Computer Science and Application 计算机科学与应用, 2026, 16(9), 85-99
Published Online September 2026 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2026.169291

Hans汉斯

物联网环境下基于SipHash代理签名的PBFT算法优化

肖　肖[1*]，邹青宏[1]，吉彦霖[2]，马海军[1]，邓宇晨[1]

[1]重庆外语外事学院智能科学与工程学院，重庆
[2]航空工业四川泛华航空仪表电器有限公司，四川 成都

收稿日期：2026年8月2日；录用日期：2026年9月1日；发布日期：2026年9月9日

摘　要

在当前物联网环境中，设备性能与能耗差异显著，加剧了分布式系统一致性问题的复杂性。区块链凭借去中心化共识与可信特性，成为应对该挑战的关键手段。然而，物联网设备常因资源受限、网络波动或电力中断而频繁离线，导致PBFT共识易失败，需频繁重启，从而增加通信开销并降低系统吞吐量。针对上述问题，本文提出一种基于SipHash代理签名的SH-PBFT共识算法，对传统PBFT进行改进。该算法首先设计了一种代理签名节点选择机制，使每个节点均可自主指定代理节点，在自身离线时代为执行签名操作，从而实现代理任务的均衡分配，避免因任务过度集中而导致的性能瓶颈，并构建起一条多级代理签名链，有效提升系统稳定性。其次，本文构建了完整的代理签名模型，涵盖高效安全的签名生成与验证流程，确保签名的有效性、安全性与可追溯性，同时优化了在存在离线节点场景下的共识执行方式，使系统在部分节点离线时仍可完成共识过程，避免因节点失效而触发频繁重启，从而显著降低额外通信时延与开销。最后，仿真实验结果表明，SH-PBFT算法在签名任务分布、系统吞吐量、响应时延及通信开销等方面均表现优良，尤其在节点离线情况下，仍能保持良好的整体性能。

关键词

物联网，区块链，PBFT，共识算法，代理签名

Optimization of PBFT Algorithm Based on SipHash Proxy Signature in the Internet of Things Environment

Xiao Xiao[1*], Qinghong Zou[1], Yanlin Ji[2], Haijun Ma[1], Yuchen Deng[1]

[1]College of Intelligent Science and Engineering, Chongqing Institute of Foreign Studies, Chongqing
[2]Sichuan Panhua Aviation Instrument Electric Co., Ltd., Chengdu Sichuan

Received: August 2, 2026; accepted: September 1, 2026; published: September 9, 2026

[*]通讯作者。

文章引用: 肖肖, 邹青宏, 吉彦霖, 马海军, 邓宇晨. 物联网环境下基于 SipHash 代理签名的 PBFT 算法优化[J]. 计算机科学与应用, 2026, 16(9): 85-99. DOI: 10.12677/csa.2026.169291

Abstract

In the current Internet of Things environment, there are significant differences in device performance and energy consumption, which exacerbate the complexity of the consistency problem in distributed systems. Blockchain, with its decentralized consensus and trustworthiness features, has become a key solution to this challenge. However, IoT devices often frequently go offline due to resource constraints, network fluctuations, or power outages, leading to the failure of PBFT consensus and frequent restarts, thereby increasing communication overhead and reducing system throughput. To address these issues, this paper proposes an SH-PBFT consensus algorithm based on SipHash proxy signature, which improves the traditional PBFT. The algorithm first designs a proxy signature node selection mechanism, allowing each node to autonomously specify a proxy node to perform the signature operation during its offline period, thereby achieving the balanced distribution of proxy tasks and avoiding performance bottlenecks caused by excessive concentration of tasks. It also builds a multi-level proxy signature chain to effectively enhance system stability. Secondly, this paper constructs a complete proxy signature model, covering efficient and secure signature generation and verification processes, ensuring the validity, security, and traceability of signatures, while optimizing the consensus execution method in the presence of offline nodes, enabling the system to complete the consensus process even when some nodes are offline, avoiding frequent restarts due to node failures, and significantly reducing additional communication delay and overhead. Finally, simulation results show that the SH-PBFT algorithm performs well in terms of signature task distribution, system throughput, response time, and communication overhead, especially in the case of node offline, maintaining good overall performance.

Keywords

Internet of Things, Blockchain, PBFT, Consensus Algorithm, Proxy Signature

1. 介绍

物联网将传感器、嵌入式系统、网络技术以及数据处理技术有机地结合在一起，使得各种物理设备、机器和物品能够相互连接和交流，从而实现了设备之间的智能化沟通和协同工作[1] [2]。然而在实际应用中面临着安全性与隐私保护、数据共享与开放性、网络拓扑与通信协议等方面的技术难题[3]，这些问题都制约了物联网技术的进一步发展和应用。区块链技术[4]为克服这些技术难题与局限性提供了有效途径。区块链[5]融合了 P2P 网络协议[6]、共识算法、非对称加密等[7]-[10]技术。在无需借助可信第三方的情况下，实现互不信任的多方对等可信的价值传输。

共识算法作为区块链技术的核心，在很大程度上决定了区块链系统的效率和可扩展性[11]，现有的共识算法大致可以为 Proof of X (PoX)共识算法[5] [12] [13]和拜占庭容错(Byzantine Fault Tolerant, BFT)共识算法[14]。PoX 共识算法通常适用于公有链的场景，通过增加服务请求提案的成本、在链上投入一定代币、放宽最终一致性确认的需求来达成共识[15]。Raft (Replication and Fault Tolerant) [16]分布式共识算法主要应用于私有链的场景来达成共识。实用拜占庭容错(Practical Byzantine Fault Tolerant, PBFT) [17]和 HoneyBadgerBFT [18]等 BFT 算法通常用于联盟的场景。

在物联网环境中，设备普遍受限于能耗与计算资源，加之网络波动、节点故障等因素，导致节点在共识过程中易意外离线甚至彻底失去响应。在此类场景下，采用 PBFT 算法进行共识面临较高的失败风险，往往需要重启整个共识流程。重启过程中，系统需重新组织节点间的信息交互，不仅带来显著的额外通信开销，也进一步加重了网络负担，从而对系统的整体吞吐量与稳定性造成不利影响[19]。

因此，本文提出了一种基于 SipHash 代理签名的 PBFT 算法优化对当前存在的问题进行优化。本文的主要贡献如下：

(1) 本文提出了一种代理签名节点选择机制，使每个节点均可自主指定代理节点，在自身离线时代为执行签名操作，从而实现代理任务的均衡分配。

(2) 本文提出了多级代理签名链的方法，在某些代理设备无法继续工作时，通过下一个代理设备接管签名任务，防止了代理设备离线而导致代理签名设计失效的问题，从而增强系统的容错性和可靠性。

(3) 本文构建了完整的代理签名模型，涵盖高效安全的签名生成与验证流程，确保签名的有效性、安全性与可追溯性，同时优化了在存在离线节点场景下的共识执行方式，使系统在部分节点离线时仍可完成共识过程，避免因节点失效而触发频繁重启，从而显著降低额外通信时延与开销。

实验表明，本文提出的算法签名任务分布、系统吞吐量、响应时延及通信开销等方面均表现优良，尤其在节点离线情况下，仍能保持良好的整体性能。

本文其余部分的结构如下。第二节回顾了近期关于 PBFT 算法的相关研究；第三节阐述了 SH-PBFT 算法的设计过程；第四节对所提共识算法进行了实验模拟与性能分析；第五节对全文工作进行总结。

2. 相关工作

许多学者都对 PBFT 进行了不同的改进和优化。一些研究者通过优化网络节点的结构以此达到降低通信开销的目的。Jiang 等人[20]提出了一种基于集群信誉的分层模型，通过声誉和波动等级将节点分配到上层或下层，降低了区块链的通信复杂度。Li 等人[21]提出了一种最优的双层 PBFT，但是 1/3 容错分布在每一个组中，现实中很难满足该条件。对此，Zhao 等人[22]在双层 PBFT[21]基础上进行改进，使用一个目标函数找到通信复杂度最小的分层结构，同时保持 1/3 拜占庭容错能力。Qushtom 等人[23]提出了一种多层多入口的 PBFT 共识算法，可以并发执行多个共识并保证提交数据的完整性。

一些研究者则建议利用可信赖的第三方来减少共识节点的数量，以提高系统的性能。Kapitza 等人[24]提出了一种 CheapBFT 算法，Veronese 等人[25]提出了一种 MinBFT 算法，两者算法都在利用可信任组件的基础上将参与共识的节点数量从 3f+1 减少到了 2f+1，其中 f 表示拜占庭节点数量。Singh 等人[26]则在这两者的基础上提出了 IBFT 算法，通过 f + 1 轮广播，在 2f + 1 个节点内达成共识。Decouchant 等人提出了 DAMYSUS [27]算法，通过 Checker 和 Accumulator 服务减少通信轮次，提高算法效率。

另外，Yu 等人[28]提出了一种基于双管理员短群签名的 GPBFT 算法，该算法将所有共识成员分为多个群，群中的任意一个节点完成的消息的确认就可代表该群进行共识完成的确认，并通过特定的共识流程确保其安全性。Wang 等人[29]提出了一种改进的匿名和代理的 PBFT 模型，利用可链接的环签名技术隐藏主节点身份，增加了 PBFT 的安全性，并且代理节点可以帮助离线节点完成共识任务。TANG 等人[30]提出了一种具有阈值代理签名的双层 TP-PBFT 共识算法，分层共识减少了通信开销，阈值代理签名机制能为离线节点进行代理签名，完成共识。Ji 等人[31]提出了一种 GV-PBFT 算法，提出了一种节点综合信誉值计算模型以及一种轻量化的组间共识，有效提高了整体共识效率。

从现有研究来看，基于信誉模型对节点进行筛选与分组、减少参与共识的节点数量，已被证实是提升共识效率的有效途径。然而，鲜有研究关注物联网环境中节点离线情形下的共识问题。为此，本文提出一种适用于物联网离线场景的共识算法，通过基于 SipHash 的多级代理签名机制，有效保障了节点离

线状态下的共识效率与系统可用性。

3. SH-PBFT 算法设计方案

在本节中，本文首先介绍了 SH-PBFT 的整体框架流程，然后详细介绍了代理签名。

3.1. SH-PBFT 概述

基于 SipHash 代理签名共识优化的整体流程如图 1 所示。其设计旨在提升物联网网络在节点离线情况下的容错能力和共识效率。以下是对该流程的详细描述。

首先，所有的节点仍然按照第三章所述的综合信誉值模型计算自己的信誉评分，然后通过代理节点选择算法选择自己的代理签名节点。

代理节点选择时节点会构建一个自身的多级代理签名链，并且生成授权证书将其发送给代理节点证明其有权进行代理签名，例如节点 A 选择 B 作为自己的一级代理节点，节点 B 选择节点 C 作为二级代理节点，形成 $A \to B \to C$ 的多级代理签名链，每层都将附带授权证书。随后进行共识。如在共识流程中无离线节点，则共识完成，更新信誉值和多级代理签名链。

如在共识中存在部分节点离线，则主节点无法按时收到节点消息，由主节点广播离线节点的相关信息，然后代理节点为该离线节点进行代理任务，生成代理签名，随后将其发送给主节点，验证该代理签名的信息是否通过，验证过程包括检查签名是否符合离线节点的授权，以及代理节点是否在多级代理签名链中具有合法地位。如果代理节点离线，则由二级代理节点完成代理任务。其余的节点通过存在离线情况下的共识流程进行共识，通过后则完成共识，如验证不通过，需要重启共识。

3.2. 代理签名

3.2.1. 代理节点选择

在代理签名节点选择中，目标是选择一个既有较高信誉值，又尽可能接近设备的设备作为代理签名者。此外，为了防止某个节点因承担过多的代理签名任务而导致负载过重，必须合理分配任务，确保系统的稳定性和效率。因此，本研究提出了一种基于 Dijkstra 算法的综合成本函数，该函数同时考虑了设备的信誉值、距离以及代理签名任务数量，以此来选择最优的代理设备。图 2 展示了该问题的图模型。其中每个节点代表一个设备，每条边表示设备之间的距离或连接质量，边的权重则结合了设备间的网络延迟和设备的信誉评分。此外，考虑到负载均衡问题，节点的当前代理签名任务数量也被纳入成本函数的计算中。通过 Dijkstra 算法，能够在综合评估这些因素后，选择出最适合作为代理签名者的设备，从而实现任务分配的优化。

设所有设备集合为 $N = \{A, B, C, D, \cdots\}$，A 为源设备，B，C 和 D 为可能的代理设备。d_{xy} 表示设备 x 和设备 y 之间的距离。每个设备 x 都有一个信誉值 C_x，较高的 C_x 表示更高的信誉，由第三章的综合信誉值模型得出。T_x 为设备 x 当前的签名任务数量，任务数量越多，设备的负载越大，处理新任务的能力就越差。由此定义一个综合成本函数如公式(1)所示。

$$Cost(x, y) = \alpha \cdot d_{xy} + \beta \cdot C_y + \gamma \cdot T_y \tag{1}$$

其中，α, β, γ 为权重因子，满足 $\alpha + \beta + \gamma = 1$，控制距离和信誉以及代理签名任务数量的相对重要性。较高的 γ 会让系统更加倾向于选择任务较少的设备。

本节主要考虑代理节点选择策略的执行，通过 Dijkstra 算法综合评估信誉值、距离以及代理签名任务数量，选择出适合的代理设备，算法 1 给出了详细过程。

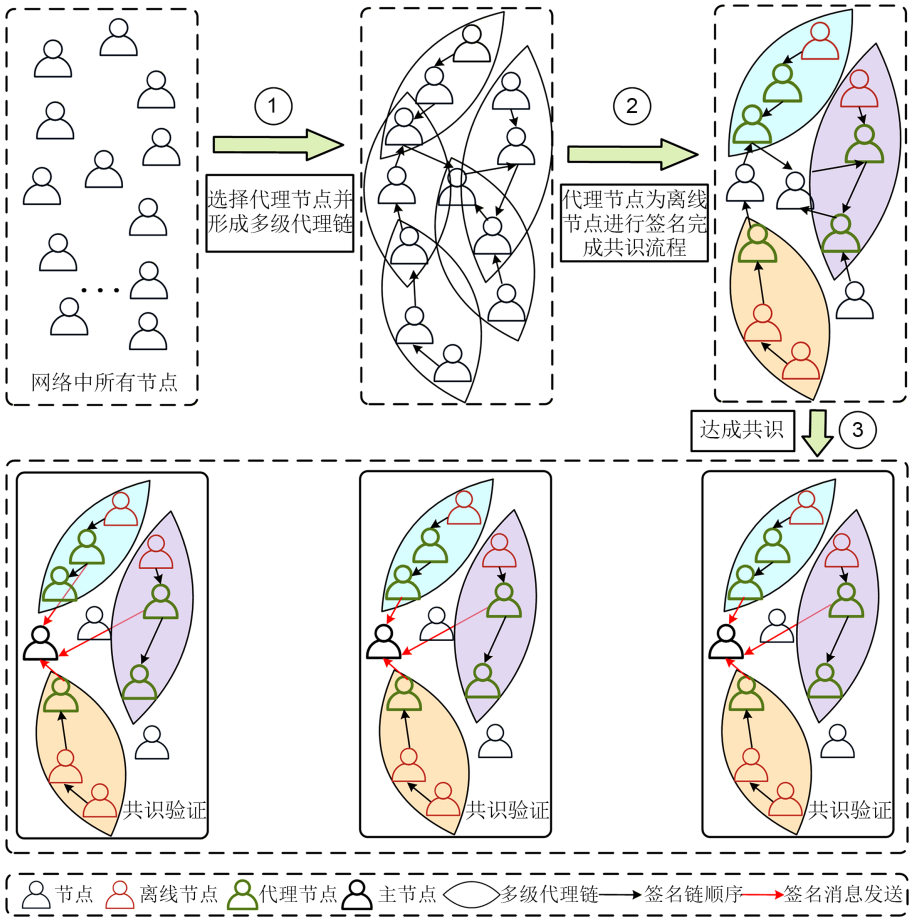


Figure 1. Proxy signature consensus scheme
图 1. 代理签名共识方案

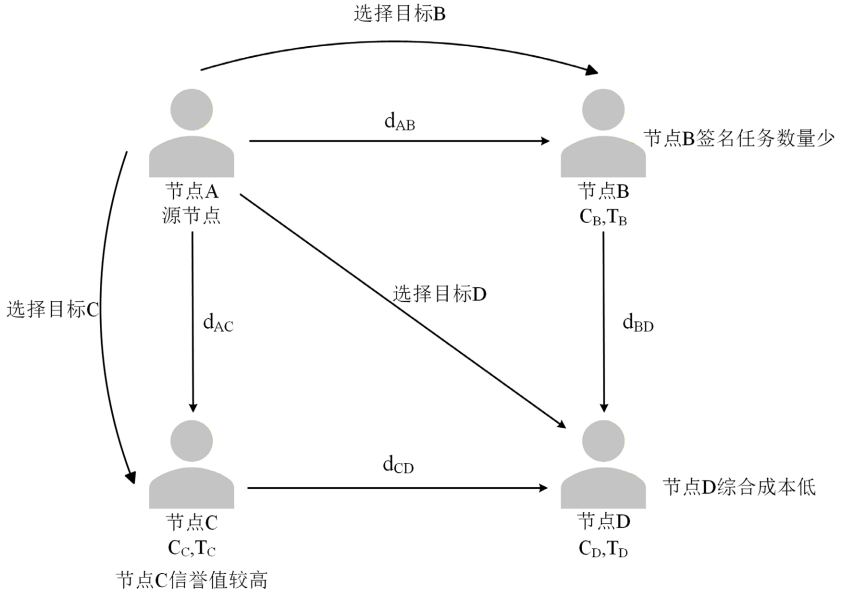


Figure 2. Modeling the proxy signature node selection
图 2. 代理签名节点选择建模

首先，初始化图，设置该节点为源节点，得到每个节点的信誉值、延迟以及其自身的代理签名任务数量，然后使用公式(1)计算与其他设备的综合成本 $Cost(x,y)$，再通过 Dijkstra 算法选择出合适的代理设备，如果设备 D 不可用(例如设备 D 离线)，则使用相同步骤选择其他设备。

算法 1 基于 Dijkstra 的代理节点选择算法

1: **输入：**设备集合 $N=\{A,B,C,D,\cdots\}$，网络图 G，权重因子 α,β,γ

2: **输出：**选定的代理设备

3: 初始化优先队列，设源设备为 A

4: 将所有距离 A 设备的距离初始化为无穷大，A 到 A 的距离为 0

5: **for** i **in** N **do**

6: 计算 $Cost(A,i)=\alpha\cdot d_{Ai}+\beta\cdot C_i+\gamma\cdot T_i$

7: 将设备 i 及其代价加入优先队列

8: **end for**

9: 运行 Dijkstra 算法，计算设备 A 到所有其他设备的最短路径

10: 选择代价最小的设备 D 作为代理设备

11: **if** 设备 D 不可用（离线）**do**

12: 使用相同算法，选择下一个设备 B 或 C 作为代理设备

13：**end if**

3.2.2. 代理签名链

代理签名链设计通过引入多个代理层级，可以在某些代理设备无法继续工作时，通过下一个代理设备接管签名任务，防止了代理设备离线而导致代理签名设计失效的问题，从而增强系统的容错性和可靠性。代理签名链的多级结构增强了签名过程的安全性。每个代理签名设备不仅要验证自己的授权证书，还要验证前一层代理的签名及其授权证书，这样就形成了一条有效的签名链，防止了单一设备滥用签名权限。代理签名链的架构如图 3 所示。

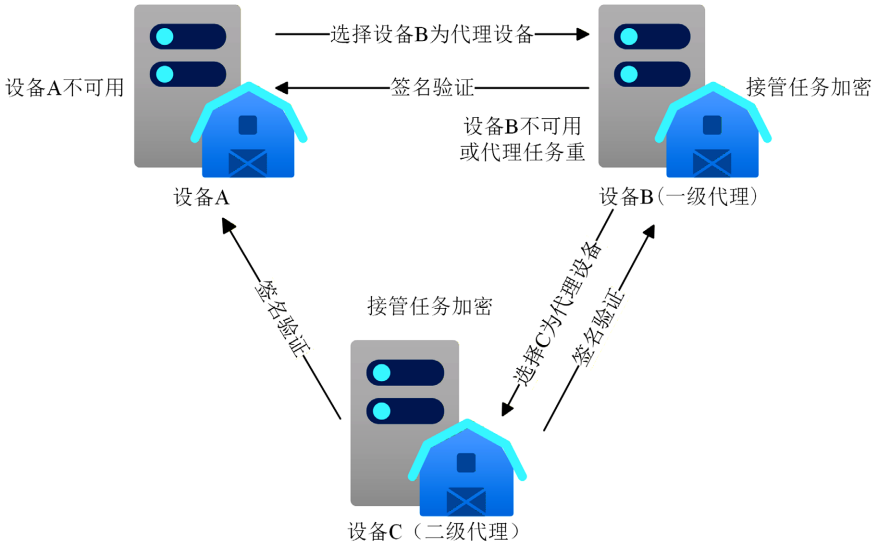


Figure 3. Proxy signature chain architecture
图 3. 代理签名链架构

一级代理签名：当设备准备离线时，选择一个综合成本最低的设备作为代理签名者。代理签名者将使用其私钥生成签名并附带授权证书。

二级代理签名：为了增加容错性，代理签名者可以选择另一个设备作为二级代理。每一层代理签名者都可以继续生成签名并附带授权证书，直到达到预定的代理签名链的最大层级，本文设计最大层级为2级。

签名链验证：每个签名链上的节点都可以验证前一个签名者的授权证书，确保代理签名链的合法性。

代理签名链的工作流程为当设备A离线时，由其选择的代理设备B接管其消息签名任务。若设备B由于某种原因无法签名或者节点B的代理任务重，则设备B通过代理节点选择算法选择设备C作为二级代理，并且每个节点在授权时会生成一个证书，证明其有权代理签名。

3.2.3. 代理签名生成

本文通过一个基于SipHash轻量级哈希的代理签名算法和消息认证来生成签名，并对SipHash签名进行了优化。采用分段加密的方式进行签名生成，该方法减少了对整个消息的计算量以及验证量，能够高效快速完成代理签名的生成与验证。并在其中添加了授权证书的设计、签名链的支持与验证以及授权追溯的验证，整个过程去除传统签名算法中的复杂操作。设计方法如下：

消息哈希：首先对消息进行轻量级哈希处理，得到固定长度的摘要。

分段加密：使用私钥加密哈希摘要的部分信息，而非整个消息，以减小计算负担，能够达到减少加密和验证时间的效果。

签名链支持：签名链中每一层的签名不仅包含生成签名的设备的标识，还附带当前设备的授权证明。每个代理签名者只需对上一层签名进行加密，而无需重新计算整个消息。

代理签名生成算法如算法2所示，代理签名生成的具体步骤如下：

1. 密钥生成

每个设备（例如设备A、B、C）生成各自的公私钥对(PK,SK)。例如私钥SK_{A}由设备A生成，公钥PK_{A}可公开。其他设备B、C等也各自生成公私钥对。公钥可用于验证签名，而私钥则用于对消息摘要进行加密，保证签名的不可篡改性。

2. 消息哈希

首先，设备对消息M进行轻量级哈希处理，得到固定长度的消息摘要。轻量级哈希算法SipHash因其计算开销小、效率高，适合在资源受限设备中使用。哈希计算公式如式(2)所示。

$$H(M)=\mathrm{SipHash}(M) \tag{2}$$

其中$H(M)$表示消息M的哈希值，长度固定为256位。

3. 消息摘要切分

将哈希值$H(M)$按照预设的分段规则切分为n段，每段长度为L。假设哈希值长度为256位，可以将其分为4个64位的段，如公式(3)所示。

$$H(M)=\{H_1(M),H_2(M),H_3(M),H_4(M)\} \tag{3}$$

这种分段方式能够有效减小加密运算的计算量，避免对整个哈希摘要进行加密所带来的性能开销。

4. 部分加密

为了减少加密计算开销，使用设备的私钥对其中一部分摘要进行加密。例如设备A仅对第一段摘要H_1进行非对称加密。加密过程如式(4)所示。

$$C_1=E_{SK_{\mathrm{A}}}(H_1) \tag{4}$$

其中，E_{SK_A} 表示使用设备 A 的私钥 SK_A 进行加密，C_1 为加密后的密文。其余摘要段可以选择明文传输或者进行对称加密，具体方式可以根据应用需求进行配置。部分加密的方式既保证了安全性，又减轻了加密计算的负担。

5. 代理签名生成

代理签名生成首先需要签名接收请求，例如代理设备 B 接收到来自设备 A 的签名请求。设备 B 验证设备 A 利用私钥签名的授权证明，确认设备 A 是否允许其代理签名。授权通过后，设备 B 使用自己的私钥 SK_B 对设备 A 加密后的摘要段 C_1 进行二次加密，得到 C_2，其过程如式(5)。

$$C_2 = E_{SK_B}\left(C_1\right) \tag{5}$$

代理设备 B 将自己的公钥 PK_B 以及设备 A 的授权证明 $Certificate_{A\to B}$ 附加到签名中，形成完整的代理签名，如式(6)。

$$\sigma = \left(C_2, PK_B, Certificate_{A\to B}\right) \tag{6}$$

签名链结构确保了代理签名的可追溯性，使得每一级代理签名都可以被验证，授权证书能证明其有权代理签名，其中授权证明 $Certificate_{A\to B}$ 的设计如式(7)。

$$Certificate_{A\to B} = \left(ID_A, ID_B, T, PK_B, Sig_A, Expiration, Noncce\right) \tag{7}$$

其中 ID_A 表示授权设备 A 的唯一标识 ID，ID_B 表示代理设备 B 的唯一标识 ID，T 表示授权时间戳，PK_B 表示设备 B 的公钥，Sig_A 表示设备 A 对授权信息的数字签名，防止消息被篡改，*Expiration* 表示授权有效期，限制代理签名的时间范围，*Nonce* 是生成的随机数，防止重放攻击。

算法 2 代理签名生成算法

输入：消息 M

输出：代理签名 σ

步骤 1：生成密钥对

每个设备生成各自的公私钥对(*PK*,*SK*)

步骤 2：消息哈希

使用轻量级 SipHash 函数计算消息的哈希值。

$$H\left(M\right) = \text{SipHash}\left(M\right)$$

步骤 3：消息摘要切分

将哈希值 *H*(*M*)拆分为个长度为 L 的片段，其中：

$$H\left(M\right) = \left\{H_1\left(M\right), H_2\left(M\right), H_3\left(M\right), H_4\left(M\right)\right\}$$

步骤 4：部分加密

使用设备 A 的私钥 SK_A 加密第一个片段 H_1，生成加密片段：

$$C_1 = E_{SK_A}\left(H_1\right)$$

步骤 5：代理签名生成，设备 B 使用其私钥对加密片段 C_1 进行加密，并附带证书

$$C_2 = E_{SK_B}\left(C_1\right)$$

$$\sigma = \left(C_2, PK_B, Certificate\right)$$

3.2.4. 代理签名验证

代理签名验证是整个方案的核心环节之一，其目的是确保签名链的完整性、签名者的合法性以及授权的有效性。针对物联网这一类资源受限的环境，需要以较低的计算复杂度实现高效的签名链验证。本节将详细阐述验证算法的步骤，从哈希完整性检查到签名链的最终确认，确保每一层代理签名的合法性与授权的有效性得以验证。代理签名验证的具体步骤如下：

1. 哈希验证

验证设备(例如设备 C)在收到消息 M 和对应的代理签名后，首先对原始消息 M 进行哈希计算，以验证消息的完整性。具体而言，使用与签名生成相同的轻量级哈希算法 SipHash，计算消息的哈希值。如果两者一致，则消息未被篡改，进入下一步；否则，验证失败，签名无效。

2. 验证加密部分

在代理签名机制中，每一层签名者(例如设备 A 和设备 B)使用各自的私钥对部分哈希摘要进行了加密。验证设备 C 需要使用对应的公钥逐步解密并验证这些加密部分。假设签名链为 $\mathrm{A}\to\mathrm{B}\to\mathrm{C}$，设备 C 使用设备 B 的公钥 PK_{B} 解密签名中的加密部分 σ_{B} 如式(8)。

$$\sigma'_{\mathrm{A}} = Decrypt\left(\sigma_{\mathrm{B}}, PK_{\mathrm{B}}\right) \tag{8}$$

其中 σ'_{A} 是设备 B 使用其私钥对设备 A 的签名部分加密 σ_{A} 的结果，也就是 C_1。解密后得到的 σ'_{A} 应与设备 A 的原始签名部分一致。

接着，设备 C 使用设备 A 的公钥 PK_{A} 解密 σ'_{A}，如式(9)。

$$H_1\left(M\right)' = Decrypt\left(\sigma'_{\mathrm{A}}, PK_{\mathrm{A}}\right) \tag{9}$$

其中，$H_1\left(M\right)'$ 是解密后得到的哈希摘要的第一段(假设加密的是第一段)。然后，将 $H_1\left(M\right)'$ 与公式(3)中计算得到的 $H\left(M\right)$ 的第一段 $H_1\left(M\right)$ 进行比较，如果相等，则设备 A 的签名有效；否则，验证失败。

这一分层解密过程确保了签名链中每一层的加密部分都能被正确追溯，同时减少了对整个哈希值进行加密的计算开销。

3. 签名链完整性与合法性验证

由于代理签名涉及多设备协作，签名链的每一层不仅包含签名本身，还附带授权证书和设备标识。为了验证签名链的完整性，设备 C 需要逐层检查每一级签名及其关联的授权信息。

签名链可表示为一个序列 $S=\{\sigma_{\mathrm{A}},\sigma_{\mathrm{B}},\cdots\}$，其中 σ_{A} 是初始签名，σ_{B} 是代理签名。每层签名还包括设备标识(如 $ID_{\mathrm{A}}, ID_{\mathrm{B}}$)和授权证书($Certificate_{\mathrm{A}\to\mathrm{B}}$)。对于每一层代理设备 i 的签名都要进行验证以确保签名链的完整性，验证过程如式(10)。

$$Valid_i = \mathrm{Verify}\left(\sigma_i, PK_i\right) \tag{10}$$

其中，Verify 是公钥验证函数，返回布尔值。如果任一层验证失败，则整个签名链无效。

对于每一层的签名都要检查授权证书的合法性，例如对于代理设备 B，其授权证书必须由设备 A 签名，验证过程如式(11)。

$$Valid_{Certificate_{\mathrm{A}\to\mathrm{B}}} = \mathrm{Verify}\left(Certificate_{\mathrm{A}\to\mathrm{B}}, PK_{\mathrm{A}}\right) \tag{11}$$

如果证书有效，则表明设备 B 确实被设备 A 授权进行代理签名。这一步骤确保了即使在多代理场景下，签名链也不会被伪造或中断。

4. 签名链最终验证

当初始设备 A 恢复在线时，它需要对整个签名链进行最终验证，以确保代理过程未被滥用。设备 A 首先重新计算消息 M 的哈希值，并验证签名链中每一层的加密部分。然后进行授权追溯，确保签名链中

每一环都符合授权规则。授权追溯过程如式(12)。

$$Valid_i = \wedge_{i=1}^{k} \text{Verify}\left(\sigma_i, PK_i\right) \wedge \text{Verify}\left(Certificate_{i-1\to i}, PK_i\right) \tag{12}$$

其中 k 是签名链的层数，$\wedge$ 是逻辑与操作，如果所有验证通过，则签名链被视为有效；否则，设备 A 可以拒绝接受该代理签名。

授权追溯的过程，确保代理签名不仅在技术上可验证即签名有效，而且在权限上合法即授权有效，提供了一种高效且安全的验证方法。

4. 实验与分析

4.1. 实验环境介绍

本文实验使用 GO 模拟底层物理网络条件，模拟实现了共识节点之间的通信过程。配置为 Intel(R) Core(TM) i7-13650HX CPU，主频 2.60 GHz，服务器运行 Ubuntu 20.04.6LTS，内存配置 12 GB。在此环境上，对 PBFT、TP-PBFT [30]、无代理方案的 GV-PBFT 以及本章节基于代理签名方案的 GV-PBFT 共识算法进行了对比分析，主要在代理签名任务数量、吞吐量、时延和通信开销四个方面进行对比分析。本文实验基于 GV-PBFT 算法[31]，在 GV-PBFT 算法组内共识的基础上对本文所提出的 SH-PBFT 算法代理签名进行广播与验证。本文主要对比了 PBFT、TP-PBFT、GV-PBFT (无代理)、GV-PBFT (有代理)几种共识算法下的性能。其中 GV-PBFT (无代理)表示没有使用本文提出的 SH-PBFT 代理算法，GV-PBFT (有代理)表示使用了本文提出的 SH-PBFT 代理算法。

4.2. 代理签名任务数量

代理签名任务数量是指一个节点为多少个其他节点进行了代理签名，从每个节点的代理签名任务数量可以看出代理签名的分布是否均匀，防止一个节点代理过多节点进行签名而导致负载较高的问题，能够有效反应代理签名节点选择方案的有效性。因此实验设置共 15 个节点，其中 5 个离线节点，随机挑选四个在线进行共识的节点观察其在共识轮次变化中代理签名任务数量的变化，如图 4 所示。

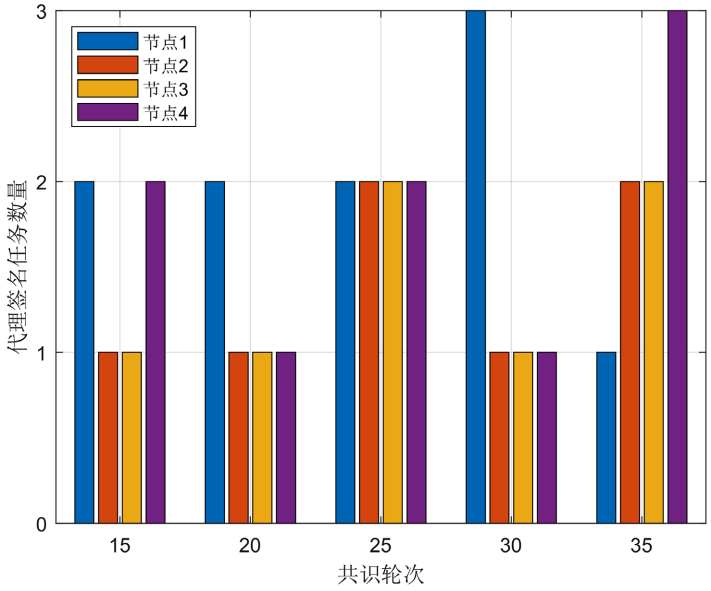


Figure 4. The number of proxy signature tasks
图 4. 代理签名任务数量

从图 4 可以看出，四个节点在不同共识轮次中的代理签名任务数量变化都较为均匀，不存在一个节点代理过多签名任务数量的问题。在面向物联网环境下一些资源和计算能力受限的设备节点时，不会因代理签名方案而产生过大负载的情况。因此，代理节点选择方案既能有效保障代理签名方案的顺利运行，也能防止整个系统中的节点产生代理签名数量负载不均衡的问题。

4.3. 时延分析

本节在时延分析方面，主要从两个方面来评估：一是平均时延，能够有效反应基于代理签名方案共识优化的整体性能，在平均共识时延中设置总节点数为 45，5 个节点为一组，每个组内存在 1 个离线节点；二是考察单个交易或事件的共识时延，这一点不仅能够有效反应出其他方案因共识失败而重启共识所造成的额外时延开销，也能反应出本节基于代理签名方案共识优化的优越性所在。在实验设置里，假设不存在拜占庭节点，节点总数为 45，离线节点数量从 2 增加到 14，步长为 2，其他对比方案设置存在离线节点过多时，共识失败会重启共识，不考虑离线节点的离线时长问题，在此基础上观察时延的变化。平均时延与单个交易或事件的共识时延分别如图 5 和图 6 所示。

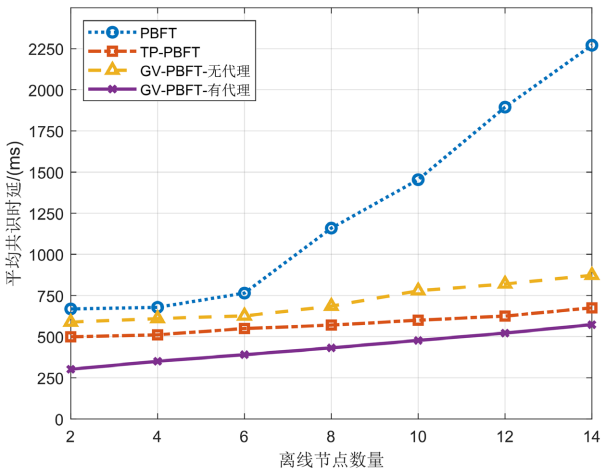
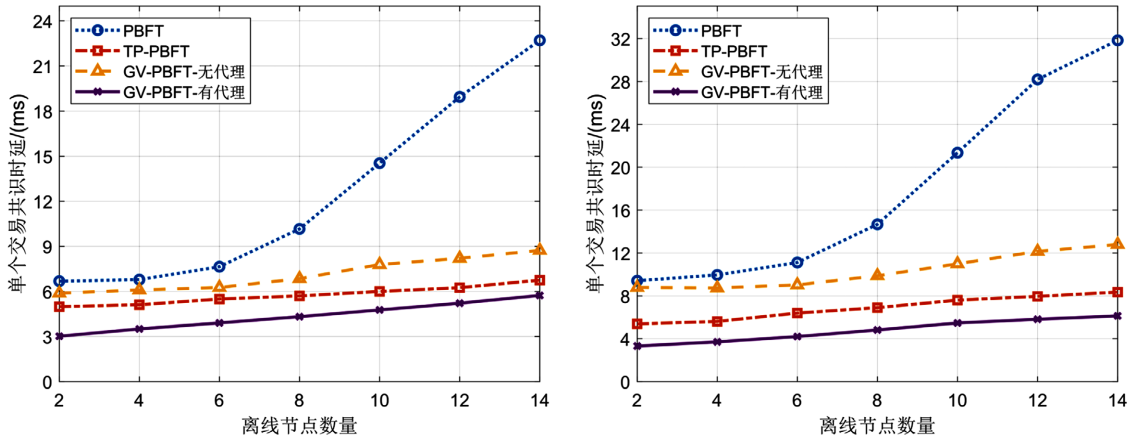


Figure 5. Average consensus delay
图 5. 平均共识时延



(a) 单个交易平均时延　　(b) 重启共识时延

Figure 6. Consensus delay for a single transaction
图 6. 单个交易共识时延

从图 5 可以看出，PBFT 算法与无代理方案的 GV-PBFT 算法的共识时延较高，因为当存在离线节点的时候都需要等待离线节点的超时重传消息，所以会产生额外的时延开销，并且随着离线节点数量的不断增加，时延将会趋于无穷大，也就是共识失败。而基于本文提出的 SH-PBFT 代理签名方案的 GV-PBFT 与基于阈值代理签名方案的 TP-PBFT 因为可以使用代理方案为离线节点进行代理签名，并且优化了共识流程，所以在时延方面表现更优，并且本文的基于 Siphash 代理方案的 GV-PBFT 由于 Siphash 签名的快速性使得时延明显更优于 TP-PBFT。

图 6 展示单个交易的时延随着离线节点数量增加的变化，其中图 6(a)由平均共识时延计算得出，实验中设置 100 个交易事件，因此将四种算法的平均共识时延除以 100 得到平均的单个交易时延图。图 6(b)则是设置 PBFT 与无代理的 GV-PBFT 可能会在共识流程中的某一个阶段共识失败，需要重启共识，而有代理的 GV-PBFT 与 TP-PBFT 则由代理节点进行代理签名任务，无需重启共识。实验表示 PBFT 与无代理的 GV-PBFT 共识时延平均增加了约 40%左右，而基于代理方案的 GV-PFBT 与 TP-PBFT 影响最小，且基于本文提出的 SH-PBFT 代理签名方案的 GV-PBFT 表现最优。

4.4. 吞吐量分析

实验设置 100 个交易事件，节点总数为 45，离线节点数量从 2 增加到 14，在此基础上观察随着离线节点增多，系统整体吞吐量的变化。一共进行 20 次实验，取均值作为实验结果。吞吐量对比分析如图 7 所示。

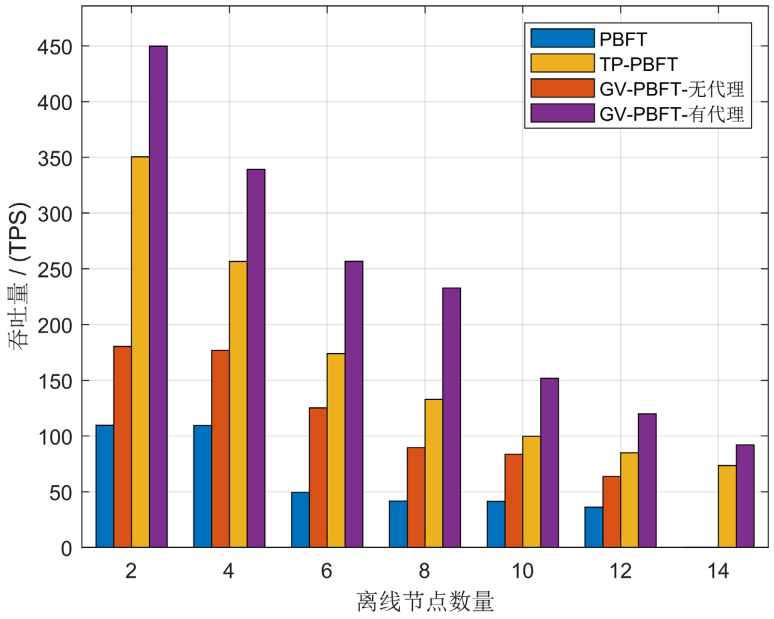


Figure 7. Comparison of throughput
图 7. 吞吐量对比

从图 7 可以看出，随着离线节点的数量增加，四种算法的吞吐量都在不断降低，其中，PBFT 与无代理的 GV-PBFT 算法在离线节点数量达到 14 个的时候，吞吐量降为了 0，因为此时离线节点数量已达到算法所能容忍的最大离线节点数量，无法再正常进行共识。而基于代理的 GV-PBFT 和 TP-PBFT 算法由于可以使用代理签名流程完成签名任务，因此可以在离线节点过多的情况下仍能完成共识，并且基于本文提出的 SH-PBFT 代理签名方案的 GV-PBFT 算法整体性能上表现最好。

4.5. 通信开销分析

本节将基于本文提出的 SH-PBFT 代理签名方案的 GV-PBFT 算法与 PBFT 和 TP-PBFT 算法的通信开销进行了对比分析，以此观察各自的通信开销表现。设总节点数为由 100 增加至 600，步长为 100，视图转化概率为由 0 增加至 1，步长为 0.2，不考虑分组的影响。当存在离线节点时，基于代理方案的 GV-PBFT 算法将增加投诉与代理签名阶段，因此在计算 GV-PBFT [31]的总通信开销公式(11)的基础上额外增加 $N+1$ 的通信次数，得到基于代理方案的 GV-PBFT 总通信开销次数如式(13)。

$$C_{\text{SH-GV-PBFT}} = \frac{11N-3}{3} + P\cdot\left(\frac{4N^2}{9} - \frac{2N}{3}\right) + 2x + N + 1 \tag{13}$$

TP-PBFT [30]总通信开销如式(14)。

$$C_{\text{TP-PBFT}} = 2\frac{N\lambda}{k}\left(\frac{N\lambda}{k} - 1\right) + 2k\left(k-1\right) \tag{14}$$

其中 k 为集群数[30]，λ 为每个集群内的代表节点数量，其中两个参数都取值 20。

PBFT 的总通信开销如式(15)。

$$C_{\text{PBFT}} = N_{\text{PBFT}} + M_{\text{PBFT}} = 2N^2 - N + P\cdot\left(N^2 - N\right) \tag{15}$$

由公式(13)、公式(14)和公式(15)得到基于代理方案的 GV-PBFT 与 PBFT 和 TP-PBFT 的通信开销比值 Q_1 和 Q_2，如式(16)和式(17)所示，其三维可视化图如图 8 所示。

$$Q_1 = \frac{C_{\text{SH-GV-PBFT}}}{C_{\text{PBFT}}} = \frac{\dfrac{11N-3}{3} + P\cdot\left(\dfrac{4N^2}{9} - \dfrac{2N}{3}\right) + 2x + N + 1}{2N^2 - N + P\cdot\left(N^2 - N\right)}, 0 \le P \le 1 \tag{16}$$

$$Q_2 = \frac{C_{\text{SH-GV-PBFT}}}{C_{\text{TP-PBFT}}} = \frac{\dfrac{11N-3}{3} + P\cdot\left(\dfrac{4N^2}{9} - \dfrac{2N}{3}\right) + 2x + N + 1}{2\dfrac{N\lambda}{k}\left(\dfrac{N\lambda}{k} - 1\right) + 2k\left(k-1\right)}, 0 \le P \le 1 \tag{17}$$

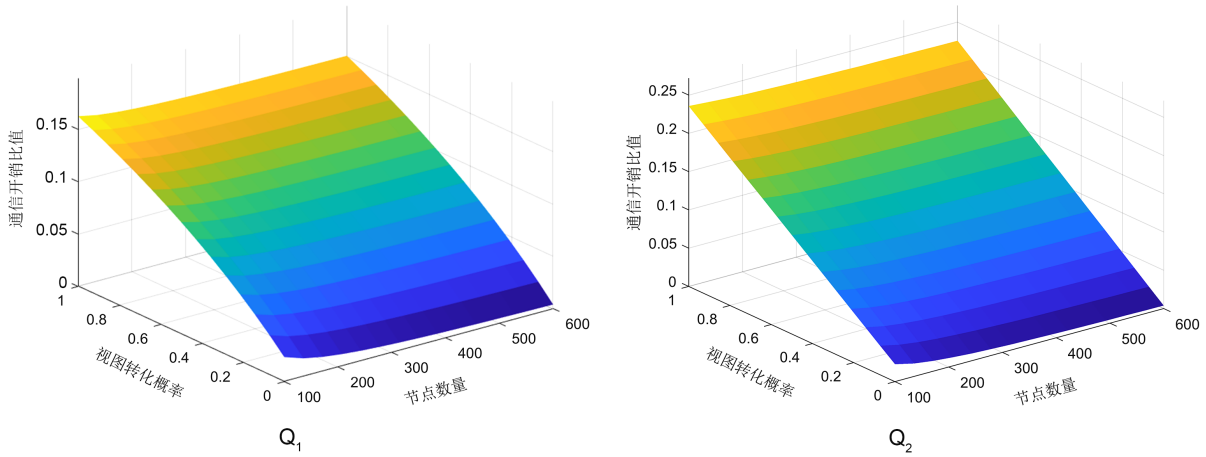


Figure 8. Comparison of communication overhead
图 8. 通信开销对比

由图 8 可以看出，无论节点数量和视图转换概率如何增加变化，比值 Q_1 和 Q_2 始终在 1 以下进行波动变化，表明基于 Siphash 代理签名方案的 GV-PBFT 的通信开销始终小于 TP-PBFT 和 PBFT 算法，并且 TP-PBFT 算法并没有考虑视图转换可能会带来的额外通信开销问题，因此在实际情况中将会有更好的表现。

5. 总结

本文主要针对物联网环境下计算能力与资源受限的设备可能会出现突然离线或宕机的问题，提出了一种基于 SipHash 代理签名的共识机制优化 SH-PBFT。首先，基于代理签名节点选择算法，每个节点可为自己选择代理签名节点，为自己在离线状态下进行消息的代理签名，实现了代理签名任务数量的均匀分布，预防了因代理任务过度而出现负载过高的问题，并形成一条多级代理签名链，增强了系统的稳定性。然后提出了代理签名模型，通过高效安全的代理签名生成与验证过程，提供了签名的有效性、安全性和可溯源检查的整套流程。并且优化了存在离线节点情况下的共识机制，使得部分节点离线仍然能完成共识流程，避免了因为部分节点离线导致的共识失败而重启共识所造成的额外通信时延和通信开销问题。最后通过实验验证了方案的有效性，实验结果表明本文方案在面对离线节点存在的情况下，仍能表现出较好的性能。

参考文献

[1] Ashton, K. (2009) That “Internet of Things” Thing. *RFID Journal*, **22**, 97-114.

[2] Wang, J., Lim, M.K., Wang, C. and Tseng, M. (2021) The Evolution of the Internet of Things (IoT) over the Past 20 Years. *Computers & Industrial Engineering*, **155**, Article ID: 107174. https://doi.org/10.1016/j.cie.2021.107174

[3] Asir, T.R.G. and Manohar, H.L. (2018) Key Challenges and Success Factors in IoT—A Study on Impact of Data. 2018 *International Conference on Computer*, *Communication*, *and Signal Processing* (*ICCCSP*), Chennai, 22-23 February 2018, 1-5. https://doi.org/10.1109/icccsp.2018.8452843

[4] Reyna, A., Martín, C., Chen, J., Soler, E. and Díaz, M. (2018) On Blockchain and Its Integration with IoT. Challenges and Opportunities. *Future Generation Computer Systems*, **88**, 173-190. https://doi.org/10.1016/j.future.2018.05.046

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

[6] Fox, G. (2001) Peer-to-Peer Networks. *Computing in Science & Engineering*, **3**, 75-77. https://doi.org/10.1109/5992.919270

[7] Mainelli, M. and Mills, S. (2016) The Missing Links in the Chains? Mutual Distributed Ledger (Aka Blockchain) Standards. https://www.longfinance.net/programmes/distributed-futures/news/the-missing-links-in-the-chains-mutual-distributed-ledger-aka-blockchain-standards/

[8] Diffie, W. and Hellman, M.E. (2022) New Directions in Cryptography. In: Slayton, R., Ed., *Democratizing Cryptography*: *The Work of Whitfield Diffie and Martin Hellman*, ACM, 365-390. https://doi.org/10.1145/3549993.3550007

[9] Rivest, R.L., Shamir, A. and Adleman, L. (1978) A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, **21**, 120-126. https://doi.org/10.1145/359340.359342

[10] McEliece, R.J. (1978) A Public-Key Cryptosystem Based on Algebraic Coding Theory. *DSN Progress Report*, **42**, 114-116.

[11] Wang, W., Hoang, D.T., Hu, P., Xiong, Z., Niyato, D., Wang, P., *et al*. (2019) A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks. *IEEE Access*, **7**, 22328-22370. https://doi.org/10.1109/access.2019.2896108

[12] King, S. and Nadal, S. (2012) PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. https://www.semanticscholar.org/paper/PPCoin%3A-Peer-to-Peer-Crypto-Currency-with-King-Nadal/0db38d32069f3341d34c35085dc009a85ba13c13

[13] Delegated Proof of Stake Consensus. https://bitshares.org/delegated-proof-of-stake-consensus/

[14] Islam, S., Islam, M.J., Hossain, M., Noor, S., Kwak, K. and Islam, S.M.R. (2023) A Survey on Consensus Algorithms in Blockchain-Based Applications: Architecture, Taxonomy, and Operational Issues. *IEEE Access*, **11**, 39066-39082. https://doi.org/10.1109/access.2023.3267047

[15] Sankar, L.S., Sindhu, M. and Sethumadhavan, M. (2017). Survey of Consensus Protocols on Blockchain Applications. 2017 4*th International Conference on Advanced Computing and Communication Systems* (*ICACCS*), Coimbatore, 6-7 January 2017, 1-5. https://doi.org/10.1109/icaccs.2017.8014672

[16] Ongaro, D. and Ousterhout, J.K. (2014) In Search of an Understandable Consensus Algorithm. In: 2014 *USENIX Annual Technical Conference* (*USENIX ATC* 14), USENIX Association, 305-319.

[17] Castro, M. and Liskov, B. (1999) Practical Byzantine Fault Tolerance. In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation* (*OSDI*'99), USENIX Association, 173-186.

[18] Miller, A., Xia, Y., Croman, K., Shi, E. and Song, D. (2016) The Honey Badger of BFT Protocols. *Proceedings of the* 2016 *ACM SIGSAC Conference on Computer and Communications Security*, Vienna, 24-28 October 2016, 31-42. https://doi.org/10.1145/2976749.2978399

[19] Wang, S. (2019) Performance Evaluation of Hyperledger Fabric with Malicious Behavior. *Blockchain-ICBC* 2019: *Second International Conference*, *Held as Part of the Services Conference Federation*, *SCF* 2019, San Diego, 25-30 June 2019, 211-219. https://doi.org/10.1007/978-3-030-23404-1_15

[20] Jiang, Y. and Guan, Y. (2023) A Cluster Reputation-Based Hierarchical Consensus Model in Blockchain. *Peer-to-Peer Networking and Applications*, **16**, 2591-2606. https://doi.org/10.1007/s12083-023-01550-5

[21] Li, W., Feng, C., Zhang, L., Xu, H., Cao, B. and Imran, M.A. (2021) A Scalable Multi-Layer PBFT Consensus for Blockchain. *IEEE Transactions on Parallel and Distributed Systems*, **32**, 1146-1160. https://doi.org/10.1109/tpds.2020.3042392

[22] Zhao, Y., Guo, B., Qin, C. and Zhao, M. (2022) A Multi-Layer PBFT Consensus Algorithm with Inter-Group Supervision. 2022 *IEEE* 24*th International Conference on High Performance Computing & Communications*; 8*th International Conference on Data Science & Systems*; 20*th International Conference on Smart City*; 8*th International Conference on Dependability in Sensor*, *Cloud & Big Data Systems & Application* (*HPCC*/*DSS*/*SmartCity*/*DependSys*), Hainan, 18-20 December 2022, 1101-1108. https://doi.org/10.1109/hpcc-dss-smartcity-dependsys57074.2022.00174

[23] Qushtom, H., Mišić, J. and Mišić, V.B. (2022) Efficient Multi-Tier, Multiple Entry PBFT Consensus Algorithm for IoT. *ICC* 2022-*IEEE International Conference on Communications*, Seoul, 16-20 May 2022, 44-49. https://doi.org/10.1109/icc45855.2022.9838616

[24] Kapitza, R., Behl, J., Cachin, C., Distler, T., Kuhnle, S., Mohammadi, S.V., *et al*. (2012) CheapBFT: Resource-Efficient Byzantine Fault Tolerance. *Proceedings of the* 7*th ACM European Conference on Computer Systems*, Bern, 10-13 April 2012, 295-308. https://doi.org/10.1145/2168836.2168866

[25] Veronese, G.S., Correia, M., Bessani, A.N., Lung, L.C. and Verissimo, P. (2013) Efficient Byzantine Fault-Tolerance. *IEEE Transactions on Computers*, **62**, 16-30. https://doi.org/10.1109/tc.2011.221

[26] Singh, J., Kumawat, A. and Venkatesan, S. (2022) Improved Byzantine Fault Tolerance with Fast Consensus. *Concurrency and Computation*: *Practice and Experience*, **34**, e6813. https://doi.org/10.1002/cpe.6813

[27] Decouchant, J., Kozhaya, D., Rahli, V. and Yu, J. (2022) DAMYSUS: Streamlined BFT Consensus Leveraging Trusted Components. *Proceedings of the Seventeenth European Conference on Computer Systems*, Rennes, 5-8 April 2022, 1-16. https://doi.org/10.1145/3492321.3519568

[28] Yu, X., Qin, J. and Chen, P. (2022) GPBFT: A Practical Byzantine Fault-Tolerant Consensus Algorithm Based on Dual Administrator Short Group Signatures. *Security and Communication Networks*, **2022**, Article ID: 8311821. https://doi.org/10.1155/2022/8311821

[29] Wang, Z., Hu, G. and You, L. (2024) An Improved Model of PBFT with Anonymity and Proxy Based on Linkable Ring Signature. In: Tari, Z., *et al*., Eds., *Algorithms and Architectures for Parallel Processing*, Springer, 491-502. https://doi.org/10.1007/978-981-97-0808-6_29

[30] Tang, F., Xu, T., Peng, J. and Gan, N. (2024) TP-PBFT: A Scalable PBFT Based on Threshold Proxy Signature for IoT-Blockchain Applications. *IEEE Internet of Things Journal*, **11**, 15434-15449. https://doi.org/10.1109/jiot.2023.3347232

[31] Ji, Y., Zou, Y., Wang, G., Li, W. and Xu, R. (2024) An Improved PBFT Consensus Algorithm Based on Reputation Grading and Voting in IoT Environment. 2024 *IEEE International Symposium on Parallel and Distributed Processing with Applications* (*ISPA*), Kaifeng, 30 October-2 November 2024, 2121-2128. https://doi.org/10.1109/ispa63168.2024.00289