计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(10): 2428−2440, 2025

# DecXClient: 一种通用、安全的轻客户端协议

邓智洪[1] 唐春明[1] 李涛涛[2] 何雅萱[1] 曾志康[1] 陈 祺[3]
[1](广州大学数字与信息科学学院 广州 510006)
[2](中山大学软件工程学院 广东珠海 528478)
[3](广州大学人工智能学院 广州 510006)
(zhihongdeng@gzhu.edu.cn)

# DecXClient: A General and Secure Light Client Protocol

Deng Zhihong[1], Tang Chunming[1], Li Taotao[2], He Yaxuan[1], Zeng Zhikang[1], and Chen Qi[3]
[1](*School of Mathematics and Information Science*, *Guangzhou University*, *Guangzhou* 510006)
[2](*School of Software Engineering*, *Sun Yat-sen University*, *Zhuhai*, *Guangdong* 528478)
[3](*School of Artificial Intelligence*, *Guangzhou University*, *Guangzhou* 510006)

**Abstract** With the rapid development of blockchain applications, light clients have become essential components in digital wallets and cross-chain communication due to their minimal storage and computational overhead. However, existing light client protocols face significant limitations in terms of storage and communication costs, security, and compatibility across different consensus mechanisms. We propose DecXClient, a general and secure light client protocol compatible with multiple consensus models. DecXClient introduces a dynamic validator set election mechanism based on chain quality attributes, eliminating reliance on trusted third parties. By integrating weighted multi-signature schemes, it enables a concise and efficient proof generation paradigm that enhances anti-fraud resilience while reducing resource consumption. Without depending on centralized relays, DecXClient maintains sublinear communication and storage complexity, effectively mitigating Eclipse attacks and other security threats. Experimental evaluations using real blockchain network environment demonstrate that the protocol produces state proofs as small as 1 248 bytes—significantly outperforming existing schemes. The average proof generation latency ranges from 266.4 milliseconds to 2.5 seconds, with negligible verification time. These results validate the feasibility and practicality of DecXClient, offering a robust solution for lightweight blockchain access across diverse application scenarios.

**Key words** light client; blockchain; digital wallet; cross-chain communication; consensus mechanism

**摘 要** 随着区块链应用的快速发展,轻客户端因其低存储和计算负担,已成为数字钱包与跨链通信中的关键技术.然而,现有轻客户端协议在存储与通信开销、安全性和共识机制兼容性方面仍存在明显局限.提出 DecXClient,一种兼容多种共识机制和具备通用性与安全性的轻客户端协议.该协议基于链质量属性设计了无需可信第三方的验证集动态选举机制,结合加权多重签名提出了一种简洁高效的证明生成范式,

**收稿日期**: 2025-06-06; **修回日期**: 2025-07-21
**基金项目**: 国家重点研发计划(2021YFA1000600); 国家自然科学基金项目(12571572); 广州大学研究生基础创新项目(JCCX2025-015); 广东省基础与应用基础研究基金项目(2024A1515012360); 狮子山网络空间安全实验室研究课题(LRL24003)
This work was supported by the National Key Research and Development Program of China (2021YFA1000600), the National Natural Science Foundation of China (12571572), the Graduate Innovative Research Funding Project of Guangzhou University (JCCX2025-015), the Guangdong Basic and Applied Basic Research Foundation (2024A1515012360), and the Open Topics from the Lion Rock Labs of Cyberspace Security (LRL24003).
**通信作者**: 唐春明(ctang@gzhu.edu.cn)

显著提升抗欺诈能力并降低轻客户端的资源消耗. 在不依赖中心化中继的前提下, DecXClient 保持低于次线性级别的通信和存储复杂度, 有效防御 Eclipse 攻击等安全威胁. 在模拟真实公链网络的实验中该协议展现出卓越性能: 状态证明大小仅为 1 248 B, 显著优于现有方案; 平均生成延迟低至 266.4 ms, 验证延迟近似为零. 结果表明, DecXClient 在安全性、效率与可适用性之间实现了良好平衡, 为多场景区块链轻量化接入提供了可行方案.

**关键词** 轻客户端; 区块链; 数字钱包; 跨链通信; 共识机制

**中图法分类号** TP393

**DOI:** 10.7544/issn1000-1239.202550429 **CSTR:** 32373.14.issn1000-1239.202550429

区块链可以被视为一种抽象的全球分布式账本, 支持上层应用用户执行数据读写操作[1]. 然而, 最为基础的“读取账本”需求在实际应用中往往隐含着用户需运行本地全节点的要求, 这意味着用户节点需要参与共识协议并维护完整的区块链副本[2]. 随着区块链技术的快速普及[3], 关注加密货币等高层应用的用户日益增加, 但应用层的终端设备或数字钱包通常受限于计算与存储资源, 难以承担全节点的持续在线与同步开销[4-5].

轻客户端(light client, LC)[6-7], 又被称为轻节点, 是区块链网络中的一类轻量化节点. 与全节点相比, 轻客户端以更低的资源消耗和存储需求提供基础的验证与交互功能. 它通过选择性下载区块链部分数据, 并结合密码学证明机制完成交易验证, 从而显著降低存储和计算负担. 例如, 比特币全节点需存储超 400 GB 数据, 而基于简易支付验证(simplified payment verification, SPV)的轻客户端仅需约 50 MB 区块头数据[8]. 作为低存储成本的节点实现方式, 轻客户端在数字钱包和跨链互操作机制的应用中充当了关键的数据验证和通信桥梁[3-4], 极大提升了区块链的可访问性与普适性.

图 1 展示了轻客户端的交互流程. 当数字钱包发起交易或查询余额时, 需通过轻客户端向全节点请求相关数据; 全节点返回完整数据支持轻客户端完成验证; 轻客户端最终完成数据验证并反馈结果至数字钱包. 数字钱包作为轻客户端协议的软件实现, 借助加密技术和多重身份认证保护用户支付信息和隐私, 并完成数据验证. 在跨链场景中, 轻客户端依

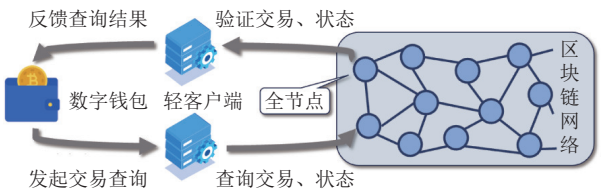



Fig. 1 Interaction logic of the light client

图 1 轻客户端交互逻辑

托源链的共识机制保证数据真实性, 同时作为目标链中的验证模块核验源链交易的合法性[3]. 跨链账户状态(如余额、资产)则通过同步验证区块头和状态证明, 实现高效的状态更新.

轻客户端协议设计的核心挑战源于区块链验证的本质: 交易有效性依赖于网络共识下最长链的确定性. 客户端若无可信的最长链副本, 必须依赖全节点转发交易数据[7]. 以 SPV 为例, 轻客户端仅存储区块头并通过默克尔证明(Merkle proof)验证交易, 安全性假设则需要满足至少连接 1 个诚实全节点. 网络层的日蚀攻击(eclipse attack)进一步加剧了安全风险: 攻击者隔离目标节点网络连接, 向轻客户端注入伪造链数据, 扭曲其链状态认知. 现有防御多依赖中继节点转发链状态, 假设轻客户端信任大多数中继节点诚实. 然而在公有链环境下, 此类强信任假设难以成立. 例如, Cardano 等人[9] 要求中继节点持续在线并依赖激励机制, 限制了方案的广泛适用性.

针对不同共识机制, 轻客户端方案之间差异显著. 在基于工作量证明(proof of work, PoW)的区块链中, 诸如 PoPoW[10]、NiPoPoW[11] 和 FlyClient[12] 等轻客户端协议利用概率抽样和默克尔山脉承诺机制, 将通信复杂度从线性降至对数级, 但受限于固定挖矿难度和硬分叉等风险, 这类协议难以实现向后兼容性. 在基于权益证明(proof of stake, PoS)的区块链中, 因后缀块有效性依赖于动态变化的权益签名集, 离线后的验证效率仍受制约[13]. 例如, 以太坊信标链中的同步委员会由 512 名验证者随机组成, 专门为轻客户端生成最新状态证明, 用以验证区块头状态[9].

密码学原语的引入推动了轻客户端技术的突破. 零知识证明(zero-knowledge proof, ZKP)通过生成链上状态转换的简洁证明, 极大减少了对原始数据下载的依赖, 例如 zk-Rollup 将海量交易压缩为单个 SNARK 证明[14]; 加密累加器(cryptographic accumulator)则通过恒定大小的证据来验证元素的隶属关系, 有效降

低存储需求[15]. Portal Network 通过分布式哈希表来实现状态证明的去中心化存储, 在保持次线性通信复杂度的同时规避单点可信中继风险, 为轻客户端的安全性与效率平衡提供了新范式[6].

综上所述, 理想的轻客户端协议应具备低通信开销、最小存储需求和简便的初始化流程. 其核心职责是作为高效的区块链状态证明验证者, 验证工作由区块链中的实体(矿工或全节点)生成证明, 且整体设计应避免显著性能损耗, 以满足数字钱包和跨区块链等多样化应用场景的需求.

因此, 针对传统轻客户端协议中存在存储与通信开销大、易受恶意全节点欺诈和 Eclipse 攻击影响, 且仅适用于特定共识机制(如 PoW 或 PoS, 且两者不兼容)的问题, 本文提出了一种通用且安全的轻客户端协议——DecXClient, 旨在提升区块链数字钱包和跨链场景中状态账本数据流通的性能. 本文主要贡献包括 3 个方面:

1)提出了 DecXClient, 这是一种通用、安全的轻客户端协议, 通用性体现为可兼容任何具备链质量属性的区块链, 涵盖且不限于比特币、以太坊和领导人轮换的拜占庭容错共识协议等; 安全性则表现为满足轻客户端的安全初始化与同步和安全验证的核心属性. 经验证, 该协议在计算、通信和存储方面的成本相较于区块链的长度低于次线性级.

2)设计了一种基于链质量属性的验证集动态选举机制, 该机制无需依赖可信第三方, 具备去中心化和灵活变更能力. 机制的安全强度是动态可调的, 且支持安全门限阈值的可插拔配置. 此外, 针对恶意全节点的欺诈攻击, 结合验证集和加权多重签名提出了一种简洁高效的轻客户端证明生成范式, 大幅降低轻客户端的验证、存储与通信负载.

3)实验结果表明, 模拟比特币真实环境且区块链的总区块长度为$6\times10^6$时, 区块链总状态的证明大小仅为 1 248 B, 相较之下, 其证明大小仅为 NiPoPoW 的 1/4 206、FlyClient 的 1/2 765, 以及 Cardano 的约 40%. 不同参数配置下的证明生成平均延迟为 266.4 ms~2.5 s, 验证延迟忽略不计, 充分验证了协议的可行性和实用性.

## 1 相关工作

轻客户端协议的雏形可追溯至比特币白皮书中提到的 SPV 协议, 随后在 BTC Relay[16] 和 ETH Relay[17] 等协议中得到具体实现. SPV 的运行机制依托默克尔证明, 该证明使仅存储最新一段区块头的轻客户端能够验证交易是否被包含在区块中. 自此以后, 学术界提出了多种类型的改进方案. 例如, 在基于权益证明的系统中, 轻客户端通常需要验证整个区块链历史中的账户余额(或验证至某个快照), 并兼顾长距离攻击的风险; 而在基于拜占庭容错(Byzantine fault tolerance, BFT)共识的系统中, 轻客户端需要效验验证者的签名并跟踪验证者的轮换状态[6].

尽管基于 SPV 的轻客户端通过缩减存储规模显著降低了区块链参与的资源成本(例如, 比特币的区块头大小约为 80 B, 而完整区块大小则通常大于 1 MB), 但其仍需处理与链长度呈线性增长的海量数据(例如, 比特币网络中的 SPV 客户端的存储需求超过 60 MB, 而以太坊则超过 4 GB), 这种存储负担催生了对各类优化方案的探索与实践.

最早的优化方案之一是 PoPoW 协议[10], 该协议将线性级别的通信存储成本降低到对数级别. 随后, NiPoPoW 协议消除了 PoPoW 的交互需求, 在乐观条件下为高达 1/2 的攻击者提供了安全性和简洁性[11]. 在后续的研究中, 通过处理乐观环境约束, 协议扩展到能够在更强的攻击者环境下保持简洁性, 并为最多 1/3 阈值攻击者提供安全保障[18]. 另一种替代方案是 FlyClient[12], 该协议为高达 1/2 攻击者提供安全性和简洁性, 并支持对可变挖矿难度的调整. 以上这些改进方案[10-12,18] 被称为超轻客户端(super-light client).

近年来, 利用通用递归零知识证明技术成为轻客户端协议的创新方向, 相关协议支持构建具有恒定通信开销的轻客户端[19]. 例如, zkBridge[20] 是一种高效的基于零知识证明的轻客户端跨链桥, 它可以在没有额外信任假设的情况下保证强大的安全性. 实验表明, zkBridge 在 Cosmos 和 Ethereum 之间的跨链传输可以在 2 min 内生成一个区块头证明, 同时链上验证证明的成本低于 220 000 gas. DendrETH 是一个去中心化且高效的基于 ZKP 的轻客户端, 通过利用零知识证明特性, 显著降低了攻击覆盖面, 从而缓解了安全问题, 并构建恒定的证明大小[19]. 然而, 基于 ZKP 的轻客户端在生成跨链中的计算开销较大, 并且通常需要初始化可信设置来生成密钥和进行验证.

为了解决初始化信任问题, 无状态 SPV(stateless SPV)的概念应运而生. 最近, Barbára 等人[21] 在 BxTB 跨链交换协议中对无状态 SPV 进行了形式化实现. 然而, Scaffino 等人[22] 发现该构造容易受到预先挖矿攻击(pre-mining attack)的影响, 存在原子性安全漏洞. 对此, 他们提出了新的 Glimpse 协议, 通过引入高

熵交易构建“新鲜”证明，有效规避了预先挖矿的风险. 此外，Glimpse 将无状态 SPV 与随机数机制结合，实现了低延迟且去中心化的轻客户端验证.

当前研究趋势已经逐步从传统基于 SPV 的轻客户端，转向无状态 SPV 及其他轻量级客户端. 这一发展方向的核心目标在于减少轻客户端验证所需的存储和通信开销，同时保障系统的安全性. 然而，迄今为止，尚未出现能够在高效性、安全性、功能全面性、实用性以及最小化共识参与者或全节点开销方面取得平衡的解决方案. 此外，现有的研究并未充分考虑轻客户端在频繁离线阶段可能引发的低效问题. 即便是采用高效引导协议的轻客户端，由于与区块链状态同步时存在延迟，频繁的离线周期依然可能显著降低其性能. 随着区块链应用场景的多样化，轻客户端的需求日趋复杂，已广泛涵盖移动设备、物联网及去中心化金融（decentralized finance, DeFi）等领域[23-25]. 未来的研究趋势[26]不仅聚焦于优化存储与通信开销，还包括增强隐私保护能力、提升抗攻击能力及实现更灵活的验证策略.

本文旨在探索将轻客户端的部分计算负载委托给主链的参与者（共识层或全节点），并通过合理的激励机制来给予参与者报酬. 该策略不仅能有效降低轻客户端的存储和计算成本，还可以提升系统整体效率，从而为轻客户端验证和状态同步问题提供更优化的解决方案. DecXClient 与其他典型轻客户端协议的工作对比如表 1 所示.

**Table 1　Analysis and Comparison of Different Light Client Protocols**

**表 1　分析比较不同的轻客户端协议**

| 协议 | 共识 | 密码源语 | 向后兼容① | 预挖矿安全② | 证明负载 | 通信复杂度③ |
|---|---|---|---|---|---|---|
| BTC Relay[16] | PoW | SPV | √ | √ | 线性增长 | $O(C)$ |
| ETH Relay[17] | PoS | SPV | √ | √ | 线性增长 | $O(C)$ |
| NiPoPoW[11] | PoW | NiPoPoWs | × | √ | 对数增长 | $O(k\cdot\mathrm{polylog}(C))$ |
| FlyClient[12] | PoW | 默克尔山脉 | × | √ | 对数增长 | $O(k\cdot\mathrm{polylog}(C))$ |
| Cosmos IBC[23] | BFT/PoS | | √ | √ | | |
| zkBridge[20] | PoW/PoS | ZKP | √ | √ | 恒定大小 | $O(1)$ |
| BxTB[21] | PoW | Stateless SPV | √ | × | 恒定大小 | $O(k)$ |
| Glimpse[22] | PoW | Stateless SPV | √ | √ | 恒定大小 | $O(k)$ |
| DecXClient（本文） | PoW/PoS/BFT | 加权多签名 | √ | √ | 恒定大小 | $O(\lvert V\rvert)$ |

注：① 向后兼容性是指区块链系统中的新版功能能够与旧版本轻客户端正常协作的能力，即使部分轻客户端未升级，也能继续参与网络操作和共识. NiPoPoW 和 FlyClient 要求工作量保持恒定，或需要对比特币进行硬分叉，因此无法实现向后兼容性.

② 预挖矿安全旨在防范预先挖矿攻击. 预先挖矿攻击是指攻击者在区块链系统正式运行之前或网络对某些关键参数公开之前，利用其对链的先验知识和计算资源挖掘区块，构造有利于自身的链状态，从而在系统正式启动或交互过程中获得恶意竞争的优势.

③ $C$表示系统的生命周期，即区块链的长度；$k$指协议执行的共同前缀参数，通常为常数；$|V|$表示验证集的大小.

## 2　准备工作

### 2.1　预备知识

本节给出区块链安全属性、加权多重签名和轻客户端属性的概念.

#### 2.1.1　区块链安全属性

**定义 1.** 安全的区块链具备以下 3 个基本属性[27]：

1）共同前缀（common prefix）. 对于 2 条链 $C_1$、$C_2$ 以及任意 2 个在时隙 $s_1$ 和 $s_2$ 上的诚实全节点（满足 $s_1\leqslant s_2$），都有 $C_1^{\neg k}\preceq C_2^{\neg k}$ 成立. 这里 $k\in\mathbb{N}$ 是公共前缀参数，$C_1^{\neg k}$ 表示去除 $C_1$ 的最后 $k$ 个区块，$\preceq$ 表示前缀关系.

2）链质量（chain quality）. 对于任意诚实全节点输出的稳定链 $C^{\neg k}$，在任意连续的 $k$ 个区块中，恶意区块所占比例不超过 $\mu$. 其中 $\mu\in(0,1]$ 为链质量系数.

3）链成长（chain growth）. 设 $C_1$ 和 $C_2$ 分别是某诚实全节点在时隙 $s$ 与 $s+\varsigma$ 输出的链，则有 $|C_2|-|C_1|\geqslant\tau\cdot\varsigma$ 成立. 这里 $\tau\in(0,1]$ 是增长速率系数.

#### 2.1.2　加权多重签名

定义 $SG:=\{p_i\}_{i\in\{1,2,\cdots,m\}}$ 是一个包含 $m$ 个参与者的签名组，加权多重签名（weighted multi-signature）允许签名组成员 $p_i$ 按照各自的权重 $w_i$ 参与联合签名，联合签名的合法性依赖于参与者的总权重是否达到特定阈值. 形式化地，在一个 $(t,x)$ 加权多重签名方案中，$t$ 表示权重阈值，$x$ 表示签名组的总权重，则 $SG$ 的权重分布 $W_{SG}$ 满足：

$$(W_{SG}:=(w_1,w_2,\cdots,w_{\mathrm{m}}))\wedge\left(\sum W_{SG}=x\right).$$

一个有效加权多重签名则由子集 $sg\subseteq SG$ 生成，

其权重分布$W_{sg}$满足:

$$(W_{sg} := (w_1, w_2, \cdots, w_{|sg|})) \geqslant t.$$

加权多重签名具有良好的兼容性,几乎适用于所有主流公链协议.在本文中,我们选取一组特定矿工组成签名组,利用该签名组为轻客户端提供与链无关的交易证明.该方法仅要求矿工具备生成和验证签名的基本能力,从而降低接入门槛.当通过智能合约实现签名验证时,其兼容性表现更为出色.

2.1.3 轻客户端属性

**定义 2.** $B_i$为区块链$C$的第$i$个区块,则一个完整的轻客户端协议运行于轻客户端$LC$和一组以$C$为输入的全节点$FN$之间,并支持以下子协议:

1)初始化协议$Init(B_0) \to (st, \pi)$. $LC$以创世区块$B_0$为输入,通过与$FN$的交互,初始化其状态$st$,并获得一份证明$\pi$,以验证初始化的正确性.

2)状态更新协议$Upd(st) \to (st', \pi)$. $LC$通过与$FN$的交互,将其状态从$st$更新为$st'$(状态是指区块链$C$中一段连续的区块头$st := (HB_i, HB_{i+1}, \cdots, HB_n)$),以反映$C$的最新视图,并获得一份证明$\pi$.

3)状态验证协议$VrfySt(st, st', \pi) \to b$.$LC$验证证明$\pi$,确认状态$st'$是从$st$正确转换而来的,输出结果$b \in \{0,1\}$;

4)查询协议$Q(st, Tx) \to (r, \pi)$. $LC$针对交易集合$Tx$发起查询,其中$Tx$包含一组需查询的交易$(tx_1, tx_2, \cdots)$,每笔交易涉及到查询类型,例如账户当前余额、交易的发送者/接收者/金额等.客户端接收到查询结果$r$和证明$\pi$.若$Tx$不在$C$中,则返回错误;

5)查询验证协议$Vrfy(st, r, \pi) \to b$. $LC$验证证明$\pi$,确认查询结果$r$的正确性,输出结果$b \in \{0,1\}$.

### 2.2 系统模型

1)区块链.区块链$C$应满足 2.1.1 节所描述的区块链安全属性.此外,假设区块链满足轻客户端属性中定义的可信初始化协议$Init(B_0) \to (st, \pi)$,即$C$可以通过可信初始化设置生成创世区块$B_0$,并在随后的$x$个连续的区块中确定一组初始的验证集$V_1$,则轻客户端通过与$V_1$进行交互确定其初始状态$st$,并得到一份初始证明$\pi$,以确保初始化的正确性.

2)验证集.验证集$V$由一组生成区块链$C$中一连串区块的矿工组成,负责观察轻客户端发起的状态查询请求并为该状态生成证明$\pi$,该证明用于证实区块链状态$st$或交易$Tx$的有效性.

3)矿工.与大多数公链网络类似,矿工可以独立生成各自的公私钥对$(pk_i, sk_i)$,在本协议中该密钥用于在成为验证集成员后参与加权多重签名与验签.此外,每位矿工拥有唯一的账户地址$ad_i$,该地址通过类似 P2PKH 的地址生成机制派生于自身的公钥.

4)轻客户端.任何资源受限的节点都可以被选为轻客户端$LC$,它们通常不具备存储所有区块头和高负载通信的能力.$LC$通过与全节点交互来执行$Upd()$、$VrfySt()$、$Q()$和$Vrfy()$等功能,查询区块链状态.

5)密码学假设.假设被建模为随机预言机$O^{Hash}$的哈希函数$Hash()$具有抗碰撞性,且所有数字签名方案均满足选择消息攻击下的存在不可伪造性抗选择消息攻击(EUF-CMA)安全.

### 2.3 敌手模型

假设协议执行过程中存在概率多项式时间(probabilistic polynomial time, PPT)敌手$\mathcal{A}$,其目标是通过伪造交易证明欺骗轻客户端$LC$.敌手可在协议开始前静态腐化一定比例的矿工,利用受控矿工实施延迟攻击等,但根据定义 1 中定义的区块链属性(共同前缀、链质量和链成长),在满足诚实大多数的情况下,腐化矿工不足以破坏链$C$的安全属性.

假设该协议中轻客户端本身是诚实的,但$\mathcal{A}$可以截获任意$LC$的输入与输出.例如,$\mathcal{A}$可以将$LC$与区块链网络隔离,并将伪造的交易和证明发送给$LC$.此外,所有诚实的全节点都可以可靠地访问区块链的最新视图.

### 2.4 安全与效率目标

**定义 3.** 一个安全的轻客户端协议需具备以下安全目标[6]:

1)安全初始化与同步.此目标要求,在已知创世区块$B_0$的情况下,恶意全节点$\mathcal{A}$无法诱导诚实的$LC$接受伪造的区块链状态$C^*$,从而使$LC$基于伪造状态处理查询,即满足不等式:

$$Pr\begin{bmatrix} B_0; \\ \mathcal{A}(C^*) \text{ and LC run } Init(B_0), \\ Upd(st): \\ (C^* \neq C) \wedge VrfySt(B_0, st, \pi) \to 1 \end{bmatrix} \leqslant negl(\lambda).$$

2)安全验证.初始化完成后,恶意敌手$\mathcal{A}$无法诱导$LC$接受伪造的交易或账户状态.例如,$\mathcal{A}$不能使$LC$相信未经验证或伪造的交易存在于区块链中,或接受错误的账户状态,还包括说服$LC$接受伪造的不存在证明,即满足不等式:

$$Pr\begin{bmatrix} B_0; \\ \mathcal{A}, Init(), Upd(), Q(): \\ \exists Tx \notin C \wedge \\ Q(Tx, st) \to (r, \pi) \wedge \\ Vrfy(r, \pi) \to 1 \end{bmatrix} \leqslant negl(\lambda).$$

**定义 4.** 定义$|C|$表示区块链的大小(即区块数量)，则一个轻客户端协议在满足定义 3 中安全目标的前提下需具备以下效率目标：

1)高效初始化. 初始化协议$Init()$和更新协议$Upd()$的计算和通信成本相对于$|C|$为次线性级.

2)高效存储. 轻客户端的存储成本(即状态大小)相对于$|C|$为次线性级.

3)高效通信. 状态更新协议$Upd()$和查询协议$Q()$的通信成本相对于$|C|$为次线性级，通信发生在$LC$和$FN$之间.

4)高效验证. 状态验证协议$VrfySt()$和查询验证协议$Vrfy()$的计算成本相对于$|C|$为次线性级.

### 2.5 协议概述

轻客户端通过下载区块链的部分数据并利用加密证明与区块链交互，适用于资源受限设备. 本文提出了 DecXClient, 这是一种通用且安全的轻客户端协议. 为便于理解后续协议细节，本节概述其核心设计.

DecXClient 致力于为轻客户端生成交易或状态的有效证明，使其即使在无可信全节点环境下，也能验证链上数据的确定性. 该协议基于区块链的链质量属性动态构建矿工验证集，保证在主链的任意连续$k$个区块中，诚实矿工比例至少为$\mu$，从而确保验证集内诚实成员的占比不低于$\mu$. 验证集负责识别主链上最终确定的区块，并为轻客户端发起的状态验证请求提供证明. 证明的生成依赖于验证成员收集达到门限的加权多重签名，权重按连续$k$个区块内单个矿工挖出的区块数分配. 轻客户端通过验证这些证明，保障从创世区块至签名区块范围内交易的完整性和正确性，有效防范恶意全节点伪造的交易证明，同时降低自身存储与通信开销. 为保障验证集活跃性并防止停滞，验证集定期更新，更新触发可为静态(如每$s$个时隙)和动态(由特定事件驱动). 最终，仅参与或贡献有效证明的活跃成员有资格获得奖励，激励其诚实持续参与.

DecXClient 的证明生成过程具备以下核心特性：无权限性、链无关性及低延迟性. 首先，验证集成员由矿工通过底层共识协议随机选取并周期轮换，无需可信第三方协调，确保无权限属性. 其次，基于链质量这一通用性质，协议兼容所有满足该属性的区块链系统(如 PoW、PoS 等)，实现链无关性. 最后，验证集成员可快速获取最新链视图并完成加权多重签名，轻客户端则通过非交互式本地验证，在无信任环境下独立确认最终链状态. DecXClient 高效实现了存储、通信及验证三大目标，且具备良好扩展性，适用于跨链互操作及复杂轻客户端需求.

## 3 DecXClient 协议

### 3.1 具体构造

在系统模型中，轻客户端通过可信初始化过程生成初始状态$st$，此处不再赘述. 本节重点介绍验证集构建机制、轻客户端查询协议及查询验证协议. 查询协议和查询验证协议是状态更新协议与状态验证协议在交易查询场景下的具体实现. 当涉及连续区块时，则被扩展为状态查询与状态验证. 二者的区别在于，状态相关功能(基于区块头)主要体现轻客户端核心能力，而交易相关功能是其细粒度表现，可进一步扩展至跨链场景. 为不失通用性，本节仅阐述查询协议与查询验证协议的执行流程. 协议的详细的算法描述和整体框架请分别参阅算法 1 与图 2.

**算法 1.** DecXClient 协议算法.

/*procedure VerificationSetConstr begin*/

① $(t,x) \leftarrow Initializing$;

② $W \leftarrow BlockWindow(B_{n-x+1},\cdots,B_n)$;

③ $ad_i \leftarrow Draw(B_i)$, where $B_i \in W$;

④ $w_i \leftarrow EvaluateWeight(ad_i)$ for each $ad_i$;

⑤ Checkout $x > k$ and $t > \lfloor(1-\mu)k\rfloor$;

⑥ $V \leftarrow ConstrV(v_1 \leftarrow ad_1,\cdots,v_m \leftarrow ad_m)$;

/*procedure Query begin*/

① $Broadcast\ Q(st,Tx)$ to network;

② for each $v_i \in V$ do

③ $r \leftarrow Evaluate(st,Tx)$;

④ $\sigma_i \leftarrow Sign(sk_i,r)$;

⑤ Broadcast $(\sigma_i,ad_i)$;

⑥ $S_{sb} \leftarrow Collect(\sigma_j)$ for each $j \neq i$;

⑦ $w_{sb} \leftarrow Evaluate\left(\sum_{\sigma_j \in S_{sb}} w_j\right)$;

⑧ Checkout $w_{sb} \geqslant t$;

⑨ $\pi \leftarrow GenProof(S_{sb},bv)$, Broadcast $(r,\pi)$;

⑩ end for

/*procedure QueryVerification begin*/

① $\{\sigma_1,\cdots,\sigma_m\} \leftarrow Draw(S_{sb})$;

② for each $bv[i] = 1$ do

③ Checkout $\sigma_i, w_{sb} \leftarrow w_{sb} + w_i$;

④ end for

⑤ Checkout $w_{sb} \geqslant t$;

⑥ $1 \leftarrow Vrfy(st,r,\pi)$;

1)验证集构建机制. DecXClient 协议的框架如

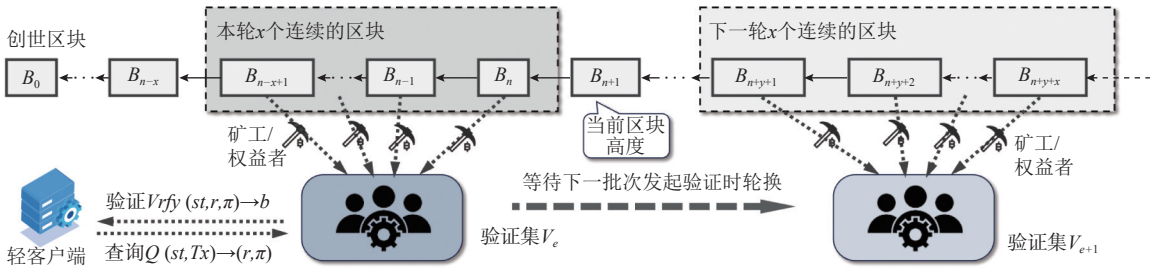



Fig. 2 Protocol framework of DecXClient

图 2 DecXClient 协议框架

图 2 所示. 假定当前的区块高度为$n+1$, 任何轻客户端均可在区块$B_{n+1}$后发起查询协议$Q(st,Tx)$. 此时, 系统将$B_{n+1}$之前生成的$x$个连续区块$(B_{n-x+1},\cdots,B_{n-1},B_n)$的矿工被定义为验证集成员, 随后每个成员初始化证明生成参数, 包括门限$t$和验证集权重分布, 每个成员$v_i$的权重大小由出块的个数决定. 轻客户端发布查询协议前, 只需提前获取$B_{n+1}$之前连续$x$个块的矿工的挖矿证明即可完成验证准备. 请注意, 在这里的连续$x$个区块中, 矿工可能是重复的, 因此, 验证集实际由$m<x$个成员组成. 参数$(t,x)$的选择依据链质量属性确定. 具体来说, 链质量意味着在链$C$的连续$k$个或者更多区块中, 诚实矿工的比率至少为$\mu$. 为了利用该属性, 轻客户端需满足$x>k$和$t>\lfloor(1-\mu)k\rfloor$的双重约束条件. 该设计具有配置灵活性, $(t,x)$允许根据不同公链的特性在可用性与安全性之间进行动态权衡.

2)查询协议. 在当前区块$B_{n+1}$确定验证集$V$后, 轻客户端可以向区块链网络发起查询请求$Q(st,Tx)$. 接收到请求后, 验证集成员根据请求生成查询结果和证明$(r,\pi)$. 具体而言, 每个验证集成员$v_i\in V$利用各自私钥$sk_i$对查询结果$r$进行加权签名, 并将签名通过网络广播. 随后$v_i$通过公共验证集网络(EIP-4337 中的公共内存池)收集其他成员的签名. 定义$S_{\text{sub}}$是$v_i$从$V$中收集的一组多重签名, $w_{\text{sub}}$为$S_{\text{sub}}$的总权重, 即集合中所有签名$\sigma_i\in S_{\text{sub}}$对应的权重之和. 当$v_i$收集到的签名权重满足$w_{\text{sub}}\geqslant t$时(其中$t$为阈值), 即可构造完整的证明$\pi:=(S_{\text{sub}},\boldsymbol{bv})$, 其中$\boldsymbol{bv}$是一个位向量, 用于指示验证者(按索引)参与了签名. 最终, 从$v_i$将$(r,\pi)$广播至网络, 以便轻客户端执行查询验证.

3)查询验证协议. 在获取到$(r,\pi)$后, 轻客户端通过检验证明$\pi$的有效性来更新最新状态. 具体地, 轻客户端从$\pi$中解析出$(S_{\text{sub}},\boldsymbol{bv})$, 其中位向量$\boldsymbol{bv}$指示的验证者地址为$ad_i$; 其次, 通过$ad_i$解析出匹配的公钥$pk_i$. 随后, 通过$pk_i$验证$S_{\text{sub}}$中每个签名$\sigma_i$的正确性. 最后, 检验$S_{\text{sub}}$的签名权重达到阈值$t$. 若以上环节都通过, 则$\pi$被证明是有效的, 轻客户端完成验证$Vrfy(st,r,\pi)$, 并更新状态或视图.

需要注意的是, 轻客户端所查询的状态或交易必须满足其所在区块位于当前最新区块$B_{n+1}$之前的$k$个区块范围内, 验证集中的诚实节点才会在确认其有效性后生成相应证明. 典型参数配置为: 比特币网络通常设定$k=6$, 而以太坊则采用$k=12$, 具体取值可根据不同区块链的共同前缀参数进行估算. 这一机制实质上为交易提供了一种“共识级不可逆性”保障, 即只有当交易在区块链中获得深度的确认后, 才能被视为稳定且不可回滚的状态, 从而有效抵御因链重组可能引发的安全风险.

与以太坊的信标链类似, 验证集$V_e$为当前周期提供给轻客户端的证明, 可以被定义为下一周期的“快照”, 因此, 轻客户端无需存储当前周期之前的完整区块头数据, 仅保存当前状态证明即可.

在敌手模型中, 假设攻击者能够拦截轻客户端的通信渠道, 进而阻止证明或区块链视图传达给轻客户端, 使其无法验证状态的有效性或及时更新链上视图. 由于这类行为属于拒绝服务攻击的范畴, 因此超出了本文的讨论范围. 然而, 基于上述前提, 在验证集保持安全可信的前提下, 攻击者无法伪造有效状态的证明. 因此, 轻客户端不会错误地将未确认的状态或非最终的链上视图认定为有效或最终结果. 具体的安全证明请参阅第 4 节.

### 3.2 验证集轮换

尽管验证集的构造机制有效保障了选出的验证集保持诚实多数, 从而确保系统安全性, 但验证集成员在被选定后长期固定, 可能导致去中心化程度不足, 进而产生安全和性能风险. 为此, 引入验证集的动态轮换机制, 以增强区块链系统中轻客户端验证的活性、去中心化水平及鲁棒性, 同时防止恶意参与者通过长期控制特定验证集而发起针对性的攻击.

图 2 右侧展示了验证集重新选举的流程. 设当前

验证周期$Epoch_e$的验证集为$V_e$，该验证集在当前区块$B_{n+1}$产生后的一段连续时隙内，为轻客户端持续提供可信且有效的区块状态证明，直至新一轮验证集$V_{e+1}$被正式选举产生. 为实现验证集动态管理，区块链$C$被划分为若干验证周期，每个周期由对应验证集负责生成并维护该周期内的跨链状态证明，确保验证流程的连贯性与安全性.

验证周期的切换策略主要分为2类：静态切换规则和动态切换规则. 静态规则基于预先设定的固定时间间隔或区块数（例如以太坊同步委员会机制中的256个周期触发1次重组），周期长度可根据具体需求灵活调整，以权衡去中心化与系统性能. 动态规则则允许轻客户端根据实际使用场景及安全需求，主动发起验证集切换请求. 具体而言，轻客户端可在新区块$B_{n+y+x+1}$上提交包含新验证集版本号及相应区块序列公钥集合的变更请求交易$TxEpoch_{e+1}$，用于替换当前验证集.

为保证验证集轮换过程的安全性与一致性，该变更请求需获得当前验证集$V_e$中成员的加权多重签名确认. 此过程不仅防止恶意节点操纵验证集切换，还保障新旧验证集之间状态的平滑过渡，降低系统遭受攻击的风险. 此外，基于加权多重签名的验证集确认机制，能够有效减轻网络负载，提升节点状态同步的效率与可靠性.

### 3.3 阈值设置

在DecXClient中，协议通过利用链质量属性构造验证集，其中验证集的总权重为$x$，权重阈值需满足$t \geqslant \lfloor (1-\mu)x \rfloor$. 本节详细分析了$(t,x)$的大小，以确保在安全的前提下适配不同的场景需求.

定义矿工参与连续挖掘$x$个区块的过程建模为伯努利随机过程，该过程中诚实矿工挖掘的区块数为随机变量$X$. 在随机过程的每个伯努利实验中，定义一个布尔随机变量$b_i$，则$X := \sum_{i=1}^{x} b_i$. 当区块$B_i$由恶意矿工创建时，$b_i = 1$，否则，$b_i = 0$. 定义单个矿工恶意的概率为$\rho$，即$\rho = Pr[b_i = 1]$，那么$X$的期望为$E(X) = n(1-\rho)$. 为评估验证集的安全保障，采用累积二项分布计算恶意矿工不超过$\delta$的概率为：

$$Pr[X \leqslant \delta] = \sum_{i=1}^{\delta} \binom{x}{i} \rho^i (1-\rho)^{x-i}. \tag{1}$$

图3展示了不同阈值和权重比值下，验证集安全概率随验证集大小的变化情况. 在固定验证集权重大小的情况下，当恶意矿工比例不超过$\delta$时，恶意矿工提供验证结果的比例也不会超过$\delta$. 这样，一旦验证结果的比例超过$\delta$，则被视为最终的正确结果. 从图3可以看出，随着$t/x$比值的增加，验证集的安全概率也同步增加. 此外，验证集的安全概率还与验证集的大小呈正相关. 例如，当总权重为99时，验证集中恶意成员阈值和权重比小于1/2的概率达到$1-0.1^5$. 而在由64个连续区块的矿工组成的PoS区块链中，假定诚实节点持有的股份超过总股份的3/4时，恶意验证者超过2/3的概率更是低于$2^{-42}$. 基于式(1)，协议可以将阈值$t$设置为48或者更高，以便达到或超越与以太坊信标链相同的安全等级.

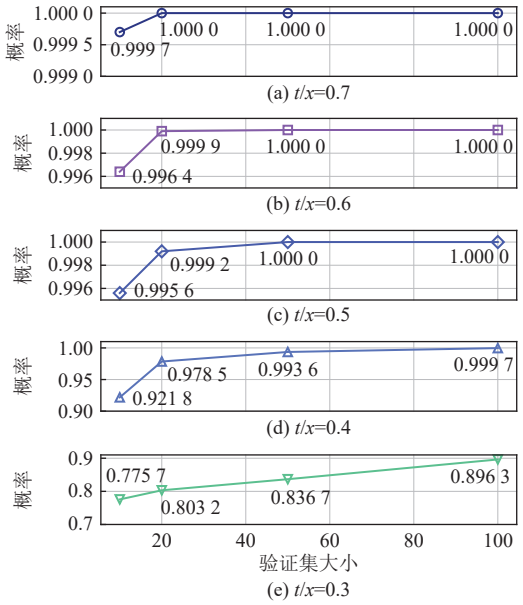



Fig. 3 Security probability of validation set under different threshold weight ratios and validation set sizes

图3 不同阈值权重比和验证集规模下验证集的安全概率

### 3.4 激励机制

在本协议设计中，轻客户端通过验证集生成交易证明来确保链上交易的安全可信性. 但需要强调的是，如果验证者在提供查询服务或生成状态证明时无法获得相应回报，其理性选择很可能会倾向于拒绝参与或实施恶意行为. 因此，建立有效的激励机制至关重要，其核心目标是使诚实行为成为验证者的最优策略选择（即达到纳什均衡状态）. 这一激励机制的设计理念与Cosmos[28]、Polkadot[29]等主流区块链网络保持高度一致. 以Cosmos为例，其中继节点通过收取交易手续费和质押Atom代币获得双重收益，从而有效保障网络的稳定运行和安全性.

验证者的行为可以从2个维度进行分析：一是与挖矿相关的行为，这由底层区块链的共识机制决定，其奖励机制与轻客户端验证过程相独立，所有成功

出块的矿工都能直接获得系统发放的 Coinbase 奖励;二是与投票相关的行为,这直接关系到轻客户端证明的生成,旨在激励验证集成员保持在线状态并参与加权多重签名过程. 在实际应用中有多种广泛采用的奖励来源方案可供选择,包括依赖协议防止数据欺骗攻击的验证者支付费用、专为协议发行的 ERC-20 代币等,这些设置可通过智能合约实现. 奖励分配严格遵循多重签名权重原则,签名权重越高的验证者获得的奖励越多. 只有在规定时间窗口内积极参与签名的验证者才有资格分享该周期的证明手续费,而未参与者将只能获得基础的挖矿奖励.

DecXClient 还可引入更细粒度的奖惩机制,激励验证者及时响应签名请求,同时惩罚包括验证者和交易发起方在内的潜在拜占庭行为. 通过灵活调整参与阈值,协议能够容忍一定比例的节点意外故障. 这种设计不仅显著提高了轻客户端验证的可靠性,还为构建去中心化的高安全性客户端通信体系提供了制度保障. 关于该激励机制在纳什均衡下的安全性分析,详见本文第 4.2 节的描述.

## 4 安全性分析

### 4.1 轻客户端安全分析

**引理 1(验证集安全).** 设区块链 $C$ 满足定义 1 中的区块链安全属性. 若敌手控制权重低于阈值 $t$,则验证集 $V$ 中诚实成员权重超越恶意权重的概率满足:$Pr[w_{\mathrm{h}} > w_{\mathrm{m}}] \geqslant 1-negl(\lambda)$,其中 $w_{\mathrm{h}}$、$w_{\mathrm{m}}$ 分别表示验证集中诚实方与恶意方的总权重.

证明. 已知 $V$ 由连续 $x$ 个区块的矿工组成,因此 $V$ 的总权重为 $x$,且 $x > k$. 根据定义 1 中的链质量属性,诚实矿工权重满足 $w_{\mathrm{h}} \geqslant \lceil \mu x \rceil \geqslant \lceil \mu k \rceil$. 因权重阈值 $t$ 满足 $t > \lfloor (1-\mu)x \rfloor \geqslant \lfloor (1-\mu)k \rfloor$,故有:

$$Pr\left[w_{\mathrm{h}} \leqslant X - t \leqslant \lceil \mu k \rceil\right] \leqslant negl(\lambda),$$

此即表明 $w_{\mathrm{h}} > w_{\mathrm{m}}$ 的概率可忽略且无限趋近 1. 证毕.

**引理 2(安全初始化与同步).** 设区块链 $C$ 满足定义 1 中的区块链安全属性,且任意 PPT 敌手 $\mathcal{A}$ 所控制的总权重小于权重阈值 $t$,若加权多重签名方案满 EU-CMA 安全性,则 DecXClient 协议满足定义 3 中的安全初始化与同步要求.

证明. 设 $\mathcal{A}$ 为任意 PPT 全节点敌手,$C$ 为合法主链,$C^*$ 为由 $\mathcal{A}$ 构造的伪造链. 在系统模型中,假设主链 $C$ 可通过可信初始化生成创世区块 $B_0$,并在其后的 $x$ 个连续区块中确定初始验证集 $V_1$,随后系统执行初始化过程 $Init(B_0) \to (st,\pi)$. 若 $\mathcal{A}$ 企图诱导 $LC$ 接受伪造链 $C^*$,则必须伪造初始状态的证明 $\pi$. 这意味着 $\mathcal{A}$ 需要伪造验证集 $V_1$ 的加权多重签名. 根据引理 1,若 $C$ 满足链质量属性,且 $\mathcal{A}$ 所掌握的权重小于阈值 $t$,则 $V_1$ 中诚实验证者占据权重多数. 因此,$\mathcal{A}$ 伪造有效签名的概率可忽略.

综上可得:

$Pr[(C^* \neq C) \wedge VrfySt(B_0, st, \pi) \to 1] \leqslant negl(\lambda)$. 证毕.

**引理 3(安全验证).** 设区块链 $C$ 满足定义 1 中的区块链安全属性,且敌手拥有的总权重小于权重阈值 $t$,加权多重签名是 EU-CMA 安全的,则 DecXClient 满足定义 3.2 中安全验证.

证明. 设 $\mathcal{A}$ 为任意 PPT 敌手,若 $\mathcal{A}$ 能构造 $Tx' \notin C$ 使得 $Q(Tx', st) \to (r,\pi)$ 返回一个可被接受的证明,则 $\mathcal{A}$ 需要满足 2 个条件:

条件①构造的 $Tx' \neq Tx$ 且满足存在 $tx_i \in Tx$, $tx_i' \in Tx'$,使得 $Hash(tx_i') = Hash(tx_i)$,即能找到新的一组交易来替换链上交易而不改变 Merkle 树结构;

条件②验证集执行 $Q(Tx', st) \to (r,\pi)$ 生成的证明是有效的.

针对条件①,$\mathcal{A}$ 需要访问预言机 $O^{Hash}$ 来构造相同哈希值,这与本文假设的 $Hash()$ 具有抗碰撞性相矛盾. 针对条件②,$\mathcal{A}$ 需要访问伪造 $V$ 的加权多重签名. 由引理 1 可知,若 $C$ 满足链质量属性,且 $\mathcal{A}$ 所掌握的权重小于阈值 $t$,则 $V$ 中诚实验证者占据权重多数. 因此,$\mathcal{A}$ 无法成功伪造有效多重签名.

综上可得:

$Pr[\exists Tx \notin C \wedge Q(Tx, st) \to (r,\pi) \wedge Vrfy(r,\pi) \to 1] \leqslant negl(\lambda)$

由引理 2 和引理 3 可知,DecXclient 满足定义 3 中轻客户端所具备的安全目标. 上述证明中,安全参数 $\lambda$ 实际由验证集 $(t,x)$ 的大小决定,具体参考第 3.3 节阈值设置. 证毕.

### 4.2 博弈安全分析

本节从博弈论视角建模验证者策略,通过量化奖惩机制参数,并分析激励机制下的纳什均衡,证明合理设计可使诚实参与成为验证者的最优策略.

设当前验证集为 $V = \{v_1, v_2, \cdots, v_n\}$,系统发起一笔轻客户端查询,期望收集一组加权签名,满足总权重超过门限 $t$. 每个验证者 $v_i$ 拥有权重 $w_i$,则验证者可选择以下 3 种策略:*Sign*-参与验证、*Skip*-离线或拒绝响应、*Fake*-返回错误验证结果. 为每个策略定义如下效用函数,衡量验证者在该策略下的期望收益.

1)诚实参与效用 $U_i(Sign)$. 奖励分配需体现权重差异与边际贡献,采用“基础收益+加权分成”形式,则

$$U_i(Sign)=R_{\text{mining}}+R_{\text{total}}\cdot\frac{w_i}{w_{\text{h}}}-k\cdot w_i,$$

其中，$R_{\text{mining}}$是挖矿奖励，$R_{\text{total}}$是按权重占比分配的投票总奖励池，$c_i=k\cdot w_i$是签名带来的计算与通信成本.

2）离线不响应效用$U_i(Skip)$. 离线者仅获得挖矿基础奖励$R_{\text{mining}}$，无加权分成或计算成本，故$U_i(Skip)=R_{\text{mining}}$.

3）作恶效用$U_i(Fake)$. 作恶者可通过伪造错误证明获取奖励，但面临惩罚风险，故有

$$U_i(Fake)=(1-p_i)\left(R_{\text{mining}}+R_{\text{total}}\cdot\frac{w_i}{w_{\text{h}}}\right)-p_i\cdot\alpha\cdot\beta\cdot w_i-k\cdot w_i.$$

若作恶者成功伪造证明，则可获得基础收益和加权分成；若未成功，则会罚没资产$p_i\cdot\alpha\cdot\beta\cdot w_i$，其中$\alpha\in(0,1]$为罚没比例，$p_i$为成功伪造概率（由引理 1 可知在满足验证集安全的情况下该概率可忽略不计）.

设全诚实策略组合为$\boldsymbol{a}=(Sign_1,Sign_2,\cdots,Sign_n)$，则需要满足对任意$v_i$，偏离策略$Sign_i$的效用不高于$U_i(Sign)$，即需要满足$U_i(Sign)\geqslant U_i(Skip)\geqslant U_i(Fake)$，代入效用函数可得：

$$R_{\text{mining}}+R_{\text{total}}\cdot\frac{w_i}{w_{\text{h}}}-k\cdot w_i\geqslant R_{\text{mining}},$$

$$R_{\text{mining}}\geqslant(1-p_i)\left(R_{\text{mining}}+R_{\text{total}}\cdot\frac{w_i}{w_{\text{h}}}\right)-p_i\cdot\alpha\cdot\beta\cdot w_i-k\cdot w_i.$$

整理化简可得诚实均衡的边际条件为：

$$k\leqslant\frac{R_{\text{total}}}{w_{\text{h}}}\leqslant\frac{p_i\alpha\beta}{1-p_i}.$$

因此，当协议设计满足该边际条件时，对任意验证者$v_i\in V$，$Sign$均为其最优响应策略，策略组合$\boldsymbol{a}$为该博弈的纯策略纳什均衡.

## 5 效率分析

本节重点分析所提协议的证明生成与验证延迟，并将其与主流轻客户端协议 BTC/ETH Relay、NiPoPoW、FlyClient 和 zkBridge 进行空间效率和时间效率对比. 实验基于概念验证（proof of concept, PoC）实现，具体来说：利用以太坊虚拟机（EVM）公共 API（eth_getBlockByNumber）模拟轻客户端获取区块链最新视图的过程，确保实验环境与真实环境相似；开发 2 个独立程序分别承载 DecXClient 的查询协议的功能与查询验证协议的功能，以评估模块化设计性能；验证集成员用于生成和验证证明的密钥对均由以太坊 API（eth_accounts 配合 web3.eth.accounts.create）生成，格式遵循 secp256k1 标准；采用多线程程序模拟验证集成员行为，每个线程对应一个独立的验证集节点，执行证明生成和响应查询等操作. 为评估协议在不同网络规模、节点分布和安全强度下的性能，实验默认采用的参数$(t,n)$分别设置为（24, 32）、（48, 64）和（96, 128），并考虑节点间的通信延迟. 由 3.3 节可知，不同阈值下的安全性强度不同，例如在阈值（48, 64）设定下恶意验证者超过2/3的概率低于$2^{-42}$. 所有测试均在本地环境中进行，硬件配置为 Intel® Core™ i7-12700KF CPU@3.60 GHz, 8 GB RAM, Ubuntu 18.04 实例.

### 5.1 通信与存储负载

通信与存储效率直观体现为证明生成的规模. 如表 2 所示，DecXClient 与其他主流协议相比，在证明规模与区块长度之间的关系展现出范式级优势：传统方案如 BTC Relay 和 ETH Relay 的证明规模随区块长度线性增长，而 NiPoPoW 与 FlyClient 呈对数（次线性）增长. 相比之下，DecXClient 的证明规模仅取决于$|S_{\text{sub}}|$的大小；在固定参数$(t,n)$时，证明规模可近似计算为常数级（比特尺度）. 尽管 zkBridge 的证明规模为更低常数，但其高昂的零知识证明生成开销及复杂的智能合约环境限制了其应用场景.

**Table 2 Comparison of Proof Paradigm**

**表 2 证明范式对比**

| 协议 | 范式大小 |
|---|---|
| BTC/ETH Relay | $\lvert BH\rvert\cdot L+\text{lb}_2(\lvert BT\rvert)\cdot\lvert H\rvert$ |
| NiPoPoW | $\log_{1/\theta}(2)\xi\cdot(\text{lb}_2(L)+1)\cdot\lvert B\rvert+\text{lb}_2(L)\cdot\lceil\text{lb}_2(\text{lb}_2(L,2),2)\rceil\cdot\lvert H\rvert$ |
| FlyClient | $\xi\log_{1/\theta}(2)\ln(L)\cdot(\lvert B\rvert+\lvert H\rvert)$ |
| Cardano | |
| DecXClient | $\lvert S_{\text{sub}}\rvert+\lvert bv\rvert$ |

注：$|B|$为区块大小，$|BH|$为区块头大小，$|BT|$为一个区块中所含平均交易个数，$|H|$为哈希大小，$L$为区块链长度，$\theta$为敌手计算比例. 关于 Cardano 的范式请参阅文献 [30].

图 4 展示了典型方案与 DecXClient 的证明规模随区块长度变化趋势（因 BTC/ETH Relay 规模过大未显示）. 为便于对比，Cardano 委员会规模默认与本方案验证集一致. 当区块链总长度为$6\times10^6$区块时，DecXClient 的状态证明规模仅为 1 248 B，相较之下，其证明大小仅为 NiPoPoW 的 1/4 206，FlyClient 的 1/2 765，以及 Cardano 的约 40%. 需要注意的是，实际场景中 Cardano 委员会大小通常设置为 512，这会导致其证明规模进一步增大.

结果表明，DecXClient 满足定义 4 的轻客户端效率目标，其存储、通信及验证成本均与区块链长度无关，且显著优于线性复杂度与次线性复杂度的方案.

### 5.2 证明生成与验证延迟

本实验基于同步网络模型模拟真实公链网络环

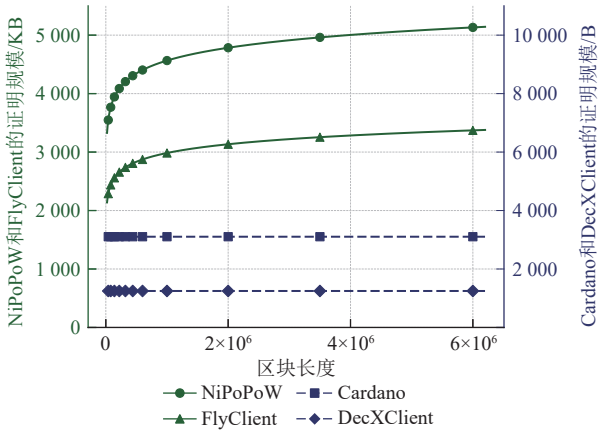



Fig. 4 Proof size for four typical protocols with different blockchain lengths

图 4 不同区块链长度下 4 种典型协议的证明大小

境, 在允许有限响应延迟的前提下, 对不同固定时间间隔(5 ms、10 ms、20 ms、50 ms、100 ms)下验证集生成有效状态证明的时间成本展开量化测量. 为简化实验设定, 假定所有验证者被赋予等效签名权重(权重值统一设为 1), 以此评估系统达到签名门限所需的实际延迟表现.

图 5 呈现了 3 组不同签名间隔条件下, 前$t$个验证者完成证明生成的延迟特性. 例如, 在门限参数为(48, 64)的情况下, 当签名间隔为 5 ms 时, 平均延迟约为 266.4 ms; 间隔增加至 50 ms 时, 平均延迟升至 1 330 ms, 最大延迟约为 2 500 ms; 当间隔进一步扩大为 100 ms 时, 平均延迟约为 2 500 ms, 最大延迟攀升至 4 850 ms. 通过分别对比 3 组阈值参数下的数据可以观察到, 在不同阈值参数下延迟随签名间隔的增大均呈现近线性增长趋势. 此外, 相同阈值参数和签名间隔下的最小延迟和最大延迟存在明显差异. 而从实际部署场景来看, 比特币等公链由于节点的地理分散性, 其网络延迟可能高于平均延迟; 而以太坊信标链等采用委员会定期检查点机制, 通过在固定时间窗口集中执行签名操作, 其时间间隔通常显著低于本平均延迟.

图 6 对比分析了在不同验证集大小下 DecXClient 的最大延迟与基于 ZKP 的轻客户端协议 zkBridge 的平均延迟. 结果显示, 在验证集规模为 32, 64 和 128 的测试场景下, DecXClient 在 3 种网络环境下均具备优势. 在最高签名间隔下的延迟范围为 3.99~7.39 s 时, 较 zkBridg 降低 69.9%~78.8%, 最低签名间隔下的延迟仅 0.43~0.92 s, 实现了 95.0%~95.6% 的降幅; 同时, 随着验证集从 32 扩大至 128, DecXClient 的 3 种配置的延迟增幅分别为 3.40 s、2.94 s 和 0.49 s, 展示出协议

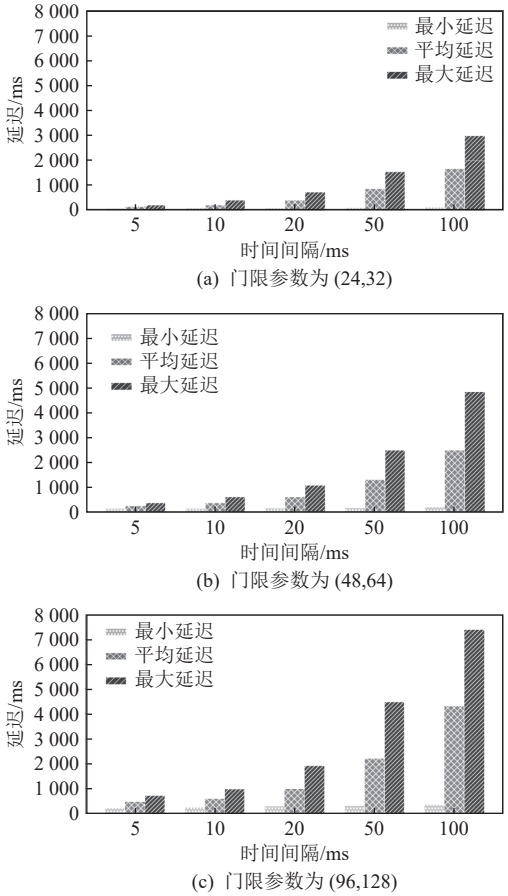



Fig. 5 Impact of different signature interval pairs on proof delay under three sets of threshold

图 5 3 组阈值下不同签名间隔对证明延迟的影响

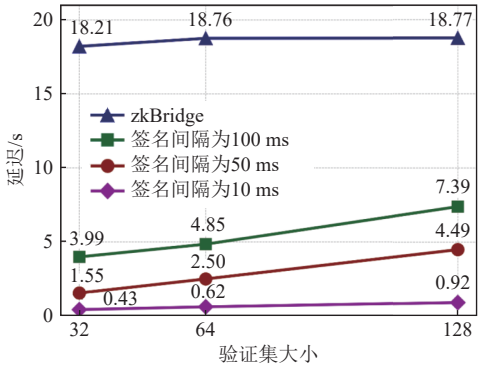



Fig. 6 Comparison of proof generation latency between DecXClient and zkBridge

图 6 DecXClient 和 zkBridge 证明生成延迟的比较

良好的扩展性.

验证延迟方面(忽略信道通信时间, 仅统计本地验证时间), 实验测量了不同$t$值下的处理耗时. $t = 48$时验证耗时约 97.6 ms; $t = 128$时验证耗时增至 178.3 ms. 尽管阈值提升增加了独立签名验证负载, 但整体验证延迟仍保持低位, 远低于证明生成及网络传输耗时.

综上, DecXClient 在证明生成与验证阶段均具备

低延迟特性. 在实际部署中, 权重分布不均可能会减少有效验证者数量, 进一步缩短证明生成时间, 印证了协议的实用性与部署可行性.

## 6 结论与下一步工作

本文提出一种通用且安全的轻客户端协议 DecXClient, 该协议解决了传统轻客户端存储和通信开销大、易受恶意节点攻击以及共识机制兼容性差等问题. 基于链质量属性, DecXClient 设计了动态验证集选举和加权多重签名机制, 实现无权限、链无关且低延迟的证明生成. 同时, 通过激励机制保障验证者诚实参与, 提升了系统的安全性和实用性.

未来工作将重点探索 DecXClient 在跨链环境中的应用, 特别是中继链[31] 和侧链[32] 场景. 作为跨链资产转移和状态同步的关键组件, 轻客户端的效率与安全性直接决定跨链系统的整体性能与可靠性. 比如, 针对侧链扩容需求, DecXClient 可通过轻客户端跨链验证实现主侧链之间的可信状态桥接, 保障跨链资产转移的强原子性及防范双重支付攻击. 凭借本协议的实用性能, DecXClient 有望推动区块链验证机制向更轻量、安全与去中心化的方向发展.

**作者贡献声明:** 邓智洪提出方案整体设计并撰写论文; 唐春明、李涛涛负责方法设计、提出指导意见并修改论文; 何雅萱、曾志康负责部分算法思路和实验方法; 陈祺提出指导意见并修改论文.

## 参 考 文 献

[1] Vranken H. Sustainability of bitcoin and blockchains[J]. Current Opinion in Environmental Sustainability, 2017, 28: 1−9

[2] Zheng Zibin, Xie Shaoan, Dai Hongning, et al. Blockchain challenges and opportunities: A survey[J]. International Journal of Web and Grid Services, 2018, 14(4): 352−375

[3] Deng Zhihong, Tang Chunming, Li Taotao, et al. Enhancing blockchain cross chain interoperability: A comprehensive survey[J]. arXiv preprint, arXiv: 2505.04934, 2025

[4] Suratkar S, Shirole M, Bhirud S. Cryptocurrency wallet: A review[C]// Proc of 2020 4th Int Conf on Computer, Communication and Signal Processing (ICCCSP). Piscataway, NJ: IEEE, 2020: 1−7

[5] Houy S, Schmid P, Bartel A. Security aspects of cryptocurrency wallets—a systematic literature review[J]. ACM Computing Surveys, 2023, 56(1): 1−31

[6] Chatzigiannis P, Baldimtsi F, Chalkias K. Sok: Blockchain light clients[C]//Proc of Int Conf on Financial Cryptography and Data Security. Cham: Springer, 2022: 615−641

[7] Paavolainen S, Carr C. Security properties of light clients on the ethereum blockchain[J]. IEEE Access, 2020, 8: 124339−124358

[8] Xie Qingqing, Dong Fan. Survey on lightweight blockchain technology[J]. Journal of Software, 2022, 34(1): 33−49 (in Chinese) (谢晴晴, 董凡. 轻量级区块链技术综述[J]. 软件学报, 2022, 34(1): 33−49)

[9] Chaidos P, Kiayias A. Mithril: Stake-based threshold multisignatures[C]//Proc of Int Conf on Cryptology and Network Security. Singapore: Springer Nature Singapore, 2024: 239−263

[10] Kiayias A, Lamprou N, Stouka A P. Proofs of proofs of work with sublinear complexity[C]//Proc of Financial Cryptography and Data Security. Berlin: Springer, 2016: 61−78

[11] Kiayias A, Miller A, Zindros D. Non-interactive proofs of proof-of-work[C]//Proc of 24th Int Conf on Financial Cryptography and Data Security. Cham: Springer, 2020: 505−522

[12] Bünz B, Kiffer L, Luu L, et al. FlyClient: Super-light clients for cryptocurrencies[C]//Proc of 2020 IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2020: 928−946

[13] Westerkamp M, Diez M. Verilay: A verifiable proof of stake chain relay[C]//Proc of 2022 IEEE Int Conf on Blockchain and Cryptocurrency (ICBC). Piscataway, NJ: IEEE, 2022: 1−9

[14] Thibault L T, Sarry T, Hafid A S. Blockchain scaling using rollups: A comprehensive survey[J]. IEEE Access, 2022, 10: 93039−93054

[15] Loporchio M, Bernasconi A, Maesa D D F, et al. A survey of set accumulators for blockchain systems[J]. Computer Science Review, 2023, 49: 100570

[16] Wright C S. Bitcoin: A peer-to-peer electronic cash system[J]. SSRN Electronic Journal, 2008, 3440802: 10−2139

[17] Frauenthaler P, Sigwart M, Spanring C, et al. ETH Relay: A cost-efficient relay for Ethereum-based blockchains[C]//Proc of 2020 IEEE Int Conf on Blockchain (Blockchain). Piscataway, NJ: IEEE, 2020: 204−213

[18] Kiayias A, Leonardos N, Zindros D. Mining in logarithmic space[C]//Proc of the 2021 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2021: 3487−3501

[19] Belchior R, Dimov D, Karadjov Z, et al. Harmonia: Securing cross-chain applications using zero-knowledge proofs[J]. IEEE Transactions on Dependable and Secure Computing. http://tinyurl.com/zkharmonia

[20] Xie Tiancheng, Zhang Jiaheng, Cheng Zerui, et al. zkBridge: Trustless cross-chain bridges made practical[C]//Proc of the 2022 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2022: 3003−3017

[21] Barbàra F, Schifanella C. BxTB: Cross-chain exchanges of bitcoins for all bitcoin wrapped tokens[C]//Proc of the 4th Int Conf on Blockchain Computing and Applications (BCCA). Piscataway, NJ: IEEE, 2022: 143−150

[22] Scaffino G, Aumayr L, Avarikioti Z, et al. Glimpse: {On-Demand} {PoW} Light Client with {Constant-Size} Storage for {DeFi}[C]// Proc of the 32nd USENIX Security Symp (USENIX Security 23). Berkeley, CA: USENIX Association, 2023: 733−750

[23] Qasse I A, Abu Talib M, Nasir Q. Inter blockchain communication: A survey[C]//Proc of the ArabWIC 6th Annual Int Conf Research Track. New York: ACM, 2019: 1−6

[24] Perazzo P, Xefraj R. SmartFly: Fork-free super-light Ethereum classic clients for Internet of things[J]. IEEE Internet of Things Journal, 2024, 11(9): 15348−15358

[25] Zetzsche D A, Arner D W, Buckley R P. Decentralized finance[J]. Journal of Financial Regulation, 2020, 6(2): 172−203

[26] Bellaj B, Ouaddah A, Bertin E, et al. Drawing the boundaries between blockchain and blockchain-like systems: A comprehensive survey on distributed ledger technologies[J]. Proceedings of the IEEE, 2024, 112(3): 247−299.

[27] Garay J, Kiayias A, Leonardos N. The bitcoin backbone protocol: Analysis and applications[J]. Journal of the ACM, 2024, 71(4): 1−49

[28] Kwon J, Buchman E. Cosmos whitepaper[J]. A Netw. Distrib. Ledgers, 2019, 27: 1−32

[29] Wood G. Polkadot: Vision for a heterogeneous multi-chain framework[J]. White Paper, 2016, 21(2327): 4662

[30] Gaži P, Kiayias A, Zindros D. Proof-of-stake sidechains[C]//Proc of 2019 IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2019: 139−156

[31] Li Taotao, Yang Changlin, Yang Qinglin, et al. Metaopera: A cross-metaverse interoperability protocol[J]. IEEE Wireless Communications, 2023, 30(5): 136−143

[32] Li Taotao, Huang Huawei, Yin Lingyuan, et al. USSC: Universal and storage-efficient sidechains[C]//Proc of the 44th Int Conf on Distributed Computing Systems (ICDCS). Piscataway, NJ: IEEE, 2024: 379−390



**Deng Zhihong**, born in 1996. PhD candidate. His main research interests include blockchain and cyberspace security.

**邓智洪**, 1996 年生. 博士研究生. 主要研究方向为区块链、网络空间安全.



**Tang Chunming**, born in 1972. PhD, professor, PhD supervisor. His main research interests include cyberspace security and applied cryptography.

**唐春明**, 1972 年生. 博士, 教授, 博士生导师. 主要研究方向为网络空间安全、应用密码学.



**Li Taotao**, born in 1992. PhD, lecturer. His main research interests include blockchain and cyberspace security.

**李涛涛**, 1992 年生. 博士, 讲师. 主要研究方向为区块链、网络空间安全.



**He Yaxuan**, born in 2001. Master candidate. Her main research interests include blockchain and cyberspace security.

**何雅萱**, 2001 年生. 硕士研究生. 主要研究方向为区块链、网络空间安全.



**Zeng Zhikang**, born in 1999. PhD candidate. His main research interests include blockchain and cyberspace security.

**曾志康**, 1999 年生. 博士研究生. 主要研究方向为区块链、网络空间安全.



**Chen Qi**, born in 1980. PhD, associate professor. His main research interests include cyberspace security and applied cryptography.

**陈　祺**, 1980 年生. 博士, 副教授. 主要研究方向为网络空间安全、应用密码学.