计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.201200148

基于 PBFT 的联盟链共识算法

周艺华[1,2,3]　方嘉博[1,2,3]　贾玉欣[1,2,3]　贾立圆[1,2,3]　侍伟敏[1,2]
1 北京工业大学信息学部　北京 100124
2 可信计算北京市重点实验室　北京 100124
3 北京工业大学区块链研究中心　北京 100124
(zhouyh@bjut.edu.cn)

摘　要　针对实用型拜占庭(PBFT)共识算法中存在的可拓展性较差、主节点选取随意、网络开销较大等问题,文中面向联盟链设计并提出了一种优化的实用型拜占庭共识算法。首先,为集群中的节点设置不同的角色,根据不同角色为节点分配不同的权限,不同权限的节点设计了动态进出网络机制。其次,在生产节点选举时,设计了投票机制与基于信誉度的 FTS 树相结合的选举算法,保证了选举的安全性和公平性。最后,在共识流程方面优化了 PBFT 共识流程,缩减了 PBFT 共识中的网络开销。实验结果表明,提出的 POC 共识算法相较于 PBFT 算法,具有高动态、选举安全、低开销等特性。

关键词:区块链;共识算法;拜占庭容错;信誉度;follow-the-satoshi 算法
中图法分类号　TP391

Consortium Blockchain Consensus Algorithm Based on PBFT

ZHOU Yi-hua[1,2,3],FANG Jia-bo[1,2,3],JIA Yu-xin[1,2,3],JIA Li-yuan[1,2,3] and SHI Wei-min[1,2]
1 Faculty of Information Technology,Beijing University of Technology,Beijing 100124,China
2 Beijing Key Laboratory of Trusted Computing,Beijing 100124,China
3 Blockchain Research Center of Beijing University of Technology,Beijing 100124,China

Abstract　Focusing on the problems of poor scalability,the random selection of primary nodes,and high network overhead in the practical-Byzantine-fault-tolerant consensus algorithm,this paper proposes a Byzantine-fault-tolerant consensus algorithm based on credit evaluation mechanism. First of all,the system sets different roles for the nodes in the cluster and assigns different permissions to the nodes according to different roles,so that the nodes with different permissions can enter and exit the network dynamically. Secondly,the voting mechanism and follow-the-satoshi algorithm based on credibility are designed to ensure the security and fairness of the election. Finally,in the aspect of the consensus process,an optimized two-stage Byzantine-fault-tolerant consensus is proposed to reduce the network overhead in the PBFT consensus process. Experiments show that the consensus algorithm based on the credit evaluation mechanism proposed in this paper has the characteristics of high dynamic,election security,and low cost compared with the PBFT algorithm which is suitable for the alliance chain.

Keywords　Blockchain,Consensus algorithm,BFT,Credit,follow-the-satoshi algorithm

2008 年中本聪[1]首次提出比特币系统,比特币底层的区块链相关技术引起了产业界和学术界的广泛关注。共识机制作为区块链系统的核心组件,为系统去中心化提供了保障,共识算法的优劣直接影响着整体区块链系统的性能效率、安全性以及可拓展性[2]。

1　相关研究

共识机制用于保证在分布式环境下节点对系统状态达成一致性确认,由于各领域对区块链技术不同的需求场景,学者们提出了众多不同的共识算法。

比特币系统采用了基于计算能力竞争(Proofs of Work,POW 工作量证明[3])的全网共识算法,同时可以在一定概率下抵抗双重支付攻击。之后,因为 POW 算法吞吐量低、资源消耗过大等问题,出现了权益证明(Proof-of-Stake,POS)协议[4-5]、委托权益证明(Delegated Proof of Stake,DPOS)[6]等协议。POS 和 DPOS 共识算法减少了算力消耗,提高了交易吞吐量,但是存在资源集中的风险,无法保证一定的公平性。

Pease 等通过对拜占庭将军问题[6]进行描述和分析,提出了著名的拜占庭容错(Byzantine Fault Tolerance,BFT)算法[7]。随后,针对 BFT 算法中网络开销过大、资源占用高等

到稿日期:2020-12-17　返修日期:2021-04-16　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61572053);北京市自然科学基金(4182006)
This work was supported by the National Natural Science Foundation of China(61572053) and Beijing Natural Science Foundation(4182006).
通信作者:方嘉博(844545494@qq.com)

问题，Castro 等[8]首次提出 PBFT(Practical Byzantine Fault Tolerance)算法，该算法依然满足可容忍不超过 1/3 的拜占庭节点。以 Linux 基金会下的 Hyperledger Fabric 项目[9]为代表的联盟链环境需要考虑集群中存在故障节点，还需要考虑集群中存在作恶节点，因此 Fabric 0.6 版本便采用了 PBFT 算法。针对 PBFT 算法性能较低等问题，研究者们提出了很多相关改进算法[10-14]，比较典型的有 RBFT(Redundant Byzantine Fault Tolerance)算法[10]和 CBFT(Concurrent Byzantine Fault Tolerance)算法[11]，它们虽然在通信开销、选举安全性方面取得了一定提升，但性能依然不足。

2014 年，Bentov 等提出了 POA (Proof of Activity)共识算法[15]，采用了 POW 与 follow-the-satoshi(FTS 树)相结合的生产节点选举算法，POA 中的选举算法可以选举出高性能节点且保证一定的公平性。

在改进共识算法方面，一种思路是通过投票表决的方式选举出节点[16]。Yu 等[17]提出了一种服务证明机制，通过对节点的存款率、错误率、活跃性等多种因子进行区域内投票选出主节点进行共识；文献[18]提出的投票选举算法，使选票最高的节点获得出块权，其实验表明该算法可有效提高系统性能。节点投票方式可以提高共识效率，但是有可能投票选出拜占庭节点，如果连续投出恶意节点将会影响系统性能。共识改进的另一种思路是给予节点“信誉权重”来评判节点行为，通过信誉度大小选举出节点，信誉度机制可有效防止贿赂攻击等情况。Huang 等[19]提出了一种动态的信誉度证明机制(Proof of Trust，POT)，其实验表明 POT 相较于 POS 共识机制可有效抵抗贿赂攻击、权益累积攻击等，提高了共识效率。Feng 等[20]提出了一种协商证明机制(proof-of-negotiation)，该机制通过信誉度评价矿工可信度，其实验表明该共识算法相较于传统共识算法出块速度更快。

目前区块链中的主流共识算法主要有基于 POW 的算力证明机制、基于 POS 的权益证明机制、基于 DAG 的图型共识机制、基于 BFT 的拜占庭共识机制。在联盟链中需要考虑存在拜占庭节点的情况且节点数量可控，拜占庭容错(BFT)类共识算法比较适用于联盟链。在 PBFT 算法中，区块提案节点一直为主节点，如果主节点长期不更新，则其他节点无法得到机会当选主节点获取系统激励，具有一定的中心化风险，不利于联盟系统长期运营，而且随着节点数量增多，PBFT 通信开销巨大。本文针对 PBFT 中主节点选取随意的问题提出了 POC(Proof of Crefit)协议。网络中的节点具有信誉权重属性，根据节点所具有的信誉权重，将 FTS 树和投票相结合依概率选择生产节点。本文的选举算法相较于 PBFT 主节点选取具有更高的可靠性，并且优化了共识流程，减少了通信开销，降低了算法的时间复杂度。

2 共识算法

2.1 系统架构模型

本文共识算法面向联盟链设计，算法整体的设计思想是首先将联盟网络中的节点进行角色分层，不同类型的节点拥有不同的权限，区块生产权和投票监督权分离，各种节点具有不同的权限，节点可以动态加入和退出网络，使系统具有高动态性。在生产节点选举算法中，提出了基于信誉权重的投票机制和 follow-the-satoshi 相结合的选举算法，使生产节点选取区块生产节点具有较高的可靠性、可控安全性及收敛可靠性。

所有网络节点需要在 CA 服务器获取数字证书，进行网络节点注册。节点注册成功之后如果要成为候选节点，需要缴纳一定的保证金并进行基于 POW 的身份证明。每个候选节点维护有全网节点的信誉度列表(根据信誉度排序)，候选节点在每一轮进行投票，选择本轮备选生产节点，再由生产节点选举算法在生产节点备选集合中选举出生产节点。生产节点发起提案，其他备选生产节点参与协商共识，区块数据共识达成后，全网节点进行本地区块链数据更新。算法模块关系如图 1 所示。

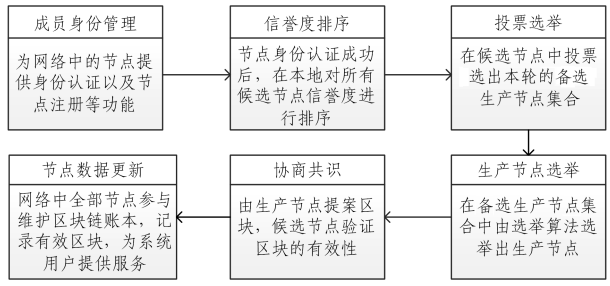

图 1 算法模块关系图

Fig. 1 Module relationship

2.2 节点角色设计

系统为网络参与者建立不同的安全身份。在共识协商过程中存在 3 个角色：普通节点、候选节点和生产节点。

普通节点(Ordinary User)：联盟委员会成员共同维护一个区块链系统，联盟中的普通成员节点需要在系统中注册用户帐户，负责对接收到的每一个区块进行验证并记录，监听并转发网络中的交易信息。新进入网络集群的普通节点不可参与到当前的共识过程中。

候选节点(Candidater)：系统中的候选节点由普通节点转换而来。为防止恶意节点在网络中大量存在并转为候选节点，首先普通节点要成为候选节点需要提交一定的保证金，此外，普通节点在完成网络注册转为候选节点时，需要基于 POW 算法提供算力证明，每个节点需要在本地计算出一个可验证的 *nonce* 值。为防止节点提前计算好满足条件的 *nonce* 值，计算时添加系统参数 $\xi_{Randomseed}$，其满足如下条件：

$$POW(IP||VK||nonce||TimeStamp||\xi_{Randomseed}) \leqslant Difficulty_{cur} \tag{1}$$

IP 和 *VK* 为节点的 IP 地址和公钥，*nonce* 为计算出满足条件的随机数，*TimeStamp* 为当前时间戳，$\xi_{Randomseed}$ 为上一轮中产生的随机数(详见 2.3 节)，$Difficulty_{cur}$ 为当前系统挖矿难度值。每轮共识过程结束后，重新在系统中选择新的候选节点参与到下一轮的共识中。候选节点对生产节点进行监督，并负责对每一个区块进行验证与记录。

生产节点(Producer)：生产节点专门负责生产区块。生产节点从每一轮的候选节点中依概率选出，参与当前共识轮的候选节点相当于备选生产节点，节点从网络中不断收集交

易信息，发现自己被选为生产节点后将交易打包进区块，全网广播进行验证。成为生产节点需要两个步骤：1）成为当前共识轮的候选节点；2）在当前共识轮的选举中成功当选生产节点。

CA 节点（Certificate Authority）：网络中 CA 服务器主要负责数字证书的颁发、管理以及归档和吊销。本文面向联盟链设计共识算法，其节点注册参考 Fabric 区块链中的网络节点注册方式，普通节点在 CA 证书管理服务器获取数字证书，在加入网络之前先进行身份识别并获得访问许可。

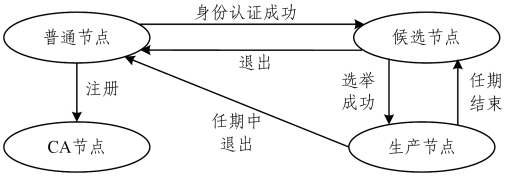

图 2 角色转换图

Fig. 2 Role transition

在联盟链网络中的每个节点的共识模块可以视为一个状态机，状态机在不同状态之间跳转。状态机基于不同类型的 RPC（Remote Procedure Call）进行状态转换。对于共识算法，可以将其视为一种状态机副本复制算法，在网络中不同的节点状态机之间复制副本进行数据认证。

在联盟网络中节点集合为 $\mathbb{N}$，候选节点集合为 $\mathbb{N}^c$，普通节点集合为 $\mathbb{N}^o$，每一个共识轮的候选节点集合 $\mathbb{N}^T$ 为 $\mathbb{N}^c$ 的子集，即满足 $\mathbb{N}=\mathbb{N}^c \cup \mathbb{N}^o$ 且 $\mathbb{N}^T \subseteq \mathbb{N}^c$。网络中节点总数量为 N，候选节点数量为 Nc，普通节点数量为 No，数量关系满足 $N=Nc+No$。每一轮的生产节点 Np 在集合 $\mathbb{N}^T$ 中产生，即满足 $\forall Np_i, Np_i \in \mathbb{N}^T$，集群中每个节点在某一时刻具有唯一状态。

定义 1 信誉度为某节点所具有的可信度数值度量，由节点行为决定数值大小，节点行为越符合系统规范，数值越大（单位：度），可信度越高，反之则越低，记全网候选节点的信誉度列表为 $C=\{Cr_1, Cr_2, \cdots, Cr_r\}$。

定义 2 每一轮备选生产节点集合由候选节点集合产生，记备选生产节点集合为 $\mathbb{C}_k=\{C_1 C_2 \cdots C_k\}$，记候选节点集合为 $\mathbb{C}_r=\{C_1 C_2 \cdots C_r\}$。其数量关系满足 $\mathbb{C}_k \subseteq \mathbb{C}_r$。

定义 3 广播消息类型分为提案、确认、投票、评价，分别表示为 Proposal，Verify，Vote，Evaluate。

2.3 基于 PBFT 的共识算法

2.3.1 PBFT 算法分析

PBFT 算法将参与共识的节点称为 Replica，一个 Replica（主节点）广播客户端请求，其他多个 Replica（副本节点）进行验证。PBFT 算法共识流程分为 5 个基本阶段。

请求（Request）阶段：客户端向主节点发送请求，附带签名等相关信息。

预准备（Pre-prepare）阶段：主节点验证客户端请求有效后，向其他副本节点广播 Pre-prepare 消息，附带签名、时间戳等信息。

准备（Prepare）阶段：节点验证 Pre-prepare 消息的有效性，如果通过则广播 Prepare 消息给其余 Replica 节点，节点接收到 $2f+1$ 个（f 为系统中拜占庭节点数量）Prepare 确认消息，则广播 Commit 消息，附带节点签名、视图编号等相关信息。

提交（Commit）阶段：节点接收到 $2f+1$ 个有效的 Commit 确认消息后，则向客户端发送 Reply 消息，附带签名、视图编号等相关信息。

回复（Reply）阶段：客户端接收到 $f+1$ 个 Reply 消息后，则认为系统对消息达成一致性共识。

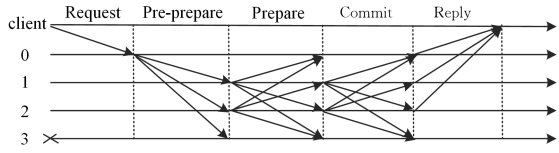

图 3 PBFT 共识流程

Fig. 3 PBFT consensus process

在 PBFT 中，主节点按照下式进行选取：

$$p=v \bmod |R| \tag{2}$$

其中，p 为主节点的编号，v 为当前视图编号（从 0 开始按照顺序递增），R 为 Replica 总数量。主节点选举是节点依次按节点序号当选主节点，此过程可靠性不高，如果存在某些恶意主节点不进行正常的消息广播，或广播错误虚假消息，频繁地触发视图变更将会严重影响系统效率。另外，PBFT 共识协议是依靠节点之间相互传递消息对提案达成确定性共识结果，需要多次进行全网广播，一旦节点数量较多，网络通信和传输开销将很大。

2.3.2 投票机制

为提升 PBFT 的主节点选举可靠性，本文提出了投票机制与基于信誉度的 FTS 树相结合的区块生产节点选举算法。其总体思想是首先通过投票选择当前共识轮的候选节点，为保障系统安全性，根据每个节点的信誉度，由高到低依次选择至少超过一半的候选节点参与本轮共识，进入本轮共识的候选节点作为当前轮的备选生产节点。每个区块的生产节点在备选生产节点集合中产生，如果一个生产节点在共识过程中生产的区块不符合系统规则，区块无法通过其他节点的验证，则该提名节点的信誉度将下降，使其在选举中获得选票的概率降低，进而在选举中失利，最终失去生产区块的机会。可以证明，企图制造非法区块的生产节点很难在选举中获胜或获得累计信誉度，可靠的生产节点更有可能在选举中获胜，而且系统也将变得更加可靠。

本文通过投票在候选节点集合中选择出部分候选节点作为备选生产节点参与到本轮共识中，再通过 follow-the-satoshi 算法（详见 2.3.3 节）根据每个节点信誉度大小依概率随机确定生产节点。对于每一个共识轮，若网络中候选节点数量为 r，选择 k 个候选节点作为一轮的备选生产节点，则每个候选节点可以投出 k 票选择不同的候选节点，k 值范围如下：

$$\lceil r/2 \rceil+1 \leqslant k \leqslant r \tag{3}$$

节点投票时依次优先将 $\lceil r/2 \rceil+1$ 票投给网络中信誉度最高的节点 $N_1, N_2, \cdots, N_{\lceil r/2 \rceil+1}$，将剩余的 $k-(\lceil r/2 \rceil+1)$ 票随机投给后面 $N_{\lceil r/2 \rceil+2}, \cdots, N_r$ 个节点。因此一轮中保证至少有 $\lceil r/2 \rceil+1$ 个高信誉度候选节点进入当前共识轮。k 值窗口

的大小限定投票数量，从而限制进入本轮候选人名单的节点数量。k 表示为：

$$k=\left[\frac{r}{2}\right]+1+\delta\cdot\left(\left[\frac{r}{2}\right]-1\right),\delta\in[0,1] \tag{4}$$

根据第 3 节实验结果分析，推荐 k 取值如下（$\delta=0.3$ 时选举具有高安全性，$\delta=0.8$ 时选举具有一定的平衡性，$\delta=1$ 时选举具有高公平性）：

$$k=\begin{cases}\left[\frac{r}{2}\right]+1+\left[\frac{1}{3}\cdot\left[\frac{r}{2}-1\right]\right], & \delta=0.3\\ \left[\frac{r}{2}\right]+1+\left[\frac{2}{5}r-\frac{4}{5}\right], & \delta=0.8\\ r, & \delta=1\end{cases} \tag{5}$$

网络中的候选节点本地维护有全网候选节点列表，其中记录有候选节点的信誉度，每次投票时根据列表选择节点进行投票。假设候选节点数量为 r，每次发送投票消息之前需要根据节点信誉度进行排序，由于节点信誉度每轮不断更新，数据乱序情况较多，本文采用快速排序法进行节点排序，之后从中选择 k 个当前轮的生产备选节点，则投票算法的平均时间复杂度为 $O(r\log_2 r)$，候选节点将选择的节点列表进行消息封装，向全网其他 $r-1$ 个节点广播投票消息，则通信复杂度为 $O(r^2)$，节点在接收到其他所有候选节点的投票信息后则认为完成投票。

k 值越小，高信誉权重节点数量在本轮候选节点集合中的占比越高，系统的安全性就越高，但时间戳信息靠后的候选节点无法得到机会当选生产者从而提升自身信誉度，系统公平性较差。为平衡安全性和公平性，实际工程中可以根据不同的安全需求选择不同的 k 值。

2.3.3 基于信誉度与 FTS 树的选举算法

本文 POC 协议在每一个共识轮（Term）产生若干区块，每一个区块由唯一的生产节点产生。生产节点在备选生产节点集合中产生，每个节点具有一定的信誉权重，节点相关行为会直接影响节点信誉度，信誉度一旦低于初始值，节点将被禁止当选生产节点，直到信誉度恢复。

信誉度初始值在系统第一轮投票结束时，根据变量 X_3 的取值完成初始化。

行为 X_1：节点是否在 T_{cut} 时间内成功出块。若节点信誉度影响因子为 0.5，则 X_1 取值如下：

$$X_1=\begin{cases}0, & \text{未当选生产节点}\\ 1, & \text{当选且正确打包出块}\\ -1, & \text{当选但未成功出块(信誉度消耗)}\end{cases} \tag{6}$$

行为 X_2：共识阶段节点是否正确发送验证消息，系统容忍时间内正确签名验证消息取值为 1，否则为 0，影响因子为 0.25。

行为 X_3：本轮节点是否正常参与投票，正常为 1，否则为 0，影响因子为 0.25。本轮结束时节点的信誉值 Y 的度量公式如下：

$$Y=0.5\cdot X_1+0.25\cdot X_2+0.25\cdot X_3 \tag{7}$$

初始值：在系统第一轮投票结束时，$X_1=X_2=0$，根据投票结果确定变量 X_3 的取值则节点信誉值 Y 初始化完成。

更新：每一轮结束，根据节点本轮行为表现得到 Y 值，所有参与本轮共识的节点信誉度更新，数据记录在评价区块中（评价区块的具体结构见 2.3.4 节表 1、表 2）。随着系统运行，节点信誉度不断变化，高信誉权重节点有更大概率当选生产节点，但本文选举算法可以使后加入系统的正常节点依然有概率当选生产节点以提升信誉度，可以在一定程度上实现去中心化。

对于每一个共识轮，每个区块的生产节点由选举函数 F_{ls} 来决定。

$$F_{ls}(R,\xi_{Randomseed})\rightarrow C_i\in\{C_1C_2\cdots C_k\} \tag{8}$$

其中，$R=\{(VK_1,Cr_1),\cdots,(VK_n,Cr_n)\}$，$VK$ 为候选节点对应的公钥地址，Cr_i 表示该节点所累积的信誉度，每个节点均可查询到全网候选节点的信誉度列表 $C=\{Cr_1,Cr_2,\cdots,Cr_r\}$，$\xi_{Randomseed}$ 为根据上一个区块所产生的随机数，C_i 节点为选举出的生产节点。在每一次选择生产节点时，函数 F_{ls} 需要输入参数 $\xi_{Randomseed}$ 来产生下一个生产节点，随机数根据上一个区块信息产生，随机数生成公式如式(10)所示。本轮共识中存在 r 个候选节点，某生产节点成功出块并有 x 个候选节点对该区块的签名，表示为：

$$\text{sign}[i]\left(0\leqslant i\leqslant x,\frac{r}{2}\leqslant x\leqslant r-1\right) \tag{9}$$

获取 $\xi_{Randomseed}$：

$$\xi_{Randomseed}=F_{getseed}\left(\sum_{i=0}^{i=r}\text{sign}[i],Term,Timestamp\right) \tag{10}$$

其中，$Term$ 为当前的共识轮次，$Timestramp$ 是区块产生时的时间戳。$F_{getseed}$ 函数通过连接 k 个签名信息、$Term$ 序号和时间戳，经过安全的 $hash$ 函数得到散列值，可以通过字符串截取的方法截取散列值后 32 位转为整型，将其映射到一个正整数上。选择出 k 个备选生产节点后，我们根据节点具有的信誉度构建 follow-the-satoshi（FTS 树），其本质依然为一种 Merkle 树，FTS 树叶子节点的左子树和右子树分别为信誉度标记。FTS 树选择叶子节点的具体步骤如下：

(1)将系统中产生的信誉度的数量换算为以聪为单位的量。假设目前已经产生了 n 份信誉度，将其换算成 n 个聪单位，按照被生产出的顺序进行排列。

(2)给定一个伪随机数生成器生成随机数，本文采用函数 $F_{getseed}$ 产生随机数 $\xi_{Randomseed}$：$\xi_{Randomseed}\sim U(1,n)$，概率密度函数为：

$$f(x)=\begin{cases}0, & x<1\text{ 或 }x>n\\ \frac{1}{n-1}, & 1\leqslant x\leqslant n\end{cases} \tag{11}$$

(3)生成随机数 $\xi_{Randomseed}$ 后，如果该随机数小于左子树的权重则选择左子树继续遍历，否则选择右子树遍历。$\xi_{Randomseed}$ 可以随机通过遍历树到达一个子节点，每个节点在树中具有一定的信誉度权重。

(4)如果没有到达 FTS 树的某叶子节点，则跳转至步骤(2)，直到选择某一个叶子节点为止，也即选择了该叶子节点所代表的权益（信誉度）所有者作为下一个出块者。

算法 1 生产节点选举算法

Input：List〈stakeholder〉候选节点列表；$F_{getseed}$ 随机种子

Output：stakeholder$_i$ 本轮生产节点

1. procedure GenerateProducer

2. r←$F_{getseed}$

3. i←0
4. tree←null
5. //根据节点信誉权重构建 FTS 树
6. tree←CreateFTSTree(List<stakeholder>)
7. While !tree[i]. isLeaf() do
8. x_1←tree[i]. getLeftNode(). getCoins()
9. x_2←tree[i]. getLeftNode(). getCoins()
10. r←$F_{getseed}$
11. /* 如果该随机数小于左子树的权重则选择左子树继续遍历，否则选择右子树遍历 */
12. if r≤x_1 then
13. i*=2
14. else
15. i=i*2+1
16. end while
17. $stakeholder_i$←tree[i]
18. end procedure

每个候选节点都具有一定的信誉度，即其所占的权重，非叶子节点的权重为左右子树的权重之和，叶子节点的权重即为某个信誉所有者的信誉度。然后节点根据随机数在左右子树中进行选择。图 4 所示为一个 4 节点的 FTS 树模型。

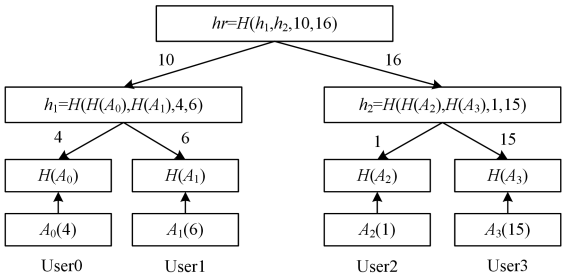

图 4 FTS 树模型

Fig. 4 FTS tree model

树中的每个叶子节点都拥有信誉权重和唯一的私钥。对于每一个成员，被选中的概率为：

$$pr=\frac{Cr_i}{\sum_{j=1}^{j=k} Cr_j} \tag{12}$$

其中，Cr_i 为某叶子节点信誉度大小，$Cr_i \in \{Cr_1, Cr_2, \cdots, Cr_k\}$，且 $H(A_0)$，$H(A_1)$分别为叶子节点 A_0，A_1 的 $hash$ 值。

$$h_1=Hash(H(A_0),H(A_1),4,6) \tag{13}$$

其中，h_1 为节点 A_0 的 $hash$ 值、A_1 的 $hash$ 值、A_0 的信誉度 4、A_1 的信誉度 6 构造的默克尔证明。右子树同理构造。

2.3.4 共识流程

本文 POC 共识协议优化了共识流程，减少了共识过程中的网络通信开销。

有效截至时间(T_{cut})：系统规定生产节点在当选时开始，到区块打包完成时刻的时间间隔为有效截至时间。

共识轮(Term)：系统规定从投票选择出 k 个候选节点开始，到最后一个生产节点产出评价区块为止为一个共识轮。

在每一轮共识开始之前，集群中已经存在 r 个候选节点，网络中候选节点广播投票信息，同时接收来自于其他候选节点的投票消息，不断更新本地的选票列表。

系统根据投票机制选择出 k 个候选节点作为本轮备选生产节点集合$\mathbb{C}_k=\{C_1, C_2, \cdots, C_k\}$，集合$\mathbb{C}_k$作为 follow-the-satoshi 算法的输入，再由生产节点选举算法确定唯一的生产节点。如果某生产节点未能在有效截至时间内生成有效的块，则重新选择生产节点。

算法 2 区块生成算法

Input：byte[] data(消息数据)
Output：poc * datapoc(poc 消息结构体)
Procedure HandleProposalMessage
1. //由 GenerateProducer 选择出生产节点 P_i
2. P_i←GenerateProducer
3. //P_i从交易池中提取交易条目打包进块，验证消息有效性
4. if isValid(data) then
5. mapProposal[hash(data)]+=1
6. //广播提案消息〈Proposal〉
7. Broadcast(proposal，poc)
8. end if
9. end HandleProposalMessage
10. Procedure HandleVerifyMessage
11. //候选节点接收到 Proposal 消息后验证消息
12. If isValid(data) then
13. mapVerify[hash(data)]+=1
14. //若检查通过广播 verify 消息
15. Broadcast(verify，poc)
16. end if
17. end HandleVerifyMessage
18. Procedure Reply
19. 〈Reply，Term，id，h(m)，〈block〉$_\sigma$〉
20. If mapReplt[data]≥[r/2]+1 then
21. Broadcast(reply，poc)
22. end if
23. end Reply
24. Procedure isValid
25. //节点在接收到消息验证该区块合法性
26. If poc. getProducer !=this. Producer then
27. return false
28. else
29. //检查 TimeCurrent−Timestamp≤T_{cut}
30. if TimeCurrent−Timestamp>T_{cut} then
31. return false
32. else
33. //检查消息散列值并验证签名。
34. if poc. getdataSign !=getDigest(P_i. pubkey)
35. then
36. return false
37. else
38. return true
39. end if
40. end isValid

假设已经存在 r 个候选节点，根据投票选择 k 个候选节点进入本轮共识，系统规定每一轮产生 4 个区块，节点 CR_1，CR_2，CR_3，CR_4 依次获得生产出块权，其分别充当生产节点进行区块创建，图 5 给出了一轮共识过程。

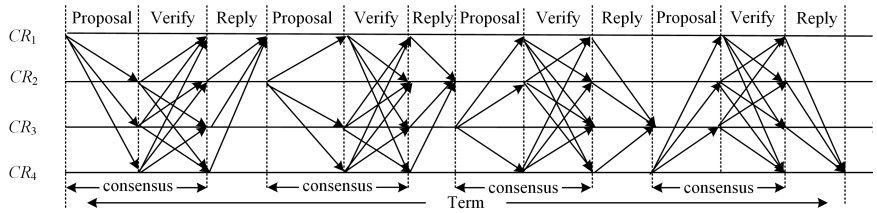

图 5 共识流程图

Fig. 5 Consensus process

一轮有效的共识过程中，每个被选中的生产节点在有效截止时间 T_{cut} 内接收到本轮中大多数的候选节点的确认后，认为该区块是一个有效确认块。在每一轮结束时会产生一个评价区块，将对本轮中每一个生产节点的行为进行评价并记录其信誉度，同时系统投票提名新的候选者节点开始下一轮的共识。

在区块链的扩展过程中，由于每次只有一个节点提名区块，因此每个有效区块都具有最终性，区块链不会产生分叉，图 6 给出了区块共识模型。

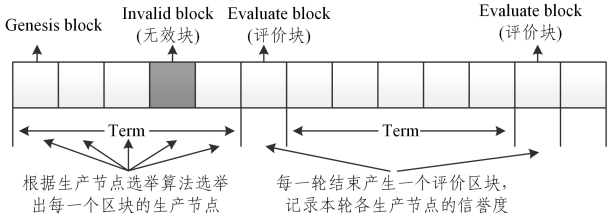

图 6 区块链模型图

Fig. 6 Blockchain model

将一轮中当前信誉度最高的节点作为评价块的生产者。一个共识轮中若候选节点 C_i 发现自己为本轮中当前信誉度最高的节点，在接收到最后一个有效普通区块后，全网广播消息$\langle Evaluate, Term, id, Credit, Timestamp, block, \langle block\rangle_\sigma\rangle$，其中 $Term$ 为当前的共识轮次，id 为节点编号，$Credit$ 为节点的信誉度，$Timestamp$ 为时间戳，$block$ 为区块信息，$\langle block\rangle_\sigma$ 为签名。网络节点在收集到 $r/2+1$（其中 r 为本轮候选人数）个确认消息后产生评价块，评价块的数据结构包含区块头和区块体两部分，其中区块头和比特币区块链类似，包含前区块 $hash$ 值、时间戳等信息。区块结构如表 1、表 2 所列。

表 1 评价块区块头结构

Table 1 Structure of evaluation block head

Version	Prev_block	Merkle	Timestamp	Block_hash
版本	前区块 hash 值	Merkle 根值	时间戳	区块 hash 值

表 2 评价块区块体结构

Table 2 Structure of evaluation block

字段	表示	描述
Producer_Addr	节点地址	记录评价区块的生产者地址
Producer_Credit	节点信誉度	记录本轮所有参与共识的节点信誉度
Satoshi_Number	聪数量	本轮结束时系统中总共产生的聪数量（信誉度总值）
transaction_list	交易列表	本轮所有参与节点更新后的信誉值作为交易列表打包进块
Sign_list	签名列表	记录本轮候选节点对评价块的签名

3 实验与分析

本文通过实现所提共识算法，以测试提出的投票模型与基于信誉度模型的生产节点选举算法的有效性、安全性以及算法性能。实验分别模拟了 1～16 个节点的网络环境，统计了 16 组投票结果，在验证生产节点选举算法过程中分别进行了 500 次随机实验来分析样本，验证了生产者选举算法的安全性。实验环境如下：操作系统为 Windows 10 家庭中文版 64 位，CPU 为 Intel(R) Core(TM) i7-4710MQ@2.50GHz，内存大小为 4 GB，jdk version 1.8，docker version 17，eclipse-2019，matlab-2017a，office 2016。

3.1 安全性分析

3.1.1 权益累计攻击

在最早的权益累计攻击中，一般为防止用户长时间累计权益进而可能对系统进行攻击，POS 权益类的共识对权益均会有所限制，如在 ppcoin 中对币龄有时间限制。本文 POC 协议的本质是 POS 与 PBFT 相结合的混合型共识协议，为防止信誉度无限制累积，在每一轮评价块生成后，如果出现某节点的信誉度超过了当前系统产生总聪数（总信誉值）的 50%，即 $producer_{Credit[i]} > satoshi_{Number} \cdot 50\%$，则系统回归初态。

3.1.2 贿赂攻击

POC 协议中恶意节点进行贿赂攻击是不可行的，因为在每一轮开始投票时，进入本轮的候选节点是不确定的，而且每次生成区块的生产节点是依概率选中，也是不确定的。贿赂攻击者需要在投票阶段提前保证生产节点 $Producer_N$ 进入一轮的候选节点集合中，之后才有概率当选出块节点。但如果被系统检测到节点出现恶意投票行为，贿赂节点会损失一定的信誉度。并且生产节点 $Producer_N$ 只对区块进行提案，如果接受贿赂提案，出现恶意提案将无法达成共识，不仅会损失保证金，而且其信誉度消耗也是极大的。累计信誉度的具体计算如下：

$$Credit_N^n = \sum_{i=1}^{i=n} Y_i \tag{14}$$

式(14)表示生产节点 $Producer_N$ 在 n 个 Term 中累计的信誉度。考虑最坏情形，在之后的 x 个区块中 $Producer_N$ 连续当选生产节点，但由于提案无法通过一致性共识，节点不断消耗信誉度，信誉度更新为：

$$Credit_N^{n+x} = \sum_{i=1}^{i=n} Y_i - 0.5x \tag{15}$$

$$\lim_{x\to+\infty} Rat_{n,n+x} = \lim_{x\to+\infty} \frac{\sum_{i=1}^{i=n} Y_i}{\sum_{i=1}^{i=n} Y_i - 0.5x} = 0 \tag{16}$$

其中，$Rat_{n,n+x}$ 是节点进行 x 个恶意提案后的更新信誉值与之

前的信誉值的比例。由于信誉度消耗极大，一旦出现节点信誉度小于初始值，即$Credit_N^{n+x}<0.25$，则该节点将被禁止发起出块提案，无法当选生产节点，只负责接收验证与投票，直到信誉度恢复初始值。

3.1.3 合谋攻击

POC 协议中由于共识基于 PBFT 算法，拜占庭节点 f 在不超过总数 1/3 的情况下，即使 f 个节点进行合谋串通，系统依然可以达成正确共识，而且一旦在共识提案中出现恶意行为，信誉度损耗巨大。节点合谋攻击只可能通过投票合谋串通，增加进入备选生产节点集合的机会，具体分析如下：

若集群中存在 r 个候选节点，分别为 $N_1,\cdots,N_{[r/2]+1},\cdots,N_r$，每个节点可以投 k 票，优先按次序投票给前 $[r/2]+1$ 个信誉度最高的节点，之后的 $k-[r/2]-1$ 票随机投给剩余的 $r-[r/2]-1$ 个候选节点。由于生产者选举算法不鼓励低信誉度的节点当选，那么一般的恶意节点不会获得较高的信誉度，其信誉度应该远低于平均信誉度。假设系统中所有候选节点的信誉度由大到小排序依次为 $\{Cr_1,\cdots,Cr_r,Cr_{f1},Cr_{f2},\cdots,Cr_{ff}\}$，其中 $Cr_{f1},\cdots,Cr_{ff}$ 为 f 个拜占庭节点所具有的信誉度，在优先给前 $[r/2]+1$ 个信誉度最高的节点投票完成后，剩余总票数为 $r\times\left(k-\left[\frac{r}{2}\right]-1\right)$，为计算方便，取 $2|r$。考虑到最坏情况，f 个拜占庭节点合谋全部进入本轮备选生产节点集合，即这 f 个节点将剩余票数全部获得，那么每个节点平均获得 t 票：

$$t=\frac{1}{f}\cdot r\cdot\left(k-\frac{r}{2}-1\right) \tag{17}$$

在随机投票过程中，一个候选节点获得一票的概率为：

$$P_1=\frac{k-[r/2]-1}{r-[r/2]-1} \tag{18}$$

那么一个候选节点获得 x 票的概率为：

$$P_2=\binom{r}{x}p_1{}^x(1-p_1)^{r-x} \tag{19}$$

那么 f 个节点同时获得 t 票的概率为：

$$P_3=\prod_{i=1}^{f}P_{2(x=t)}(i)=\binom{r}{t}^t p_1{}^{t\cdot f}(1-p_1)^{(r-t)\cdot f} \tag{20}$$

取 $r=16$，$k=[r/2]+1+f$，模拟不同的 f 取值，即 $f=[0.1r,0.2r,0.33r]$ 下 P_3 的变化，其结果如表 3 所列。

表 3 恶意投票合谋概率

Table 3 Collusion probability of malicious voting

r	$f=0.1r$	$f=0.2r$	$f=0.33r$
10	9.5×10^{-7}	9.5×10^{-7}	7.5×10^{-5}
14	1.2×10^{-11}	4.3×10^{-14}	7.9×10^{-9}
16	3.0×10^{-14}	1.6×10^{-12}	2.5×10^{-8}
20	7.4×10^{-27}	6.6×10^{-29}	5.7×10^{-24}
30	6.2×10^{-61}	5.9×10^{-67}	1.0×10^{-57}

如表 3 所列，这是考虑最坏情况下合谋进行投票攻击的结果，虽然理论上攻击可能成功，但其概率极小。在节点数目较多的情况下，如超过 16 个节点时，即使 $f=1/3r$，在实际中成功的概率基本也可以忽略，因此 POC 协议可以防止合谋投票攻击。

3.1.4 容错性

POC 共识协议是基于 PBFT 算法，在 Verify 阶段节点需要接收到 $2f+1$ 个确认消息即可达成确认，因此容错性为 $r\geqslant3f+1$（f 为拜占庭节点数量，r 是候选节点数量）。在投票阶段，需要保证 k 个候选节点进入本轮备选生产节点集合中（$k\geqslant[r/2]+1$），即保证超过一半的高信誉度节点参与到共识中。考虑最优情况，如果有一半的候选节点是良好节点，拜占庭节点数量 $f=r/2$，拜占庭节点的信誉度均小于良好节点的累积信誉度，信誉度由高到低排序，那么大多数良好节点会进入备选生产节点集合中，共识依然有效。因此在最优情况下，容错可达到 $f\leqslant r/2$。

3.2 实验结果分析

实验模拟了候选节点投票选举过程，网络集群中每个候选节点都有一定的信誉度，由于每个节点的信誉度均记录在区块链上，全网可以公开验证，可一定程度上防止篡改。生产节点选举算法使用投票机制与 follow-the-satoshi 相结合的方法，目的是保证系统安全性的同时兼顾一定的公平性，不鼓励低信誉度节点当选生产者，但是表现良好的低信誉度节点依然有机会成功当选，成功出块获得信誉度提升。实验中取 $r=16$，$k=13$，测试时对于所有节点初始信誉度，根据第一轮开始投票时投票行为 X_3 的数值完成初始化，节点信誉度初始值一般为 0.25 或 0。表 4 为选举测试结果，其统计了不同信誉度节点当选生产节点的次数，表中 8 号节点与 11 号节点随机出现投票超时、共识过程中断消息广播、生产出块超时等行为，经过多轮出块后信誉度列表更新，文中列举了其中一组测试数据，信誉度为经过 500 个区块之后各个节点所累积的信誉度。

表 4 选举测试结果

Table 4 Election test results

序号	节点编号	信誉度/聪	成为候选节点次数/次	当选生产节点次数/次
1	1	20	500	112
2	2	10	500	47
3	3	5	500	33
4	4	16	500	116
5	5	8	500	49
6	6	4	500	22
7	7	2	500	12
8	8	1	500	5
9	9	4	500	15
10	10	2	274	13
11	11	1	281	2
12	12	4	293	17
13	13	2	285	8
14	14	1	306	6
15	15	4	290	23
16	16	2	271	20

如图 9 所示，为保证共识可靠性，节点 1－节点 9 参与所有共识轮，同时为保证一定的公平性，对低信誉度节点（节点 10－节点 16）进行随机投票，确定是否进入本轮共识。从图 10 可以发现，节点 1 和节点 4 的信誉度高于平均值，当选次数分别为 112 和 116，所占获选比例分别为 22.4%和 23.2%，总共为 45.6%。一般来说，高信誉度节点当选的占比越高，系统的安全性就越高，实际工程中可以根据不同的安全需求选择不同的 k 值。

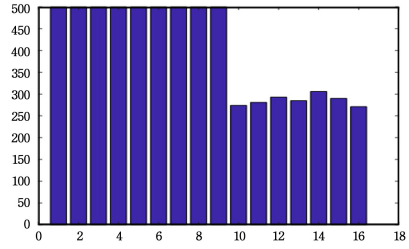

图 9 节点进入共识圈次数对比

Fig. 9 Number of times each node enters the consensus circle

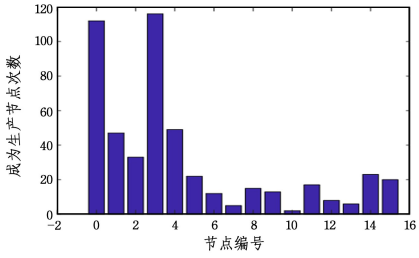

图 10 节点当选生产节点次数对比

Fig. 10 Number of production nodes selected for each node

3.3 性能分析

实验仿真对比了 PBFT 算法[8]、RBFT 算法[10]、CBFT 算法[11]以及本文提出的基于信誉度模型的 POC 协议(Proof of Credit)在网络开销、吞吐量和容错性等方面的性能。PBFT 算法的通信开销为$(2(n^2-n))$,时间复杂度为 $O(n^2)$,其中 n 为网络中全部节点的数量。POC 协议的通信开销为(n^2-1),时间复杂度为$O(r^2)$,其中 r 为本轮候选节点的数量($r\leqslant n$),相较于 PBFT 算法,本文算法在通信开销和时间复杂度方面均有所降低。

3.3.1 网络开销

在网络中,各节点共识过程中需要进行数据传输进行验证,数据传输所占用的网络带宽表示为:

$$Bandwidth=n*(n-1)*Blocksize \tag{21}$$

其中,$Bandwidth$ 为所需的网络带宽,n 为节点数目。$Blocksize$ 为数据块大小,以比特币系统为例,目前区块平均大小约为 1.1M。式(21)中,在 $Blocksize=1$Mbps 的情况下,网络带宽随着 n 的增加而增加。本文通过实验对比了不同算法所需的网络带宽,结果如图 11 所示。

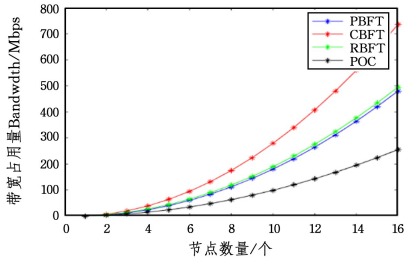

图 11 带宽占用对比图

Fig. 11 Bandwidth occupancy comparison

3.3.2 吞吐量

吞吐量定义为单位时间内系统能处理的事务总量。吞吐量的大小反映了系统处理事务的能力。在区块链系统中,一般用系统每秒成功接收的交易数量 TPS(Transaction PerSecond)来表示吞吐量:

$$TPS=\frac{Tx_{\Delta_{time}}}{\Delta_{time}} \tag{22}$$

其中,Δ_{time} 表示交易创建完成到交易上链的时间间隔,$Tx_{\Delta_{time}}$ 表示Δ_{time}时间内上链的区块包含的交易数量。各算法吞吐量对比如图 12 所示。

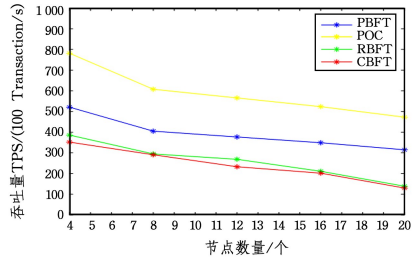

图 12 吞吐量对比

Fig. 12 Throughput comparison

结束语 本文分析了 PBFT 等相关算法的优缺点,针对其中的不足之处,提出了 POC 共识算法。该算法将网络中的节点进行角色分层,节点自由进出网络,网络状态可以进行动态调整。对于生产节点选举,本文设计了投票机制与基于信誉度的 follow-the-satoshi 算法相结合的方法,保证了安全性的同时兼顾公平性。在共识流程方面,所提算法优化了共识流程,提高了共识效率。实验结果表明,生产节点选举算法可以保证系统的安全性,同时具有一定公平性,缩减了网络开销,提升了交易吞吐量。POC 协议在共识流程上是基于 PBFT 算法的,其本质为一种拜占庭容错(BFT)类共识,理论上 BFT 共识算法可用于公链,但是在共识过程中需要多次全网广播,网络节点数量较多时,网络通信消耗巨大,因此公链系统一般不采用 BFT 类共识。从应用角度考虑,本文 POC 算法更适合用于联盟链或私有链。

参 考 文 献

[1] NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system [EB/OL]. https://bitcoin.org/bitcoin.pdf.

[2] YUAN Y, NI X C, ZENG S, et al. Blockchain consensus algorithms: the state of the art and future trends[J]. Acta Automatica Sinica, 2018, 44(11): 2011-2022.

[3] JAKOBSSON M, JUELS A. Proofs of work and bread pudding protocols (extended abstract). Secure Information Networks. Boston, MA[M]. Germany: Springer, 1999: 258-272.

[4] BUTERIN V. A next-generation smart contract and decentralized application platform[J]. Etherum, 2014(1): 1-36.

[5] NADAL S, GRAMOLI S. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake[C]// Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2017: 1-27.

[6] LARIME D. Delegated Proof-of-Stake (DPOS)[OL]. https://how.bitshares.works/en/master/technology/dpos.html.

[7] LAMPORT L, SHOSTAK R, PEASE M. The Byzantine generals problem[J]. ACM Transactions on Programming Languages and Systems, 1982, 4(3): 382-401.

[8] CASTRO M, LISKOV B. Practical byzantine fault tolerance and proactive recovery[J]. Acm Transactions on Computer Systems, 2002, 20(4): 398-461.

[9] ELLI A, ARTEM B, VITA B, et al. Hyperledger Fabric: a distributed operating system for permissioned blockchains[C]// Proceedings of the Thirteenth Euro -Sys Conference. Porto, Portugal, ACM, 2018: 30.

[10] AUBLIN P, MOKHTAR L, BEN S, et al. RBFT: Redundant byzantine fault tolerance[C]// Proceedings International Conference on Distributed Computing Systems. 2013: 297-306.

[11] ZHANG H. Concurrent Byzantine Fault Tolerance for Software-Transaction-Memory Based Applications [C] // International Journal of Future Computer and Communication. 2012: 47-50.

[12] MENG W T, ZHANG D W. Optimization scheme for hyperledger fabric consensus mechanism[J]. Acta Automatica Sinica, 2020, 46: 1-14.

[13] ZHANG S X, WEN J. A group-base blockchain consensus algorithm [J]. Computer Applications and Software, 2020, 37(3): 261.

[14] HAN Z Y, GONG N S. An Improved blockchain practical byzantine fault tolerance algorithm[J]. Computer Applications and Software, 2020, 37(2): 226.

[15] BENTOV I, LEE C, MIZRAHI A, et al. Proof of Activity[J]. ACM SIGMETRICS Performance Evaluation Review, 2014, 42(3): 34-37.

[16] ZHANG P Y, SONG J. Research Advance on Efficiency Optimization of Blockchain Consensus Algorithms[J]. Computer Science, 2020, 47(12): 296-303.

[17] YU B, LIU J, NEPAL S, et al. Proof-of-Qos: Qos based blockchain consensus protocol [J]. Computers & Security, 2019, 87(11): 101580. 1-101580. 13.

[18] LI K, LI H, HOU H, et al. Proof of Vote: A High-Performance Consensus Protocol Based on Vote Mechanism & Consortium Blockchain [C] // Proceedings-2017 IEEE 19th Intl Conference on High Performance Computing and Communications, DSS 2017: 466-473.

[19] HUANG J H, XIA X, LI Z C, et al. Proof of Trust: Mechanism of Trust Degree Based on Dynamic Authorization [J]. Journal of Software, 2019, 30(9): 2593-2607.

[20] FENG J Y, ZHAO X Y, CHEN K X, et al. Towards random-honest miners selection and multi-blocks creation: Proof-of-negotiation consensus mechanism in blockchain networks [J]. Future Generation Computer Systems, 2020, 105: 248-258.

ZHOU Yi-hua, born in 1969, Ph.D, associate professor. His main research interests include cryptography, blockchain theory and technology, privacy protection technology and the interdisciplinary of information security and other disciplines.

FANG Jia-bo, born in 1993, postgraduate. His main research interests include information security, blockchain and cloud storage.