诈行为的新兴市场中尤其如此。

以太坊和 Orchid 市场 ERC20 代币

以太坊网络上部署的大多数代币都符合 ERC20 标准 [44]。 该标准规定了用于传输代币和元数据的紧凑、简单 API，该 API 可以轻松快速地将代币集成到硬件或软件用户钱包中。

例如，Augur [37] 和 Gnosis [38] 平台使用 ERC20 代币，通过代币数据创建预测市场。 ERC20 可使任意智能合约轻松与 Orchid 协议对接。 对于希望安全访问 互联网 端点的 IoT 设备而言，这非常重要。

符合 ERC20 标准还使代币交换和应用程序更容易受益于新型 ERC20 代币提供的扩展功能。 这继而使不同应用程序可以使用符合 ERC20 的代币来交换信息和状态。