

面向区块链的轻量级抗量子分布式共识方案

张德俊, 李剑, 王乃乐, 张春晖

(北京邮电大学网络空间安全学院, 北京 100876)

摘要: 现有抗量子算法存在计算复杂度高、存储开销大的问题, 且当前区块链系统在多节点环境下共识吞吐量压力显著。针对上述问题, 提出了面向区块链的轻量级抗量子分布式共识方案, 通过改进基于哈希的安全后量子互联网通信签名增强版 (secure post-quantum hash-based internet communication signatures plus, SPHINCS+) 抗量子签名算法中的随机子集森林 (forest of random subsets, FORS) 结构, 减少签名体积和存储负担, 设计改进 Winternitz 一次签名 (Winternitz one-time signature, WOTS+) 结构, 降低算法计算复杂度。提出了分组实用拜占庭容错 (practical byzantine fault tolerance, PBFT) 共识机制, 并结合积分与加权随机森林模型设计主节点选举机制, 降低通信开销、提升网络可靠性与吞吐量。实验结果显示, 与传统 SPHINCS+ 抗量子签名算法及改进 PBFT 的方案相比, 该方案签名平均验证速率和共识吞吐量分别提升 25% 和 28%, 签名存储压力和共识时延分别降低 31% 和 25%, 表明该方案能够提升区块链系统的签名、验证效率和系统吞吐量, 并降低共识时延。

关键词: 抗量子; 区块链; 分布式; PBFT

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.AQ25268

Lightweight quantum-resistant distributed consensus scheme for blockchain

Zhang Dejun, Li Jian, Wang Naile, Zhang Chunhui

School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract: Existing post-quantum cryptographic algorithms have been confronted with high computational complexity and large storage overhead, while current blockchain systems have suffered from significant consensus throughput pressure in multi-node environments. To address these issues, a lightweight post-quantum distributed consensus scheme for blockchain systems was proposed in this work. The scheme was constructed by improving the FORS structure in the SPHINCS+ post-quantum signature algorithm, through which the signature size and storage burden were reduced, and by designing an improved WOTS+ structure to decrease the computational complexity of the algorithm. Furthermore, a grouped PBFT consensus mechanism was proposed, and a primary node election mechanism was designed by integrating a credit-based scoring strategy with a weighted random forest model, thereby reducing communication overhead and improving network reliability and throughput. Experimental results demonstrated that, compared with the traditional SPHINCS+ post-quantum scheme and improved PBFT algorithms, the proposed scheme achieved improvements of 25% and 28% in average signature verification rate and consensus

收稿日期: 2025-12-27; 修回日期: 2026-05-01

通信作者: 李剑, lijian@bupt.edu.cn

基金项目: 国家自然科学基金资助项目 (No.U1636106)

Foundation Item: The National Natural Science Foundation of China (No.U1636106)

throughput, respectively, while the signature storage pressure and consensus latency were reduced by approximately 31% and 25%. These results indicate that the proposed scheme effectively enhances signature and verification efficiency as well as overall system throughput of blockchain systems, while reducing consensus latency.

Key words: post-quantum, blockchain, distributed, PBFT

0 引言

随着区块链技术在金融、医疗、政务等关键领域的深入应用,其安全性和可扩展性面临着前所未有的挑战^[1]。一方面,当前主流区块链系统依赖基于数论的传统公钥密码体制,如 RSA (Rivest-Shamir-Adleman) 和椭圆曲线签名 (elliptic curve cryptography, ECC),而这些算法的安全性在量子计算飞速发展的环境下受到严重威胁^[2]。针对此问题,相关学者在签名方案领域展开了相关研究^[3-4]。在诸多签名方案中,SPHINCS+ 哈希抗量子签名方案具备无状态、高效、易实现等特点,尤其适用于区块链这类交易频繁、对性能敏感的环境,但该方案的结构存在签名体积大、计算复杂度高等缺陷。另一方面,实用拜占庭容错 (practical byzantine fault tolerance, PBFT) 作为当下区块链系统的主流共识机制,随着区块链节点规模的持续扩张,在通信开销和系统吞吐量方面难以满足高性能应用需求。因此,研究面向区块链的轻量级抗量子分布式共识方案具备重要意义。

基于哈希的数字签名方案最早由 Diffie 和 Hellman^[5]提出,他们设计了一个利用单向函数实现 1 位数字签名的算法。Lamport^[6]基于哈希签名方案设计了 Lamport 一次性签名方案 (LOTS),但该方案存在密钥过大、易伪造等缺点,于是 Merkle^[7]对 LOTS 进行改进,提出 Winternitz 一次签名 (WOTS) 方案,WOTS 方案通过将消息分块并多次迭代哈希函数来生成签名,并在消息后添加校验和,显著缩短了密钥大小,提升了签名生成的效率和安全性。但上述方案均为一次性签名方案,为了解决这一问题,Merkle^[8]利用树型结构将大量一次性签名算法 (OTS) 的公钥组织起来,从而实现了多个消息的签名,然而上述方案仍然为一次性签名方案。直到 Reyzin 等^[9]提出哈希获取随机子集 (hash to obtain random subsets, HORS) 方案,才真正实现了多次签名。随后, Bernstein

等^[10]提出基于哈希的安全后量子互联网通信签名增强版 (secure post-quantum hash-based internet communication signatures plus, SPHINCS+) 方案,该方案将 HORS 与超树结构结合,使用密钥种子代替传统密钥,大大增加了密钥对的利用率。然而 HORS 本身的局限性限制了方案的发展,于是 Bernstein^[11]利用随机子集森林 (forest of random subsets, FORS) 替换了 SPHINCS 方案中的 HORS,提出美国国家标准与技术研究院 (national institute of standards and technology, NIST) 标准化的 SPHINCS+ 方案,后来 SPHINCS+ 方案逐渐成为基于哈希的后量子签名方案研究基础,并有诸多学者基于此方案进行不断优化。Wang 等^[12]利用国产 SM3 哈希函数实现了 SPHINCS+SM3 方案,使得方案具备更可靠的安全性。Shahid 等^[13]对基于 SPHINCS+ 方案的改进 Winternitz 一次签名 (Winternitz one-time signature, WOTS+) 结构进行优化,提出新型哈希抗量子方案 WOTS-S,并大幅度降低了传统签名的存储开销。Zhang 等^[14]提出 SPHINCS+ α 方案,它通过改进 Winternitz 一次性签名方案和引入一种新的多次签名方案随机链森林 (forest of random chains, FORC),提高了签名的效率。虽然上述方案均从不同角度对哈希签名方案进行了优化改进,但其所具备的 FORS 结构与 WOTS 结构仍然存在较大的改进空间。

共识机制作为区块链系统的核心组件,是保障区块链系统安全性和一致性的关键。随着节点规模扩大,传统 PBFT 的交易吞吐量显著下降,而通信开销和交易时延大幅增加。为了解决这些问题,国内外诸多学者针对 PBFT 的不足进行了改进。DBFT^[15]将 PBFT 与委托权益证明算法^[16]相结合,为降低通信过程的复杂性,该方案共识节点由委托权益证明 (delegated proof of stake, DPOS) 算法中的股份投票方式选举产生。Fu 等^[17]采用聚合签名的方式降低共识通信量。基于分布式账本 Raft 的实用拜占庭容错 (distributed ledger raft-based practical byzantine fault toler-

ance, DL-RBFT^[18]) 将 Raft^[19] 共识方案与 PBFT 相结合以提升网络吞吐量。Zhong 等^[20] 提出了分片信任实用拜占庭容错 (sharded trust practical byzantine fault tolerance, ST-PBFT), 这是一种基于一致哈希的分区方法, 旨在降低 IP 交易场景中的通信复杂性。此外, Ji 等^[21] 引入了群体价值实用拜占庭容错 (group value practical byzantine fault tolerance, GV-PBFT), 这是一种基于群体信誉值的 PBFT 共识优化算法, 旨在提高系统效率。然而, 上述方案均存在一定的缺陷, DBFT 方案的容错率较低, 聚合签名在签名生成阶段计算成本较高, 不适用于资源受限的环境, 而 Raft 与 PBFT 结合的方案存在无法大规模扩充节点的问题。ST-PBFT 和 GV-PBFT 方案则并未解决节点扩充带来的通信量爆炸问题。为有效解决上述问题, 本文引入 PBFT 分组共识机制, 降低通信开销的同时, 提高网络吞吐量。

综上所述, 本文提出面向区块链的轻量级抗量子分布式共识 (lightweight post quantum distributed consensus scheme for blockchain, LPDC) 方案。本文的主要贡献如下。

1) 改进了基于哈希的抗量子签名方案 SPHINCS+, 通过重构叶子节点索引改进 FORS 结构, 解决树节点过多导致的存储压力问题。

2) 为 WOTS+ 结构设计哈希链合并构造算法, 减少算法存储开销的同时降低算法的计算复杂度。

3) 设计分组共识机制, 实现“先组内, 后组间”的共识模式, 在保证安全的前提下, 显著提升了 PBFT 共识吞吐量, 并结合积分投票与加权随机森林模型设计主节点选举机制, 提升系统可靠性。

1 预备知识

1.1 哈希函数

哈希函数是一种单向映射函数, 能够将任意长度的输入数据 m 转换为固定长度的哈希值 $h(m)$, 且具备抗原像性、抗第二原像性、抗碰撞性。

本文所使用的伪随机函数包含 $\text{PRF} = \{0,1\}^n \times \{0,1\}^{256} \rightarrow \{0,1\}^n$, $\text{PRF}_{\text{msg}} = \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^n$ 。PRF 伪随机函数用于处理伪随机密钥的生成, PRF_{msg} 用于构建压缩消息的随机性。本文所使用的哈希函数为 $H_{\text{msg}} = \{0,1\}^n \times \{0,1\}^n \times$

$\{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^n$, H_{msg} 用于压缩任意长度的消息。

1.2 FORS

FORS 是 SPHINCS+ 方案中用于支持有限次签名的结构, 可从消息摘要中提取索引并生成初级签名。FORS 结构将消息摘要 m 分为 k 个子块, 每个子块表示一棵高度为 a 的二叉 Merkle 树中的一个叶子索引, 其中 r 为各个子树的根节点。每棵树的签名由对应叶子的私钥和从该叶子到树根的认证路径组成, 最终将这 k 棵树的根节点哈希组合为一个统一的输出, 并作为后续签名的输入。一个 $k=6$, $a=2$ 的 FORS 签名结构如图 1 所示, 其对应的签名消息为 100 100 111 011。

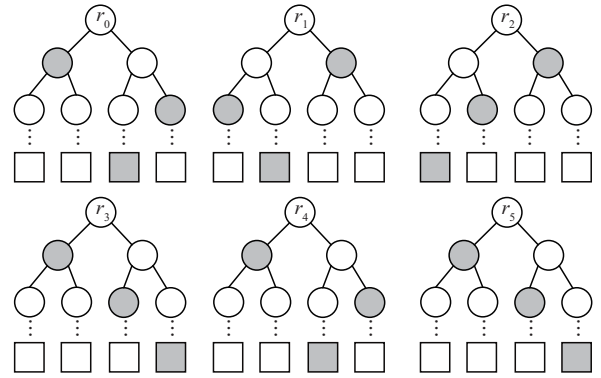


图1 FORS签名结构

1.3 WOTS+

WOTS 是一种基于哈希函数构建的单次使用数字签名方案。它主要通过构造哈希链来生成签名值, 每条链仅签署消息的一个片段。由于完全基于哈希函数构建, WOTS 具备良好的抗量子攻击能力, 是构建后量子签名系统的重要基础模块。而 WOTS+ 是 WOTS 方案的改进版本, WOTS+ 在原方案的基础上引入了伪随机掩码和地址参数, 使得相同输入不会产生可预测的输出, 从而增强了抗结构攻击和相关消息攻击的能力, 现在 WOTS+ 已经成为 SPHINCS+ 方案的标准底层签名组件之一。

1.4 PBFT

PBFT 是一种针对拜占庭容错问题设计的实用型共识算法, 旨在在分布式系统中容忍部分节点出现任意类型的故障, 包括系统崩溃、网络异常, 甚至恶意作恶。该算法假设系统中最多存在 f 个拜占庭节点, 并要求系统总节点数满足 $n \geq$

$3f+1$ 的条件, 从而在多数诚实节点的协作下实现安全和一致性。

PBFT 按照如下结构运行, 包含一个主节点和多个备份节点, 每轮共识过程分为 3 个阶段: 预准备 (pre-prepare)、准备 (prepare)、提交 (commit) 和回复 (reply)。客户端首先向主节点发送请求, 主节点排序并广播预准备消息, 随后各节点通过准备与提交消息相互验证, 当一个节点收到来自 $2f+1$ 个不同节点的匹配消息后, 即可认为请求达成一致并执行操作。执行完成后, 节点将执行结果作为回复消息发送给客户端。PBFT 流程图如图 2 所示, X 代表拜占庭节点, 也就是破坏节点。

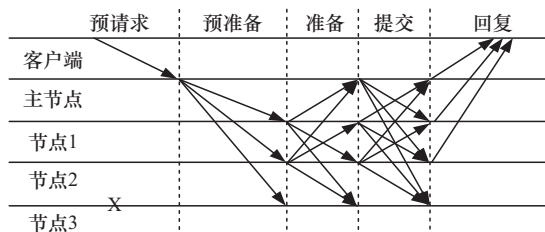


图2 PBFT 流程

2 面向区块链的轻量级抗量子分布式共识方案

2.1 方案模型

当前区块链系统普遍采用基于数论难题的数

字签名方案, 然而随着量子计算的快速发展, 此类签名方案在理论上面临被量子算法有效破解的潜在风险, 如何保障区块链系统在后量子环境下的签名安全性已成为亟须解决的关键问题。同时, PBFT作为当前联盟链中广泛采用的共识机制, 在节点规模不断扩大的场景下, 其通信复杂度随节点数量的增多呈平方级增长, 导致通信开销显著增加, 系统吞吐量和可扩展性受到严重限制, 难以满足高性能区块链应用的需求。

针对上述安全性与性能瓶颈, 本文提出了面向区块链的轻量级抗量子分布式共识方案。首先, 方案基于传统 SPHINCS+ 抗量子签名方案, 对构造 FORS 结构的方法加以改进, 采用叶子节点索引重构动态生成子树的方法降低存储开销。同时, 改进 WOTS+ 签名算法的签名流程, 降低算法计算复杂度, 最后设计分组共识机制与主节点选举机制降低传统 PBFT 方案的通信复杂度并提升系统吞吐量。面向区块链的轻量级抗量子分布式共识方案结构如图 3 所示, h/d 指每一层子 Merkle 树的高度。

2.2 SPHINCS+改进方案

传统 SPHINCS+ 算法由 HyperTree、WOTS+、FORS 3 个结构组成。FORS 作为算法的底层结构负责对消息摘要进行签名, FORS 结构由多棵

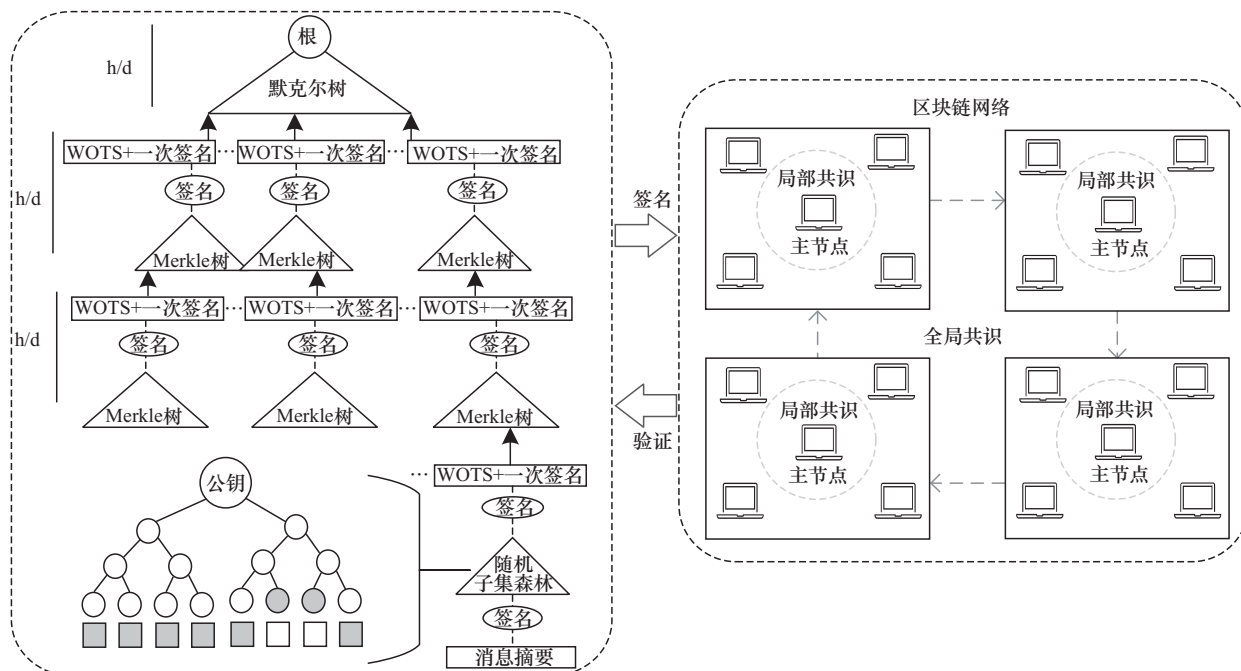


图3 面向区块链的轻量级抗量子分布式共识方案结构

Merkle子树组成，每棵Merkle子树在一次签名中仅有一个叶子节点被选中，并与其余认证节点共同参与FORS根节点的计算，在计算得到FORS根节点后，利用WOTS+结构对FORS根节点进行签名。

WOTS+是一种哈希链结构，该结构首先对输入值进行校验和计算，随后对输入值及校验和 C 进行哈希链长度映射，根据映射长度进行对应次数的哈希操作，哈希结束后便得到输入值的WOTS+签名，此时继续对签名进行哈希操作直至达到哈希链的长度，便得到WOTS+公钥。

HyperTree是超树结构，即Merkle树中的Merkle树，下层Merkle树的根节点作为WOTS+的输入，经过哈希链的运算后，得到WOTS+公钥，WOTS+公钥将作为上层Merkle树的叶子节点与该树的认证节点组合向上层不断进行哈希运算，直至计算得到整个Merkle树的根节点。如此递归下去，便可得到整棵HyperTree的根节点。SPHINCS+方案所使用的参数含义见表1。

表1 SPHINCS+方案设计的参数

参数	含义
n	安全参数
k	FORS结构数
t	FORS叶子节点数
M_{FORS}	FORS结构输入值
MD	消息摘要
FORS _{PK}	FORS公钥
FORS _{SK_{i, j}}	FORS叶子节点私钥
FORS _{Sign}	FORS签名
PK _{.root}	超树根节点
R	随机数
$M_{\text{WOTS}+}$	WOTS+输入值
WOTS _{PK}	WOTS公钥
WOTS _{SK_{i}}	消息分片 i 的WOTS私钥
WOTS _{Sign}	WOTS签名
w	Winternitz基数
C	校验和
d_i	消息分片 i 的大小
len_1	消息编码对应的哈希链条数
len_2	校验和对应的哈希链条数
p	合并哈希链的数目
q	校验和进位

在传统SPHINCS+算法中，FORS结构的每棵子树仅使用一个叶子节点，且需要计算大量认证节点，造成存储与计算开销较大。同时算法包含大量WOTS+结构，而WOTS+的哈希链结构需要进行大量的哈希操作，这导致算法的运行效率直线下降。

基于上述缺陷，本文基于传统SPHINCS+方案的FORS和WOTS+进行优化。方案的核心思想是对FORS结构进行叶子节点索引聚合，将多棵树各用一个叶子的模式，转变为不同索引的叶子节点聚合为一棵树，对于索引冲突的叶子节点则构造新的树进行容纳。改进后的方案不再直接构造 k 棵子树结构，而是按需构造。对于WOTS+结构，本文引入新的参数 p 、 q 。新参数用于压缩哈希链的条数，并保存校验和进位精度，从而在保证安全性的同时，大幅降低算法计算复杂度。方案具体流程如下。

(1) 参数初始化

首先定义安全参数 n 、FORS结构数量 k 、叶子节点数 t 。随后，利用安全参数 n 定义以下伪随机函数和哈希函数： PRF 、 PRF_{msg} 、 H_{msg} 。构建完上述函数^[22]后，则需要使用 PRF 伪随机函数生成 SK.seed 、 PK.seed 、 SK.prf 。 SK.seed 和 PK.seed 分别为私钥种子和公钥种子， SK.prf 为生成签名的随机化因子，用于防止重放攻击。接下来基于消息和 SK.prf 伪随机地生成一个随机器 R ，通过添加额外的随机性 OptRand 抵御多个跟踪的侧信道攻击，定义如式(1)所示：

$$R = \text{PRF}(\text{SK.prf}, \text{OptRand.key}) \quad (1)$$

在得到随机生成器 R 后，便可使用 R 计算消息摘要MD，其中 PK.root 为超树节点的根哈希值，具体定义如(2)所示：

$$\text{MD} = H_{\text{msg}}(R, \text{PK.seed}, \text{PK.root}, M_{\text{FORS}}) \quad (2)$$

(2) FORS签名

在计算FORS密钥之前，需要先利用消息摘要MD构造FORS子树。给定 ka 位的消息MD，提取 k 个 a 位的分片 $\text{md}_i, i \in [0, k-1]$ ，每一个分片 md_i 都是FORS子树的叶子节点。 FORSConstruct 函数是构造FORS子树的功能函数，该函数的输入为：分片消息 md_i ，分片位数 a 。对每个分片进行如下构造：

$$(\text{treeNums}, \text{ADRS}_{ij}) = \text{FORSSConstruct}(\text{md}_i, a), \quad (3)$$

$$i \in [0, k-1], j \in [0, 2^a-1]$$

消息 M 的每个分片都会计算得到不同的叶子节点索引 ADRS_{ij} ，其中 i 代表 FORS 子树的索引， j 代表叶子节点的索引。不同的叶子节点索引聚合为一棵 FORS 子树，以节点索引最大重复度 treeNums 作为 FORS 子树的数目。接着根据得到的叶子节点索引，构建不同叶子节点的 FORS 密钥 FORSSK_{ij} 。 FORSSK_{ij} 由 kt 个随机的 n 位值组成，其中， k 代表 FORS 结构数， t 为 FORS 叶子节点数， kt 为二者乘积，这些值是利用 PRF 函数和分片索引从 SK.seed 上派生出来的。

$$\text{FORSSK}_{ij} = \text{PRF}(\text{SK.seed}, \text{ADRS}_{ij}), \quad (4)$$

$$i \in [0, k-1], j \in [0, 2^a-1]$$

综上所述，索引重构的 FORS 结构如图 4 所示。

如图 4 所示， $k=6$ 、 $a=2$ 时，对消息 100 100 111 011 的 FORS 签名结构进行前后对比，索引重构算法首先计算出消息的最大重复度为 2，随后构建 2 棵 FORS 子树，根据每个消息分片的索引 ADRS_{ij} ，将对应分片分配到 i 棵子树的 j 个叶子节点上，再利用密钥种子 SK.seed ，结合伪随机函数 (pseudo random function, PRF) 为每个消息分片计算密钥 FORSSK_{ij} ，利用 FORSSK_{ij} 对叶子节点进行签名并与其兄弟节点进行组合哈希，直至得到根节点。对于未分配消息分片的空叶子节点则不进行密钥计算，而是向上按需计算整棵子树的认证节点，并与其他节点共同组成认证路径。

FORS 签名包括认证路径和叶子节点。若 FORS 子树的叶子节点全部被索引，则利用 FORSSK_{ij} 签名分片 md_{ij} ，并将所有叶子节点签

名进行组合哈希操作；否则先利用 FORSSK_{ij} 对分片 md_{ij} 进行签名。随后，构建认证路径 proofPath ，再将认证路径与签名进行组合哈希，便得到若干 FORS 子树的根哈希节点 root_i ，对 root_i 执行合并哈希操作，输出 FORS 的根节点 FORSPK 。将上述 FORSSK_{ij} 、叶子节点与认证路径结合，共同组成签名 FORSSign 。具体算法流程如算法 1 所示， M 为输入的消息。

算法 1 FORS 签名算法

输入 $M, \text{SK.seed}, \text{SK.prf}, \text{PK.seed}, k, a, t$

输出 $\text{FORSSign}, \text{FORSPK}$

// Step 1: 计算消息摘要

1) $R \leftarrow \text{PRF}(\text{SK.prf}, \text{OptRand_key})$

// 生成随机数 R ，防止重放攻击

2) $\text{MD} \leftarrow \text{H_msg}(M, R, \text{PK.seed}, \text{PK.root})$ //

计算消息摘要 MD

// Step 2: 将消息摘要分段，得到 k 个索引

3) **for** $i=0 \rightarrow k-1$ **do**

4) $\text{md}_i \leftarrow \text{ParseBits}(\text{MD}, i, a)$

// 将 MD 切分为 a 位索引

5) **end for**

// Step 3: 构造叶子节点及认证路径

6) **for** $i=0 \rightarrow k-1$ **do**

7) $\text{addr} \leftarrow \text{ComputeAddress}(i, \text{md}_i)$

// 计算叶子索引地址

8) $\text{leaf_sk} \leftarrow \text{PRF}(\text{SK.seed}, \text{addr})$ // 生成叶子私钥

9) $\text{auth_path} \leftarrow \text{ComputeAuthPath}(\text{SK.seed}, \text{addr}, a)$ // 计算认证路径

10) $\text{save}(\text{leaf_sk}, \text{md}_i, \text{auth_path})$

11) **end for**

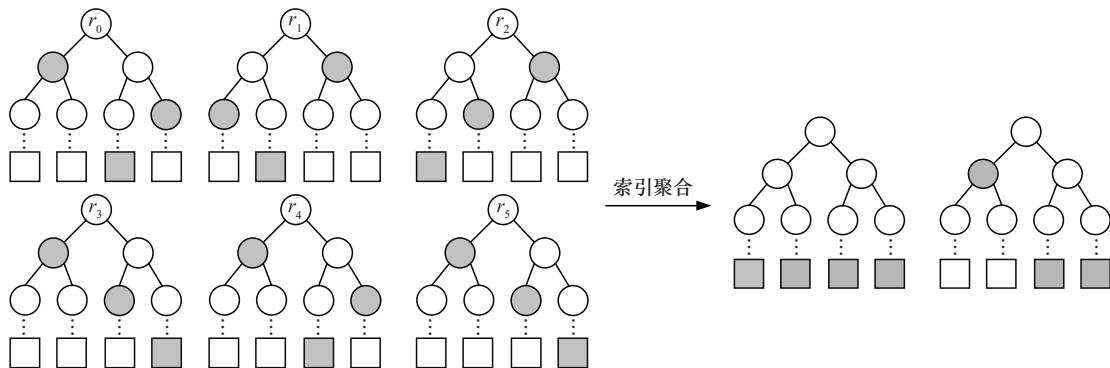


图 4 索引重构的 FORS 结构

// Step 4: 计算每棵子树的根节点
 12) **for** $i=0 \rightarrow k-1$ **do**
 13) $\text{roots}[i] \leftarrow \text{ComputeLeafRoot}(\text{leaf_sk}, \text{auth_path}, \text{md_i}, a)$
 14) **end for**
 // Step 5: 合并所有子树根节点, 得到FORSPK
 15) $\text{FORSPK} \leftarrow \text{HashAggregate}(\text{roots}, k)$
 // Step 6: 输出签名
 16) $\text{FORSSign} \leftarrow \{R, \text{所有叶子节点签名}, \text{所有认证路径}, \text{FORSPK}\}$
 17) **return** $(\text{FORSSign}, \text{FORSPK})$.

(3) WOTS+签名

如图5所示, 本文对原WOTS+与所提优化方案的结构进行对比。原方案中, w 为单条哈希链的最大迭代步数, 同时也代表单条哈希链的长度, 一般为4、8、16等。首先, 将WOTS+的输入值 $M_{\text{WOTS+}}$ 拆分为 $\text{len}_1 = n/\log_2^w$ 个分片 $d_i, i \in [0, \text{len}_1 - 1]$, 随后计算校验和 C :

$$C = \sum_{i=0}^{\text{len}_1 - 1} (w - 1 - d_i) \quad (5)$$

再将校验和 C 编码为 len_2 个分片 $d_i, i \in [\text{len}_1, \text{len}_1 + \text{len}_2 - 1]$, 得到总计 $\text{len}_1 + \text{len}_2$ 条哈希链。分片 d_i 本身的大小决定了在其对应的哈希链中进行哈希迭代的次数, 同时利用PRF函数为每个消息分片生成对应的私钥元素 WOTSSK_i 。对每个分片进行 d_i 次哈希运算便得到了消息的签名 WOTSSign , 随后继续对每个消息分片进行 $w - d_i - 1$ 次哈希运算, 便得到公钥 WOTSPK 。

本文设计的WOTS+签名方案旨在保证安全性的前提下降低计算复杂度与存储开销。具体方案为: 引入参数 p, q 。 p 为数据哈希链的合并

数, q 为校验和进位精度, 合并后的数据哈希链为 $\text{len}_1 - p + 1$ 条。对 len_2 条校验和链同样执行合并操作, 并且将校验和链合并为一条哈希链。经过上述操作后, 新方案的哈希链条数仅为 $\text{len}_1 - p + 2$ 条, 大大降低了哈希链的数目。方案具体流程如下。

在得到FORSPK后, 需要利用WOTS+对FORSPK进行签名, 首先需要设定Winternitz参数 w , 它定义了哈希链的长度。然后计算消息 $d = H(\text{FORSPK})$, 随后将 d 拆分为 d_i , 并分别映射到WOTS+的哈希链上, 即: $d_0, d_2, \dots, d_{\text{len}_1 - 1} \in w$, 随后计算校验和 C 并映射为校验和链: $d_{\text{len}_1}, \dots, d_{\text{len}_1 + \text{len}_2 - 1} \in w$, 再定义参数 p , 执行合并哈希操作, 校验和进位精度保存为参数 q , 合并哈希具体计算过程为:

$$d_{p-1} = \sum_{i=0}^{p-1} d_i \bmod w, d_{\text{len}_1} = \sum_{i=\text{len}_1}^{\text{len}_1 + \text{len}_2 - 1} d_i \bmod w \quad (6)$$

对于 $i \in [p-1, \text{len}_1 + 1]$, 计算 $\text{WOTSSign}_i = F^{d_i}(\text{WOTSSK}_i)$, 最后输出 $\text{WOTSSign} = (\text{WOTSSign}_{p-1}, \dots, \text{WOTSSign}_{\text{len}_1 + 1})$ 。对 WOTSSign_i 继续执行哈希操作, 直至每个部分完成 w 次哈希, 再将这部分组合起来便得到WOTS+公钥: $\text{WOTSPK} = F^{w - d_i - 1}(\text{WOTSSign}_i)$ 。WOTSPK 将作为Hyper-Tree的叶子节点与认证路径共同组合成Hyper-Tree签名, 并层层向上递归, 直至得到Hyper-Tree的根节点值。

上述优化方案在保留了校验和机制的情况下, 大大压缩了哈希链的数目, 同时极大地降低了系统的开销与计算复杂度。方案具体算法如算法2所示。

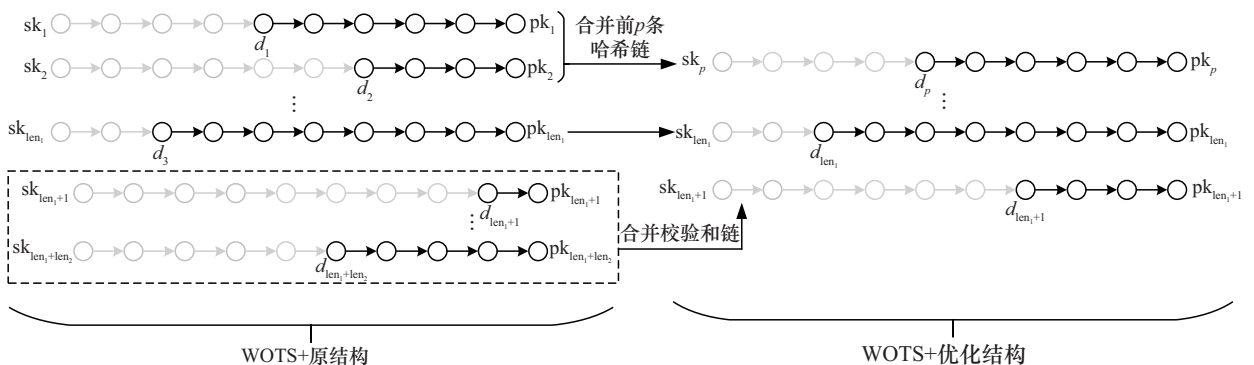


图5 WOTS+签名优化方案

算法2 WOTS+签名算法输入 FORSPK, SK_seed, w , p

输出 WOTSSign

// Step 1:计算消息摘要向量

1) digest ← Hash(FORSPK)

//对 FORSPK 进行哈希

2) $a[] \leftarrow \text{BaseW_Convert}(\text{digest}, \text{len}_1, w)$ //将摘要转为 base- w 向量

// Step 2:计算校验和

3) $C \leftarrow \Sigma(w-1 - a[i])$ 4) for $i=0 \rightarrow \text{len}_1-1$ 5) checksum[i] ← BaseW_Convert(C , len_2, w)

// Step 3:合并哈希链

6) for $i = p-1 \rightarrow \text{len}_1-1$ do7) sum ← $\Sigma a[i-r], r=0 \rightarrow p-1$ 8) $a_merged[i-(p-1)] \leftarrow \text{sum mod } w$

9) end for

10) $d_chk \leftarrow \Sigma \text{checksum}[i] \text{ mod } w$ // 校验和链压缩11) $q \leftarrow \Sigma \text{checksum}[i] \div w$ // 校验和进位精度

// Step 4:为每个消息分片生成私钥并迭代哈希

12) for $i = 0 \rightarrow \text{len_new}-1$ do13) $\text{addr} \leftarrow \text{ComputeWOTSAddress}(i)$ 14) $\text{sk}_i \leftarrow \text{PRF}(\text{SK.seed}, \text{addr})$ // 生成分片私钥15) $\text{tmp} \leftarrow \text{sk}_i$ 16) for $j = 0 \rightarrow a_merged[i]-1$ do $\text{tmp} \leftarrow \text{WOTS_ChainHash}(\text{tmp},$ $\text{addr}, j)$ // 哈希链迭代

17) end for

18) $\text{WOTSSign}[i] \leftarrow \text{tmp}$

19) end for

20) return WOTSSign

(4) 签名验证

签名需要从底层 FORS 结构开始进行验证, 使用签名中的随机值 R 、PK.seed 和 PK.root 计算消息摘要 MD, 随后将 MD 拆分为不同的 a 位分段, 进行叶子节点索引重构并构造子树。使用签

名中的 FORSSign 对叶子节点进行签名, 并层层递归直到获得 FORSPK。将 WOTS+签名中的 WOTSSign、WOTSPK 进行解析, 计算 $d = H(\text{FORSPK})$ 及校验和 C_{verify} , 随后对校验和进行比对, 验证方式如下:

$$\sum_{i=0}^{p-1} d_i \text{ mod } w = d_{p-1}, C_{\text{verify}} = q \cdot w + d_{\text{len}_1} \quad (7)$$

对于 $i \in [q-1, \text{len}_1]$ 都有 $\text{WOTSPK} = F^{w-d_i-1}(\text{WOTSSign}_i)$ 。随后继续验证 HyperTree 的根节点是否正确, 若上述步骤均无误, 则签名正确。具体流程如算法3所示。

算法3 签名验证算法输入 消息 M , 签名 Sig, 公钥 PK

输出 签名验证结果

// Step 1:重新计算消息摘要

1) MD ← H_msg(M , Sig.R, PK.seed, PK.root)2) indices[] ← Parse_MD(MD, k , a)

3) //Step 2:验证 FORS 签名

4) FORSPK ← FORS_Verify(Sig.FORS-Sign, indices, PK.seed)

5) if FORSPK == NULL then return false

6) // Step 3:验证 WOTS+签名

7) $d_hash \leftarrow \text{Hash}(\text{FORSPK})$ 8) $a[] \leftarrow \text{BaseW_Convert}(d_hash, \text{len}_1, w)$ 9) checksum ← $\Sigma(w-1 - a[i])$ 10) for $i = 0 \rightarrow \text{len}_2-1$ doif Sig.WOTSSign[len_1+i] ≠ checksum[i] then

return false

end if

11) end for

// Step 4:计算公钥并验证超树根节点

12) for $i = 0 \rightarrow \text{len}_1-1$ do13) $\text{pk_elem}[i] \leftarrow \text{WOTS_ChainHash_To_End}(\text{Sig.WOTSSign}[i], a[i], w)$

14) end for

15) WOTSPK_prime ← HashAggregate(pk_elem, len_1)

16) node ← Merkle_ComputeRoot(WOTSPK_prime, Sig.authPath, PK.seed)

- 17) **if** node == PK.root then
- 18) **return** true
- 19) **else**
- 20) **return** false
- 21) **end if**

2.3 分组共识机制

PBFT作为当前区块链系统中广泛应用的一类拜占庭容错共识协议,具备良好的安全性和可用性。然而,传统PBFT在节点规模扩展时存在严重的通信开销问题。PBFT的通信复杂度为 $O(N^2)$,全网节点需要在每个共识阶段进行全网广播,导致网络通信压力剧增^[23-24]。当节点数量超过一定规模后,系统的吞吐量显著下降,难以满足大规模区块链系统对于高性能、高扩展性的需求^[25-26]。针对上述问题,本文设计了一种基于分组共识机制的PBFT改进算法分组实用拜占庭容错 (grouped practical byzantine fault tolerance, G-PBFT)。其中G-PBFT方案使用的参数含义见表2。

表2 G-PBFT方案参数含义

参数	含义
N	网络总节点数
G	网络分组数
P_i	组内节点
L_i	组间节点
L_{main}	组间主节点
HD	哈希摘要
v	当前视图编号
s	当前请求序列号
R_i	局部共识结果
R_{global}	全局共识结果
σ_i	节点 i 的签名
f_{local}	组内拜占庭节点数
f_{global}	组间拜占庭节点数

G-PBFT方案将全网节点划分为多个独立小组,首先在组内通过传统PBFT机制达成局部共识,然后由各组主节点在组间运行简化的PBFT进行全局共识。通过将节点划分为规模更小的子集,组内通信复杂度由原始的 $O(N^2)$ 降低为 $O\left(\frac{N^2}{G}\right)$,组间通信复杂度控制在 $O(G^2)$,其中 G

为分组的个数, N 为全网节点总数。算法具体流程如下。

(1) 节点初始化

系统启动节点后,设整个区块链网络由 N 个共识节点组成,这些节点被划分为 G 个分组,每组包含 $n = N/G$ 个节点。对于第 i 组,定义节点集合为:

$$P_i = \{P_{i,1}, P_{i,2}, \dots, P_{i,n}\}, 1 \leq i \leq G \quad (8)$$

在每个子组内,根据主节点选举机制,选出一个组内主节点,记为 $L_i \in P_i$ 。这些主节点组成全局主节点集合:

$$L = \{L_1, L_2, \dots, L_G\} \quad (9)$$

每个节点在初始化阶段均被分配一个唯一标识ID及签名密钥对 (PK_{ij}, SK_{ij}) ,用于共识过程中消息的认证与验证。此外,每个节点需要在本地维护如下信息:当前视图编号 v ,组内成员列表 P_i ,组间节点列表 L 、请求缓存池 M_{ij} 、局部及全局状态日志等。

(2) 主节点选举机制

主节点选举时,将考虑积分投票、加权随机森林 (weighted random forest, WRF) 两部分因素。每个选举周期首先由链上或联盟内的可验证子协议计算并公布每个节点的积分 $S_i(t)$,该积分是节点长期与近期贡献的综合量化指标,并作为积分投票部分的基础。系统基于同一周期内采集的多维实时特征训练一份随机森林模型并对每个节点输出模型可信度分数 $M_i(t)$ 。两部分分别给出节点的贡献与可靠性评估,最后把这两部分进行结合形成最终选举权重,并得到最终主节点。通过以上方式在鼓励长期诚实贡献和防止被少数节点操控之间取得平衡。

节点积分由3部分组成:周期积分、奖励量、惩罚量。具体的计算公式为:

$$S_i(t+1) = \eta \cdot S_i(t) + \alpha_b \cdot B_i(t) - \alpha_p \cdot T_i(t) \quad (10)$$

其中, $S_i(t)$ 为节点 i 在周期 t 结束时的积分, η 是历史保留系数,用于使历史贡献逐步衰减以避免永久, $B_i(t)$ 是当前周期的行为奖励量, α_b 为奖励放大系数, $T_i(t)$ 为当前周期的惩罚量, α_p 为惩罚系数。

行为奖励 $B_i(t)$ 本身由若干子指标加权得到,将相关指标采用归一化线性组合便得到最终的奖

励值。具体来说, 其计算过程如下:

$$B_i(t) = \sum_{j=1}^m w_j f_{ij}(t) f_{ij}(t) \in [0,1], \sum_{j=1}^m w_j = 1 \quad (11)$$

其中, $f_{ij}(t)$ 表示第 j 个原始指标在周期 t 的标准化值, 如 t 个周期内节点在线时长占比、交易打包率、响应时延的倒数、签名/验证成功率、历史诚信评分等。权重 w_j 由协议初始参数确定。

惩罚 $T_i(t)$ 则是按事件集合加权累加, 若周期内节点触发若干违规事件, 则进行如下计算:

$$T_i(t) = \sum \rho \cdot \delta, \delta \in [0,1] \quad (12)$$

其中, ρ 为惩罚事件的基准罚分, δ 为共识确认率, 即有多少比例的节点共同确认该惩罚事件, 从而使惩罚与证据强度呈正比。

而对于加权随机森林, 系统构建加权随机森林模型 WRF 来评估节点当期的运行能力与鲁棒性。对节点 i 构造经准化的特征向量 $\mathbf{x}_i(t)$, WRF 的输出定义为模型分数:

$$M_i(t) = \sum_{k=1}^T w_k \text{tree}_k(\mathbf{x}_i(t)) \quad (13)$$

其中, T 为森林中树的数量, $\text{tree}_k(\mathbf{x}_i(t))$ 为第 k 棵树对样本的打分, w_k 为第 k 棵树的权重, 且满足 $\sum_k w_k = 1$ 。

有了积分 $S_i(t)$ 与模型分数 $M_i(t)$ 后, 方案将二者合成为原始选举权重。在此定义两部分标准化后值分别为 $S'_i(t)$ 与 $M'_i(t)$ 。综上所述, 将两者的分数进行合成得到最终选举主节点的分数:

$$U_i(t) = S'_i(t) + M'_i(t) \quad (14)$$

(3) 客户端请求

当客户端 C 希望在区块链网络发起一项交易请求 m 时, 生成如下请求消息:

$$\text{request} = \langle \text{request}, m, t_c, \sigma_c \rangle \quad (15)$$

其中, m 为交易内容, t_c 是客户端时间戳, 用于防止重放攻击, $\sigma_c = \text{Sign}_{\text{SK}_c}(m \| t_c)$ 是对客户端请求消息的签名。客户端可将请求广播至全部主节点集合 L , 也可依据策略发送至部分主节点。主节点 L_i 接收到请求后, 首先验证签名 σ_c 的合法性及时间戳有效性。验证通过后, 主节点将请求存入本地缓存池, 并利用函数 $\text{Digest}(m)$ 对应消息的哈希摘要 HD, 主节点随后启动组内共识流程, 将该请求分发至其所在小组 P_i 中的其他节

点, 并记录当前视图编号 v 及当前共识实例标识。

(4) 组内共识

主节点 L_i 作为组内 PBFT 流程的发起者, 向小组内所有节点广播预准备 (pre-prepare) 消息: $\text{pre-prepare}_{\text{local}} = \text{pre-prepare}(v, s, \text{HD}, m, L_i, \sigma_{L_i})$ (16) 其中 s 为当前请求的序列号, σ_{L_i} 为组内主节点 L_i 对函数输入部分内容的签名, $\sigma_{L_i} = \text{Sign}_{\text{SK}_{L_i}}(v \| s \| \text{HD})$ 。所有接收该消息的节点 P_{ij} 将验证消息的正确性, 若消息验证无误, 则广播准备消息:

$$\text{prepare}_{\text{local}} = \text{prepare}(v, s, \text{HD}, P_{ij}, \sigma_{P_{ij}}) \quad (17)$$

当节点收到至少 $2f_{\text{local}}$ 个有效 prepare 消息后, 组内节点开始进入 commit 阶段, 并广播提交消息:

$$\text{commit}_{\text{local}} = \text{commit}(v, s, \text{HD}, P_{ij}, \sigma_{P_{ij}}) \quad (18)$$

当节点收到至少 $2f_{\text{local}} + 1$ 个有效的 commit 消息后, 即认为该请求在组内已达成局部共识, 随后组间节点 L_i 记录局部共识结果并进行组间共识:

$$R_i = \langle m, \text{HD}, v, s, \text{Signatures}_{L_i} \rangle \quad (19)$$

(5) 组间共识

在所有小组完成局部共识后, 各组主节点 $L_i \in L$ 利用局部共识结果 R_i 完成组间共识。首先, 由当前全局视图的主节点 L_{main} 发起组间预备消息:

$$\text{pre-prepare}_{\text{global}} = \text{pre-prepare}(v, s, \text{HD}, R_i, L_{\text{main}}, \sigma_{L_{\text{main}}}) \quad (20)$$

其他主节点收到后验证 R_i 的合法性, 若验证通过, 则返回组间准备消息:

$$\text{prepare}_{\text{global}} = \text{prepare}(v, s, \text{HD}, R_i, L_i, \sigma_{L_i}) \quad (21)$$

当节点收到至少 $2f_{\text{global}}$ 个有效 prepare_{global} 消息后, 会进入 commit 阶段, 并广播提交消息:

$$\text{commit}_{\text{global}} = \text{commit}(v, s, \text{HD}, L_i, \sigma_{L_i}) \quad (22)$$

当节点收到至少 $2f_{\text{global}} + 1$ 个有效的 commit_{global} 消息后, 即认为该请求在全局范围内已达成最终共识, 记录最终共识结果:

$$R_{\text{global}} = \langle m, \text{HD}, v, s, \text{Signatures}_{L'} \rangle, L' \in L, |L'| \geq 2f_G + 1 \quad (23)$$

此结果将被该主节点同步返回其所在小组, 并通知客户端。

(6) 最终确认

当组间共识结果 R_{global} 达成后, 各组主节点 L_i 将最终结果广播给其所在组 P_i 中的所有节点:

$$L_i \rightarrow P_i: \text{Final-Reply}(v, s, R_{\text{global}}, \sigma_{L_i}) \quad (24)$$

组内节点在验证主节点签名后, 将结果记录入本地账本状态, 并生成回复消息发送给客户端:

$$P_{ij} \rightarrow C: \text{Reply}(v, s, R_{\text{global}}, P_{ij}, \sigma_{P_{ij}}) \quad (25)$$

客户端收到来自至少 $f_{\text{global}}+1$ 个不同节点的签名一致回复后, 即可确认该交易请求 m 被全网正式接受。客户端可采用以下条件进行结果验证:

$$\exists S \subseteq P, |S| \geq f_{\text{global}} + 1, \forall P \in S, R_P = R_{\text{global}} \quad (26)$$

此时, 客户端将交易视为不可逆确认, 进入链上状态更新阶段。分组共识机制整体流程, 如图6所示。

3 安全性分析

本文通过以下定理来证明 LPDC 方案的安全性。为形式化证明方案的安全性, 本文假设底层哈希在量子随机预言机模型 (quantum random oracle model, QROM) 下满足单向困难, 任何 PRF 在经典与量子情形下均保持伪随机性, 攻击者可进行任意多项式时间的签名查询, 查询上限记作 q_s , 目标安全度记为 λ 。

定理1 在 QROM 下, 若存在多项式时间伪造器 A, 在至多 q_s 次签名查询后以概率 ε 对 LPDC 构造出对未查询消息的有效签名, 则存在多项式时间算法 B, 使得:

$$\varepsilon \leq L_{\text{opt}} \cdot \text{Adv}_{\text{preimage}}(\text{B}) + \delta(q_s, q_{\text{RO}}) \quad (27)$$

其中, L_{opt} 为哈希链条数, $\text{Adv}_{\text{preimage}}(\text{B})$ 表示 B 成功求解哈希预像的概率, q_{RO} 表示随机预言机查询次数, $\delta(q_s, q_{\text{RO}})$ 是与签名查询数和 QROM 模拟相关的可控模拟失败项。该不等式表明, 若底层哈希的预像攻击在给定资源下被证明是不可行的, 则伪造成功概率 ε 亦可忽略。

证明 为了找出 x 使得 $H(x)=Y$, B 接收一个预像挑战 Y , B 随机选取一个索引 $j^* \in \{0, \dots, L_{\text{opt}} - 1\}$, 并在公钥构造阶段将第 j^* 条链的值作为挑战值 Y , 对其他索引正常生成私钥和对应的公钥。随后 B 模拟签名并回应 A 的查询: 对于每次签名请求, B 使用已知的私钥分量构造签名分量, 只在 A 要求使用第 j^* 个私钥分量时产生模拟困难, 上述事件的失败概率记为 $\delta(q_s, q_{\text{RO}})$ 。当 A 最终输出对某个消息 m^* 的有效伪造签名 σ^* 时, 该签名在验证过程中必须使每条链经过一定次数的哈希迭代才能获取对应的公钥, 因此 σ^* 在索引 j^* 上隐含一个与 Y 相关的链中间态信息, B 可根据此信息逆推出 Y 的预像 x 。由于 j^* 是随机选择的, 因此 B 在 A 成功伪造的条件下提取到预像的概率约为 $1/L_{\text{opt}}$, 由于精确概率需要减去模拟失败的概率, 因此上述不等式成立。

证明中 QROM 用于在模拟过程中编程随机预言机以满足所需的哈希响应, 同时保证模拟失败概率 q_s, q_{RO} 形成上界, LPDC 方案在实际应用过程中限制了 p 的取值, 将局部进位与合并值进

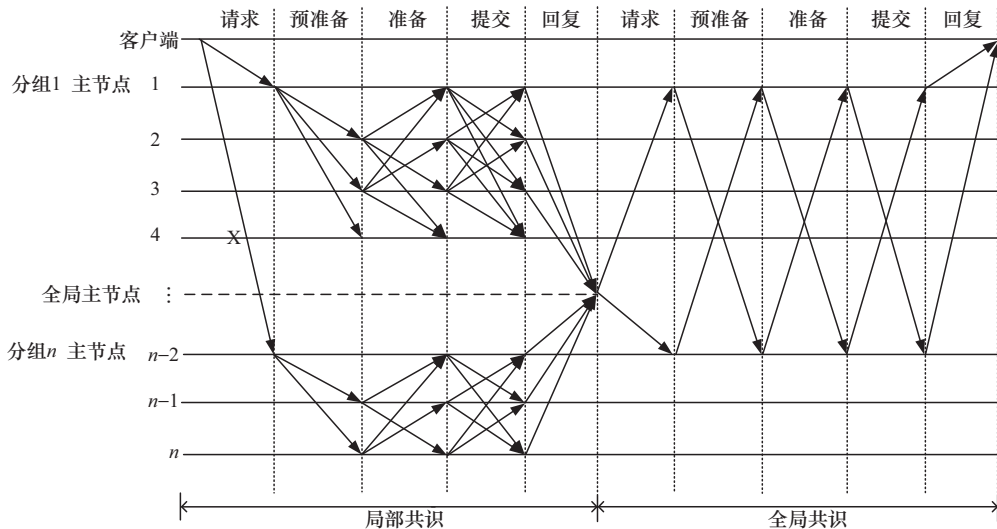


图6 分组共识机制流程

行哈希绑定并在签名中保留局部进位信息以降低哈希碰撞的风险,防止出现不同原始向量映射到相同位置的情形,避免出现可利用的多解空间。同时在后量子设置下利用 Grover 加速挑战哈希输出长度 n 以维持预像难度。基于上述归约,LPDC 方案在不可伪造性方面可被约化为底层哈希函数的预像难度,并在形式上具备与 SPHINCS+ 同级别的抗量子特性。

定理 2 在 G-PBFT 分组共识机制中,若每个子组内的拜占庭节点数不超过 f_{local} ,且全局组间主节点集合中的拜占庭节点数不超过 f_{global} ,则无论是否存在恶意主节点或跨组勾结攻击,系统均能保证安全性,且攻击者无法通过伪造消息、双花攻击或重放攻击破坏系统的一致性和正确性。

证明:在组内共识阶段,G-PBFT 继承了传统 PBFT 的安全性约束条件,即在 n 个节点的子组中若拜占庭节点数 $\leq f_{\text{local}}$,则必然存在至少 $f_{\text{local}} + 1$ 个诚实节点能够完成 prepare 与 commit 流程并共同生成局部共识结果 R_i 。由于消息需要附带数字签名和哈希摘要,恶意组内主节点即便试图伪造请求或篡改内容,也无法获得足够的合法签名以通过多数验证,因此局部共识结果 R_i 具有唯一性和不可抵赖性。在组间共识阶段,各组主节点基于局部共识结果 R_i 运行简化 PBFT 流程。当组间拜占庭主节点数 $\leq f_{\text{global}}$ 时,仍然存在至少 $f_{\text{global}} + 1$ 个诚实主节点参与 prepare_{global} 和 commit_{global} 阶段,因此任一全局共识结果 R_{global} 必然包含多数诚实节点的签名,保证了结果的正确性与一致性。

对于跨组攻击场景,假设若干组主节点联合企图发起双花攻击或构造相互冲突的全局结果,由于全局共识结果需要至少 $2f_{\text{global}} + 1$ 个不同组主节点的签名集合才能生效,攻击者必须控制超过阈值的多数组主节点才能成功伪造 R_{global} ,而这一条件超出了系统设计的容错范围。消息的时间戳和哈希摘要机制可以有效防止重放攻击和消息篡改,从而消除了跨组勾结在信息一致性层面的影响。因此,在恶意主节点单独作恶或多组之间协同攻击的情况下,攻击均无法破坏最终共识的一致性。

针对组间勾结攻击,假设攻击者试图通过控

制多个分组内的部分节点,使若干组主节点在组间共识阶段协同作恶,从而构造冲突的全局共识结果或实施双花攻击。在 G-PBFT 机制中,每个分组需先在组内通过 PBFT 达成唯一且不可抵赖的局部共识结果,该结果必须包含不少于 $2f_{\text{local}} + 1$ 个组内节点的有效签名,因此任何未获得组内多数诚实节点支持的结果均无法进入组间共识阶段。在组间共识过程中,全局决策需由至少 $2f_{\text{global}} + 1$ 个不同分组主节点共同确认,这意味着攻击者必须同时突破多个分组内部和组间两级阈值约束,才能成功推动非法结果生效,该条件在系统参数设计中被明确限制在攻击者能力之外,从而从结构上抑制了跨组勾结攻击的可行性。

针对主节点轮换攻击,即攻击者试图在连续视图中通过操纵主节点选举过程或利用主节点更替窗口实施拒绝服务或一致性破坏,LPDC 通过积分投票与加权随机森林相结合的选举机制,使主节点选择同时受历史行为、实时性能与随机扰动因素约束,攻击者难以在多个周期内稳定控制主节点身份;即便某一轮次中出现恶意主节点,其行为仍需通过组内或组间多数节点的签名验证才能推进共识流程,若其发送冲突消息或拒绝正常广播,将触发视图切换机制并被快速替换,从而保证系统活性与安全性不受持续影响。此外,所有共识消息均绑定唯一序列号、视图编号与哈希摘要,并经抗量子签名认证,能够有效防止重放、篡改及伪造消息在主节点轮换过程中的传播。

综合上述分析可知,在分组规模与容错阈值满足设计假设的条件下,即使面对组间勾结攻击或主节点轮换相关的主动攻击场景,G-PBFT 仍能够保证全局共识结果的唯一性、一致性与不可逆性,从而保证 LPDC 方案在复杂对抗环境下的整体安全性。

4 实验分析

为了测试 LPDC 方案的性能,本文共设计了 4 组对比实验,针对签名方案,本文设计从签名时间与签名大小方面与不同方案进行对比。本文用于比较的方案如下:SPHINCS+[¹¹],SPHINCS+SM3[¹²],WOTS-S[¹³]。对于 PBFT 分组共识机制,

本文设计在共识时延与网络吞吐量方面与PBFT进行比较。针对共识机制，本文用于比较的方案如下：CE-PBFT^[27]、DBFT^[15]、ST-PBFT^[20]。

本文的实验设备配置为：CPU I9 14 900 KS、内存 64 GB，GPU 为 RTX 4070、操作系统为 Ubuntu 20.24，并基于 Go 语言搭建 2.4 版本的 HyperLedger Fabric 区块链实验平台，采用 LevelDB 作为键值数据库。实验数据的交易字段包括：交易 TxID、交易发起者、链码方法 function、链码参数 args、时间戳 timestamp、数字签名等。该交易格式对应典型联盟链资产转移与状态更新场景，符合现实区块链应用场景。

4.1 签名时间测试

针对本文签名方案和 SPHINCS+ 方案在签名时间方面的测试，统一设置安全参数 n 为 128 位， $w=16$ ， $k=33$ ， $t=64$ 。分别在不同交易数量条件下测量签名结构的生成时间、签名验证时间以及二者的总耗时。不同交易数量下的总时间开销如图 7 所示。

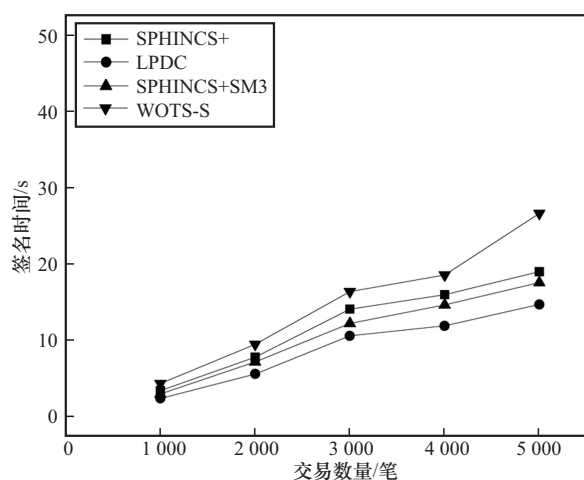


图7 不同交易数量下的总时间开销

上述实验结果表明，在不同交易数量下，LPDC 在签名时间开销上均表现优异，相较于 SPHINCS+，签名时间降低幅度约为 33%；相较于 SPHINCS+SM3，签名时间降低幅度约为 21%；相较于 WOTS-S，总时间开销则降低了约 67%。

造成上述实验结果的主要原因是 LPDC 在 WOTS+ 结构中通过哈希链合并操作，压缩了哈希迭代次数，从而降低每笔交易的签名计算量。相比之下，传统 SPHINCS+ 方案固定使用多棵

FORS 子树，每棵子树的叶子节点和认证路径均需完整计算，即使部分叶子未被使用，也会产生额外计算开销，导致签名时间显著增加。

SPHINCS+SM3 方案虽然采用 SM3 哈希函数提高单次哈希效率，但其 FORS 和 WOTS+ 结构仍沿用原始设计，缺乏跨交易批量优化机制，因此在多笔交易场景下签名时间仍保持较高水平。而 WOTS-S 方案虽然在签名体积压缩上具备一定的优势，但每条哈希链仍需完整迭代计算，且没有引入任何索引聚合或链条合并机制，因此在批量交易处理下签名时间明显高于 LPDC。

4.2 签名大小测试

FORS 结构和 WOTS+ 结构是 SPHINCS+ 方案的签名结构之一，其大小是衡量 SPHINCS+ 抗量子签名方案的重要参考标准，并且会直接影响签名的效率。为评估 LPDC 方案在 FORS 结构和 WOTS+ 结构上的优化效果，本文选取不同交易规模下生成的签名大小作为评估指标，对比两种方案在签名处理效率方面的差异。

实验共设置 5 000 笔交易，每完成 1 000 次交易统计一次历史签名结构的大小信息，实验结果如图 8 所示。

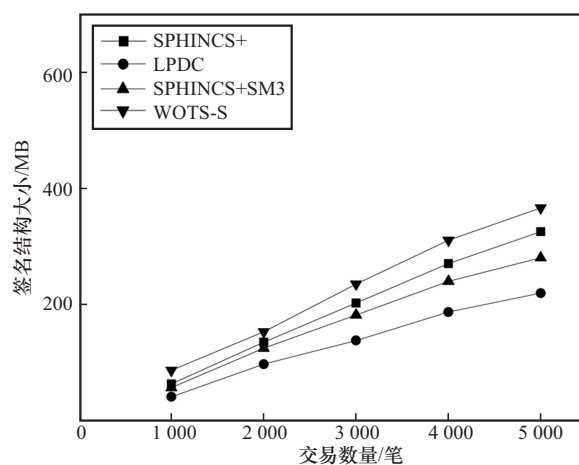


图8 不同交易数量下的签名结构大小对比

从实验结果可以看出，LPDC 在不同交易数量下的签名大小始终低于其他方案。在 5 000 笔交易的场景下，LPDC 签名总大小仅为 219.6 MB，而 SPHINCS+、SPHINCS+SM3 和 WOTS-S 分别达到 325.3 MB、280.2 MB 和 365.8 MB。且随着交易数量增加，所有方案的签名大小均呈增长

趋势, 但 LPDC 的签名结构明显较低, 表示该方案在大规模交易场景下具备更高的存储效率。

造成上述结果的主要原因是 LPDC 在 FORS 结构上的优化。在 FORS 结构中, LPDC 通过叶子索引聚合和按需构造策略, 将原本分布在多棵子树中的叶子节点映射到较少的子树上, 仅生成必要的认证路径, 从而有效减少了冗余节点的存储开销。

相比之下, SPHINCS+ 固定采用 k 棵 FORS 子树和完整的 WOTS+ 哈希链, 即便许多叶子节点未被使用, 也需生成完整认证路径, 导致签名大小随交易数量线性增加。SPHINCS+SM3 虽然优化了哈希计算效率, 但子树数量和哈希链结构未做改变, 因此签名大小依旧较大。WOTS-S 由于采用多路径认证, 每个叶子节点额外增加多条认证路径, 存储开销的增加最为显著, 尤其在交易量大时, 签名大小增长最快。

4.3 共识时延测试

共识时延是衡量区块链共识协议性能的重要指标之一, 为验证所提 LPDC 方案中的 G-PBFT 机制在共识效率方面的优势, 本文设计了共识时延对比实验, 分别在不同节点规模下 (节点数量从 5 个扩展至 100 个) 进行测试, 并记录在每种规模下完成一次完整共识流程所需的时间。实验环境在本地搭建的 fabric 区块链网络环境下构建, 每个节点均运行一致的共识逻辑与通信接口, 确保测试过程中不存在性能瓶颈或网络拥塞等额外干扰, 实验结果如图 9 所示。

实验结果表明, 在节点规模从 5 个扩展至

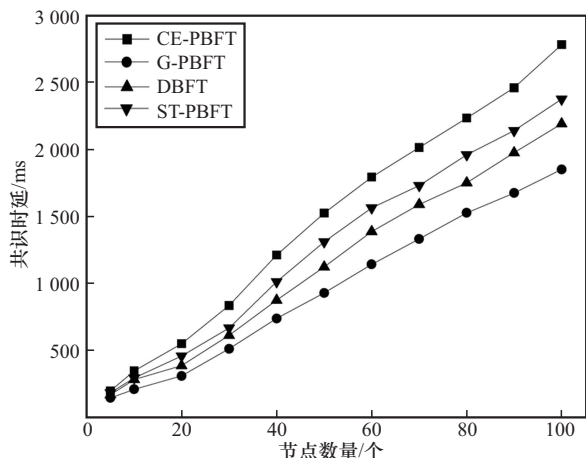


图9 共识时延对比

100 个的测试中, 4 种共识机制的时延均随节点数量的增加而上升, 但 G-PBFT 在所有规模下的时延最低。例如, 在 100 个节点场景中, G-PBFT 的共识时延为 1 854.8 ms, 而 CE-PBFT、DBFT 和 ST-PBFT 的共识时延分别为 2 784.5 ms、2 194.3 ms 和 2 377.9 ms。

造成上述结果差异的根本原因在于 G-PBFT 将全网节点划分为多个小组, 仅组内节点参与完整的消息广播与投票流程, 组间通过代表节点进行结果汇总与确认。这种结构有效减少了全网广播的消息总量, 从而显著缩短了节点规模较大时的消息传递和处理时间。

相比之下, DBFT 虽引入部分通信优化, 但在投票与确认阶段依旧维持全网消息交互, 因此高节点规模下时延依然较大; ST-PBFT 在安全性上增加了额外的状态一致性检查与确认轮次, 这在小规模节点下影响不大, 但在节点规模扩大时, 便会显著增加共识时延。CE-PBFT 虽然简化了一致性协议并优化了通信开销, 但随着节点的不断拓展, 该方案仍然需要所有节点进行全网广播和签名验证, 造成通信复杂度升高, 时延急剧上升;

4.4 共识吞吐量测试

共识吞吐量是衡量区块链系统处理能力的重要指标, 通常以每秒可完成的交易数量 (transactions per second, TPS) 表示, 反映了系统在单位时间内完成共识并成功写入区块的交易数。较高的吞吐量意味着系统具备更强的交易处理能力, 对于高频交易场景与大规模部署的区块链平台具有关键意义。计算 TPS 的公式如式 (19) 所示, 其中 transaction 为 ΔT 时间内打包进入区块的交易数, ΔT 为区块的出块时间。

$$TPS = \frac{\text{transaction}}{\Delta T} \quad (28)$$

实验利用客户端向节点发送 1 000 次交易。节点初始数为 5 个, 逐渐递增到 100 个。最终实验结果如图 10 所示。

实验结果显示, 随着节点数量的增加, 各共识算法的吞吐量均呈下降趋势, 但 G-PBFT 的吞吐量表现相对较好。具体来看, 当节点数为 10 个时, 其他方案的平均吞吐量为 622 TPS, 而 G-PBFT 的吞吐量达到 717.3 TPS, 表明 G-PBFT 相

比其他方案具有明显优势。

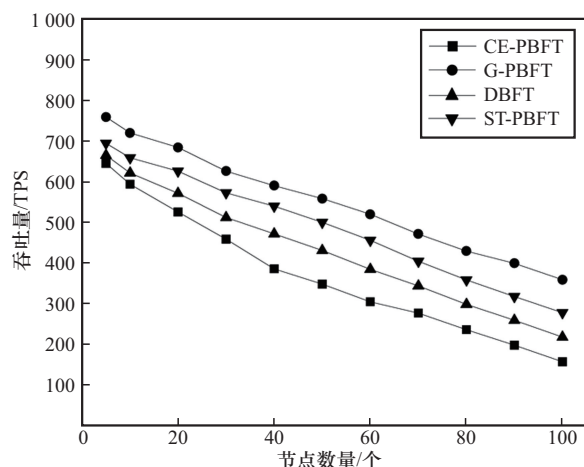


图10 共识吞吐量对比

造成上述实验结果的主要原因在于各算法在通信复杂度与共识流程上的差异。G-PBFT通过将节点划分为多个组进行局部共识，并在组间仅进行主节点汇总的方式，能够显著降低全网通信轮数，从而减少网络负载与时延，保证较高的吞吐量。而DBFT结合DPoS的投票选举机制，虽然在通信过程中可减少部分通信量，但其跨组节点投票仍引入了额外时延，导致吞吐量低于G-PBFT。而ST-PBFT在节点同步过程中需进行多轮全网同步，随着节点数量增加，节点之间同步所需的通信时延和验证开销逐渐累积，从而影响吞吐量。CE-PBFT在节点数量增加时，需要进行全网广播，导致其通信开销迅速增加，造成吞吐量显著下降。

综上所述，G-PBFT的分组共识设计在节点规模扩展时能够有效提升系统吞吐量，而CE-PBFT、DBFT和ST-PBFT在大规模节点网络下受到通信瓶颈限制，吞吐量下降较为明显。

4.5 最优分组数测试

本文提出的分组共识机制优点在于能够将通信复杂度降低为 $O\left(\frac{N^2}{G} + G^2\right)$ ，但降低程度取决于分组

G 的数量，为探究不同分组数对于系统吞吐量的影响，方案利用HyperLedger Fabric网络，分别搭建不同数量的区块链节点，分别为：10个、20个、30个、40个、50个，并设置1 000笔交易，针对不同分组的系统吞吐量进行测试，测试结果如图11所示。

从上述测试结果能够得出以下结论：相同数

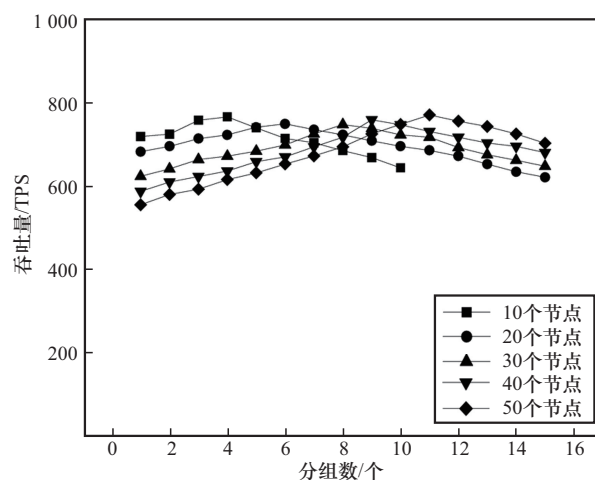


图11 最优分组数对比

量的节点下，分组数对于吞吐量有着一定的影响，随着分组数越来越多，系统吞吐量均呈现先上升再下降的趋势，且不是分组数越多，吞吐量越高，说明不合理的分组数反而会降低系统的吞吐量。随着系统节点数的增加，合理的分组能够显著提升系统整体吞吐量。测试发现，不同的节

点数，合理的分组数均趋向于 $\left(\frac{N^2}{2}\right)^{\frac{1}{3}}$ ，超过此阈值，则会降低系统吞吐量。

4.6 共识参与率测试

本文的主节点选举机制在于将明确、可审计的积分机制与随机森林模型评估进行结合来选举主节点。该设计既保留了对诚实贡献节点的激励与保护，又通过模型稳健性显著提升了对人为干预与拜占庭扰动的抗性；优良的选举机制能够极大提升区块链网络的共识效率，为了测试该机制的有效性，实验选取了30个网络节点，并将节点的带宽、网络时延、在线时长、打包率等作为积分与决策模型的评估标准，通过上述条件，测试10个周期内的节点共识参与率，并进行20次测试，选取平均值作为最终实验结果。共识参与率对比如图12所示。

基于上述实验结果可以得出以下结论：将积分激励机制与加权随机森林结合的主节点选举策略显著提高了节点共识参与率并展现出较好的稳定性，G-PBFT的共识参与率稳定居于80%左右，相对其他方案提升了约5%，表明该方案能够有效激励节点参与网络共识，并在平均参与率

与鲁棒性之间取得平衡。

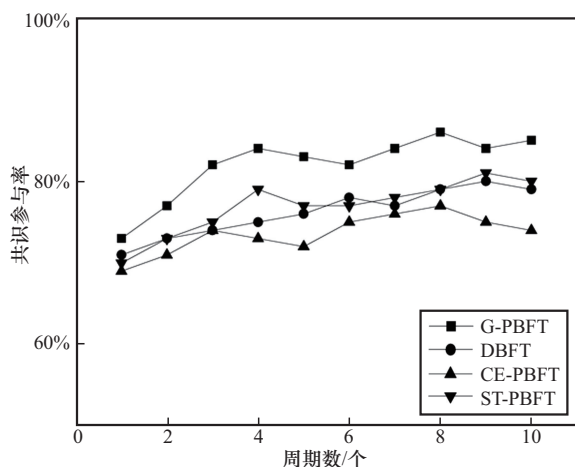


图 12 共识参与率对比

5 结束语

本文针对当前区块链系统在抗量子签名方案存储开销大、效率低、PBFT 共识机制吞吐量受限等方面的问题,提出了一种面向区块链的轻量级抗量子分布式共识方案。该方案首先对 SPHINCS+抗量子签名方案的 FORS、WOTS+结构分别进行优化,显著降低了存储开销并提升了算法效率。其次,设计了 G-PBFT 共识机制与主节点选举机制,显著减少共识时延的同时提升了系统吞吐量并保证区块链网络的活跃性。实验结果表明,该方案在存储效率、共识性能方面均优于传统设计,具备良好的实际应用潜力。未来的工作将聚焦于挖掘该方案在多元化应用场景中的实践潜力,进一步优化系统的扩展能力与容错机制,以应对区块链网络规模持续扩大所带来的挑战。

参考文献:

- [1] Gharavi H, Granjal J, Monteiro E. Post-quantum blockchain security for the internet of things: survey and research directions[J]. IEEE Communications Surveys & Tutorials, 2024, 26(3): 1748-1774.
- [2] Hossain Faruk M J, Tahora S, Tasnim M, et al. A review of quantum cybersecurity: threats, risks and opportunities[C]//Proceedings of the 2022 1st International Conference on AI in Cybersecurity (ICAIC). Piscataway: IEEE Press, 2022: 1-8.
- [3] 张振琦, 朱秋池, 王志伟. 带鲁棒性的子分组多重 Schnorr 签名方案[J]. 网络与信息安全学报, 2024, 10(4): 132-142.

Zhang Z Q, Zhu Q C, Wang Z W. Robust schnorr-based subgroup

multi-signature scheme[J]. Chinese Journal of Network and Information Security, 2024, 10(4): 132-142.

- [4] 李继国, 方淳. 基于 SM9 的指定验证者聚合签名方案[J]. 网络与信息安全学报, 2024, 10(4): 63-71.
- [5] Li J G, Fang C. Designated verifier aggregate signature scheme based on SM9[J]. Chinese Journal of Network and Information Security, 2024, 10(4): 63-71.
- [6] Diffie W, Hellman M. New directions in cryptography[J]. IEEE Transactions on Information Theory, 1976, 22(6): 644-654.
- [7] Lamport L. Constructing digital signatures from a one way function[R]. 1979.
- [8] Merkle R C. Secrecy, authentication, and public key systems[D]. Stanford: Stanford University, 1979.
- [9] Merkle R C. A certified digital signature[C]//Advances in Cryptology-CRYPTO' 89 Proceedings. New York, NY: Springer, 1990: 218-238.
- [10] Reyzin L, Reyzin N. Better than BiBa: short one-time signatures with fast signing and verifying[C]//Information Security and Privacy. Berlin, Heidelberg: Springer, 2002: 144-153.
- [11] Bernstein D J, Hopwood D, Hülsing A, et al. SPHINCS: practical stateless hash-based signatures[C]//Advances in Cryptology -- EUROCRYPT 2015. Berlin, Heidelberg: Springer, 2015: 368-397.
- [12] Bernstein D J, Hülsing A, Kölbl S, et al. The SPHINCS+ signature framework[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2019: 2129-2146.
- [13] Wang X A, Liu H, Jiang W W, et al. A SM3 hash-based post-quantum signature scheme and its application to food source authentication[J]. IEEE Internet of Things Journal, 2025, 12(23): 49233-49247.
- [14] Shahid F, Khan A, Malik S U R, et al. WOTS-S: a quantum secure compact signature scheme for distributed ledger[J]. Information Sciences, 2020, 539: 229-249.
- [15] Zhang K Y, Cui H R, Yu Y. SPHINCS- α : A compact stateless hash-based signature scheme[Z]. 2022.
- [16] Crain T, Gramoli V, Larrea M, et al. DBFT: efficient leaderless Byzantine consensus and its application to blockchains[C]//Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA). Piscataway: IEEE Press, 2018: 1-8.
- [17] Sun Y Y, Yan B W, Yao Y, et al. DT-DPoS: a delegated proof of stake consensus algorithm with dynamic trust[J]. Procedia Computer Science, 2021, 187: 371-376.
- [18] Fu J H, Zhou W H, Xu M X, et al. New public blockchain protocol based on sharding and aggregate signatures[J]. Journal of High Speed Networks, 2021, 27(1): 83-99.
- [19] 袁昊天, 李飞. 基于改进 Raft 共识算法和 PBFT 共识算法的双层共识算法[J]. 计算机应用研究, 2024, 41(5): 1314-1320.
- [20] Yuan H T, Li F. Double layer consensus algorithm based on improved raft consensus algorithm and PBFT[J]. Application Research of Computers, 2024, 41(5): 1314-1320.
- [21] Zhu J, Lu C F, Li J J, et al. Secure consensus control on multi-agent

- systems based on improved PBFT and raft blockchain consensus algorithms[J]. IEEE/CAA Journal of Automatica Sinica, 2025, 12(7): 1407-1417.
- [20] Zhong W, Feng W L, Huang M X, et al. ST-PBFT: an optimized PBFT consensus algorithm for intellectual property transaction scenarios[J]. Electronics, 2023, 12(2): 325.
- [21] Ji Y L, Zou Y, Wang G, et al. An improved PBFT consensus algorithm based on reputation grading and voting in IoT environment[C]//Proceedings of the 2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA). Piscataway: IEEE Press, 2025: 2121-2128.
- [22] Mu Z Q, Liu Y Y, Yang Y L. A large-scale group decision making model with a clustering algorithm based on a locality sensitive hash function[J]. Engineering Applications of Artificial Intelligence, 2025, 140: 109697.
- [23] Ding J W, Wu X, Tian J, et al. RE-BPFT: an improved PBFT consensus algorithm for consortium blockchain based on node credibility and ID3-based classification[J]. Applied Sciences, 2025, 15(13): 7591.
- [24] Chang Y X, Wang Q, Li Q L, et al. Performance and reliability analysis for PBFT-based blockchain systems with repairable voting nodes[J]. IEEE Transactions on Network and Service Management, 2024, 21(4): 4039-4060.
- [25] Yao Z Y, Fang Y H, Pan H, et al. A secure and highly efficient blockchain PBFT consensus algorithm for microgrid power trading[J]. Scientific Reports, 2024, 14: 8300.
- [26] Liu X, Fan X, Niu B N, et al. 5G-practical Byzantine fault tolerance: an improved PBFT consensus algorithm for the 5G network[J]. Information, 2025, 16(3): 202.
- [27] Xiao J, Luo T, Li C Q, et al. CE-PBFT: a high availability consensus algorithm for large-scale consortium blockchain[J]. Journal of King Saud University - Computer and Information Sciences, 2024, 36(2): 101957.

[作者简介]



张德俊（2001-），男，北京邮电大学硕士生，主要研究方向为区块链技术、网络空间安全。



李剑（1976-），男，博士，北京邮电大学教授、博士生导师，主要研究方向为量子密码、区块链技术、人工智能安全、网络空间安全等。



王乃乐（2002-），女，北京邮电大学硕士生，主要研究方向为人工智能安全、网络空间安全等。



张春晖（2002-），男，北京邮电大学硕士生，主要研究方向为人工智能安全、网络空间安全等。