源石币：一种零交易手续费的分布式加密货币

Colin LeMahieu
clemahieu@gmail.com

***Abstract*—目前，高需求和有限的可扩展性增加了普通加密货币的平均交易时间和费用，产生了不尽人意的体验。在这里，我们介绍源石币，一种基于区块点阵(BlockLattice)结构的新型加密货币，其中每个账户都有自己的区块链，提供近乎瞬时的交易速度和无限的可扩展性。每个账户(即地址)都有自己的区块链，允许他们异步地更新到网络的其余部分，从而以极小的资源开销获得快速的交易确认。交易记录帐户余额而不是交易金额，使得系统可以在不牺牲安全性的情况下进行大幅度的数据库修剪。到目前为止，RaiBlocks网络已经处理了420万笔交易，这个数据库账本在未进行修剪的情况下只有1.7GB的大小。RaiBlocks的无交易手续费、瞬时交易特性使其成为消费交易领域的最优加密货币。**

***Index Terms*—索引：加密货币，区块链，源石币，分布式账本，数字化，交易**

I. 引言

互联网上的商业活动几乎完全依赖于金融机构作为可信任的第三方来处理电子支付。虽然这个系统运作良好足以应付大部分交易，但仍然存在着固有的、基于信任模型的缺陷。完全不可逆转交易是不可实现的，因为金融机构不能避免调解纠纷。调解成本的增加导致了交易成本的增加，限制了最低实际交易规模和切断了小额交易的可能性。由于存在逆转交易的可能，对信任的需求增加，商家必须警惕他们的客户并且通过收集过剩的信息来最小化财务风险。一定比例的欺诈事件被认为是不可避免的。这些成本和交易的不确定性可以通过使用物理现金来避免，但目前还不存在可以绕过可靠的第三方来进行支付的通信渠道。

解决方案是采用基于密码学电子支付系统，使得交易双方可以在不需要可靠的第三方的情况下直接进行交易。从算力的角度来看要逆转一个合法的交易是不切实际的，这保护了卖家免受欺诈，而常规的托管机制则作为一个保护买家的机制。本文将介绍一种具有无限可扩展性和零交易费用的、低延迟加密货币——源石币。

本文引用的加密货币统计数据精确至本文的发布日期。

II. 背景

2008年中本聪（Satoshi Nakamoto）发表了比特币白皮书，介绍了世界上第一个分布式加密货币比特币 [1]。比特币带来的核心创新是区块链：一种公开的，不可篡改的分布式数据结构，用作货币交易的帐本。然而，随着比特币的发展，协议中的几个问题使得比特币越来越难以应用于实践中：

1) 可扩展性差：区块链中的每个区块可存储的数据有限，这意味着系统每秒只处理一定数量的交易，使得每个区块中的位置都变成了商品。目前的中位交易费用是$10.38 [2].
2) 高延迟：平均确认时间为164分钟 [3].
3) 高能耗：比特币的网络消耗估计每年27.28TWh，平均每笔交易消耗260KWh [4].

比特币和其他加密货币通过对其全局的分布式账本达成共识的方法来验证合法性交易，同时抵制恶意行为者。比特币通过一种称为工作量证明（PoW）的经济措施来达成共识。在一个PoW系统中，参与者竞争计算一个称为*nonce*的随机数，使整个区块的哈希值落在一个目标区间内。该有效区间与整个比特币网络的累计算力成反比，以便使找到有效的随机数的平均时间保持一致。找到有效的随机数的矿工被允许将该块添加到区块链中，因此，那些耗费更多计算资源来计算随机数的人在区块链的状态中扮演着更重要的角色。PoW提供对Sybil攻击的抵制。在Sybil攻击中，单个实体表现为多个实体以在去中心化系统中获得额外的能力，并且还大大减少了在访问全局数据结构时固有存在的竞争条件。

Peercoin在2012年首次提出了另一个共识协议，即权益证明(PoS) [5]。在PoS系统中，参与者投票的权重等于他们在特定加密货币中拥有的货币数量。在这种协议中，投资人投资的额度越大，其获得的权力也越大。他们将倾向于维护一个诚信的系统，否则会面临失去投资的风险。PoS消除了浪费计算能力的竞争模式，只需要在低功耗硬件上运行轻量级软件。

最初的RaiBlocks白皮书和第一个Beta测试版本于2014年12月发布，使其成为最早基于定向非循环图(DAG)的加密货币之一 [6]。不久之后，其他DAG加密货币开始发展，最著名的是DagCoin / Byteball和IOTA [7], [8]。这些基于DAG的加密货币打破了区块链模式，提高了系统性能和安全性。Byteball依靠由诚实的，有信誉的，值得信赖的“证人”组成的“主链”达成共识，而IOTA则通过堆叠式交易的累计PoW达成共识。RaiBlocks通过余额权重对冲突交易的投票达成共识。这个共识体系提供了更快、更确定的交易，同时保持强大的，去中心化的系统。RaiBlocks继续这一发展，并将自己定位为性能最高的加密货币之一。

III. RaiBlocks 组件

在描述RaiBlocks的整体架构之前，我们先定义构成系统的各个组件。

A. 账户

账户是数字签名密钥对的公钥部分。公钥也被称为地址，而私钥则保密。数字签名的数据包确保内容得到私钥持有者的认可。一个用户可以控制多个帐户，但每个帐户只能有一个公共地址。

B. 区块/交易

“区块”和“交易”两个术语通常可以交替使用，一个块包含一个交易。交易特指动作，而区块特指交易的数字编码。交易由执行交易的帐户的私钥签名。

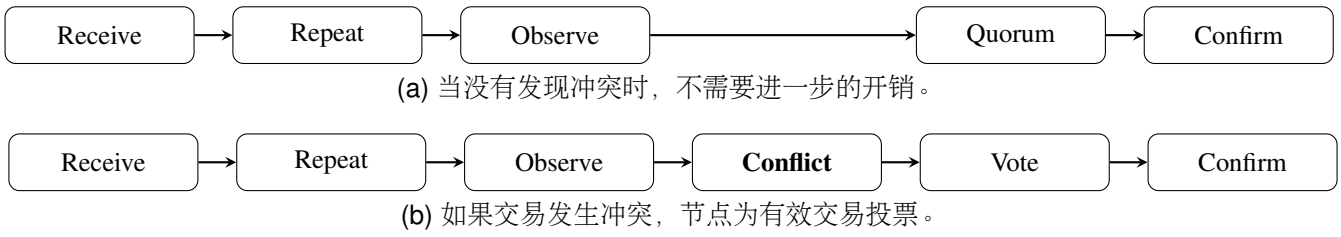

图 1. RaiBlocks对于典型的交易不需要额外资源的开销。在交易冲突的情况下，节点必须投票支持交易。

C. 账本

账本是一个全局集合，每个账户拥有自己的交易链(图 2)。这是一个关键的组件，其使用设计时协议取代运行时协议;通过签名检查，每个人都同意只有帐户所有者可以修改自己的链。这将看似共享的数据结构（分布式账本）转换为一组非共享数据结构。

D. 节点

节点是在计算机上运行的一个软件，它服从RaiBlocks协议并参与RaiBlocks网络。该软件管理着账本和节点可能控制的任何账户（如果有的话）。一个节点可以存储整个账本或者只存储每个账户的最后几个区块的修剪历史记录。建立新节点时，建议验证整个历史记录并在本地修剪。

IV. 系统总览

不同于许多其它使用区块链的加密货币，RaiBlocks使用区块点阵结构。每个账户都有自己的区块链（账户链），相当于账户的交易/余额历史记录(图 2)。每个账户链只能由账户所有者更新，这允许每个账户链立即且异步地更新到晶格的其余部分，从而实现快速交易。RaiBlocks的协议是极其轻量级的，每个交易都能够被放进符合在互联网上传输所需的最小UDP数据包里。节点的硬件要求也是最小的，因为节点只需要为大部分交易记录和重播区块(图 1)。

系统通过一个包含初始金额的创世账户实现初始化。初始金额（总供应量）是一个固定的数字，永远不能增加。初始金额通过在创世账户链上注册的发送交易发送到其他账户。所有账户余金额总和不会超过最初的初始金额，从而给系统一个数量的上限，不存在增加的可能。

本节将介绍如何在整个网络中构建和传播不同类型的交易。

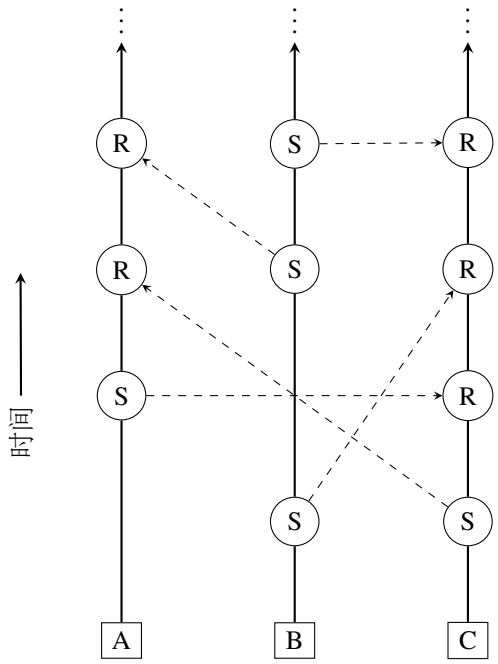

图 3. 区块点阵的可视化。每笔资金转移都需要一个发送块（S）和一个接收块（R），每个块由其账户链的所有者（A，B，C）签署。

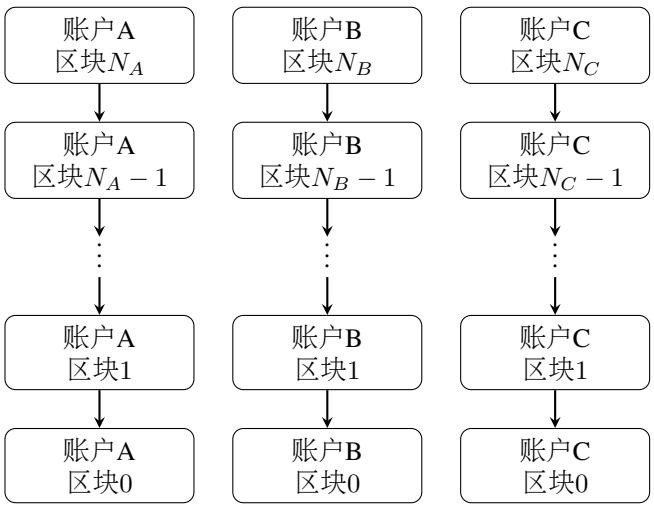

图 2. 每个账户都有自己的区块链，包含账户的余额历史记录。区块0必须是一个初始化交易(IV-B节)。

A. 交易

将资金从一个账户转移到另一个账户需要两个交易：一个发送交易从发送方的余额中扣除金额，一个接收交易将该金额添加到接收账户的余额(图 3)。

在发送方和接收方的账户中将金额作为单独的交易进行转账有以下几个重要目的:

1) 对固有的异步传入转账进行排序。
2) 保持交易小到可以放进UDP数据包。
3) 通过最小化数据足迹来促进账本修剪。
4) 将已结算的交易与未结算的交易分开。

多个账户转账到同一目标账户是异步操作;网络延迟和发送账户不一定相互通信意味着没有普遍适用的方法来知道哪一个交易首先发生。由于加法是混合运算，输入序列的顺序并不重要，因此我们只需要一个全局协议。这是将运行时协议转换为设计时协议的关键设计组件。接收账户可以决定哪一笔转账率先到达，并且由传入块的已签名顺序来表示。

如果一个帐户想要进行一笔大额转账，而这笔金额本身是通过接收很多笔小的转账而得到的，我们想用一个适合UDP包的方式来表示这个转账。当一个接收账户对输入数据进行排序时，它会实时保持一个账户的总额，以便在任何时候都能够以固定大小的数据交易任何金额。这与比特币等其他加密货币使用的输入/输出交易模型不同。

一些节点对花费资源来存储账户的完整交易历史不感兴趣;他们只对每个账户的当前余额感兴趣。当账户进行交易时，会对其累计余额进行编码，这些节点只需要跟踪最新的块，这样可以在保持正确性的同时丢弃历史数据。

即使重点关注设计时协议，由于需要识别和处理网络中的不良参与者，验证交易时也会有一个延迟窗口。由于RaiBlocks中的协议快速达成（约为毫秒级到秒级），我们可以向用户展示两个相似的传入交易类别：已结算交易和未结算交易。已结算交易是帐户已经为其生成接收块的交易。未结算交易尚未纳入接收方的累计余额。这是对其他加密货币中复杂、不直观的确认度量的替代。

B. 创建帐户

要创建一个帐户，您需要生成一个初始化交易(图 4)。初始化交易永远是每个账户链的第一笔交易，可以在第一次收到资金时创建。*account*字段存储用于签名的私钥所派生的公钥（地址）。*source*字段包含发送资金的交易的哈希值。在创建账户时，必须选择一位可以代你投票的代表，代表可以稍后改变(IV-F 节)。该帐户可以声明选择自己作为代表。

```
open {
   account: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C...182A0E26B4A,
   representative: xrb_1anr...posrs,
   work: 0000000000000000,
   type: open,
   signature: 83B0...006433265C7B204
}
```

图 4. 初始化交易剖析

C. 账户余额

帐户余额记录在帐本本身内。系统通过检查发送块的余额和前一个块的余额之间的差额来进行验证(IV-I 节)，而不是通过记录交易金额的方式。接收账户然后可以将之前的余额增加，并将其作为最终余额记录到新的接收块中。这是为了提高下载大量区块时的处理速度。在请求帐户历史记录时，金额已经给出。

D. 从帐户发送交易

要从一个地址发送，地址必须有一个已经存在的初始区块，因此有一个余额(图 5)。*previous*字段包含帐户链中前一个块的哈希值。*destination*字段包含要发送到的帐户。一个发送区块被确认后是不可变的。一旦广播到网络，资金立即从发送方的帐户的余额中扣除，并处于等待接收的状态，直到接收方签署一个区块接受这些资金。等待接收不应被视为等待确认，因为资金相当于已经被发送方使用，发件人无法撤销交易。

```
send {
   previous: 1967EA355...F2F3E5BF801,
   balance: 010a8044a0...1d49289d88c,
   destination: xrb_3w...m37goeuufdp,
   work: 0000000000000000,
   type: send,
   signature: 83B0...006433265C7B204
}
```

图 5. 发送交易剖析

E. 接收交易

要完成一笔交易，接收方必须在自己的账户链上创建一个接收区块来接收资金(图 6)。*source*字段引用关联的发送交易的哈希值。一旦创建并广播了该区块，账户余额就会更新，并且资金已经正式转入接收方账户。

```
receive {
   previous: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C6...182A0E26B4A,
   work: 0000000000000000,
   type: receive,
   signature: 83B0...006433265C7B204
}
```

图 6. 接收交易剖析

F. 指定代表

账户持有人有能力选择代表为其进行投票，这是一个强大的去中心化工具，在工作量证明（PoW）或权益证明（PoS）协议中没有能与之相提并论的机制。在传统的PoS系统中，账户拥有者的节点必须运行参与投票。连续运行节点对许多用户来说是不切实际的，给予代表权力代表一个帐户投票放宽了这一要求。账户持有人有权随时重新分配共识给任何账户。变更交易通过从旧代中表减去投票权重并将权重添加到新代表处来改变账户的代表(图 7)。这笔交易中没有资金被转移，代表也没有账户资金的消费权。

```
change {
   previous: DC04354B1...AE8FA2661B2,
   representative: xrb_1anrz...posrs,
   work: 0000000000000000,
   type: change,
   signature: 83B0...006433265C7B204
}
```

图 7. 变更交易剖析

G. 分叉和投票

当j签署区块$b_1, b_2, \ldots, b_j$时,如果它们声明的前一区块相同则发生分叉(图 8)。这些块会导致帐户状态发生冲突，必须予以解决。由于只有账户的所有者才有能力在他们的

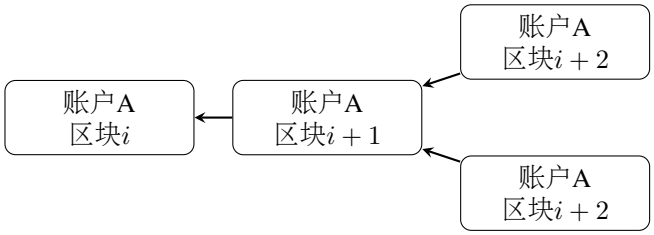

图 8. 两个（或更多）已经签署的区块引用的前一个区块是同一个块时则发生分叉。老块在左边，新的块在右边。

账户链上签名，所以一个分支必然是账户所有者的糟糕编程或恶意意图（双重花费）造成的。

一旦检测到，代表将创建一个投票，在其账本中引用块$\hat{b}_i$，并将其广播到网络。节点投票的权重w_i是指定其代表的所有账户余额的总和。节点将观察来自其他M个在线代表的进来的投票，并保持总计1分钟的4个投票期的累积计数，并确认获胜块(方程式1)。

$$v(b_j) = \sum_{i=1}^{M} w_i \mathbb{1}_{\hat{b}_i = b_j} \tag{1}$$

$$b^* = \arg\max_{b_j} v(b_j) \tag{2}$$

最流行的块b^*将获得最多的票数(方程式2)，且被保留在该节点的账本中。丢失投票的区块将被丢弃。如果代表取代了账本中的一个区块，它将创建一个更高序列号的新投票，并将新的投票广播给网络。这是代表投票的**唯一**场景。

在某些情况下，短暂的网络连接问题可能会导致一个广播块不被所有节点接受。这个帐户的任何后续块将被那些没有看到最初广播的节点所忽略。一个这个块的重播将被其余的节点接受，随后的块将被自动检索。即使出现分叉或缺失块，也只会影响交易中引用的帐户。网络的其余部分继续处理所有其他帐户的交易。

H. 工作量证明

所有四种交易类型都有一个必须正确地填充的*work*字段。*work*字段允许交易创建者计算一个随机数，使得该随机数的哈希值与接收/发送/变更交易中的*previous*字段或初始化交易中的*account*字段串联计算后低于某个阈值。与比特币不同的是，RaiBlocks中的PoW被简单地用作反垃圾交易工具，类似于Hashcash，耗时约为几秒 [9]。一旦发送了一个交易，由于之前的块的*previous*字段是已知的，所以可以预先生成后续块所需的PoW，只要两笔交易之间的时间大于生成PoW所需的时间，用户的交易将在瞬时完成。

I. 交易验证

对于被认为合法的区块，它必须具有以下属性:

1) 区块不能已经存在在账本中（重复交易）。
2) 必须由账户所有者签名。
3) 前面的块是账户链的头块。如果它存在但不是在头部，那么这是一个分叉。
4) 账户必须有一个初始化区块。
5) 计算的哈希值符合PoW阈值要求。

如果是一个接收区块，检查源区块哈希值是否处于待完成状态，这意味着它尚未被兑付。如果是一个发送区块，余额必须小于之前的余额。

V. 攻击途径

像所有去中心化加密货币一样，RaiBlocks可能会遭到恶意者攻击，企图获得经济利益或使系统崩溃。在本节中，我们将概述一些可能的攻击场景、这些攻击的后果以及RaiBlocks协议如何采取预防措施。

A. 区块间隙同步

在第 IV-G节中，我们讨论了一个块可能无法正确广播的情况，导致网络忽略后续块。如果一个节点观察到一个没有引用前一个块的块，它有两个选择:

1) 忽略该块，因为它可能是一个恶意的垃圾块。
2) 请求与另一个节点重新同步。

在重新同步的情况下，必须与引导节点构建TCP连接，以匹配重新同步增加的通信量。但是，如果该块实际上是一个坏块，那么重新同步是不必要的，反而会增加网络的流量。这是网络放大攻击，将导致阻断服务。

为了避免不必要的重新同步，在启动到引导节点的连接以进行同步之前，节点将等待，直到观察到对潜在的恶意块的投票达到一个特定的阈值。如果一个块没有得到足够的票数，可以认为是垃圾数据。

B. 泛滥交易

恶意者可以在其控制下的帐户之间发送许多不必要的但有效的交易，试图使网络饱和。由于没有交易费用，他们能够无限期地继续这种攻击。然而，每次交易所需的PoW限制了恶意者在不显著投入计算资源的情况下可能产生的交易率。即使在这种企图膨胀账本的攻击下，不包含完整历史的节点也能够从链中删除旧的交易，这几乎为所有用户限制了该类攻击所使用的存储。

C. Sybil 攻击

一个实体可以在一台机器上创建数百个RaiBlocks节点。然而，由于投票系统中的权重是基于账户余额的，所以增加额外的节点到网络中将不会让攻击者获得额外的投票权重。因此，Sybil攻击不会获得任何额外优势。

D. Penny-Spend 攻击

Penny-Spend攻击是指攻击者向大量的账户发送极小的金额，已达到浪费节点存储资源的目的。由于区块广播的速率是受PoW限制的，所以这在一定程度上限制了账户和交易的创建。非完整历史节点可以对那些很有可能不是有效帐户的账户在一个统计指标下进行裁剪。最后，RaiBlocks被调定为使用最小的永久存储空间，因此存储一个额外账户所需的空间与初始化区块 + 索引 = 96B + 32B = 128B的大小成正比。这相当于1GB可以存储800万个Penny-Spend帐户。如果节点想要进行更加激进的修剪，它们可以根据访问频率来算出统计分布，将不经常使用的帐户委托给较慢的存储。

E. 预先生成*PoW*攻击

由于账户的所有者将是唯一向账户链添加区块的实体，因此在向网络广播之前可以预先生成连续的区块以及它们所需的PoW。在这里，攻击者在很长一段时间内产生大量的连续块，每个块都是最小值。在某个时刻，攻击者通过向网络倾注大量有效的交易来执行DoS攻击，其他节点将尽可能快地处理和回应。这是 V-B节所述的泛滥交易的高级版本。这样的攻击只能短暂地工作，但可以与其他攻击结合使用，例如>50%攻击(V-F节)以提高效率。目前正在研究交易限速等其他技术来减轻攻击。

F. *>50%* 攻击

RaiBlocks的共识机制是一个余额加权的投票系统。如果攻击者能够获得超过50%的投票权，就会导致网络振荡，系统崩溃。攻击者可以通过DoS攻击来阻止诚信的节点进行投票，从而降低自身所需要的投票权重。RaiBlocks采取以下措施来防止这种攻击：

1) 针对这种攻击的主要防御机制是，在系统中投票权是与投资挂钩的。账户持有人本质上倾向于去维护一个诚实的系统，以保护他们的投资。试图操纵账本将对整个系统造成破坏，从而造成投资损失。
2) 这种攻击的成本与RaiBlocks的市值成正比。在PoW系统中，相较于投入金钱的方式，新技术可能被发明出来对系统造成不成比例的操纵，如果攻击成功，这种技术还可以在攻击完成后重新利用。而在RaiBlocks中攻击系统的成本随系统本身而扩大，如果攻击成功，为攻击付出的投资成本就无法回收。
3) 为了维持最大的合法投票人数，下一道防线是代表性投票。由于连接原因而无法可靠地参与投票的账户持有人可以将余额权重委托给一个代表来代为投票。最大化代表的数量和多样性可以提高网络的弹性。
4) 在RaiBlocks中，分叉从来不会意外地发生，所以节点可以对如何与分叉区块交互做出策略性决定。非攻击者帐户唯一一个易受分叉影响的情景是当他们从攻击帐户收到余额的时候。希望从分叉中获得保护的账户可以在向生成分支的账户接收资金前稍作等待，或等待更长的时间，也可以选择永远不去接收它。接收者可以生成独立的账户用以接受可疑账户的资金，以达到隔离风险的目的。
5) 最后一道防线是粘合(目前还没有实施)。RaiBlocks通过投票快速解决分叉问题。节点可以被配置为粘合区块，这将防止它们在一段时间之后被回滚。通过专注于快速建立确认来防止不明的分叉，网络得到充分的保护。

Fig.9详细描述了一个更为复杂的> 50%攻击。*Offline*是指已经被指定但不在线投票的代表的百分比。*Stake*是攻击者用来进行投票的已投入投资金额。*Active*是按照协议在线投票的代表。攻击者可以通过网络DoS攻击使其他选民脱机，从而抵消他们需要获得的权重。如果这种攻击能够持续下去，被攻击的代表将会变得不同步，这在图中用*Unsync*来表示。最后，攻击者可以在被攻击的代表在同步的时候将DoS攻击目标切换到一批新的代理身上，获得短暂且爆发式的相对投票强度，这在图中用*Attack*表示。

如果攻击者能够通过这些情况的组合导致Stake >Active，那么他们将能够以牺牲他们的份额为代价成功地在账本上操纵投票。我们可以通过考察其他系统的市值来估计这种攻击的成本。如果我们假设有33%的代表离线或遭受DoS攻击，攻击者需要购买33%的市值才能通过投票来攻击系统。

Offline	Unsync	**Attack**	Active	Stake

图 9. 一个潜在的投票协议，可以降低51%的攻击要求。

G. 引导中毒

攻击者持有一个具有余额的旧私钥的时间越长，余额在当时不具有参与性代表的可能性越高，因为他们的余额或代表可能已经转移到较新的账户。这意味着，如果一个节点被引导到网络中的一个旧版本，而该版本中攻击者拥有与当时代表相比足够的投票权，则他们将能够对该节点的投票进行振荡。如果这个新用户想要与攻击节点之外的任何人进行交互，所有交易都将被拒绝，因为他们有不同的头块。最终的结果是节点可以通过发送不良信息来浪费网络中新节点的时间。为了防止这种情况，可以将节点与一个包含账户和合法头块数据的初始化数据库进行配对，这是作为下载整个数据库的一个替代方案。下载越接近当前，准确防御这种攻击的概率就越高。最后，这种攻击可能并不比向引导中的节点发送垃圾信息高明多少，因为攻击者无法与任何具有已更新数据库的人发生交易。

VI. 实现

目前的所有参考实现皆是通过C++实现的，并且自2014年以来一直在Github [10]上发布。

A. 设计特性

RaiBlocks的实现遵循本文中概述的体系结构标准。其他规范在这里描述。

1) 签名算法*:* RaiBlocks使用改进的ED25519椭圆曲线算法和Blake2b哈希算法处理所有的数字签名 [11]。采用ED25519是因为其具有快速签名、快速验证和高安全性的特点。

2) 哈希算法*:* 由于哈希算法仅用于防止网络垃圾交易，与基于挖掘的加密货币相比，算法的选择并不那么重要。我们的实现使用Blake2b作为对块内容的摘要算法 [12]。

3) 密钥派生功能*:* 在参考钱包中，密钥通过密码加密，并且密码通过密钥派生函数馈送以防止ASIC破解。目前，Argon2 [13]是旨在创建弹性密钥派生函数的公共竞争中的赢家。

4) 区块间隔*:* 由于每个帐户都有自己的区块链，更新可以与网络状态异步执行。因此不存在区块间隔，交易可以立即广播。

*5) UDP*通信协议*:* 我们的系统被设计为尽可能使用最少量的计算资源来无限期地运行。系统中的所有消息被设计为无状态，并且能被容纳进单个UDP数据包。这也使得具有间断连接性的轻节点能更容易地参与到网络中，而不需要重新建立短期的TCP连接。TCP连接仅在新的节点想要大量引导区块的时候建立。

通过观察来自其他节点的交易广播流量，节点可以确定它的交易已由网络接收，因为它应该看到多个副本回送给自己。

B. IPv6及多播

基于无连接UDP协议的构建，使未来的实现可以使用IPv6多播来替代传统的泛滥交易和投票广播。这将减少对网络带宽的消耗，并给予运行节点带来更多的策略灵活性。

C. 性能

在写这篇文章的时候，RaiBlocks网络已经处理了420万笔的交易，产生了1.7GB大小的账本，交易时间为秒级。运行在市面固态硬盘上的当前实现可以每秒处理超过10,000笔交易，这个数据主要受限于硬盘的IO速率。

VII. 资源利用

这里是RaiBlocks节点使用资源的概况。此外，我们考察了一些方案，以在特定的应用场景中减少资源的使用。减少使用资源的节点通常称为轻节点、修剪节点或简化支付验证（SPV）节点。

A. 网络

网络活动的量取决于网络节点对网络的健康有多大贡献。

1) 代表节点*:* 代表节点需要最大的网络资源，因为它需要观测来自其他代表的投票流量和广播自己的投票。

2) 去信任节点*:* 一个信任节点观测一个它信任的代表节点的投票流量，以正确地达成共识。这减少了代表节点向该节点传输的入站投票流量。

3) 信任节点*:* 一个信任节点观测一个它信任的代表节点的投票流量，以正确地达成共识。这减少了代表节点向该节点传输的入站投票流量。

4) 轻节点*:* 轻节点也是信任节点，仅观察它感兴趣的账户的流量，这使它只占用最小的网络带宽。

5) 引导节点*:* 引导节点向那些上线节点提供部分或整体账本。这通过TCP连接完成，而不是UDP，因为它涉及到大量的数据传输，需要高级流量控制。

B. 硬盘容量

根据用户的需求，不同的节点配置需要不同的存储容量。

1) 历史配置*:* 有意保留所有交易的完整历史记录的节点将需要的最大存储量。

2) 当前配置*:* 得益于区块保存累计余额信息的设计，节点只需要保留每个帐户的最新或第一个区块，以参加共识。如果一个节点对保存一个完整的历史记录不感兴趣就可以选择只保留第一个区块。

3) 轻节点配置*:* 轻节点不在本地存储帐本数据，而是仅参与到网络中观察账户活动。它只对以它持有的私匙创建交易感兴趣。

C. CPU

1) 交易生成*:* 节点创建新的交易时必须计算出一个PoW随机数nonce，以通过RaiBlocks的限制机制。各种硬件的计算表现见表1。

2) 代表节点*:* 代表节点必须验证区块及投票的签名，同时也生产自己的签名去参与共识。相较于生成交易所需的CPU资源，代表节点需要的资源大幅降低，当代计算机中任何单个CPU都应能满足其要求。

3) 观察者*:* 一个观察者节点不会产生自己的选票。由于生成签名的开销是极小的，对CPU资源的需求代表节点相当。

VIII. 结论

在本文中，我们展示了一个去信任化、零交易费、低延迟的加密货币框架，该框架应用了一种新型的区块点阵结构和DPoS投票机制。该网络只需要极少的资源，没有大功率采矿硬件，并且可以处理高交易吞吐量。这一切是通过让每个帐户拥有自己的区块链来实现的，这消除了全局数据结构的访问和低效问题。我们识别了在系统中可能出现的攻击途径，同时论证了RaiBlocks对这些形式的攻击能进行有力的抵抗。

APPENDIX A
POW硬件基准

正如前面提到的，在RaiBlocks中PoW的作用是为了减少网络垃圾交易。我们的节点实现使得可以利用兼容OpenCL的GPU进行加速。表 I提供了各种常见硬件的基准。目前PoW阈值是固定的，但由于平均计算能力的逐年提升，将来可能会应用自适应阈值。

表 I
硬件POW性能

设备	每秒交易笔数
Nvidia Tesla V100 (AWS)	6.4
Nvidia Tesla P100 (Google,Cloud)	4.9
Nvidia Tesla K80 (Google,Cloud)	1.64
AMD RX 470 OC	1.59
Nvidia GTX 1060 3GB	1.25
Intel Core i7 4790K AVX2	0.33
Intel Core i7 4790K,WebAssembly (Firefox)	0.14
Google Cloud 4 vCores	0.14-0.16
ARM64 server 4 cores (Scaleway)	0.05-0.07

致谢

感谢Brian Pugh对本文的编辑和排版。

REFERENCES

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[2] “Bitcoin median transaction fee historical chart.” [Online]. Available: https://bitinfocharts.com/comparison/bitcoin-median_transaction_fee.html

[3] “Bitcoin average confirmation time.” [Online]. Available: https://blockchain.info/charts/avg-confirmation-time

[4] “Bitcoin energy consumption index.” [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption

[5] S. King and S. Nadal, “Ppcoin: Peer-to-peer crypto-currency with proof-of-stake,” 2012. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf

[6] C. LeMahieu, “Raiblocks distributed ledger network,” 2014.

[7] Y. Ribero and D. Raissar, “Dagcoin whitepaper,” 2015.

[8] S. Popov, “The tangle,” 2016.

[9] A. Back, “Hashcash - a denial of service counter-measure,” 2002. [Online]. Available: http://www.hashcash.org/papers/hashcash.pdf

[10] C. LeMahieu, “Raiblocks,” 2014. [Online]. Available: https://github.com/clemahieu/raiblocks

[11] D. J. Bernstein, N. Duif, T. Lange, P. Shwabe, and B.-Y. Yang, “High-speed high-security signatures,” 2011. [Online]. Available: http://ed25519.cr.yp.to/ed25519-20110926.pdf

[12] J.-P. Aumasson, S. Neves, Z. Wilcox-O’Hearn, and C. Winnerlein, “Blake2: Simpler, smaller, fast as md5,” 2012. [Online]. Available: https://blake2.net/blake2.pdf

[13] A. Biryukov, D. Dinu, and D. Khovratovich, “Argon2: The memory-hard function for password hashing and other applications,” 2015. [Online]. Available: https://password-hashing.net/argon2-specs.pdf