Blockstack

Blockstack技术白皮书 v2.0 中文版

2019年5月

By Blockstack PBC

本白皮书中的部分之前由下述的同行评审会议和杂志发布：

- M. Ali, J. Nelson, R. Shea and M. J. Freedman, *"Blockstack: A Global Naming and Storage System Secured by Blockchains"*, 2016 USENIX Annual Technical Conference, Denver, CO, June 2016.
- J. Nelson, M. Ali, R. Shea and M. J. Freedman, *"Extending Existing Blockchains with Virtualchain"*, Workshop on Distributed Cryptocurrencies and Consensus Ledgers, Chicago, IL, July 2016.
- M. Ali, J. Nelson, R. Shea and M. J. Freedman, *"Bootstrapping Trust in Distributed Systems with Blockchains"*, USENIX ;login: Issue: Vol. 41, No. 3, Pages 52-58, Fall 2016.

本版(2.0)白皮书描述了自2017年发布1.0版白皮书以来出现的主要变化。更早的白皮书请参见：

- M. Ali, R. Shea, J. Nelson and M. J. Freedman, *"Blockstack: A New Internet for Decentralized Applications"*, Whitepaper Version 1.1, Oct 2017.

本白皮书中描述的某些系统和理念也在此白皮书各作者的下述博士论文中进行了讨论：

- M. Ali, *Trust-to-Trust Design of a New Internet*, PhD dissertation, Princeton University, June 2017.
- J. Nelson, *Wide-area Software-defined Storage*, PhD dissertation, Princeton University, June 2018.

免责声明：Blockstack代币是由Blockstack Token LLC. 正在开发的一种加密资产。Blockstack Token LLC. 是一家特拉华州有限责任公司，其网址为：www. stackstoken. com。

本白皮书不构成关于Stacks代币的要约或出售以及不构成关于购买Stacks的任何其他机制。关于Stacks代币的任何要约或出售或任何相关文件将仅基于关于Stacks代币的最终发行文件。

该文件可以被视为1933年《证券法》项下条例A项下的“试水”资料。我们不承担根据条例A完成发行的任何义务。我们可能选择向表示有兴趣进行投资的某些人（但不是所有人）进行发行，而且发行可能并不根据条例A进行。只有美国证券交易委员会（SEC）“批准”我们向美国证券交易委员会报备的发行说明书后，我们才可能进行出售。发行说明书中包含的信息要比我们现在提供的信息更全面，而且在某些重要方面可能有所不同。您必须在投资前阅读在美国证券交易委员会报备的文件。

我们未在招揽任何款项或其他对价，而且如果作为响应支付了任何款项或其他对价，我们不会接受该等款项或其他对价。在美国证券交易委员会批准公司向美国证券交易委员会报备的发行说明书之前，我们不会接受购买证券的要约而且不会接收任何购价部分。在批准日之后发出接受通知之前的任何时间，均可撤销或取消任何该等要约，无需承担任何形式的责任或义务。

表示有兴趣不涉及任何形式的责任或义务。

有兴趣投资Stacks代币发行的任何人士应审阅我们与该发行有关的披露文件以及公开报备的发行说明书，发行说明书可在www. sec. gov. 获得。

Blockstack未作为经纪交易商或投资顾问在美国证券交易委员会（SEC）、金融业监管局（FINRA）或者任何其他金融监管机构进行登记、获得许可，或者受到监管，而且Blockstack未被许可提供任何金融建议或服务。

中文版免责声明：本文件为对英文原件的中文翻译件。如果本翻译件与英文原件间有任何不一致之处，以英文原件为准。

中文版翻译：苏波

Blockstack 去中心化计算网络

Muneeb Ali Jude Nelson Aaron Blankstein

Ryan Shea Michael J. Freedman*

https://blockstack.org

白皮书 2.0.4版

2019年5月20日

摘要

传统互联网应用的客户端/服务器模型，以及通常所说的云计算，都是在服务器端保持应用状态以达到扩容目的。近些年，大型科技公司保存着所有的用户数据，导致了几个问题：大规模的数据泄露，单点失败和单点控制，以及用户隐私的丧失。

这篇白皮书将展示Blockstack去中心化的计算网络。Blockstack为传统的云计算提供一个全栈的替代，来打造安全、隐私的应用程序。基于Blockstack的去中心化应用与传统互联网应用的一个关键不同在于，绝大部分业务逻辑和数据处理在客户端运行，而不是运行在应用提供商托管的中心化服务器上。转向去中心化的计算与20世纪80年代从大型主机转向桌面电脑在许多方面很相似。

Blockstack遵循端到端的设计原则，保持网络核心的尽可能简单，而将复杂性推送到系统边界（用户设备和用户控制的存储）。Stacks区块链是我们网络的基础，其设计成 (a)扩展去中心化的应用，并且(b)激励开发者在网络上打造高质量的应用。Stacks区块链采用新颖的可调谐工作量证明（Tunable Proof-of-Work）选举系统来安全的启动一条新区块链。我们的新智能合约语言优化了智能合约的安全和可预测性，容许对所有的交易进行静态分析。

我们架构中的一个关键组件称为 Gaia，是一个高度可扩展、性能优越的去中心化存储系统，提供用户控制的私人数据锁柜(data locker)。用户将其私人数据锁柜连接到Blockstack客户端软件，应用直接将用户数据写到数据锁柜里。Blockstack Auth提供了全球通用ID和认证系统，不必为每个应用单独注册用户，也不必使用密码登录，这种方式不如加密认证安全。

*普林斯顿大学计算机科学教授，Blockstack PBC. 技术顾问

我们的SDK和开发者工具使得开发一个Blockstack应用并不比开发传统的互联网应用更复杂，而且开发者无需担心运行服务器或数据库。截至2019年初，Blockstack在生产环境中正运行着超过100个独立的应用。得益于数年生产环境经验和应用开发者的反馈，Blockstack的架构演进至今。这篇白皮书是我们2017年早期那篇白皮书的一个主版本改动。

1 介绍

今天的互联网，已经发展了四十多年，从一个小型的科研项目，成长为数字触角伸向世界的庞然大物。尽管核心的底层互联网协议自20世纪90年代以来基本保持一致，互联网的应用层和服务架构却发生了极大的变化，以支持互联网应用的爆发式增长。

建设互联网应用的基本模型是上世纪90年代流行起来的客户端/服务器模型[1]。该模型短期利好，但长期带来很多负面后果。Web应用顺势而起，但是导致Web服务愈加依赖于远程服务器。云计算是基础的客户端/服务器模型的一个演进。今天，在云端存储着私有的用户数据，运行着应用业务逻辑和计算，管理着访问权限，等等。

最近的十年里，我们开始看到云计算带来的负面后果，人们由此开始质疑基于客户端/服务器开发软件的完整模型。大规模的数据泄露[2]， 用户隐私的丧失[3]，数据缺乏可移植性，以及根植于客户端/服务器模型核心设计带来的科技巨头[4]间广泛的互不信任。鉴于计算在人类社会中的重要性与日俱增，我们不能让过时的计算模型来定义我们现在的生活方式。

云计算的下一步演进将利用更强劲的客户端设备、边缘计算和全球连接以减少对这些中心化平台的依赖。趋势已经朝着去中心化计算演进，我们相信这是计算机工业自大型机转向桌面电脑以来最重大的技术变迁。去中心化计算可以改变软件如何构造和使用。其提供给开发者一系列新的工具，改变了用户和软件之间的关系：软件的存在是保护用户，软件的优化是为用户利益至上。

Blockstack是一个开源软件产品，在开源社区中设计、开发、成长为一个去中心化计算网络，为传统的云计算提供一个全栈的替代。Blockstack正在重新构想传统互联网的应用层，为去中心化应用提供一个全新网络；构建于Blockstack之上的应用将使用户拥有和直接控制他们自己的数据[5]。Blockstack使用现存的互联网传输层协议以及底层的通信协议，但是移除了应用层里的中心点。我们遵循端到端的设计原则[6,7]，以保持网络核心简单，而将复杂性推送到客户端。为了应用的可扩展性，我们将全局状态变化最小化，提供一个可靠的去中心化存储系统，其相比云存储性能相当。而且，我们的全栈方法为所有开发者构建去中心化应用必须的栈组件提供了默认的选项。Blockstack是模块化的，开发者可以轻易定制和集成其他替代技术。

这篇白皮书是我们2017年早期那篇白皮书的一个主版本改动，包含了我们从生产部署中得到的教训和应用开发者的反馈而做出的设计演进。2016年同行评审的出版物[8,9,10]中的很多部分也过时了，我们鼓励读者以本篇白皮书为准了解Blockstack的最新设计。本篇白皮书介绍了我们新的Stacks区块链的设计，以扩展去中心化的应用，

并且激励开发者在网络上打造高质量的应用（第2章）。我们创造了一种新的智能合约语言， 叫做Clarity，并为它起了一个中文名字， 为“清”。 清语言优化了安全和可预测性（第3章）。我们描绘了Gaia去中心化存储系统的设计（第4章），我们的认证协议（第5章），开发者工具（第6章），重点强调了几种目前应用开发者使用Blockstack的方式（第7章）。

1.1 去中心化计算概述

去中心化系统是一种特殊类型的分布式系统，不存在单一实体控制底层基础设施，节点有经济激励加入网络。最近对于去中心化网络的兴趣起始于比特币白皮书的发布[11]。在同时期的去中心化系统中，区块链和加密数字货币扮演了中心的角色。我们推荐读者查看文献[12]了解区块链和加密数字货币的背景。

今天，有许多不同类型的去中心化系统在生产环境中运行。作为第一，而且是当前最大的区块链网络，比特币的首要目标是跟踪和解决该种数字货币的权属问题。以太坊[13]的目标更宽泛：构建一台“世界计算机”，运行智能合约和去中心化应用。Filecoin[14]尝试构建一个去中心化文件托管和存储的网络。相比之下，Blockstack试图实现一个去中心化计算的*全栈（full-stack）*，聚焦于为安全、隐私应用赋能，而将区块链层处理的状态和逻辑最小化。

1.2 设计目标

Blockstack的设计优化源于下列属性：

1. **易使用。**去中心化应用应该像现在的互联网应用一样容易被终端用户所使用。此外，开发去中心化应用应该像在今天的云上开发一样容易。
2. **可扩展。**去中心化应用应该可以支持互联网级别的用户量，也就是数亿到十亿的用户量。为了达到这点，网络（包括区块链）必须可以随用户数和运行的应用数量进行扩展。
3. **用户控制。**采用去中心化计算的应用应该默认由用户控制。用户应该可以提供自己的计算和存储资源，而不是依赖于应用运营的服务器。

带着这些设计目标，Blockstack做出了自己的设计选择，将其与其他的“重”区块链和“世界计算机”设计哲学[13,15,16,17]的去中心化计算方案区别开来。

最小化区块链层的逻辑和状态：为了取得可扩展性，Blockstack在我们的“轻”区块链层最小化了应用的逻辑和数据。使用区块链操作记录应用逻辑和存储，本质上要比“链下”方法要慢。需要在全网范围和设备间同步和验证状态，显示出这种操作在吞吐量上极大的局限性。限制因素在于底层的全局连通带宽和典型网络节点上可用的内存/存储，也就是物理限制（而不是任何协议的限制）。

本地状态变化 vs. 全局状态变化：Blockstack平台使用全栈方法确保构建在Blockstack上的应用是*可扩展的*：与应用的交互尽可能改变本地的状态，而不是全局的

状态。正因为此，我们的存储系统（Gaia，见第4章）和认证协议（见第5章）是我们平台的基础组件——其使得应用不发起一个区块链交易，就可以和用户的私人数据存储交互，并且完成用户认证。Stacks区块链仅用于在去中心化的环境中，以一种一致的方式协调全局状态的变换（例如：注册一个全局唯一的用户名）。

可靠的云存储 vs. 对等存储：在Blockstack上构建的应用，其数据存储和用户是一体的（使用用户自己的私人数据锁柜），不需要在服务器端保存任何用户访问凭证。这种方式不仅将用户数据交由用户自己控制，而且为开发人员降低了复杂度：开发人员无需运行服务器和数据库，从而代替用户支付云服务的账单。此外，我们避免了点对点存储[18]固有的可靠性和性能问题，在一个去中心化的广域文件系统中改变了现有云存储提供商的位置——区块链层只存储指向用户数据锁柜的指针。

适用开发者的全栈SDK：Blockstack提供全栈方法，为开发去中心化应用所需的所有层提供了默认的选项。开发者SDK将区块链的复杂性和其他开发技术抽离；应用开发者能够使用Blockstack SDKs（第6章）轻松构建他们的应用。技术栈的不同层次是模块化的，可以根据需要使用其他技术。

与同期的去中心化计算方法除了这些不同外，我们的智能合约语言也做了独特的设计决策来优化智能合约的安全和可预测性（详情见第3章）。

1.3 新应用模型

Blockstack为开发者构建应用提供了一个新的模型，确保应用是去中心化的，而且默认是由用户控制的：

1. **无不透明数据库：**在客户端/服务器模型中，数据库是所有应用的核心组成部分，因为服务端需要存储和查询大量的用户数据。在去中心化计算中，开发者无需担心数据库的维护和安全问题，因为他们从一开始就不做数据托管。开发者更多聚焦在应用逻辑上，用户下载应用后，接入他们的私人数据锁柜。如果使用数据库的话，其功能等同于过去互联网的“搜索检索器”——索引公共数据的服务。任何人都可以使用底层（去中心化）的数据创建这些索引。
2. **无服务器：**在客户端/服务器模型中，应用通过增加服务器扩容，因为所有的计算都在服务端执行。在去中心化计算中，应用在客户端运行，每个用户将自身的计算和存储能力带入网络（而不是依赖于应用开发者）。开发者只需提供最少的基础设施托管应用代码，因为每个用户自带所需的存储和计算资源使用应用。
3. **智能合约：**在客户端/服务器模型中，全局状态变化由一个中央服务器协调，其是网络真理的唯一权威。在去中心化计算中，这些状态的变化是通过执行于一个开放区块链之上的智能合约解决的。
4. **去中心化认证：**传统的互联网中，用户认证通过使用某种信任的认证流程进行。如果应用维护了一个用户数据库，该应用通过密码认证用户，有时加入第二因子。如果应用依赖于某个第三方认证服务，像Google或Facebook，该应用将使用OAuth[11]协议从第三方认证服务获得验证结果。显然，在所有这些方法中用户自

己无法控制认证流程。在去中心化计算中，认证由用户的客户端执行，通过加密数字签名证明对区块链上注册的某个用户名的控制权。任何应用都可以独立验证证明。

5. **原生代币：**在传统的互联网应用中，支付通常采用像信用卡一样的第三方服务。数字代币是去中心化计算平台上的原生资产，如Blockstack和以太坊。用户对代币有直接的所有权，可以使用它们直接注册数字资产和智能合约，也可以支付智能合约的运行费用。这种原生代币的使用可通过智能合约编程，构建软件订阅服务，也可以自动化其他的应用功能。这种可编程的代币是传统互联网应用开发者无法得到的能力。

1.4 去中心化计算层次结构

Blockstack去中心化计算网络逻辑上位于传统互联网架构的“应用层”。然而，Blockstack网络自身由多个系统组成，共同为实现去中心化应用提供必需的组件：

1. **Stacks区块链：**Stacks区块链是Blockstack网络的基础。Stacks区块链使用户可以注册和控制数字资产，如通用用户名，并且可以注册/执行智能合约。像通用用户名这样的数字资产，允许用户接下来控制他们的存储以及更多功能——用户将其私有数据锁柜的访问凭证与其通用用户名进行连接。
2. **Gaia：**Gaia是一个用户控制的存储系统，使应用可以和私人数据锁柜交互。私人数据锁柜可以在一个云服务提供商，或者是其他的数据存储服务商托管。重要的是，用户控制使用哪一个提供商。Gaia上的数据经过加密，并使用用户密钥在客户端侧签名。用户的数据锁柜（data locker）可以通过查询Stacks区块链上的信息发现。
3. **Blockstack认证：**Blockstack认证协议是应用的去中心化认证协议。通过该协议用户可以使用自己拥有的ID进行认证，并且设置使用哪个Gaia服务器保存该用户的应用数据。
4. **Blockstack程序库和开发包：**开发者程序库（**Libraries**）和开发包（**SDKs**）位于平台堆栈的顶端，应用开发者和用户以此和Blockstack网络的不同组件进行交互。例如，Blockstack客户端软件允许用户注册并管理自己的ID。Blockstack的开发者程序库使开发人员构建Blockstack应用像构建传统的Web应用*一样简单*。

2 Stacks区块链

Stacks区块链是Blockstack网络的基础层。Stacks区块链为网络提供了全局共识和协调层，产生了Blockstack网络的原生代币，称为“*Stacks token*”。当用户注册通用用户名，软件许可，存储锁柜的指针等数字资产时，需要消耗Stacks代币作为“燃料”。当注册/执行智能合约时，Stacks代币也被用于支付给矿工。

本章我们展示Stacks区块链的高阶设计。关于这些设计如何实现和演进的细节，我们建议您阅读不同组件的SIP（Stacks Improvement Proposals）[1]。当更多的SIP被Stacks改进程序接受时，我们将更新本篇白皮书。Stacks区块链体现了如下的设计决策：

1. 一个可调谐的工作量证明（tunable proof-of-work）机制用于领导人选举
2. 一个燃烧证明（proof-of-burn）挖矿算法来重用现有区块链的算力
3. 一个新颖的对等网络（Atlas），节点连通采用基于图的随机游走算法，减少了取得共识需要的数据量
4. 一种智能合约语言，Clarity （清），非图灵完备，*解释型*语言

区块链版本：当前生产环境中运行的Stacks区块链是“版本1”，是部署基本功能的一个初始实现。Stacks区块链v1使用比特币网络实现其共识算法，支持Stacks代币操作，比如转账。Stacks区块链v1为一些用例实现了智能合约，比如Blockstack Naming System[8]。关于版本1功能和实现的更多细节，请查看Github上的已有实现[20]。本章中的剩余部分将讨论Stacks区块链“版本2”的设计。Stacks区块链v2实现了我们新共识算法和智能合约语言的完整功能，将是对版本1的一个主要升级。

2.1 领导人选举

Blockstack的第一代区块链逻辑上在Layer-1（L1）之上操作，每一个交易1：1对应于一个L1的比特币交易。这样做的原因是确保重组Blockstack区块链的难度就像重组比特币区块链的难度一样大——这是我们从Namecoin，一个更小型区块链网络上得到的一个安全问题的教训[8]。

Stacks区块链采用一个**可调谐的证明机制（Tunable Proofs）**用于领导人选举过程。可调谐证明机制是一个有附加功能的工作量证明（PoW）系统，可以重用另一个更成熟区块链的算力。采用可调谐证明机制，我们的目标是安全地启动一条新的区块链，慢慢转换到使用自身的PoW机制。可调谐证明机制有两部分：（a）自身的PoW 和（b）另一种加密数字货币的燃烧证明。

[1]网址：https://github.com/blockstack/blockstack-core/tree/develop/sip

在初始时，燃烧证明部分的挖矿有更大的权重。通过燃烧证明，矿工燃烧加密数字货币表明他们参与挖矿程序的兴趣。为了竞选领导人，候选人燃烧底层链的代币（这里是比特币），在领导人的候选（would-be）块里提交了一个初始的交易集。该次提交*同时也*表明了该领导人的分叉选择：当前块的共识哈希必须包含前一个块的头部。当出现有多个竞争的分叉时，那些选择在失败分叉上“挖矿”的领导人无法收到区块奖励和交易手续费，也不能恢复已烧掉的加密数字货币。

Stacks区块链中采用的燃烧证明机制可以达成：

高验证吞吐量。处理的Stacks交易数量与底层“燃烧链”（这里是比特币）的交易处理速率间解耦。使用燃烧证明选举允许Stacks交易的“*全部区块*”用底层燃烧链的每一个新区块来确认。

低延迟块采纳。通过采用单一领导人选举，我们的燃烧证明共识算法允许当前领导人在Stacks区块中立即包含一个来自于交易池（mempool）新交易。这个*区块流*模型允许用户在几秒钟内得知一个交易被区块采纳。

开放领导人集合。燃烧证明选举允许*任何人*成为领导人。这个机制确保Stacks区块链是一个开放的区块链（相对于依赖固定领导人集合的封闭区块链，或者委托权益证明（DPoS）系统，其行为功能上与封闭的集合类似）。而且，通过执行*单一领导人选举*，我们的共识算法确保潜在领导人间无需协调。

无挖矿硬件可参与。作为领导人参与，所需的工作涉及燃烧某种加密数字货币，而不是像传统的工作量证明挖矿方案。因此，参与领导人选举不需要挖矿硬件。任何能获得要燃烧的加密数字货币的人都能参与挖矿，哪怕只能负担得起最低限度的数量。

公平矿池。Stacks区块链天然地支持公平的矿池。任何参与到网络中的人可以燃烧加密数字货币支持某个指定领导人的选举。提交这样的“用户投票型燃烧”（user support burns）的用户将与该领导人分享等量份额的Stacks区块奖励。

故障恢复能力。这个设计确保发生燃烧链不稳定，或者不适合Stacks区块链挖矿时，Stacks区块链可以使用另一条不同的燃烧链。

关于燃烧证明组件的更多细节参见文献[21]。将来可能发生的情况是，一旦在Stacks区块链上有足够的自身算力，燃烧证明组件就不再需要了。

2.2 可调谐证明

除了燃烧证明之外，Stacks区块链的共识算法中包含了一个内置的工作量证明（PoW）组件。这种组合分担了在SIP-001中描述的燃烧证明选举系统所在区块链的安全职责。这种内置的工作量证明和燃烧证明的组合在我们系统里被称为可调谐证明。

允许引入内置的PoW挖矿，而在当前PoW利益较低时，通过燃烧证明确保区块链的稳定。当底层燃烧链走向衰落时，可调谐的功能给我们更灵活的迁移。可协调机制的设计的灵活性还扩展到今后新的共识机制。我们可以研究其它PoW和PoS的共识机制， 如果未来有必要采取新的共识机制， 也可以安全合理的引入

PoW组件在领导人选举中是这样工作的，其允许候选人在他们的燃烧交易中可选地包含一个PoW随机数。产生此随机数需要的工作量（就是某种函数，其计算结果哈希前置多少个0）对应成候选人的“燃烧数量”。在初始时，内置的PoW将占比5%（相对于已提交的燃烧数量）。内置的PoW组件仍在大量的设计和开发之中。随着更多的细节具体化，这部分内容（以及一个对应的SIP）将被更新。

2.3 Atlas对等网络

Atlas对等网络是一个内容可寻址的对等网络，实现了一个Gossip协议，每个节点跟踪哪些其他节点当前在网络中，每个节点试图保存网络中所有数据的一个完整副本。该网络的容量受到了Stacks区块链的限制：数据集中的每一条新纪录，都必须和Stacks区块链上的一个交易相关联。Atlas对等网络是作为Stacks区块链的一个子系统工作的。其设计成一个无结构对等网络（unstructured peer network）以避免节点加入或离开网络引起的问题[18,22]。而且，既然每个节点都保留所有数据的一个副本，数据的索引在Stacks区块链上可用，那么新的Atlas节点可以快速同步其需要存储的数据，因为事先知道应该从其他节点存储什么数据（通常在点对点网络中这对节点是未知的）。

Atlas网络作为Stacks区块链的“扩展存储”子系统运行。我们的设计方案是*尽可能少*地依赖于直接与Stacks区块链自身的交互，尽可能少地在其上存储数据。对于许多在Blockstack上的应用，例如Blockstack Naming System（BNS)的智能合约[8]，其本质上是有一个机制存储*不可改变*和*带有时间戳*的数据。在BNS里，这被用来将用户名和路由信息关联，通过路由信息可以发现用户个人资料和应用数据。大多数的区块链*直接*将这种数据存储在区块链上。然而，我们相反选择将哈希存储在区块链上（空间昂贵），并实现了一个单独的对等网络来交换对应于这些哈希的数据。

2.4 Stacks代币用途

Stacks区块链实现的原生Stacks代币激活了Blockstack网络上的几项基础操作：

1. **注册数字资产的燃料。**Stacks代币用来注册不同种类的数字资产，例如：用户名，域名，软件授权，播客，还有一些其他的。
2. **注册/执行智能合约的燃料。**执行智能合约需要燃料以支付验证合约正确性并执行合约的开销。Stacks代币也被用来核销在Stacks区块链上存储智能合约的成本。
3. **交易手续费。**Stacks代币被用来支付交易手续费，以此Stacks区块链才能记录该交易。
4. **锚定的应用链。**对于在Blockstack上广受欢迎的应用，我们的区块链有一个可扩展的入口，应用可以在Stacks区块链上初始化自己的区块链。这样的“应用链”

燃烧Stacks代币挖矿。

上面的列表并不是全部——随着Blockstack网络的成熟，我们期待网络参与者会发现和开发出Stacks代币的其他用处。我们当前正积极研究一个“应用权益”机制，使代币持有者可以潜在参与我们的“应用挖矿”开发者激励计划中。

3 Clarity （清）智能合约语言

Stacks区块链支持加载和执行智能合约，以对数字资产进行编程控制。“清”这个新智能合约语言优化了安全和可预测性，强调了其不同于先前的智能合约系统的一些关键设计目标：

1. 该语言必须允许快速、精确的运行时和空间需求的静态分析。为了支持这一点，该语言在单一交易的执行中是非图灵完备的。但就完整的交易历史来说，该语言是图灵完备的。

2. 智能合约应该是由我们的虚拟机*解释*执行的，而非编译。开发者编写的合约代码必须直接部署到区块链上。

为了实现以上两个性质，我们创造了一个新的LISP语言变种，特殊设计成智能合约的编写语言。在SIP-002[24]中有关于Clarity（清）语言设计的更详细讨论。

3.1 语言概览

智能合约语言Clarity（清）与其他的LISP变种（例如：Scheme）相似，但是有如下的不同：

1. 递归是非法的，并且没有lambda函数

2. 循环只能通过map, filter，或fold执行

3. 唯一的原子类型是布尔，整型，定长数组，以及控制主体（principals，是Blockstack智能合约语言中特有的数据类型）

4. 对原子类型的列表提供额外的支持，但该语言中唯一变长的列表只能作为函数输入出现（也就是说，对类似append或join的列表操作不支持）。我们也支持名称和类型指定（named-and-typed）的元组。

5. 变量仅能使用let绑定创建，不支持类似set的可变功能

6. 允许使用define语句定义常量和函数来简化代码。但这纯粹是句法意义上的。如果一个定义不能被还原（inlined），该合约将被视为非法被拒绝。这些定义同时也是私有的，在某个函数中这样的定义，只能被指定智能合约中定义的其他函数

所调用。

7. 通过define-public语句指定的函数是公有函数。传给这些函数的参数必须指定类型。

智能合约有下面的权力：

1. 从其他的智能合约调用公有函数。这些智能合约以其哈希标识，被调用智能合约必须在调用者智能合约发布时已存在。结合递归的非法性，这样做将防止函数重入，这在现存的智能合约平台上是一个通用的攻击途径。
2. 拥有并控制数字资产。就像公钥或多重签名地址一样，智能合约是一级主体。

每个智能合约有自己的数据空间（data-space）。在此数据空间中的数据被存储在map里。这些存储将一个typed-tuple与另一个typed-tuple相关联（很像一个有类型的键值对数据库）。相对于表（table）数据结构，map只关联一个指定的key到一个唯一确定的value。

每个智能合约可以从任何其他智能合约的map里取数据。但是，只有一个智能合约可以直接在其自己的map里更新数据。

由于以下两点原因，我们选择使用map，而不是其他的数据结构：

1. 由于map结构的简单性，使得虚拟机内实现更简单，函数推导更容易。通过检查一个指定函数的定义，可以清楚的看到哪些map将被修改；甚至在那些map中，哪些key会受到一个给定调用的影响。
2. Map接口确保其操作的返回类型是定长的，这对智能合约运行时、成本和其他属性的静态分析是必须的。

3.2 图灵不完备和静态分析

创造一个非图灵完备语言是一个重要的设计考虑。在区块链这种恶意的网络环境中，这一点为编程带来了许多好处。

1. 图灵不完备使静态分析能够决定执行一个指定交易的成本。这允许网络*预先*清楚地知道向一个指定交易收取多少手续费。这也会提升客户端的体验，因为对客户端来说广播一个交易的成本可知了，所以能容易地传达给用户。
2. 图灵不完备允许静态分析可以快速决定一些重要属性，例如单个交易可能调用了哪几个合约。这提升了用户体验，因为客户端可以警告用户关于一个给定交易的任何潜在副作用。
3. 改进的、精确的静态分析将允许程序员充满信心地分析他们的智能合约，在上线之前发现任何可能的缺陷和错误。

基本上我们认为，像对待其他编程形式一样对待智能合约编程是个错误。区块链的性质造就了智能合约非常重要的特性。我们认为牺牲编程的简易，来换取增进的人和计算机对智能合约行为的全面理解，是一个好的权衡。已有的智能合约使用实践证实了这点——图灵完备智能合约的历史基本上是智能合约bug的历史。

在智能合约语言“清”中，在广播智能合约之前运行的静态分析可以提供如下的信息：

1. 广播指定交易的成本是其输入大小的函数

2. 可以修改任意特定表的交易集合

未来的工作可以支持甚至更高级的分析功能，例如自动检查智能合约代码上的证据的能力。

3.3 解释型语言 vs. 编译型语言

在我们智能合约语言Clarity（清）中的另一个关键设计决策是，选择一门解释型语言，而不是编译型语言（例如，编译成WASM）。与同时期的其他方法相比，我们不采用编译器的设计决策是一个根本的不同。采用这种设计决策的主要原因，是对程序实现的bug归因的能力。

程序实现的bug是无法更改的事实，即使有最好的编码规范，也无从避免。智能合约的bug（区块链）是如此，其他代码也一样。智能合约的bug处理起来更复杂。不同的区块链社区奉行“代码即法律”的哲学，提交到区块链上的法则是最终真相的来源（source of ultimate truth）。编写智能合约的开发者通过源代码表达他们的意愿，然而是编译将他们的意愿转换为实际的法则。这会导致因为编译器的bug使实际法则偏离开发者意愿的情况。会产生令人不快的情况，人们将争论是开发者意愿更重要还是法则更重要。在Stacks区块链里，我们移除编译步骤，直接提交开发者意愿到区块链，避免这种状况，以使开发者意愿绝不会偏离法则。

让我们考虑一下在智能合约语言（也就是虚拟机）的实现里出现bug的情况。如果智能合约语言使用解释器，那么解决bug相对容易实施。世界上的所有合约代码都在区块链上，只需要对解释器打一个补丁，然后从创世块开始重启区块链（重新应用到所有的交易）。

但是如果智能合约语言是编译的，该bug是在编译器内部，而非虚拟机，那补救措施就没那么明显了，因而可能会发生更多的争议。这是因为编译器里的一个bug可以导致其生成的代码（最终广播到区块链上）产生与开发者初衷背离的行为。在加密数字货币社区里，在对“代码即法律”哲学的认同下，这种情况愈加复杂。由开发者编写的代码是正确的，但是区块链上产生的交易是错误的。收集每个开发者的源代码并重新编译是不现实的，尤其在无法验证源代码是否已改变的情况下。我们怀疑，在实践中这种情况下在区块链上发布的代码多数是最终真相的源。如果是这样，开发者应该

归因和验证此代码，而不是他们的源代码。我们相信，使用一个高级的解释型语言对于确保正确的智能合约执行是至关重要的。

4 Gaia：用户控制的存储

Blockstack使用*Gaia*存储系统给用户对其数据的控制权。这是一个用户控制的存储系统，使应用可以与私人数据锁柜（private data lockers）交互。私人数据锁柜可以托管在一个云服务提供商，或者是其他的数据存储服务商。重要的是，用户控制使用哪一个提供商。Gaia上的数据经过加密，并使用用户控制的密钥签名。逻辑上，Gaia像是一个广域的文件系统，可以被挂载以保存文件。

使用Gaia存储系统，用户要指定一个Gaia存储位置的地址，在此保存数据。只是Gaia存储位置的“指针”被保存到Stacks区块链上（以及Atlas子系统上）。当用户使用Blockstack认证协议登录应用和服务时，将此位置传递给应用；有了这个信息，应用知道如何与指定地Gaia数据锁柜交流，如此应用数据被保存到用户指定的存储里。

Gaia的设计哲学是，以一种终端用户无需信任底层云服务提供商的方式重用现有的云服务提供商和基础设施。我们看待云存储服务提供商（像Amazon S3, Google Cloud Storage，甚至一个本地磁盘）只是作为一个通道（dumb drives），在上面存储加密过的，和/或签名过的数据。云服务提供商看不到用户的数据；他们只能看到加密的数据块。

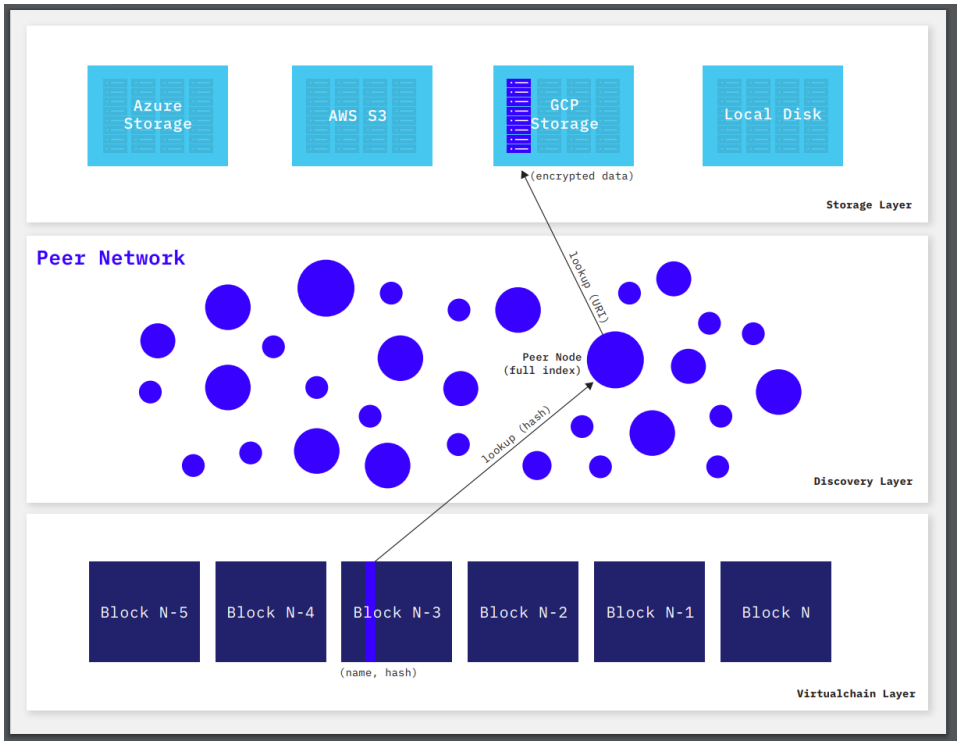

图 1：Gaia概览及数据查找步骤

而且，因为通过Stacks区块链可发现相关联的公钥或数据哈希，云服务提供商没有篡改用户数据的能力。

向Gaia服务器写数据涉及到POST数据到服务器的合适位置。这些POST在服务器端被验证，检查是否此写请求附带了一个签名的*认证*token。这个token使用私钥签名，控制了是否对特定的bucket有写权限。为了给用户的每一个应用提供分离的bucket，用户将为每一个应用推导出不同的私钥。每一个私钥仅赋予Gaia服务器上特定bucket的访问权限。

在Gaia里，用户的区块链验证的路由信息中包含一个URL，指向一个签名过的JSON对象（由该用户的owner key签名）。这个签名过的JSON对象内包含指向该用户Gaia数据锁柜的URL。一旦应用知道用户的Gaia数据锁柜的位置，就可以使用普通的HTTP请求向那个位置请求一个文件。为了查询*不同*用户创建的文件，应用可以完全在客户端顺序地执行查询。这在初始查询时会有时间延迟的开销，但是路由信息中的许多数据可以由浏览器本地缓存（或者由本地应用），所以后续的查询就像传统的互联网数据访问一样快了。

图1显示了Gaia的总览。查询一个像werner. id一样的名字，其工作流程如下：

1. 在Stacks区块链上查询该名字，取得（name, hash）对
2. 在Blockstack的Atlas对等网络里查询hash(name)，得到该名字的路由信息文件。
3. 从路由信息文件中取得该用户的Gaia URL，接着查询此URL，连接至存储后端。
4. 从指定的Gaia服务处GET/PUT数据（如果需要且读者有此访问权限，则进行解密），并验证相应的签名或哈希。

上面的步骤1和步骤2可通过对blockstack-core在/v1/names/<name>访问点上的一个简单调用即可执行。这些重复的读写操作已在我们的开发者程序库中自动处理了。

性能。我们架构的目标是相对构建于云服务提供商之上的传统互联网应用提供可比的性能。通过移除控制和失效的中心点，我们引入了有意义的安全和容错的好处——在读写性能上付出一些小的额外开销是值得的，只要这些额外开销对普通用户不那么重大或显而易见。我们评估了Gaia的读写性能，显示出其在读写底层存储的文件上是有竞争力的。由于加密的缘故，Gaia为每个文件增加了一个可忽略的固定大小的存储空间开销（粗略为文件大小的5%）。加密也会有CPU的额外开销；但是因为文件尺寸变化非常小，读写的网络性能与直接访问底层的存储服务是相似的。

系统扩展性。我们架构的存储层不是一个可扩展性的瓶颈。同期的云存储系统是高度可扩展的[25]。Atlas网络也是扩展性良好，因为其并不索引单独的用户文件或文件块，只索引指向用户存储后台的指针。存储后台处理批量的数据读写操作，Atlas网络只在

下述情况下参与 (a)用户改变或更新了他的存储后台或者公钥映射，或(b)新用户在系统中注册。当注册新域名/用户时，路由文件的哈希必须在区块链上广播。区块链可能是可扩展性的一个瓶颈（相对于Atlas网络），但用户极少写区块链。另外，*链下*名字注册的使用可以在一个单一的区块链交易中注册超过100个用户，由此可以支持每天数十万个用户注册（可以与传统云上平台的每日新用户数量相比）。提升Gaia到10亿用户量级在实践中可能会暴露出可扩展性问题，但明显现在不会发生，解决这些挑战是正在进行和将来工作的一个研究领域。

5 认证

使用互联网应用，用户账户是至关重要的。Blockstack提供给用户一个通用的用户名，无需任何密码，可用于所有的应用。不像基于密码的认证，用户使用公钥密码学进行认证：一个本地运行的软件客户端处理来自特定应用的登录请求，并对认证请求签名。

Blockstack Auth是我们的认证协议，其将应用与用户的Gaia hub以及任何应用相关的私钥连接在一起。应用使用这些信息将用户和其数据保存，验证其他用户产生的数据是真实的。

5.1 单点登录

Blockstack Auth使用公钥密码学进行认证。用户登录一个应用以使此应用可以产生和存储签名过的数据，其他用户可以读取和验证其数据。这反过来向其他用户证明了该登录用户是合法的。

在Blockstack中，登录的目的是向应用客户端提供足够的信息，来产生和存储真实的数据。这意味着，认证功能可以以一个认证器（authenticator）应用的形式，*独立运行在用户的计算机上*。因为所有的名字都是在Stacks区块链上注册的，每个应用和认证器一直有一个最新的视图 （1）所有存在的名字，和（2）所有名字的公钥及Gaia hub。这消除了对一个服务端ID提供者的需要。

为了认证用户数据，应用客户端只需能够联系一个Stacks区块链节点。为此用户在登录时向应用提供其首选Stacks节点的网络地址。

用户通过点击“登录”按钮来登录一个Blockstack应用。此应用（调用 *blockstack.js* SDK）将用户重定向到Blockstack认证器应用，请求登录。用户将看到可选择的用于登录的Blockstack ID，同时还有一个应用所需的权限列表。选择一个ID，认证器则将用户导回到应用，并向应用传递三个信息：

1. 用户的用户名（或者是公钥的哈希，如果还没用户名的话）
2. *应用特定*的私钥，用来加密和签名用户的数据。这是使用用户主私钥、登录使用的ID和应用的HTTP Origin所生成的确定性密钥。
3. 用户Gaia hub的URL，以及用来查询其他用户和数据的首选Stacks区块链节点。

有了这些，用户展现了其用户名，通知应用哪里可以找到和存储其数据。在那里，应用可以持久化地读写应用特定的数据，访问其他用户的应用特定数据——所有这些不需要提供其自身的存储或ID解决方案。

登出操作简单地清除应用的本地状态，因此导致Web浏览器和客户端忘记应用特定的私钥。

6 Blockstack程序库和开发包

Blockstack PBC是一个公益公司（Public Benefit Corp），和开源贡献者一起开发了Blockstack的核心协议和开发者程序库。开发者程序库使开发人员在Blockstack网络上构建应用更简单，而Blockstack客户端使用户可以和Blockstack网络的不同组件以及不同应用进行交互。

6.1 开发者程序库

Blockstack设计成让开发人员开发去中心化应用尽可能地简单。与Stacks区块链或去中心化存储的交互复杂性大多向应用开发者隐藏了，他们可以只关注应用的逻辑。Blockstack开源代码库包含了一些不同平台的开发者程序库：一个Javascript Web SDK（blockstack.js），iOS和Android的移动SDK。所有这些程序库都是在MIT许可协议下可用，访问此链接可获取 `https://github.com/blockstack`。

这些程序库提供所有必须的API接口，以及实现我们认证协议的代码，直接与Gaia服务器交互，生成Stacks交易。使用这些程序库允许开发者创建尊重用户安全和隐私的去中心化应用，就像开发传统的应用一样容易。

Radiks 对于希望穿透复杂的社交图谱分享数据的应用来说，对数据建立*索引*通常是有用的和最有效的。Radiks系统是一个服务器和客户端的程序库，用来构建并与这样的索引交互。Radiks程序库使开发人员可以在应用内创建跨用户的结构化数据集，可以通过字段的值查询。这要求一个服务器端的组件处理索引和查询，可关键这*不*是用户信任的计算环境的一部分。其只能看到数据的密文和一些必要的元数据，后者用于构建索引以及通过索引应答查询。关于Radiks的更多信息参见文献[26]。

6.2 用户软件

虽然应用开发者将使用开发包和程序库与Blockstack网络进行交互，但用户还需要软件来执行诸如注册用户、指定其Gaia服务器以及应用用户认证等功能。Blockstack生态系统目前提供两个可使用户与网络进行交互的开源项目：

1. **Blockstack浏览器**。这是目前推荐的认证器应用的一个开源实现，而且其允许用户浏览可用的Blockstack应用、注册用户名以及认证应用用户。Blockstack浏览器可以在桌面上进行本地安装，也可以采用web部署。

2. **Blockstack CLI**。这是一个命令行实用程序，允许高级用户和开发者与Blockstack协议交互。除了提供认证功能，其允许用户创建原始交易，以及通过Gaia进行高级数据管理任务。

6.3 文档和社区资源

Blockstack开源社区维护有学习指南、API文档和系统设计文档，可在Github和https://docs. blockstack.org上获得。

Blockstack用户和开发者可获得下述官方社区资源。

- **Github**：所有软件开发在Github进行，

  网址：https://github.com/blockstack

- **论坛**：高级用户和开发者经常在Blockstack论坛上回答技术问题、分享想法并且帮助彼此。网址：https://forum. blockstack.org

- **Slack**：Blockstack社区采用公开Slack小组实时交流，

  网址：https://blockstack.slack.com

7 应用和服务

截至2019年初，Blockstack上已经搭建了100多个应用。开发者正在搭建各种不同类型的应用，在app. co上可找到持续增多的Blockstack应用的完整清单。由于Blockstack是模块化的，不同的应用可独立地使用不同的组件。以下是我们对一些示例用例做出的简要概述。

目前，Blockstack上的办公效率应用[27,28,29,30]使用Blockstack认证和Gaia存储，用户可以创建、编辑以及共享文件。为帮助用户发现彼此的文件，这些应用使用Blockstack个人资料检索器。该检索器是去中心化的——因为个人资料集是全球可见的，是可以被发现的，任何人都可以部署及运行个人资料检索器。

Blockstack生态系统还包含许多社交应用[31,32,33,34]。通常情况下，这些社交应用使用 Blockstack认证，同时部署一个Radiks服务器，以使用户高效地发现并获取其他用户的资料。在至少一个用例中，应用使用一个专用的中继通道在众多用户间路由加密的信息[31]。Blockstack上的发布和存储应用[35,36,37,38,39,40]不仅使用Gaia存储用户资料，而且还将其通过传统的HTTP URL与非Blockstack用户分享。

开发者奖励。Stacks区块链扩大了挖矿的概念，应用开发者可以通过在网络上发布高质量的应用 “挖”Stacks代币。这一机制被称作应用挖矿，其被设计成一种激励机制，以期在网络上获得高质量应用。应用挖矿计划目前由拥有多名独立审阅人的Blockstack PBC运营。开发者可以将其应用每月提交一次以审阅，并基于其应用在应用排名机制中的表现获得奖励。应用由一组独立审阅人审阅，每名审阅人对于什么样的应用才是好

应用有自己的评定标准。应用得到的*总分*决定了其排名情况。排名和奖励费按月发布。有关应用挖矿计划的细节超出了本白皮书范围，读者请参见以下链接获取详情 https://app.co/mining。

8 结论

Blockstack是一个去中心化计算网络，向开发者提供了用于搭建去中心化应用的全栈。迄今为止，我们的网络上已经搭建了100多个去中心化应用。Blockstack无需开发者运行服务器和数据库：取而代之的是用应用将数据写到用户控制的私人数据锁柜里。这一去中心化存储系统与传统云存储在性能上相当，只因加密/解密引入一点开销。我们的认证协议无需采用基于密码的登录方式，那种方式不如加密认证安全。用户可以使用单一账户访问所有服务和应用，不必持续不断的为新服务创建新账户。我们的开发者程序库使得在该平台上开发去中心化应用与搭建传统互联网应用一样简单。

在本白皮书中，我们呈现了Blockstack的最新设计。自2016年和2017年产品的早期实现以来，Blockstack的核心设计一直在演进，吸取了从产品部署中得到的经验教训以及去中心化应用开发者的反馈。与早期（2017年）的白皮书相比，主要变化包括(a)对Stacks 区块链的说明，其使用新颖的可调谐证明机制来安全地启动一条新的区块链，和(b)对新智能合约语言Clarity（清）的说明，关注智能合约的安全和可预测性。我们已经以开源的方式发布了Blockstack [41]。

致谢

多年来，许多人为Blockstack的设计和实现做出了贡献。我们值此想要特别感谢以下人士在其专长领域所做出的贡献：

早期贡献和理念：Larry Salibra、Ken Liao、Guy Lepage、Patrick Stanley和John Light；

开发者程序库和开发包：Hank Stoever、Shreyas Thiagaraj和Matthew Little；

产品设计和开发者文档：Jeff Domke、Mark Hendrickson、Thomas Osmonson、Jasper Jansz和Mary Anthony；

基础架构开发：Jesse Wiley、 Virginia Hickox和Tim Wells；

有益意见和反馈：Brittany Laughlin 和Diwaker Gupta。

有关更多Blockstack贡献者的信息，请参见 https://github.com/blockstack。总之， 太多的人为这个项目做出了贡献，无法一一致谢——我们对整个Blockstack开源社区所给予的支持心怀感激。

引用

[1]S. Sulyman, “Client-server model,” *IOSR Journal of Computer Engineering*, vol. 16, pp. 57–71, 01 2014.

[2]N. Perlroth, “Yahoo says hackers stole data on 500 million users in 2014,” Sept. 2016. http://nyti.ms/2oAqn0G.

[3]K. Granville, “Facebook and cambridge analytica: What you need to know as fallout widens,” Mar. 2018. https://nyti.ms/2HP4Dr3.

[4]R. Mcnamee, “I mentored mark zuckerberg. i loved facebook. but i can’t stay silent about what’s happening.,” Jan. 2019. http://time.com/5505441/mark-zuckerberg-mentor-facebook-downfall/.

[5]“Blockstack website,” 2019. http://blockstack.org.

[6]J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-to-end arguments in system design,” *ACM Trans. Comput. Syst.*, vol. 2, pp. 277–288, Nov. 1984.

[7]D. D. Clark and M. S. Blumenthal, “The end-to-end argument and application design: The role of trust,” *Federal Comm. Law Journal*, vol. 63, no. 2, 2011.

[8]M. Ali, J. Nelson, R. Shea, and M. Freedman, “Blockstack: A global naming and storage system secured by blockchains,” in *Proc. USENIX Annual Technical Conference (ATC ’16)*, June 2016.

[9]J. Nelson, M. Ali, R. Shea, and M. J. Freedman, “Extending existing blockchains with virtualchain,” in *Workshop on Distributed Cryptocurrencies and Consensus Ledgers (DCCL’16)*, (Chicago, IL), June 2016.

[10]M. Ali, J. Nelson, R. Shea, and M. J. Freedman, “Bootstrapping trust in distributed systems with blockchains,” *USENIX ;login:*, vol. 41, no. 3, pp. 52–58, 2016.

[11]Satoshi Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” tech report, 2009. https://bitcoin.org/bitcoin.pdf.

[12]A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton, NJ, USA: Princeton University Press, 2016.

[13]V. Buterin, “A next-generation smart contract and decentralized application platform,” tech. rep., 2017. https://github.com/ethereum/wiki/wiki/White-Paper.

[14]“Filecoin: A Cryptocurrency Operated File Network,” tech report, 2014. http://filecoin.io/filecoin.pdf.

[15] https://eos.io.

[16]T. Hanke, M. Movahedi, and D. William, "Dfinity technology overview series consensus system rev.1," 2018. https://dfinity.org.

[17]"Ethereum 2.0 specifications," 2019. https://github.com/ethereum/eth2.0-specs.

[18]Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer- to-peer overlay network schemes," *IEEE Communications Surveys Tutorials*, vol. 7, pp. 72–93, Second 2005.

[19]"Oauth." https://oauth.net.

[20]"Blockstack Core: Stacks blockchain v1," 2018. https://github.com/blockstack/blockstack-core/tree/v20.0.8.1.

[21]"SIP 001: Burn Election," 2019. https://github.com/blockstack/blockstack-core/blob/develop/sip/sip-001-burn-election.md.

[22]I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pp. 149–160, 2001.

[23]"App mining." https://app.co/mining/.

[24]"SIP 002: Smart Contract Language," 2019. https://github.com/blockstack/blockstack-core/blob/develop/sip/sip-002-smart-contract-language.md.

[25]"Google Cloud Storage SLA." Retrieved from https://cloud.google.com/storage/sla in May 2017.

[26]"Radiks." https://github.com/blockstack-radiks.

[27]J. E. Hunter, "Graphite docs," Feb. 2019. https://app.graphite-docs.com.

[28]D. Travino, "Noteriot," Feb. 2019. https://note.riot.ai/.

[29]"Forms.id," Feb. 2019. https://forms.id.

[30]"Blockusign," Feb. 2019. https://blockusign.io.

[31]P. Bhardwaj and A. Carreira, "Stealthy," Feb. 2019. https://www.stealthy.im.

[32]A. Sewrathan, R. Adjei, and F. Madutsa, "Afari," Feb. 2019. https://afari.io.

[33]T. Alves, "Recall," Feb. 2019. https://app.recall.photos/.

[34]T. Alves, "Travelstack," Feb. 2019. https://app.travelstack.club.

[35]J. E. Hunter, "Graphite publishing," Feb. 2019. https://publishing.graphitedocs.com.

[36]"Decs," Feb. 2019. https://app.decs.xyz.

[37]"Sigle," Feb. 2019. https://app.sigle.io.

[38]"Xorbrowser," Feb. 2019. https://xorbrowser.com.

[39]"Mevaul," Feb. 2019. https://mevaul.com/.

[40]"Xordrive," Feb. 2019. https://xordrive.io.

[41]"Blockstack source code release v20.0.8," 2019. http://github.com/blockstack/blockstack-core.