一种基于双链模型的分区共识协议 *

黄建华[1†]，黄雪茹[1]，季钰翔[1]，唐瑞琮[2]

(1. 华东理工大学 信息科学与工程学院, 上海 200237; 2. 香港 DAEX 区块链有限公司, 上海 200120)

摘 要：针对目前分区模型中区块链的存储容量不能随着分区的增加而同步扩展以及分区算法存在的安全性问题，提出一种基于双链模型的分区共识协议 DB-SCP (Dual Blockchain-based Sharding Consensus Protocol)。首先通过基于哈希链和交易链的双链分区存储模型来设计验证信息共享机制和交易差异化存储机制，实现了区块链的存储容量随分区的增多而同步增加；其次，采用基于节点投票份额的分区方法把节点权益拆分到不同的分区，有效防止了分区中权益过多节点的出现；最后采用 VRF 函数改进分区内共识算法，保证了验证者选取的随机性，且使用密钥演变技术保证了交易的前向安全性。安全性分析表明，基于投票份额的分区方式既稳定又安全，实验结果表明该协议有着良好的性能优势，存储容量较传统区块链模型提升了大约 30%~70%。

关键词：区块链；分区；共识协议

中图分类号：TP doi: 10.19734/j.issn.1001-3695.2020.01.0002

Sharding consensus protocol based on dual blockchains

Huang Jianhua[1†], Huang Xueru[1], Ji Yuxiang[1], Tang Ruicong[2]

(1. *School of Information Science & Engineering*, *East China University of Science & Technology*, *Shanghai* 200237, *China*; 2. *HONG KONG DAEX Blockchain Limited*, *Shanghai* 200120, *China*)

Abstract: In view of the fact that the storage capacity of the blockchain in the current shard model cannot be expanded synchronously with the increase of shards and the security problems of the shard algorithm, this paper proposed a Dual Blockchain-based Sharding Consensus Protocol (DB-SCP) . Firstly, it used a dual-blockchain shard storage model based on hash chain and transaction chain to design the verification information sharing mechanism and differentiated transaction storage mechanism. Therefore, the storage capacity of the blockchain increases synchronously with the increase of shards. Secondly, it adopted the sharding mechanism based on voting shares, through spliting node stakes into different shards to prevent a node from having excessive stakes in a shard. Finally, it used the VRF function to improve the intra-shard consensus algorithm to ensure the randomness of verifier selection, and the key evolution technology introduced ensures the forward security of transactions. The security analysis shows that the sharding mechanism based on voting shares is both stable and safe. Experimental results show that the DB-SCP protocol has good performance advantages, and its storage capacity is improved by about 30%~70% compared with the traditional blockchain model.

Key words: blockchain; sharding; consensus algorithm

0 引言

区块链是一种融合了密码学、分布式系统、网络协议等一体的综合技术，具有去中心化、不可窜改、不可否认、可追溯等特点，在金融[1]、供应链、医疗保健、物联网等领域均展现了广阔的应用前景[2]。

由于没有中心机制，区块链需要通过共识算法选择一个节点出块，以保证所有节点状态的一致性和区块链的不可窜改性，因此，区块链的出块效率和安全性显得尤为重要。基于区块链的比特币系统采用的是工作量证明(PoW)共识算法[3]，节点之间通过计算一个难度值来决定谁来出块，参与其中的节点数越多，安全性越高，其问题是计算难度值的过程需要消耗大量的电力资源[4]。为了弥补此不足，Sunny King 在 PPCoin[5] 中引入了权益证明(PoS)共识算法，既能防止女巫攻击[6]，又不产生任何能耗。权益证明有很多不同的变种，如 DPoS[7] 、PoSV[8]和 PoA[9]，但都在一定程度上引入新的安全性问题[10]。以上两种都是属于基于证明类的共识算法，要求加入网络的节点能证明自己比其他节点更有资格添加一个区块到链上，主要应用于进行金融价值转移的公有链，其存在的问题是交易处理速度低，增加节点数量并不能显著提高系统吞吐量，且本质上都存在共识终结性问题，容易出现区块链分叉情况；另一类是基于投票的共识算法，要求网络中的节点交换当前新区块或者交易的验证结果，然后作出最终的决定。最典型的是实用拜占庭算法 PBFT[11]，PBFT 解决了之前拜占庭容错算法[12]效率不高的问题，可提供共识终结性，避免交易确认的延迟，可以容忍小于 1/3 个无效或者恶意节点，但它容易受到 DoS 攻击，存在通信复杂性和可扩展性问题。除了 PBFT，还有很多进一步提升性能或鲁棒性的 BFT 算法被提出，主要有 DBFT、HoneyBadger-BFT [13] 和 Tendermint[14]等。这类算法解决了证明类算法效率不高的问题，交易处理能力可达上千 TPS，且具有强一致性，不易分叉。其不足是节点扩展性差，通信复杂度高，对加入节点有许可要求，通常只能用于联盟链和私有链。

研究和实践表明分区是提高区块链扩展性和性能的有效途径。Zilliqa[15]是第一个提出使用分区解决可扩展性问题的公共区块链；以太坊[16]为扩容也在做分区方面的尝试，其方

收稿日期：2020-01-05；**修回日期：**2020-03-07 **基金项目：**国家自然科学基金面上项目(61472139)

作者简介：黄建华(1963-)，男(通信作者)，湖南麻阳，副教授，博士，主要研究方向为计算机网络与信息安全(jhhuang@ecust.edu.cn)；黄雪茹(1995-)，女，安徽亳州，硕士，主要研究方向为区块链；季钰翔(1996-)，男，江苏连云港，硕士，主要研究方向为区块链；唐瑞琮(1990-)，男，浙江杭州，硕士，主要研究方向为区块链与金融信息技术.

式是通过将区块分成子区块进行分而治之；Elastico[17]使用的分区协议可实现交易速度随着全网计算能力的提高呈线性增长。但是目前大多数的分区模型中每个节点存储的都是整个区块链的交易，并不能实现随着分区数的增多而同步扩展区块链存储容量,给区块链在物联网、金融等领域的应用带来挑战。同时现有的基于 PoW 的分区方式计算难度难以把握，导致分区时间不稳定，而基于节点的 PoS 分区方式则会出现节点在分区中权益过现象，导致分区的不安全性。

为了解决以上问题，本文提出一种基于双链模型的分区共识协议 DB-SCP(Dual Blockchains-based Sharding Consensus Protocol)，该协议包括地址分区和权益分区，并通过构建由哈希链和交易链组合的双链(dual blockchains)模型来提高区块链的存储能力，其中哈希链存储全网共享的验证信息；而交易链只存储与本分区有关的交易。为了提高分区过程和分区内共识的安全性，采用了基于节点投票份额的分区机制[18]，通过将节点的权益拆分成多个份额来实现节点投票份额的随机分区，从而有效防止单个节点在分区中权益过高的现象。在共识算法方面，将 VRF[19]引入 PBFT 共识算法来改进算法的安全性，保障了选取验证者的随机性和公平性。此外，为了保证前向安全性，共识算法引入了密钥演变技术[20,21]，以一个 epoch 为周期进行私钥的改变，即使节点的私钥泄露，也能防止恶意节点窜改过去的交易，可在一定程度上抵御长程攻击。

1　DB-SCP 分区共识协议概述

1.1　系统设置与假设

DB-SCP 中有两种角色：用户和权益人。用户使用区块链基础设施完成转账或合约调用；权益人负责整个区块链架构的运行，保证其安全性。为了成为权益人，参与者需缴纳一定的押金以加入权益集 VaSet，该集合的每一项元素包括押金缴纳者的公钥地址和押金数。作为权益证明，押金既防范女巫攻击又可对恶意节点进行惩罚，如果成功出块，权益人会得到与其押金成比例的奖励。假设系统共有 m 个权益人，P_i 表示第 i 个权益人($1\le i\le m$)；系统分为 n 个分区，分区编号用 s 位表示($n=2^s$)；每个权益人要想参加当前 epoch 的共识过程必须在上一个 epoch 时间内缴纳押金。

对于网络模型，假设网络通信信道是部分同步的，网络中信息传输存在一个未知的上界，即时延是任意的但也是有限的。对于攻击模型，本文假设全网恶意节点权益比少于 1/4，每个分区的投票份额数大于 150，在此假设下可保证每个分区的恶意节点权益比少于 1/3。

1.2　DB-SCP 分区共识协议

在 DB-SCP 中共有 2 种分区：地址分区和权益分区，前者是实现了区块链存储容量的扩展，后者则实现了分区内共识的安全。地址分区基于权益人地址将网络划分成一系列分区，各个分区独立并行地处理分区交易，每个权益人仅存储与本分区有关的交易，实现了交易的差异化存储，从而扩展了整个网络的存储容量；权益分区是将权益人所拥有的权益拆分成一个个份额并随机分到不同分区，以有效防止单个分区节点权益过大问题。

为实现交易的分区存储，本文提出了如图 1 所示的由哈希链和交易链组合的双链模型。不同分区拥有不同交易链，哈希链由全网共有，所有分区保持一致。交易链由与本分区有关的交易块构成，由分区内的所有权益人保存，权益人所属分区由其地址哈希值的低 s 位决定，即根据函数 $LSB_s(hash(address_{P_i}))$ 的值来确定所属分区。哈希链主要记录相关验证信息(如 merkle 根[22])和用于权益分区的随机种子等信息，由全网所有权益人保存。

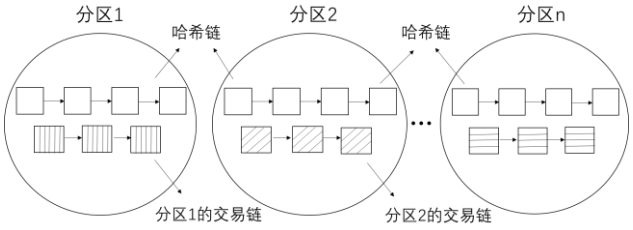

图 1　双链模型

Fig. 1　Dual-Blockchain model

整个系统以纪元(epoch)作为大周期运行，每个 epoch 再划分多个轮次(round)，在每一个 round 时间内，每个分区都会产生一个区块，可以用 epoch、round 和区号来标志某个区块。在每个 epoch 开始的时候，将根据最新哈希块中的随机数 random 来划分权益人投票份额所属分区，从而确定权益人在每个分区所拥有的投票值。分区共有两种类型，分为创块区和组合区。创块区有 n-1 个，负责并行产生区块；组合区只有一个，负责对各个创块区提交的哈希块(区块头)进行组合和验证，然后将组合好的子哈希链广播给整个区块链网络。为了保证协议启动，最初的 VaSet 和 random 会被硬编码到创世区块中，一个 epoch 的运行过程如图 2 所示。

在一个 epoch 中，权益人做以下操作：

a) 更新 VaSet 和建立权益分区。VaSet 记录押金缴纳者的公钥地址和其押金数，可以看做是权益列表。为了允许参与者随时加入和退出 VaSet，每个 epoch 结束后权益人都要更新 VaSet 以掌握系统最新的权益关系。在每一个 epoch 开始时，权益人获得与其所缴纳押金成正比的投票份额，这些投票份额会根据分区算法随机分配到各个分区，投票份额分到哪个区，则代表权益人在哪个区具有投票权。权益分区完成后，各个权益人和同一分区内其他成员建立连接。

b) 每一 round 各个创块区会产生一个区块。分区内使用改进的基于 VRF 的 RBFT 算法进行共识，分区成员每一 round 选取一个 leader 来提议区块，利用 VRF 选出一部分验证者对提议区块投票。如果区块得到足够多的投票，则 leader 发送区块体给分区内的权益人，权益人以交易块的形式保存，同时 leader 提交哈希块(区块头部分)到组合区。

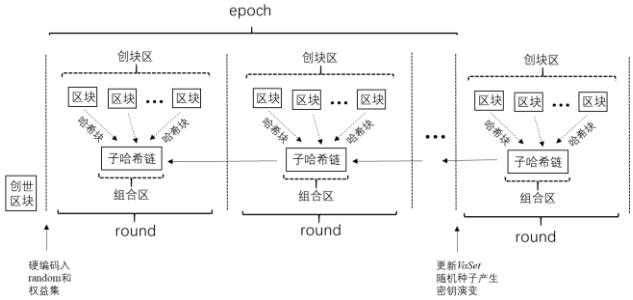

图 2　DB-SCP 运行示意图

Fig. 2　DB-SCP operation diagram

c) 组合区也使用 RBFT 算法确定哈希块的排列顺序。与创块区不同的是，分区内选取的 leader 负责将自己收到的每一个哈希块进行哈希，按照哈希结果从小到大对哈希块进行排列，最后形成一条一定长度的子哈希链，长度由 leader 节点收到哈希块的个数决定。之后通过 VRF 选取的验证者对哈希链进行投票，达成共识后 leader 将哈希链广播到整个区块链网络中，所有权益人将其添加到哈希链上。

d) 重复步骤 b)直到整个 epoch 结束，此时系统进入下一个 epoch 周期。新一轮 epoch 之初，权益人获得新的投票份额并将其重新分区，其分区所用的种子存在于最新的哈希块中，为了保证交易的前向安全性，随着 epoch 发生改变，权益人密钥随之进行演变。

2　权益分区

2.1　权益分区算法

目前的分区方法主要是基于 PoW 和 PoS 算法。基于 PoW 的分区方法不容易把握难度值的大小，会导致分区效率不高；而基于 PoS 的分区方法存在某些节点在分区所占权益比重过大的情况，若这些节点存心作恶，将无法保证分区内诚实节点所占比重超过规定阈值。针对这些问题，本文采用基于投票份额的分区算法[15]，将分区的安全考虑从最小节点数转移到最小投票份额数。如图 3 所示，权益人根据其缴纳的押金获得与之成比例的投票份额，这些投票份额会被随机分配到不同分区，从而权益人可以在对应的分区中获得相应的投票权，一个投票份额对应一票。通过这种投票份额分区方式，即使恶意节点拥有较大的权益，也难以在某一分区内进行集中攻击，而对于诚实节点而言，所拥有的权益越大，生成的投票份额越多，在各个分区得到的收益也会越多，不会导致诚实节点的利益受损。

图 3　基于投票份额分区

Fig. 3　Voting share- based sharding

系统共有 n 个分区，分区编号用 s 位表示($n=2^s$)，$q=LSB_s(hash(random \| address_{p_i} \| j_{p_i}))$，其中 $address_{p_i}$ 为 P_i 的账户地址，$random$ 为用于分区的随机种子，j_{p_i} 为权益人 P_i 的第 j 个投票份额编号，$LSB(\cdot)$ 表示最低 k 位，P_i 的第 j 个投票份额将根据 q 值分配到指定分区，j_{p_i} 的取值范围由份额大小和节点所拥有的权益值决定。份额大小 $share=total_e/n*se$，$total_e$ 为在 $epoch_e$ 中总的权益值，n 为分区数，se 为安全系数，它可以通过算法根据全网的权益变化情况动态设定。$sum_{p_i}=total_{p_i}/share$ 为节点 P_i 所拥有的投票份额数，前文所提到的 j_{p_i} 满足 $0\le j_{p_i}\le sum_{p_i}$。

每个参与权益分区的权益人会管理一张分区映射表，此表记录了各个权益人投票份额的分区情况。假设有 4 个权益人和 4 个分区，根据图 3 可得到表 1 所示投票份额分区结果，该表详细描述了每个权益人的投票份额被随机分到哪个区，以及每个权益人在每个分区所拥有的投票份额数。

表 1　基于投票份额的分区结果

Tab. 1　Sharding results based on voting shares

分区 1				
权益人 P_i	P_1	P_2	P_3	P_4
投票份额编号	{1,3,9}	{2}	{2}	{1}
投票份额数	3	1	1	1
分区 2				
权益人 P_i	P_1	P_2	P_3	P_4
投票份额编号	{2,6,10}	{1,6}	0	{5}
投票份额数	3	2	0	1
分区 3				
权益人 P_i	P_1	P_2	P_3	P_4
投票份额编号	{5,7}	{4,5}	0	{2,4}
投票份额数	2	2	0	2
分区 4				
权益人 P_i	P_1	P_2	P_3	P_4
投票份额编号	{4,8}	{3}	{1}	{3,6}
投票份额数	2	1	1	2

注：$P_1\rightarrow\{1,2,3,4,5,6,7,8,9,10\}$，$P_2\rightarrow\{1,2,3,4,5,6\}$，$P_3\rightarrow\{1,2\}$，$P_4\rightarrow\{1,2,3,4,5,6\}$，集合内为每个权益人的所有投票份额编号

2.2　分区算法安全性分析

本文中分区内使用改进的 PBFT 算法 RBFT，所以必须保证分区内恶意节点投票份额占比小于 1/3。设事件 $X=k$ 表示某个分区内恶意节点所拥有的投票份额数为 k，则其概率为

$$P(X=k)=\frac{C_M^k C_{N-M}^{n-k}}{C_N^n} \tag{1}$$

N 为全网总的投票份额数，$M=N/4$ 是恶意节点所拥有的最多投票份额数，$num=N/n$ 为每个分区所拥有的投票份额数，由此可以看出某分区内恶意节点所拥有的投票份额数为 k 的概率 $P(X=k)$ 服从超几何分布，当 N 无限大的时候，超几何分布转换为二项分布，则恶意节点所拥有的投票份额数不超过 k 的概率为

$$P(X<k)=\sum_{i=0}^{k}C_n^i p^i(1-p)^{n-i} \tag{2}$$

从表 2 可以看出，当 $k=50$ 时，$P(X<k)$ 高达 0.991，且随着 n 的不断增加，每个分区内恶意节点所占比重小于 1/3 的概率越来越高，分区内的安全性也越来越高。

表 2　k 与 $P(X<k)$ 的正比关系

Tab. 2　Proportional relationship between k and $P(X<k)$

k	n	$P(X<k)$
50	150	0.991
100	300	0.9995
150	450	0.99997
200	600	0.999998
250	750	0.99999987
300	900	0.99999999
350	1050	0.999999999
400	1200	0.99999999995
450	1350	0.999999999997
500	1500	0.99999999999979

接下来证明为什么假设整体恶意节点所拥有的权益比< 1/4 。由于每个分区采用的共识算法是基于 PBFT 算法，因此需保证分区内恶意节点所占比重少于 1/3，即 p 值满足不等式：

$$\sum_{i=0}^{k}C_{3k}^i p^i(1-p)^{3k-i}>0.99 \tag{3}$$

由图 4 可以看出，区内恶意节点所拥有的份额为 50 时，$p\approx 1/4$。当 $p=1/4$，$k>50$ 时，随着 k 的增多，其分区内恶意节点占比小于 1/3 概率更高，则整个系统也更安全。

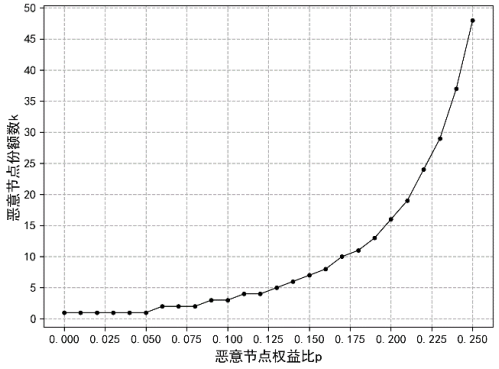

图 4　p 与 k 的关系

Fig. 4　Relationship between p and k

2.3　种子生成

前面提到的权益分区算法中有一个参数为随机种子，其作用是保证分区的随机性和公平性。Omniledger[23]使用 RandHound[24]协议产生种子，其涉及 PVSS(可公开验证的秘密共享)和拜占庭协议，复杂度较高；RapidChain[25]允许每个参与者都执行 VSS(可验证的秘密共享)[26]，最后使用组合的秘密共享作为结果，但是这个协议并不安全，因为恶意节点

可以将不一致的共享发送到不同的节点。基于以上考虑，本文采用 VRF 算法来产生随机种子，在保证种子随机产生的同时，也便于验证。

在介绍种子产生之前，首先介绍下 VRF 可验证随机函数，它以私钥和公开值作为输入，生成一个随机值和一个证明，验证者可以在不获知输入者任何私有信息的情况下证明此随机值的正确性，且验证速度很快。

在本文中创块区产生区块的区块头中会有一个专门的字段来存放种子及其证明，这个种子由 leader 节点在本地运行 VRF 产生，输入的公开值为本分区上一轮的 seed 值和本轮轮数，即：

$$\{seed_r, proof_r\} \leftarrow VRF_{sk_{leader}}(seed_{r-1} \| r) \tag{4}$$

如果本轮没有产生区块，则：

$$seed_r = Hash(seed_{r-1} \| r) \tag{5}$$

在下一个 epoch 开始时，随机节点分区所用的随机数是哈希链中最新块中的随机数，因为组合区 leader 组合的哈希链无法被提前预知，所以没有任何节点会提前预知随机数的值，同时用 VRF 产生的随机数也方便验证，防止某些节点随意编造。

3　区块链及其交易处理

3.1　区块结构

不同于传统的区块链模型，本文共有 2 种链：哈希链和交易链。哈希链由网络中所有节点保存，便于重新分区后验证交易，交易链由各个分区内的存储者保存，其内容为某个分区内交易记录。与链相对应，本文区块也分为哈希块和交易块 2 种，哈希块数据结构如图 5 所示。

图 5　哈希块结构

Fig. 5　Structure of hash block

另一种区块为交易块，主要由已经确认的交易组成，由分区内的存储者保存，从而实现每个分区只存储本分区内交易，即存储分区。但其内确认的交易形成的 merkle 根存储在哈希块中，以便跨区交易的验证。交易块数据结构如图 6 所示。

图 6　交易块结构

Fig. 6　Structure of transaction block

将区块分为哈希块和交易块之后，各分区之间的通信内容只有哈希块，相比于连同交易一起广播，减少了通信成本。

3.2　交易处理

为了防止交易在不同创块区中被重复处理，将根据发送方地址来分配交易所属分区，根据 $z = \mathrm{LSB}_s(\mathrm{hash}(\mathrm{address}_{p_i})) \bmod (n-1)$ 决定交易所属分区，即取发送方地址哈希值的最低 s 位并将其对创块区数目 $n-1$ 进行取余得到 z 值，z 为 0 则为 0 号创块区，为 1 则为 1 号创块区，依此类推。

如果发送方地址和接收方地址在同一分区则直接打包进区块，若发送方和接收方不在同一分区则会出现跨区交易的情况。已有的区块链系统中，RSCoin[27]和 OmniLedger 采用 2 阶段提交协议[28]处理跨区交易，其主要方式是“锁定”，这在一定程度上会限制交易的并发，造成交易堵塞。

为最大化交易的并发程度，本文引入“中继交易[29]”的概念。若交易的发送方和接收方不在一个分区，此交易在发送方会产生一个中继交易，携带一定的验证信息被发送到接收方所在分区。中继交易数据结构如图 7。

一个分区内被确认的跨区交易所产生的中继交易会形成一个列表，所以每个中继交易都会有一个索引，也就是图 7 中列表索引字段。所有的中继交易组成一个 merkle 树，其树根存储在哈希块中，而 merkle 路径则存储在中继交易中，便于跨区交易的验证。

图 7　中继交易

Fig. 7　Relay transaction

中继交易的出现最大化分区的优势，即允许交易以异步和无锁的方式进行，在很大程度上保证分区交易的并发，并能保持用户状态的最终一致性。

3.3　交易验证

本文中交易分为 2 种，普通交易和中继交易。普通交易又包含区内交易和跨区交易，区内交易在本分区内被快速打包上链；若是跨区交易，则会产生一个相应的中继交易，原来的交易也正常被打包，此时交易的发送方状态发生改变，但只有当相应的中继交易在目的分区被打包后，交易接收方的账户状态才会发生改变。

假设分区 1 的用户 A 转一笔 X 元的钱给分区 2 的用户 B：

1)分区 1 的 leader 收到此交易首先对用户 A 的余额进行检查，如果 A 的余额小于所转金额 X，此交易作废，否则 leader 会将其放进自己的交易池中,收集到一定的交易后则会打包交易成块，并将此区块发送给其他验证者。经过 2 轮投票之后，共识区块内的所有交易将被确认，相应的用户 A 的余额会减少 X 元。由于转账交易为跨区交易，所以会产生一个中继交易发送到分区 2 中，在分区 2 中完成后面的交易验证操作。

2)接收到中继交易的分区 2 会把它当普通交易进行处理。首先对其进行验证，若验证通过则打包上链。验证的流程如下，首先根据中继交易中的发送区号、块号用来标识产生此交易的哈希块，从而找到哈希块中的中继交易的 merkle 根，通过此中继交易中的 merkle 路径验证其交易的正确性。若通过验证，则分区 2 中的 leader 会把此交易打包进新的区块，提交到哈希链上，此时用户 B 的余额会增加 X 元，交易框架如图 8 所示。

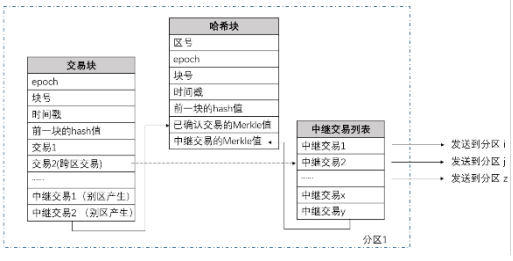

图 8　交易框架

Fig. 8　Transaction frame

4　区内共识算法

本文对 PBFT 算法进行了改进，提出基于 VRF 的算法 RBFT (random byzantine fault tolerance)，用于创块区和组合区共识。与传统的 PBFT 相比，RBFT 使用 VRF 算法随机选择验证者，并确保验证者数量控制在一定范围内，增强了随机性和安全性。同时 RBFT 引入密钥演变技术来保证过去交易的不可窜改，增强了抵御长程攻击的能力。

分区内成员每 round 基于 follow-the-satoshi 算法选举 leader，参与投票的验证者由权益人运行 VRF 算法选出，选中的验证者与 leader 建立连接。新区块由 leader 打包并广播给各个验证者，leader 和验证者运行 RBFT 共识算法就所提交的区块达成共识，区块头(哈希块)提交到组合区，区块体(交易块)在本分区内广播，由本分区的权益人保存。组合区选举的 leader 收到一定数量的哈希块，对其进行验证和组合，形成一定长度的子哈希链，发给各个验证者，达成共识后，leader 将子哈希链在全网进行广播。区内共识流程如图 9 所示。

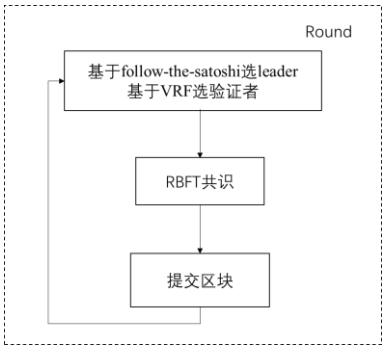

图 9　区内共识流程

Fig. 9　Intra-shard consensus process

4.1　leader 选举

RBFT 在每个 round 之初选取 leader。如果仅根据权益大小选取 leader，攻击者就会预先知道 leader 的顺序，并有针对性地对以后的 leader 进行攻击。为了避免这个问题，RBFT 采用 follow-the-satoshi 算法选取 leader, 这里 satoshi 表示聪，是加密货币的最小单位。follow-the-satoshi 最早在 PoA 中被提出，该算法的目标是使得每个权益人被选为 leader 的概率与其权益成正比，由于每轮的 leader 又是不确定的，只能说权益越大，被选中的可能性越大，安全性也更高。

RBFT 对 follow-the-satoshi 的实现为：权益人将本分区的成员权益表按公钥地址顺序排列，取上一个 round 哈希块中随机种子哈希值最后几位，由其所处的权益区间决定本轮的 leader。例如某分区投票份额表为 $(P_1,4),(P_2,3),(P_3,5),(P_4,4)$，划分的权益区间如图 10 所示。权益总数为 16 个 satoshi，需要 4bit 表示，则取上一个哈希块中随机种子哈希值最后 4 位，假设值为 10，计算 10 mod 16=10，属于 P_3 区间，则 P_3 为本轮的 leader。

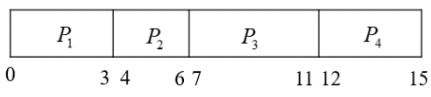

图 10　权益区间

Fig. 10　Stake range

4.2　随机选举验证者

利用 VRF 选取验证者，随机选择验证者算法见算法 1。选出委员会成员的个数由参数 τ 决定，在本文中 τ=16 ，可保证选出验证者个数在 4 和 30 之间的概率达到 0.999，详细证明参照文献[30]。

算法 1　随机选择算法

输入： $(SK, seed, \lambda, blockNum, w, W)$

$\langle random, \pi\rangle \leftarrow VRF_{SK}(seed \| blockNum)$

$p \leftarrow \frac{\lambda}{W}$

$i \leftarrow 0$

while $\frac{random}{2^{randomLen}} \notin [\sum_{k=0}^{i} B(k;w,p), \sum_{k=0}^{i+1} B(k;w,p)]$ *do*

　i++

输出： $\langle random, \pi, i\rangle$

基于权益的共识算法中，被选取的概率应该和其所拥有的权益成正比，从算法 1 可以看出，拥有的权益值越多，运算出来的 i 越大[27]。

各个权益人算出 $\langle random, \pi, i\rangle$ 后，若 $i>0$，则会主动和 leader 节点建立连接，leader 收到后首先会对 *random* 值进行验证，验证算法见算法 2。经过一段时间后 leader 将自己提议的

区块广播给通过验证的节点，即验证者。leader 和验证者随后运行 RBFT 算法，就新提出的区块达成共识。一轮结束之后，开始下一轮的 leader 和验证者选举，重复上述操作。

算法 2：验证算法

输入： $(PK, random, \pi, seed, \lambda, blockNum, w, W)$

if $\neg VerifyVRF_{PK}(random, \pi, seed, blockNum)$

then return 0;

$p \leftarrow \frac{\lambda}{W}$

$i \leftarrow 0$

while $\frac{random}{2^{randomLen}} \notin [\sum_{k=0}^{i} B(k;w,p), \sum_{k=0}^{i+1} B(k;w,p)$ *do*

　i++

输出： i

4.3　密钥演变

基于权益的共识算法不需要消耗算力，产生区块几乎没有成本，因此导致攻击者可以从历史上很远的一个区块开始制造分叉,也就是所谓的长程攻击。制造分叉最常用的方式就是获取某些节点的私钥，窜改全部或部分交易历史，从而代替原有的主链。为了在一定程度抵御长程攻击，本文共识算法引入密钥演变技术，此技术意义在于保证密钥具有前向安全性，即使密钥在未来被破坏也能够保护过去进行的交易不可窜改。密钥演变的最大特点是公钥固定，而私钥随着时间进行更新，且这个更新过程是单向的。

本文使用的密钥演变算法基于椭圆曲线 $E: y^2+xy=x^3+ax^2+b$，基点 $G=(x_G,y_G)\in E$。每个 epoch 之后，所有节点以 epoch 为参数进行密钥演变，其过程共分为四个阶段：

a)密钥产生：初始私钥随机选取 $SK_0<q$，q 为一无穷大素数，$PK=SK_0^{\tau}G$。假设一个 epoch 为一周，则令 T=52。

b)密钥演变：$SK_{epoch}=SK_{epoch-1}^2 \bmod (q-1)$。此算法具备单向安全性，即知道 SK_{epoch} 无法推断出 $SK_{epoch-1}$。

c)签名阶段：如算法 3，消息 M 在本文中即为 leader 节点提出的区块。恶意节点无法从 $(d,e,m,M,R_{epoch},epoch)$ 中获取私钥，因为 c 的值没办法获取。

d)验证阶段：如算法 4 所示，验证者收到签名后，首先对 R_{epoch} 进行验证，验证节点是否进行了密钥演变，之后会对收到的参数一一验证，验证通过后签名验证流程过程结束，在下一个 epoch 后重复密钥演变、签名和验证 3 阶段。

在 DB-SCP 协议中节点可以任意加入和离开，如果每个新节点的私钥都从 S_0 开始，验证时会造成一定的混乱，因此每个新加入的节点 $SK_{epoch}=SK_0^{2^{epoch}}$，epoch 就是加入时的 epoch 数。

算法 3　签名算法

输入：基点 G ，公钥 PK ，私钥 SK_{epoch} ，消息 M

$m \leftarrow H(M)$

随机选 $c \subset [1,n]$ ， $d \leftarrow (c - SK_{epoch})m^{-1}$ ，

$R_{epoch} \leftarrow SK_{epoch}G$ ，

$e \leftarrow (c - dSK_{epoch}^{2^T - epoch}) \bmod q$

输出：$(d,e,m,M,R_{epoch},epoch)$

算法 4　验证算法

输入：$(d,e,m,M,R_{epoch},epoch)$ ，公钥 PK

如果 $PK = R_{epoch}^{2^T - epoch}G$ ，继续；否则返回 `0`

$X = mdG + R_{epoch} = (u,v)$

$Y = eG + dPK = (x,y)$

$u' = u \bmod q$

$x' = x \bmod q$

如果 $u' = x'$ ，返回 `1`；否则返回 `0`

输出：`1`(成功)或`0`(失败)

5　实验评估

本节主要从吞吐量、共识时延、交易处理时间、存储量等方面进行实验评估。

5.1　实验环境与参数设置

实验使用 Docker 虚拟化技术来搭建区块链的多节点环境，Docker 的运行环境为 1 台 DELL R320 服务器，整个实验的软硬件环境如表 3 所示。

表 3　实验环境配置

Tab. 3　Experimental environment configuration

软硬件	配置
Docker Engine	版本 18.09.0
CPU	Intel Xeon(R) E5-2407@2.2GHZ
内存	64GB
操作系统	Ubuntu server 14.04

实验初始分区个数为 2，每个分区内共有 5 个节点，之后依次增加分区个数到 20，共进行 10 组实验，所有数据都取在相同参数下的平均数。

5.2　性能测试

分区性能主要从吞吐量和时延两个方面来评估。这里吞吐量指单位时间内整个系统处理的交易量，分区时延指所有参与共识的权益人进行投票份额随机分区所花费的时间。为了便于仿真，本文将区块大小设置为 1KB，且由于交易内容的不同导致交易大小不是常数，同时每个区块中的具体交易数并不影响实验结果中的变化趋势，所以简单假设每个区块中有 2 笔交易。

吞吐量实验分别测试了 DB-SCP 协议和 PoW 协议，结果如图 11 所示。由实验结果可以看出，DB-SCP 的吞吐量要明显高于 PoW，且 DB-SCP 的吞吐量随节点数量的增长呈近线性的增加，而节点数量的增长却对 PoW 的吞吐量影响不大。造成这种现象的原因是 DB-SCP 中每个分区可以并行处理交易，随着网络中的节点数量的不断增加，可以划分出更多的分区，就能同时处理更多的交易请求，因而系统的吞吐量也就在不断提高。与之相对，基于 PoW 的共识机制中每个共识周期只能生成一个共识区块，因此即使网络中节点数量增加，也不会对系统的吞吐量有任何的贡献。

分区时延实验对比测试了基于投票份额、基于 PoS 和基于 PoW 三种分区方式的分区时延。基于投票份额分区所用的随机种子从哈希链中最新的区块中获得，在新一轮 epoch 之初，权益人获得与押金成比例的投票份额，实验假设每个权益人拥有的投票份额数为 10。权益人获取随机种子之后对自己的公钥、随机数和份额编号进行哈希运算得到每个份额所属分区，然后向同一分区的其他权益人请求建立连接，当分区内的成员形成连通图后分区过程结束，所以分区时延为计算所有份额的哈希时长加上节点建立连接时长。基于 PoS 的分区方式中，权益人直接对自己的公钥、随机数进行哈希运算，从而得到所属分区，因此分区时延为哈希时长加上节点建立连接时长。基于 PoW 的分区方式中节点根据难度值消耗算力计算出一个符合条件的随机数，由随机数的最后几位得出自己所属分区，接着与同一分区内其他节点进行连接。与计算随机数的时耗相比，节点建立连接时长可以忽略不计，所以实验中 PoW 分区时延仅考虑计算随机数时长。

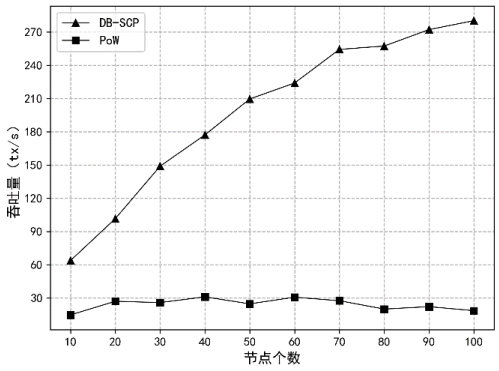

图 11　吞吐量实验

Fig. 11　Throughput experiment

在分区数不变的情况下，实验测试三种分区方式的时延，共进行了 10 组实验，实验结果如图 12、13 所示。从图 12 可以看出基于投票份额的分区时延要小于 PoW，且相对稳定。PoW 分区时延呈现出一种不稳定的状态，原因是每次 PoW 计算满足条件的随机数的时间都不一样，而基于 PoS 份额的分区时延只需要进行哈希运算，所以相对稳定。由图 13 可以看出，基于投票份额的分区时延和基于 PoS 的分区时延差不多且都比较小，原因是两者都是仅需要进行哈希运算。由以上分析可以看出基于投票份额的分区方式在提高分区安全性的同时也很好的控制了分区时延。

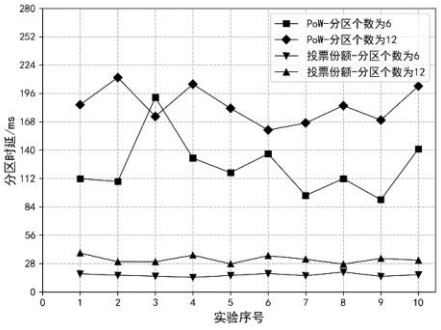

图 12　基于投票份额与 PoW 分区耗时对比

Fig. 12　Comparison of sharding delay between voting shares and pow

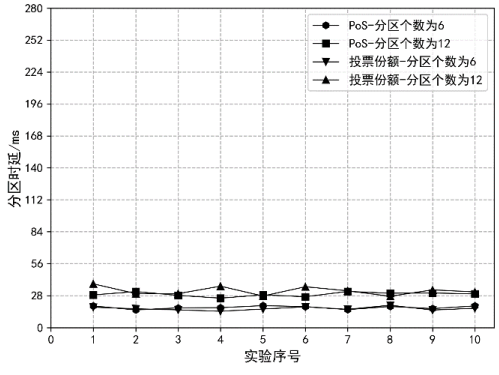

图 13　基于投票份额与 PoS 分区耗时对比

Fig. 13　Comparison of sharding time Consumption between voting shares and pos

此外，实验还测试了基于 PoW、基于 PoS 和基于投票份额三种分区方式分区时延随着分区数增加的变化情况，如图 14 所示。由实验结果可知，随着分区数的不断增加，三种方式分区时延都在增加，原因是更多的分区会造成整个分区过程所花费的时间增加，但基于投票份额和基于 PoS 的分区时延要明显低于 PoW 的分区时延，时延降低了大约 70%~90%。随着分区数的不断增多，可以看到基于投票份额的时延要稍大于 PoS，因为前者权益人需要计算每个份额的所属分区，而后者只需要计算一次所属分区即可，但通过实验结果可以看出基于投票份额时延仅仅比基于 PoS 时延多了 3%。

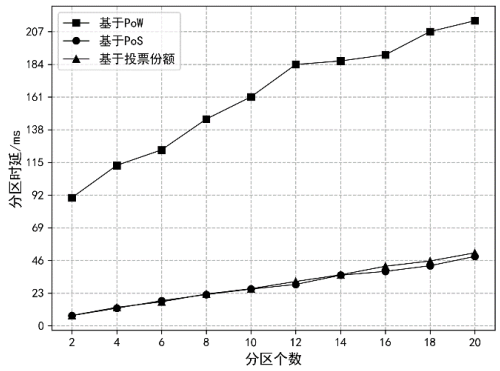

图 14　三种分区方式时延对比

Fig. 14　Sharding delay comparison of three methods

5.3　交易处理速度测试

交易性能测试实验同时考虑区内交易和跨区交易，在实验中设定跨区交易占总交易数的 3/4，所以整个系统处理的交易数实际上是正常交易数的 1.75 倍，实验结果如图 15 所示。可以看出，虽然引入中继交易来处理和验证跨区交易会导致总体交易数量增多，但是其处理速度仍然大于正常处理交易的速度，且随着交易数量的不断增多，相比于不分区，分区的处理速度增加得越来越多，造成这一现象的主要原因是分区处理交易速度增长的幅度远远大于交易数增长的幅度。由此可以看出中继交易的引入既保持了跨区交易的安全性，同时也提高了交易处理的速度。

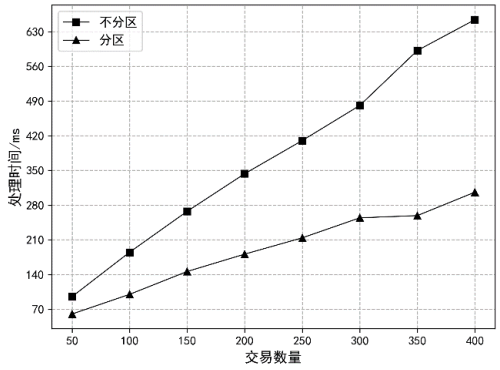

图 15 分区与不分区处理交易时间对比

Fig. 15　Comparison of transaction processing time between sharding and non-sharding

5.4　交易处理速度测试

在双链模式下，每个权益人仅存储与本分区有关的交易，从而减少了本地的存储量，实验对进行存储分区和不进行存储分区的区块链存储量进行了对比。假设每个分区提交 5 个区块，每个分区有 10 个权益人，每一个区块中区块头和区块体的比例为 1：3，根据不同的分区个数可计算出分区存储方式与不分区存储方式的存储量，结果如图 16 所示。从图中可以看到，分区存储较不分区存储，存储量降低了 30%~70%，从扩容的角度来说，采用存储分区方式的区块链容量提高了 30%~70%。存储效率提高的主要原因是每个分区的权益人只存储与本分区有关的交易，从而实现交易记录的分区存储，增加了区块链的存储容量，减少了存储成本。

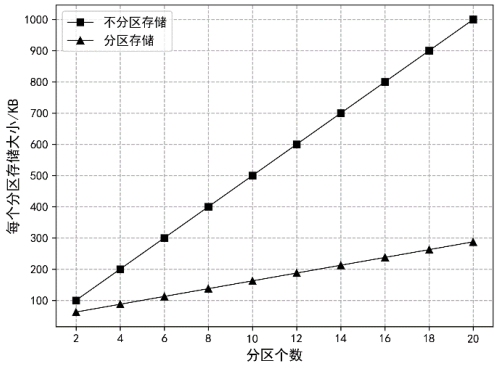

图 16　分区存储与不分区存储容量对比

Fig. 16　Comparison between sharding storage and non-sharding storage capacity

6　结束语

区块链交易记录的存储需求与日俱增，为了扩展区块链的存储容量，本文提出一种基于双链模型的分区共识协议 DB-SCP，该协议采用由哈希链和交易链组成的双链架构，其中哈希链全网共享，存储相关验证信息，交易链仅存储与本分区有关的交易，不同分区拥有不同交易链，分区越多，则区块链存储容量越大。此外，用于分区内共识的 RBFT 算法使用 VRF 保证了选举验证者的随机性和安全性，引入的密钥演变技术保证了交易的前向安全性。针对现有的基于 PoW 分区方机制存在的分区效率较低和基于 PoS 的分区方式存在单个节点权益过高等问题，提出了基于权益投票份额的分区机制，可有效防止节点在分区中权益过高的问题。安全性分析表明，基于投票份额的分区机制既稳定又安全，实验结果表明 DB-SCP 有着良好的性能优势，在存储容量方面较传统区块链提升了大约 30%~70%。

参考文献：

[1] 王硕. 区块链技术在金融领域的研究现状及创新趋势分析 [J]. 上海金融, 2016 (2): 26-29. (Wang Shuo. Analysis on the research status and innovation trend of blockchain technology in the financial field [J]. Shanghai Finance, 2016 (2): 26-29.)

[2] 袁勇, 王飞跃. 区块链技术发展现状与展望 [J]. 自动化学报, 2016, 042 (004): 481-494. (Yuan Yong, Wang Feiyue. Development status and prospect of blockchain technology [J]. Acta Automatica Sinica, 2016, 042 (004): 481-494.)

[3] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system [EB/OL]. (2008) . http://bitcoin. org/bitcoin. pdf.

[4] Gervais A, Karame G O, Wyust K, *et al*. On the security and performance of proof of work blockchains [C]// Proceedings of the 2016 ACM SIGSAC conference. ACM, 2016: 3-16.

[5] King S, Nadal S. PPcoin: peer-to-peer crypto-currency with Proof-of-Stake [J]. Self-published Paper, 2012.

[6] Douceur J R. The sybil attack [C]// International Workshop on Peer-to-Peer Systems. Berlin: Springer, 2002: 251-260.

[7] Luo Y, Chen Y, Chen Q, *et al*. A new election algorithm for DPos consensus mechanism in blockchain [C]// 2018 the 7th International Conference on Digital Home (ICDH) . IEEE, 2018: 116-120.

[8] Usman C. The narcotized blockchain: a potcoin case study [J]. SSRN Electronic Journal, 2018.

[9] Busse A, Eberhardt J, Frost S, *et al*. A Response to the United Nations

CITES Blockchain Challenge: Incremental and Integrative PoA-based Permit Exchange [C]// 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) . 2019: 320-328.

[10] Zheng Zibin, Xie Shaoan, Dai Hong-Ning, *et al.* Blockchain challenges and opportunities: a survey [J]. International Journal of Web and Grid Services, 2018, 14 (4): 352-375.

[11] Castro M, Liskov B. Practical byzantine fault tolerance [C]// In Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI) . New Orleans, 1999.

[12] 范捷，易乐天，舒继武. 拜占庭系统技术研究综述. 软件学报, 2013, 24 (6): 1346-1360. (Fan Jie, Yi Le-Tian, Shu Ji-Wu. Research on the technologies of Byzantine system. Journal of Software, 2013, 24 (6): 1346-1360) .

[13] Makhdoom I, Abolhasan M, Ni W. Blockchain for IoT: the challenges and a way forward [C]// Proceedings of the 15th International Joint Conference on e-Business and Telecommunications. Portugal, 2018.

[14] Kwon J. Tendermint: consensus without mining [EB/OL]. (2016) . https://tendermint. com/static/docs/tendermint. pdf.

[15] The Zilliqa Team. The zilliqa technical whitepaper [EB/OL]. (2017) . https://docs. zilliqa. com/whitepaper. pdf.

[16] Albert. On sharding blockchains [EB/OL]. (2019) . https://github. com/ethereum/wiki/wiki/Sharding-FAQs.

[17] Luu L, Narayanan V, Zheng Chaodong, *et al*. A secure sharding protocol for open blockchains [C]// In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York, 2016: 17–30.

[18] The Harmony Team. Harmony technical whitepaper [EB/OL]. (2018) https://harmony. one/pdf/whitepaper. pdf.

[19] Micali S, Rabin M, Vadhan S. Verifiable random functions [C]// In Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS) . New York, 1999.

[20] Bellare M, Miner S K. A forward-secure digital signature scheme [C]// In Advance in Cryptology-CRYPTO. Berlin: Springer, 1999: 431-448.

[21] Bo L B L, Lin Yinlin. A new forward-secure digital signature scheme based on elliptic curve [C]// In the 2nd International Conference on Industrial and Information Systems. China, 2010.

[22] Wikipedia. Merkle tree [EB/OL]. https://en. wikipedia. org/wiki/Merkle tree.

[23] Kokoris-Kogias E, Jovanovic P, Gasser L, *et al.* Omniledger: A secure, scale-out, decentralized ledger via sharding [C]// In 2018 IEEE Symposium on Security and Privacy (SP) . IEEE, 2018: 583-598.

[24] Syta E, Jovanovic P, Kokoris-Kogias E, *et al.* Scalable bias-resistant distributed randomness [C]// In the 38th IEEE Symposium on Security and Privacy. IEEE, 2017.

[25] Zamani M, Movahedi M, and Raykova M. RapidChain: a fast blockchain protocol via full sharding [C]// In the 2018 ACM SIGSAC Conference. Cryptology ePrint Archive, 2018.

[26] Feldman P. A practical scheme for non-interactive verifiable secret sharing [C]// In Proceedings of the 28th Annual Symposium on Foundations of Computer Science. Washington: IEEE Computer Society, 1987: 427-438.

[27] Danezis G, Meiklejohn S. Centrally banked cryptocurrencies [C]// In Proceedings of the 23rd Annual Network and Distributed System Security Symposium. 2016.

[28] Lamport L. How to make a multiprocessor computer that correctly executes multiprocess programs [J]. IEEE Transactions on Computers, 1979, 28 (9): 690–691.

[29] Wang Jiaping. Monoxide: scale out lockchains with asynchronous consensus zones [C]// In the 16th USENIX Symposium on Networked Systems Design and Implementation. 2019.

[30] Gilad Y, Hemo R, Micali S, *et al.* Algorand: scaling byzantine agreements for cryptocurrencies [C]// In Proceedings of the 26th Symposium on Operating Systems Principles. New York: ACM, 2017: 51–68.