Computer Science and Application 计算机科学与应用, 2021, 11(2), 383-393
Published Online February 2021 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2021.112038

Hans汉斯

基于可验证随机函数的权威证明共识算法改进

周　宇，胡振宇，杨振国，刘文印

广东工业大学计算机学院，广东 广州
Email: yuzhou1su@foxmail.com, yuzhenhu98@outlook.com, yzg@gdut.edu.cn, liuwy@gdut.edu.cn

收稿日期：2021年1月19日；录用日期：2021年2月13日；发布日期：2021年2月23日

摘　要

权威证明(Proof of Authority, PoA)共识算法中的领导节点需要提出和确认区块，针对该算法中领导节点选取容易导致权威节点公开易遭受攻击、权威节点不能变更等问题，提出一种基于可验证随机函数领导节点随机选取和动态变更权威节点的可验证权威证明(Verifiable Proof of Authority, VPoA)改进算法。首先，改进PoA共识算法中领导节点的选取方法，在所有权威节点集合中随机选取一个领导节点，其他节点可以验证该领导节点的身份，但无法预知下一阶段中的领导节点，防止权威节点的公开被攻击。其次，PoA中其他节点不经过验证直接接受新区块，VPoA在共识流程中改进区块接收阶段，可以通过投票验证区块的有效性，防止领导节点作恶的情况。此外，VPoA实现节点状态动态变更，即可以通过已参与共识的权威节点来投票决定新节点加入和恶意节点的移除。最后实现了VPoA方案，并进行了实验和安全分析，结果表明VPoA共识改进方案能够正确有效随机选取领导节点，提供稳定的区块出块时间，实现区块链网络共识的一致性。

关键词

区块链，共识算法，权威证明，可验证随机函数

Improved Proof of Authority Consensus Based on Verifiable Random Functions

Yu Zhou, Zhenyu Hu, Zhenguo Yang, Wenyin Liu

School of Computers, Guangdong University of Technology, Guangzhou Guangdong
Email: yuzhou1su@foxmail.com, yuzhenhu98@outlook.com, yzg@gdut.edu.cn, liuwy@gdut.edu.cn

Received: Jan. 19th, 2021; accepted: Feb. 13th, 2021; published: Feb. 23rd, 2021

文章引用: 周宇, 胡振宇, 杨振国, 刘文印. 基于可验证随机函数的权威证明共识算法改进[J]. 计算机科学与应用, 2021, 11(2): 383-393. DOI: 10.12677/csa.2021.112038

Abstract

Proof of Authority (PoA) needs leader node to propose block and confirm block. The selection of leader node may expose the authorities, which is vulnerable to attacks and result in the authorities cannot be updated. In this paper, we propose a Verifiable Proof of Authority (VPoA) algorithm based on Verifiable Random Function, aiming to dynamic update the leader node. Firstly, a leader node is randomly selected from the set of all authorities, which can prevent the publicity of authorities from being attacked. The other nodes can verify the identity of the leader node, but they cannot predict the leader node in the next phase. Secondly, other nodes in PoA directly accept new blocks without verification. VPoA adds a Block Acceptance phase to the consensus process, which can verify the validity of the block by voting to prevent the leader node from conducting evil. In addition, VPoA realizes the dynamic change of node status, that is, authority nodes can determine adding new nodes and deleting malicious nodes. Finally, we implement the VPoA and conduct experiments and security analysis. The VPoA consensus can correctly and effectively select the leader node randomly, ensure a stable block generation time, and achieve the consensus of the blockchain network.

Keywords

Blockchain, Consensus Algorithm, Proof of Authority, Verifiable Random Functions

1. 引言

2008 年，Nakamoto 首次提出比特币[1]，数字货币逐渐得到各界人士的重视，因此区块链技术成为了研究热点。区块链技术通过链状结构、P2P 网络协议、非对称加密、共识算法等技术的融合，在不可信环境中解决了数据的可信问题。区块链的本质是一种分布式数据库，在分布式数据库中，首先要解决的就是多节点如何保证一致性和如何达成共识等问题，可以说，共识算法是区块链技术的基础和核心。随着区块链技术在不同领域的应用，共识算法的研究也成为区块链中的一个研究热点，相继出现了多种主流共识算法[2]。

比特币中使用的工作量证明(Proof of Work, PoW)算法[3]是目前最常见的共识算法，该算法依赖于机器来进行高难度的数学运算来获取记账的权利，最先计算出哈希难题(也称挖矿)的节点会获取区块链的记账权，同时获得系统奖励，这个过程被称作工作量证明。但是，PoW 达成共识的过程需要大量的算力资源用在无用哈希计算上，因此系统在每秒交易量(Transaction Per Second, TPS)方面的性能有限，同时也会消耗大量的资源。

为了提高 TPS，第二代区块链系统以太坊[4]中使用了权益证明(Proof of Stake, PoS)算法，2012 年，King 等人在点点币中使用了权益证明算法[5]，节点获得记账权利的难度和节点所掌控的权益成反比，权益越大越容易获得记账权，因此权益小的节点失去了挖矿的动力，因而影响到整个系统去中心化程度和安全性。

PBFT (Practical Byzantine Fault Tolerance)共识算法[6]为了达成数据的一致性采用了三阶段共识流程

的方式，而在 PREPARE 阶段与 COMMIT 阶段的过程达到 O(n^2)的通信复杂度，此处 n 为参与共识的总节点数量。随着节点规模的增大，达成共识需要的时间大大增加，整个网络的通信复杂度会随着节点的增加而呈指数级增大，可扩展性较差。

Gavin Wood 等提出权威证明(Proof of Authority, PoA)算法[7]，该算法通过可信身份授权来提供快速的交易，因为其高性能和对故障的高容错能力而被人关注，目前，以太坊的两个著名客户端 Parity [8]和 Geth [9]中实现并使用了这种共识算法。目前的 PoA 共识算法仍然存在一些缺陷，De Angelis 等分析了 PBFT 和 PoA 在联盟链上 CAP 理论的分析[10]，发现在联盟链中的使用 PoA 的确比 PBFT 有更好的性能，同时分析了两种主要的 PoA 算法，即 Aura [11]和 Clique [12]，最终发现 PoA 无法为数据完整性提供足够的一致性保证。在 PoA 共识中，认为参与共识的节点都是可信的，但实际中却是不可能的，也缺乏对区块链网络节点的有效管理。Ekparinya 等[13]人发现在 PoA 共识中存在克隆攻击(The Cloning Attack)，即可以通过克隆一个权威节点来实现对 PoA 算法的攻击，致使这种攻击的原因在于，在 PoA 中的领导节点取余法易暴露领导节点的选取顺序，导致下一轮权威节点被选取的过程在网络中是完全公开的，权威节点的公开可能导致安全性和第三方操纵，从而容易遭受被拒绝服务攻击(Distributed Denial-of-service attacks, DDos)攻击和审查攻击(Censorship attacks)，即攻击者可以通过恶意攻击导致权威节点的不诚实行为。

本文基于 PoA 共识算法，提出一种改进方案 VPoA 算法，其优势在于：1) 通过改进取余法这一简单领导节点轮换方式，使用可验证随机函数的签名和验证算法保证选取领导节点的随机性和可验证性，其他节点可以验证该领导节点的身份，保证领导节点的安全性。2) 通过改进共识流程中的区块接收阶段，通过验证算法和投票算法保证领导节点打包的交易将会得到其他节点的验证，验证通过后该区块的交易才会最终得到确认，保证了交易的有效性。3) 设置了基于投票的动态变更方案，增加和删除权威节点提高了系统的去中心化程度，防止权威节点作恶的情况。

2. 预备知识

2.1. 共识算法

定义 1 (共识算法)系统中有 n 个节点，其中最多有 f 个恶意节点，也就是说，最少有 $n-f$ 个节点是正常节点。节点 i 从一个输入值 v_i 开始。所有节点必须要从全部输入值中最终选择一个值(决策值)，并且满足下面的条件：

- **一致性(Agreement)**：所有正确的节点的决策值必定相同。
- **可终止性(Termination)**：所有正确的过程都在有限的时间后结束决策过程。
- **有效性(Validity)**：选择出的决策值必须是某个节点的输入值。

共识算法的基本流程[14]如下：

1) 选举出块者：出块者是指区块链中负责产生区块的节点，又称领导者。

2) 生成区块：出块者主要完成生成区块的工作，即将一段时间内网络中产生的交易数据打包放到当前区块中。一个领导者在其“任职”期间，能够生成多个区块，一般将一个领导者的任职时间称为一个时期(epoch)，每个时期由多个轮(round)组成，每一轮生成一个区块。

3) 节点验证更新区块链：出块者生成区块后，将区块在网络中广播，收到区块的节点验证区块正确性并更新本地区块链。节点可能还需要验证区块中交易合法性和出块者身份合法性等。

定义 2 (共识算法的评价标准)良好的共识算法有益于区块链技术在理论和实践中的推广[15]，不同的区块链采用不同的共识算法，在满足一致性和有效性的同时会对系统整体性能产生不同影响[16]，综合现有共识算法的特点，主要存在以下评价指标：

- **安全性**：这是共识算法最基本、最重要的属性。区块链共识算法的安全性主要指在不安全的网络环境下，诚实用户是否能达成最终一致性。并且能够抵抗一些针对共识算法的攻击，如女巫攻击、审查攻击、克隆攻击。
- **可扩展性**：随着网络节点增多，交易的处理速度和性能是否能相应增加，通常以网络吞吐量来衡量。
- **去中心化**：去中心化指系统中没有可信第三方存在，区块由参与的节点共同决定，不是集中在少数几个节点上。
- **资源消耗**：在达成共识的过程中，系统所要耗费的计算资源大小，包括 CPU、内存等，主要关注节点的通信复杂度和计算复杂度。

2.2. 可验证随机函数

可验证随机函数(Verifiable Random Functions, VRF)由 Silvio Micali 等[17]于 1999 年提出，是一种基于公私钥的密码学哈希函数，只有 VRF 私钥的持有者才能计算哈希，但是具有相应公钥的任何人都可以验证哈希的正确性。在此应用中，证明人 Prover 持有 VRF 私钥并使用 VRF 哈希在输入数据上构建基于哈希的数据结构，由于 VRF 的性质，只有 Prover 才能给出有关哈希正确性的证明，知道其 VRF 公钥的任何人都可以验证 Prover 的证明是否正确，却不能对存储在数据结构中的数据做出推断。可验证随机函数可以基于私钥对一个输入，产生一个唯一的固定长度的输出，以及一个对应的证明。其他人通过公钥、输出、证明之后就一定能验证这三者的正确性，并且也只有在知道这三者之后才能验证其正确性。

VRF 附带一个密钥生成算法，可以生成公钥 PK 和私钥 SK。Prover 通过其私钥对某个输入 α 进行哈希处理可得到随机输出 β，其公式(1)如下：

$$\beta = \text{VRF\_hash}(\text{SK}, \alpha) \tag{1}$$

Prover 还使用私钥来构造零知识证明 π，用来证明 β 是正确输出，如公式(2)所示：

$$\pi = \text{VRF\_prove}(\text{SK}, \alpha) \tag{2}$$

任何人都可直接通过 π 获得 β，如公式(3)所示：

$$\beta = \text{VRF\_prove\_to\_hash}(\pi) \tag{3}$$

从而可写成公式(4)：

$$\text{VRF\_hash}(\text{SK}, \alpha) = \text{VRF\_prove\_to\_hash}(\text{VRF\_prove}(\text{SK}, \alpha)) \tag{4}$$

零知识证明 π 允许 Verifier 持有公钥 PK 以验证 β 是否基于 PK 下的 α 的正确输出 β。因此，VRF 还附带了算法，首先计算公式(3)，若计算出的 β 等于 PK 下的正确 β，输出 VALID，否则输出 INVALID：

$$\text{VerifyVRF}(\text{PK}, \alpha, \pi) \tag{5}$$

VRF 具有以下性质：

1) 唯一性：对于任何固定的 VRF 的公钥 PK 和任何的输入 α，都有一个唯一的 VRF 输出 β，并且β是可以被验证是否有效的。

2) 抗碰撞性：给定一个消息 α_1，很难找到另一个消息 α_2，使得 $\text{Hash}(\alpha_1) = \text{Hash}(\alpha_2)$，即破坏者知道私钥，要找到具有相同输出的两个不同的输入 α_1 和 α_2，在计算上是不可能的。

3) 伪随机性：伪随机性确保当攻击者在没有相应的 VRF 证明 π 的情况下看到 VRF 哈希输出 β 时，β 看起来就是随机值，不可能找出某种确定的规律。

2.3. 权威证明共识算法

权威证明共识算法依赖于 N 个经过身份认证过的可信节点，也称权威节点(Authorities)，然后在每一轮中选举出一个或多个权威节点担任该次共识过程中的领导者(Leader)，负责提出新的区块，当此次提议被至少 $N/2+1$ 个验证节点(Validators)确认后，该打包区块被最终确定下来，从而实现整个系统的分布式共识，此算法几乎不需要任何算力去竞争，因此可以通过减少每个区块的间隔时间和处理更多的交易。PoA 算法中的共识采用轮换模式，通过取余法在权威节点之间公平地选取领导节点。

PoA 取余法的 Leader 选取方法如算法 1：总共有 N 个节点，第一个区块生成的时间为 first，当前时间 time，区块生成间隔的秒数 step_duration，每一轮的索引 $s=(\text{time}-\text{first})/\text{step\_duration}$，在第 i 次权威选取过程中，第 A_i 被选举为领导节点，即 $A_i = s \bmod N$，A_i 负责区块提议和打包的任务，其他节点成为验证节点则负责投票验证该领导节点的区块提议。

Algorithm 1. The Leader node selection of PoA
算法 1. PoA 领导节点选取

Algorithm 1 The Leader node selection of PoA

Parameters:
N: *Number of nodes at the current block generation interval*
time: *Current time(UNIX)*
first: *First block generation time(UNIX)*
step_duration: *Number of seconds in the block generation interval*
Leader Selection:

$$A_i = ((time - first)/step\_duration)\%N$$

3. 改进方案

本文基于联盟链的应用场景，对原有的 PoA 算法进行部分改进，引入基于可验证随机函数的领导节点选取和基于投票的权威节点动态变更的 VPoA 算法。VPoA 算法通过加密抽签算法来选取领导节点，验证抽签算法可以验证该领导节点的身份，改进区块接收阶段，通过验证的区块才会最终得到确认。VPoA 算法的核心流程图如图 1：

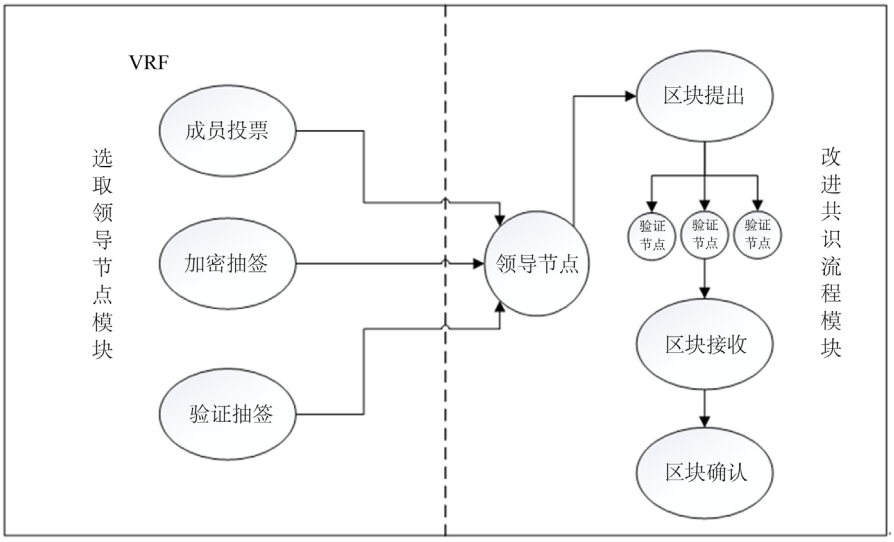

Figure 1. VPoA 核心流程图
图 1. The main structure of VPoA

3.1. 选取领导节点的改进

引言中针对 PoA 的分析得出，PoA 共识流程使用取余法从权威节点集合(ValidatorSet)中选取一个领导节点，限制消息传输的次数，但因此权威节点的信息被在多轮区块链共识之后，权威节点信息容易被公开，导致权威节点被攻击后的共识的效率和安全问题。针对这一存在问题，设计一个基于可验证随机函数的领导节点抽签选取算法，每次领导节点的选取和验证区块都不会被公开，保证了领导节点的不可预测性和随机性，同时可以通过验证抽签算法验证领导节点的身份。

3.1.1. 领导节点的抽签算法

首先，在加密抽签算法中，权威节点通过输入本节点的私钥以及所有节点都能得到的参数，经过计算可以得到本轮 VRF 随机哈希值与中签结果。如算法 2：

Algorithm 2. The cryptographic sortition algorithm
算法 2. 加密抽签算法

Algorithm 2 The cryptographic sortition algorithm

Input:
sk：用户私钥
α：随机种子
$type$：成员类型
$round$：共识阶段
τ：成员中签数量
ω：用户权重
W：总权重

Leader Selection:
$function\ VrfSortition(sk,\ \alpha,\ round,\ type,\ t,\ w,\ W)$:
$<hash,\pi> \leftarrow VRF_{sk}(\alpha||type||round)$
$p \leftarrow \frac{\tau}{W}$
$j \leftarrow 0$
$while\ \frac{\pi}{2^{\beta len}} \notin (\Sigma_{k=0}^{j} B(k;w,p),\ \Sigma_{k=0}^{j+1} B(k;w,p))\ do$
$j++$
$end\ while$
$return\ (hash,\ \pi,\ j)$
$end\ function$

Output:
VRF 哈希值：$hash$
零知识证明：π
抽签结果：j

3.1.2. 验证抽签算法

验证抽签算法是其他节点对中签节点进行验证的算法，某节点收到中签节点的消息后，中签节点的 pk 与 π 是公开的。验证过程先验证 π 是否合法有效，然后依据与抽签类似的过程获得用户被选中的子用户数，如果结果是 0 表示没被抽中，从而与用户随消息广播出来的 j 值做比较以验证其正确性。如算法 3。

3.2. VPoA 共识流程的改进

3.2.1. 状态定义

NEW ROUND：上一轮的领导节点发送新的区块提案，验证节点等待 LEADER SELECTION 消息。

LEADER SELECTION：验证节点已收到 LEADER SELECTION 消息，计算上一轮的哈希从而验证自己的 VRF 签名是否成为新的领导节点。

Algorithm 3. The verifying sortition algorithm
算法 3. 验证抽签算法

Algorithm 3 The verifying sortition algorithm

Input:

pk：用户公钥
π：零知识证明
α：随机种子
$type$：成员类型
$round$：共识阶段
τ：成员中签数量
ω：用户权重
W：总权重

Leader Validation:

$function\ VrfVerifySortition(pk,\ \alpha,\ round,\ type,\ \pi,\ t,\ w,\ W)$:
  $if\ \neg\ VerifyVRF_{pk}(hash,\ \pi,\ \alpha||type||round)$
    $return\ 0$
  $else$
    $p \leftarrow \frac{\tau}{W}$
    $j \leftarrow 0$
  $while\ \frac{\pi}{2^{\beta len}} \notin (\Sigma_{k=0}^{j} B(k;w,p),\ \Sigma_{k=0}^{j+1} B(k;w,p))\ do$
    $j++$
  $end\ while$
  $return\ j$
$end\ function$

Output:

被验证的抽签结果：j

BLOCK PROPOSAL：抽签成功的领导节点提议新的区块，并向其他验证节点广播。

BLOCK ACCEPTANCE：验证节点确认已收到 BLOCK PROPOSAL 消息，向其他验证节点广播。

FINAL COMMITTED：领导节点确认已收到 BLOCK ACCEPTANCE 消息，成功地将新块插入到区块链中，准备进行下一轮。

ROUND CHANGE：新的一轮开启，重新选取新的领导节点。

3.2.2. VPoA 算法共识流程

首先，在初始状态下，所有的权威节点都是待选举状态，当前的 LEADER 打包区块结束后，所有的节点在收到 LEADER SELECTION 消息时，开始新一轮的领导节点选取。新一轮的时间间隔为当前时间戳和第一次区块生成时间的时间差，所有待选举的节点开始加密抽签，当某权威节点被选取给新的领导者 LEADER 时，其他权威节点称为验证节点 VALIDATORS，参与验证该轮抽签是否正确。当所有的验证节点成员投票结束后，新节点开始 BLOCK PROPOSAL 过程。VPoA 算法共识流程如图 2，假设一次共识中存在 4 个权威节点，序号分别为 1，2，3，4。则 VPoA 算法共识流程的四个步骤如下：

1) 领导节点选取：通过公式 1 和公式 2 分别计算随机哈希值 *hash* 与零知识证明π，其中哈希值由私钥和输入参数决定，他人不可伪造。节点 1 对当前区块高度和上一区块中的 VRF 值做签名，然后计算哈希，根据哈希值独立判断自身能否成为当前高度区块的潜在 Leader。通过抽签算法节点 1 得知自己成为新的领导节点后，其他节点 2、3、4 将使用验证抽签算法进行结果验证，同意节点 1 为领导节点后，将信息传回节点 1。

2) 提出新的区块：节点 1 成为当前区块高度的 Leader，则将包还未得到共识的新交易请求，构造新

块，并对此区块数据和区块哈希进行签名，将区块和签名向共识网络中的所有权威节点进行广播。

3) 区块接收：节点 2，3，4 成为该轮的验证节点 Validators。验证节点会检查区块头的有效性，并对该区块进行对应签名，然后将签名信息发送回当前 Leader 节点 1。

4) 区块确认：领导者 Leader 节点 1 至少要等待 $N/2 + 1$ 个 Validators 的有效的签名(可以与步骤 3 的签名者不同)，然后提交该区块，然后将新的块广播给整个区块链网络。

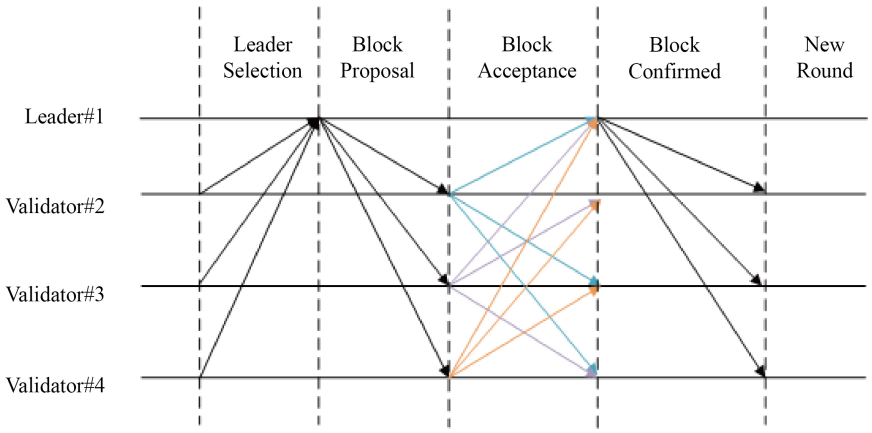

Figure 2. The improved consensus process
图 2. 改进后的共识流程

3.3. 权威节点动态变更

目前联盟链中的主流 PoA 实现算法如 Aura 和 Clique，但是这两者的算法实现并不支持动态变更权威节点，所以本文设计了一种基于投票的权威节点动态变更方案。如果需要新增一个权威节点，设置开启一个投票机制，通过网络中权威节点集的共同投票，投票通过后才能加入权威节点集合，成为新的权威节点。如果出现权威节点作恶或失效的情况，则剔除该领导节点，选取新的领导节点完成该轮区块打包任务。增加和剔除权威节点的投票都是立即生效的，而且都需要超过当前权威节点的 50%的投票数方能生效。

3.3.1. 加入权威节点

新节点通过向全网广播，表明自己想成为权威节点。并向已有权威节点提交自己的认证信息，当原有的权威节点收到该提议后，开启一轮投票，当该节点收到超过 50%个权威节点的票数时，此时将该新节点加入到权威节点的列表中。

3.3.2. 剔除权威节点

某个区块交易的进行到确认过程中，如果权威节点在一段时间内发生作恶或无响应的情况，则可以重新投票将其从权威节点列表中剔除，通知所有的权威节点，进行新的共识确认。如果立马删除节点，则可能丢失未提交的事务。因此，在本文的动态变更方案中，选出新的领导节点将其所有事务转移到新的轮次，然后再将所有的交易数据重新提交，权威节点的剔除可以防止领导节点作恶或者被攻击后作恶。

4. 实验与分析

为了验证本文提出的 VPoA 算法的有效性，基于 Golang 语言实现了该 VPoA 算法，采用多机器多节点搭建测试区块链进行共识模拟[18]，本次实验在 1 台 Intel i7-7700 CPU、16G 内存、512 硬盘、操作系统为 CentOS 7.6 64 位的 4 个虚拟服务器上进行。

4.1. 领导节点选取实验结果

对于领导共识节点的选取的改进，我们用实验证明该选取方案是有效的，首先，我们在创世配置文件中设置我们的权威节点个数，分别为 1、2、4、8、10 个，查看在权威节点个数不同时是否能够在每一轮次中正确有效的选取领导节点。在第一轮中，只有一个权威节点 0x00⋯⋯597c2 时，领导节点选取就为该权威节点。身份验证也是正确的，此时的区块，也被称为创世区块，会被正确提交到系统中。在第二轮，会开始新的领导节点选取，抽签算法还是选取 0x00⋯⋯597c2 作为该轮的领导节点。同理，在不同的轮次中设置不同的权威节点个数，查看每轮领导节点选取的结果，发现改进的 VPoA 方案可以完成领导节点的正确选择。尤其在第 4 次试验中，人为将 0x00⋯⋯f548a 的权威节点下线，发现该次领导节点选取失败，在下一轮中重新选取 0x00⋯⋯7183d 为领导节点成功。实验结果如表 1：

Table 1. Leader node selection result experiment
表 1. 领导节点选取结果

轮次	权威节点个数	领导节点选取	验证结果
1	1	0x00⋯⋯597c2	True
2	2	0x00⋯⋯597c2	True
3	4	0x00⋯⋯77c9e	True
4	8	0x00⋯⋯f548a	False
5	8	0x00⋯⋯7183d	Ture
6	8	0xf1⋯⋯03a43	True
7	10	0x00⋯⋯b8c9e	True
8	10	0xf1⋯⋯a1A75	True
9	10	0x00⋯⋯597c2	True
10	10	0x00⋯⋯f548a	True

4.2. 领导节点出块时间

在搭建的区块链测试网络中运行一段时间后，首先设置 10 个权威节点，在 100 个区块中，以 5 个区块作为间隔，统计 Aura、Clique 和 VPoA 三种算法的出块时间，结果如图 3 所示。从图中可以看出，三者出块时间增长基本比呈线性增长，而且都在 1 s 左右出一个块，因此可以看出 VPoA 算法的出块速度是稳定的，改进的领导节点选举和区块接收阶段并不影响整个出块过程。

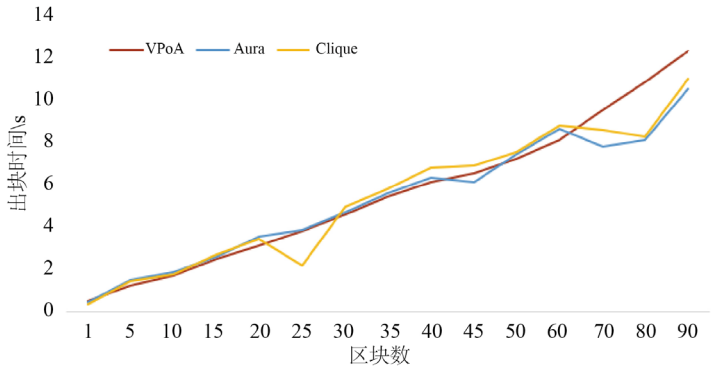

Figure 3. Block mining statistics of top 100 blocks
图 3. 前 100 个区块的出块时间统计

4.3. 安全与性能分析

本节针对改进的 VPoA 算法进行常见的区块链网络攻击的安全分析。VPoA 算法可以抵御广播许多无效节点的 DDoS 攻击，由于普通节点首先通过身份验证后才能成为权威节点。此外根据算法 2 中可知，VPoA 使用抽签算法选取的领导节点，只有当领导身份验证通过后再有区块提议和区块确认的权限，因此无效节点并不会对共识网络产生影响。最后，如果某个权威节点在一段时间内不可用或在区块验证阶段未通过验证过程，则可以将其从权威节点列表中排除，所以 VPoA 方案可以抵御 DDoS 攻击。

在原来的 PoA 算法中，通过身份认证后成为的权威节点具有决定新区块的全部权力，因此意味着说它们有可能会选择或终止某些特定交易，甚至危害网络安全性。而 VPoA 改进后的区块接收阶段，不仅仅是验证区块的有效性，更让每个权威节点都能密切监视其他权威节点的行为，因此鼓励每个领导者以诚实的方式履行其职责。而且通过权威节点动态变更的方案让每个权威节点在发生恶意行为时会得到惩罚甚至剔除，因此可以有效抵抗审查攻击。

对于 51%攻击，要求攻击者获得对一半以上的网络节点的控制权。首先 PoA 区块链中的权威节点都要通过一定的身份证明，进一步在 VPoA 中，因为可验证函数选取方案的伪随机性几乎无法预知下一轮的领导节点，所以要获得 VPoA 算法中的权威节点的控制权比获得计算能力更要困难。对于克隆攻击，VPoA 算法验证抽签算法具有唯一性和不可碰撞，用来验证领导节点的身份可以抵御领导节点身份克隆。

5. 总结与展望

区块链共识算法是区块链技术的核心[19]，PoA 共识算法与 PoW 共识相比，无需花费计算资源来解决复杂的数学哈希，不需要高性能的系统也能提供较好的性能和对故障的高容错能力。针对现有联盟链中常见共识算法 PoA 的弊端，本文设计了一种改进的可验证权威共识算法 VPoA，通过可验证函数做到选取领导节点的随机性，防止领导节点被恶意攻击；通过改进区块接收阶段来提高原有方案的有效性，防止作恶节点攻击；设置动态变更的方案，如果领导节点作恶，反而会被取消参与共识的过程，提高了整个网络的安全性。而且经过实验验证，该改进并无明显增加区块生成时间，做到按一定的时间间隔按顺序生成交易块，保证稳定的区块生成；通过对 VPoA 的安全分析，发现此方案能抵御常见的攻击方式，使得更多的区块链系统能够得以运用此种算法。

目前，VPoA 仍然存在不足，下一步工作将会在权威节点的状态变更和基于信用机制的权威节点选取等方面深入研究，不断完善该共识算法，进一步提升其健壮性和可用性。

基金项目

国家自然科学基金资助项目(91748107)；广东省基础与应用基础研究基金(No. 2020A1515010616)；广东省引进创新科研团队计划资助项目(2014ZT05G157)。

参考文献

[1] Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. http://bitcoin.org/bitcoin.pdf

[2] Du, M.X., *et al*. (2017) A Review on Consensus Algorithm of Blockchain. 2017 *IEEE International Conference on Systems*, *Man*, *and Cybernetics* (*SMC*), Banff, 5-8 October 2017, 2567-2572.

[3] Gervais, A., Karame, G.O., Karl, W., *et al*. (2016) On the Security and Performance of Proof of Work Blockchains. *ACM SIGSAC Conference on Computer & Communications Security*, Vienna, 24-28 October 2016, 3-16. https://doi.org/10.1145/2976749.2978341

[4] Wood, G. (2014) Ethereum: A Secure Decentralized Generalised Transaction Ledger. Ethereum Project Yellow Paper Vol. 151, 1-32.

[5] Saleh, F. (2018) Blockchain without Waste: Proof-of-Stake. Social Science Electronic Publishing.

https://doi.org/10.2139/ssrn.3183935

[6] Castro, M. and Liskov, B. (1999) Practical Byzantine Fault Tolerance. *Proceedings of the* 3*rd Symposium on Operating Systems Design and Implementation*, Vol. 99, 173-186.

[7] Gavin, W. (2015) PoA Private Chains. Github.

[8] Parity: Fast, Light, Robust Ethereum Implementation, Parity Technologies, 2017-12-12. https://www.parity.io

[9] Go-Ethereum. https://geth.ethereum.org

[10] De Angelis, S., Aniello, L., Baldoni, R., *et al*. (2018) PBFT vs Proof-of-Authority: Applying the Cap Theorem to Permissioned Blockchain.

[11] Aura. https://github.com/paritytech/parity/wiki/Aura

[12] Clique. https://github.com/ethereum/EIPs/issues/225

[13] Ekparinya, P., Gramoli, V. and Jourjon, G. (2019) The Attack of the Clones against Proof-of-Authority. *Network and Distributed Systems Security* (*NDSS*) *Symposium* 2020, San Diego, 23-26 February 2020, 1-14. https://doi.org/10.14722/ndss.2020.24082

[14] Xian, X., Zhou, Y., Guo, Y., *et al*. (2019) Improved Consensus Mechanisms against Censorship Attacks. 2019 *IEEE International Conference on Industrial Cyber Physical Systems* (*ICPS*), Taiwan, 6-9 May 2019, 718-723. https://doi.org/10.1109/ICPHYS.2019.8780370

[15] 刘懿中, 刘建伟, 喻辉. 区块链共识机制研究: 典型方案对比[J]. 中兴通讯技术, 2018, 24(6): 6-11.

[16] 韩璇, 刘亚敏. 区块链技术中的共识机制研究[J]. 信息网络安全, 2017, 17(9): 147-152.

[17] Gilad, Y., Hemo, R., Micali, S., *et al*. (2017) Algorand: Scaling Byzantine Agreements for Cryptocurrencies. *Proceedings of the* 26*th Symposium on Operating Systems Principles*, Shanghai, 28-31 October 2017, 51-68. https://doi.org/10.1145/3132747.3132757

[18] Deploy Ethereum Proof-of-Authority Consortium Solution Template on Azure. https://docs.microsoft.com/en-us/azure/blockchain/templates/ethereum-poa-deployment

[19] 袁勇, 王飞跃. 区块链技术发展现状与展望[J]. 自动化学报, 2016, 42(4): 481-494.