C2BR-VDS：面向链上链下混合存储的流数据黑盒实时验证方案

林 玮, 孙 奕, 杨佳硕, 李宇杰

中国人民解放军信息工程大学 郑州 中国 450000

摘要 为解决流数据外包存储场景中存在的数据丢失、损坏和恶意用户抵赖等安全问题, 本文借助区块链技术的去中心化和不可篡改性的特点, 结合传统流数据外包存储结构和认证, 提出了一种面向链上链下混合存储的流数据黑盒实时验证方案(C2BR-VDS)。该方案基于陷门哈希函数构造链上链下混合存储可认证数据结构, 链下基于陷门哈希函数自适应地存储完整流数据, 链上轻量的存储认证根节点作为关键验证信息, 验证由去中心化结构的智能合约执行, 提升了流数据外包场景下验证节点的可信度。既实现了流数据即来即验证, 提高流数据实时验证的效率, 又降低了区块链的 Gas 消耗。通过触发智能合约实现第三公平审计方对外包流数据进行完整性验证, 有效监管了云服务器和数据使用者, 防止恶意用户向云服务器对验证结果抵赖, 并且在区块链智能合约上应用 zk-SNARKs 零知识证明算法, 实现流数据链上隐私保护的黑盒验证。分析 C2BR-VDS 的安全性, 并且通过实验对比, 对本方案进行了评估, 将链上验证的复杂度控制到对数时间内。

关键词 链上链下混合存储; 陷门哈希函数; 数据流验证; zk-SNARKs

中图法分类号 TP393 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2026.01.05

C2BR-VDS: A Black Box Real-time Verification Scheme for Streaming Data for on Chain and off Chain Hybrid Storage

LIN Wei, SUN Yi, YANG Jiashuo, LI Yujie

PLA Information Engineering University, Zhengzhou 450000, China

Abstract In order to address the security challenges associated with data loss, corruption and malicious user denial in the context of streaming data outsourcing storage, this paper presents a black box real-time verification scheme for streaming data for on chain and off chain hybrid storage (C2BR-VDS). The scheme makes use of the characteristics of blockchain technology, namely decentralisation and immutability, in conjunction with the conventional structure of streaming data outsourcing storage and the existing authentication mechanisms, with the aim of providing a robust solution for secure data management and verification in this domain. The scheme establishes an on-chain and off-chain hybrid storage system that can be authenticated through the use of the trapdoor hash function. The complete stream data set is stored off-chain in accordance with the trapdoor hash function. Furthermore, the scheme employs an on-chain lightweight storage authentication root node as the key for verifying data. The verification is conducted by a smart contract with a distributive structure, thereby enhancing the credibility of the validator node in the context of streaming data outsourcing. The scheme's functions can be broadly divided into three categories: facilitating the instant verification of the stream data set; improving the efficiency of verifying the data set in real-time; and reducing the gas consumption of the blockchain. Upon triggering the smart contract, the third impartial auditor verifies the integrity of the outsourced streaming data, effectively supervising the cloud server and data users. This prevents malicious users from denying verification results to the cloud server and applies the zk-SNARKs zero-knowledge proof algorithm on the blockchain smart contract to achieve enhanced privacy protection on the streaming data chain through the process of verifying the data in a black box. The security of the C2BR-VDS was evaluated through an analysis of its performance and a comparison with the experimental results. The complexity of the on-chain verification process was controlled to logarithmic time.

Key words hybrid storage on and off chain; trapdoor Hash function; data flow verification; zk-SNARKs

通讯作者: 孙奕, 博士, 教授, Email: 11112072@bjtu.edu.cn。

本课题得到国家密码科学基金(No. 2025NCSF02020), 河南省自然科学基金-面上科学基金项目(No. 242300420297)资助。

收稿日期: 2024-05-10; 修改日期: 2024-07-16; 定稿日期: 2025-11-11

1 引言

近几年随着物联网的快速发展和5G网络的普及, 万物互联, 网络边缘设备急剧增加, 随之产生的数据呈指数级增长。P2P 网络、云计算、传感器网络等大规模复杂网络的快速发展, 也对数据安全提出了新的挑战。我们需要处理的数据不仅是静态的、已知大小的数据, 还有一种是连续的、无限的、快速的、随时间变化的流(如实时监控、在线视频、环境温度监测、股票交易等数据)。有限的用户资源无法存储和处理如此大数量级的流数据, 将流数据外包给第三方存储、管理、分析、计算等应用逐渐进入大众视野。与传统的内部存储基础设施相比, 其拥有更高的扩展性以及更低的管理成本, 吸引了越来越多的用户选择将各类数据存储到云存储服务器上。传统基于云的数据存储与共享方案中, 物联网设备将采集的数据, 流式上传到云服务器中进行存储, 这样既解决了物联网设备数据存储能力有限的问题, 又方便用户共享数据。但同时外包存储也带来了额外的安全和隐私问题。开放的网络环境中, 没有任何节点是绝对可信的, 例如, 云服务器存储的数据一旦从根节点被破坏, 所有的叶子数据信息都无法验证通过的单点故障问题, 以及用户向云服务器抵赖等问题。在上述情况下, 引入一个可信的第三方进行公平审计尤为重要。因此如何提升外包流数据完整性验证的可信度成为一个研究热点。

随着 P2P 网络技术的发展, 2008 年, 中本聪[1]提出了一种去中心化的分布式存储结构——区块链。由于其去中心化和不可篡改的特性, 广泛应用于金融、物联网和文件管理, 被认为是安全数据存储和检索的有前途的解决方案。区块链可以被视作分布式账本, 由网络中不信任节点共同维护。随着智能合约功能的出现与应用, 区块链技术也逐步由简单的交易记录式账本, 转变为一种有效可靠的数据可信存储载体, 无需第三方, 即可以数字方式实现可信交易, 这些交易可追踪且不可逆转。在有效解决服务公平性和存储正确性问题的同时, 由于区块链固有的透明性和开放性, 基于区块链构建数据存储平台仍受到潜在隐私攻击问题的严重阻碍。另外由于数据需要在整个网络中复制, 区块链有限的可扩展性, 开销等问题在设计数据外包存储方案中不可忽视。区块链作为数据存储载体, 其开销成本和数据隐私保护均是需要考虑的问题。

为解决上述问题, 本文提出了一种面向链上链下混合存储的流数据黑盒实时验证(C2BR-VDS)方案。由区块链和云服务器共同维护的外包流数据可认证数据结构, 链下存储完整可认证数据结构, 链上由智能合约公平的执行数据查询结构的验证。完整认证数据结构基于陷门哈希函数, 结合流数据即来即验的数据特点, 设计了链上链下混合存储认证数据结构, 认证规模无需提前固定, 根据数据流规模变化而自适应拓展。将根节点作为验证信息存储在区块链上, 并由智能合约执行验证。本方案旨在提升大数据流的实时验证效率, 降低区块链的成本开销, 有效提升流数据外包场景下验证结果的公正性。另外本方案基于 zk-SNARKs 算法结合智能合约实现黑盒验证, 有效保护链上数据隐私。将链上有效的验证信息进行隐藏, 向区块链节点提供的信息, 均为根节点集合经过计算的隐秘验证证据。验证过程中, 区块链公开节点窥探不到验证的有效根节点的真实信息。黑盒验证防止用户窥探到链上作为验证信息的根节点, 私自验证数据完整性, 并欺骗智能合约, 进而得到错误的验证结果。这种欺骗行为使用户既享受了正确数据带来的价值, 又向云服务器抵赖, 逃避了数据查询使用应支付的成本。因此, 黑盒验证通过隐藏链上有效的验证信息, 防止用户抵赖, 保护各方利益, 进一步提高区块链作为第三审计方的公平性与可信性。

本文的主要贡献如下:

(1) 在大数据流外包存储验证中, 针对规模不可预测、验证实时性高、延迟敏感等问题, 提出了面向链上链下混合存储的实时验证结构。链下基于陷门哈希函数自适应地存储完整流数据, 链上轻量的存储认证根节点, 作为查询数据的验证信息, 随着大数据流的动态增长和变化进行实时构建, 实现了数据流的即来即验证。

(2) 提升了流数据外包场景下验证节点的可信度。基于区块链结构, 构造了链上链下混合存储架构, 链下存储完整的可认证数据结构, 链上智能合约触发执行公正的结果验证, 防止云服务器欺骗和用户抵赖, 保证了数据验证结果的可信性。

(3) 解决了云平台下第三方审计存在的验证数据隐私泄露问题, 基于 zk-SNARKs 算法提出一种链上黑盒认证方法, 面向区块链节点, 隐藏上链的验证信息, 有效维护链上认证数据的安全。使审计方在不泄露任何数据隐私的情况下进行黑盒验证, 实现数据在共享过程中的完整性和正确性, 保护数据各方的利益。

本文其余部分组织如下: 第 2 节讨论流数据外包存储验证的研究背景与现状; 第 3 节介绍构造

C2BR-VDS 方案的预备知识; 第 4 节对方案的系统模型设计, 可认证数据结构和形式化定义进行详细阐述; 第 5 节对方案进行安全性分析; 第 6 节对方案进行了性能分析; 最后, 在第 7 节进行总结。

2 研究现状

流数据往往具有连续性和不可预测性, 传统的可认证数据结构不能直接应用于流数据[2], 需要针对流数据设计专用的可认证数据结构。随着流数据的完整性验证技术不断突破和发展, 目前构造流数据可验证数据结构的方法主要有默克尔哈希树(Merkle Hash Tree, MHT)、变色龙认证树和密码累加器等。Wang 等[3-4]最先将默克尔哈希树引入到云计算中外包数据的完整性验证并设计了第一个同时支持公开审计和全动态数据操作可证明数据持有方案。Papamanthou 等[5]基于密码累加器和默克尔哈希树的思想提出了一种无交互的数据验证结构, 称为广义哈希树(Generalized Hash Tree, GHT)。GHT 基于默克尔哈希树, 支持各种动态操作对数级别的复杂度和累加器无需交互即可完成更新, 来确保数据值和范围查询结果的正确性, 但是, 该方案过于复杂在实际应用中难以设计和实现。Yu[6]结合全同态加密技术和默克尔哈希树结构提出了全同态默克尔哈希树(Fully Homomorphic Encryption-based Merkle Tree, FHMT)结构来实现流数据的完整性验证。FHMT 利用全同态加密的性质几乎将所有的计算任务转移到服务器上, 使得用户的开销很小, 更适用于具有资源受限设备的系统。此外, Xu 等[7]基于 FHMT 结构, 提出了一种隐私保护数据完整性验证模型。但是, GHT 和 FHMT 都是静态的数据结构, 需要预先确定结构的规模, 不使用于流的数据场景。于是, 文献[8]通过改进 FHMT, 提出了一种支持动态流数据的完整性验证结构(Dynamic Fully Homomorphic Encryption-based Merkle Tree, DFHMT), 在初始化时不需要确定 DFHMT 的规模, 在树中添加新数据时可以向上的自适应扩展。但由于全同态加密技术的效率较低且不支持范围查询, 这些基于同态加密的方案目前还无法应用在现实流数据外包存储的场景中。为了找到更适用于流数据的认证数据结构(Authenticated Data Structure, ADS), Schroeder 等[9]提出了可验证数据流(Verifiable Data Streaming, VDS)的概念。他们构造了一种新的数据结构变色龙认证树(Chameleon Authentication Tree, CAT), CAT 本质上是在 MHT 的构建过程中引入陷门哈希。陷门哈希具有抗碰撞性以及一个陷门, 掌握陷门信息的用户无需获取所有的叶子节点即可构建认证树的结构, 来实现对数据流的动态操作和实时验证。Sun 等[10]则在 CAT 的基础上构造了一种基于属性加密的双陷门哈希认证树, 实现了数据的细粒度的授权验证。与 MHT 方案相比, 基于 CAT 构造的完整性验证方案不需要一次性接收全部数据来构造整棵认证树进行验证, 新增叶子节点时, 也无需重新计算先前所有的节点信息, 支持数据实时更新和验证, 其根节点是利用陷门哈希函数生成的, 只有持有陷门的用户才可添加叶子节点, 防止恶意用户新增或修改叶子节点。但是无论是 CAT 还是其改进方案, 都需要在初始化阶段就要确定认证树的深度, 即认证树最多能容纳的数据量上限。文献[11]提出了支持数据动态操作与动态可扩展公共审计方案, 初始化不需要确认认证树深度, 通过结合陷门哈希函数和布隆过滤器签名技术实现自适应扩展认证数据结构, 但其实现的安全性依赖于不可信的第三方。

区块链因其去中心化和不可篡改的特性, 常常被设计成有效解决第三方信任委托的角色。区块链本身可以作为一种数据库被使用, 确保数据不被篡改, 其服务过程由智能合约进行计算, 可以保证审计结果的公开性与公正性。文献[12-13]在以区块链为典型的去中心化结构的存储服务器返回的查询结果证明中, 允许智能合约公平并正确性地执行验证服务。

由于链上存储空间限制, 区块链上运行系统所花费的共识和验证费用, 会随着需求量的增加而增加, 使得在区块链上的数据审计方案因为巨大的链上成本无法持续运行。Xu 等[14]为解决区块链数据库的可验证查询处理问题, 提出了一种新的可验证处理框架 vChain。该框架将存储和查询服务委托给一个功能强大的全节点, 而查询用户仅作为一个轻节点接收结果, 来解决区块链数据库的外包数据完整性验证问题。Wang 等[15]又在 vChain 的基础上进行改进, 提出了一种新的可验证范围查询区块链系统 vChain+。他们设计了一种滑动窗口累加器, 在每个块中构建 ADS, 使数据的查询处理更加高效。Zhang 等[16]基于此架构提出了一种名为 GEM2-tree 的认证数据结构, 将消耗较高的更新操作替换为计算操作, 降低了 ADS 在区块链上的维护成本, 但是不支持数据更复杂的操作。为了扩大区块链存储的规模, 轻量级混合存储区块链的架构已经被现有的行业和研究界广泛应用[17-18]。轻量级混合存储即链上仅存储完整数据中的少量摘要作为元数据, 而原始数据集被外包给链下存储服务器。文献[19]在

轻量级混合存储架构中实现关键字搜索的数据完整性验证。Cui 等[20]提出了一种混合型索引, 既支持动态更新, 又保证了前向的隐私性。Sun 等[21]基于 MHT 的结构进行独立子树设计, 提出了一种区块链上适用于流数据高效认证的数据结构, 文献[19,21]中的数据认证审计计算过程均由用户进行计算, 无法防止恶意用户抵赖。然而以上研究中没有方案可以在公正隐秘的第三方审计中实现对于流数据的完整性验证。

针对流数据的可验证外包数据结构的研究中, 第三方审计节点不可信、用户抵赖等问题, 本文提出了具有针对性的面向链上链下混合存储的流数据黑盒实时验证方案(C2BR-VDS)。基于智能合约实现去中心化的完整性验证, 确保了验证结果的可信性, 同时基于 zk-SNARKs 算法设计了链上具有隐私保护的流数据查询验证, 有效保护了链上验证信息的隐私性, 防止用户抵赖。

3 预备知识

3.1 陷门哈希函数

陷门哈希函数的概念最早来源于 Brassard 等[22]提出的陷门承诺的思想, Krawczyk 和 Rabin[23]首次构造了一种陷门哈希函数, 并基于该陷门哈希函数构造了一种变色龙签名方案。

陷门哈希函数也称为变色龙哈希函数, 是一种特殊的默克尔哈希函数, 具有哈希函数抗碰撞的特性[23]。对于持有陷门的人, 存在一个算法输出一个与 m' 匹配的随机数 r' ,使得 $Th(m,r)=Th(m',r')$, (m,r) 和 (m',r') 被称为是一对碰撞。陷门哈希函数是抗碰撞的, 如果不知道陷门则找到两对不同的 (m,r) 和 (m',r'), 使得 $Th(m,r)=Th(m',r')$, 在计算上是不可行的。

定义 1: 陷门哈希函数, 一个陷门哈希函数由以下多项式时间算法所组成 $(TrapGen,Th,Col)$:

$TrapGen(1^{\lambda})\to(tpk,tsk)$: 初始密钥生成算法, 输入安全参数 1^{λ} 成陷门密钥 tsk 和哈希密钥 tpk。

$Th(tpk,m,r)\to Th(m)$: 陷门哈希值计算算法, 输入陷门哈希函数的公钥, 消息 $m\in\{0,1\}^{in}$, 随机数值 r 通过哈希密钥计算出对应的陷门哈希值。

$Col(tsk,m,r,m')\to r'$: 陷门哈希碰撞计算算法, 输入单向陷门哈希函数的私钥和消息 m, 随机数 r 及需要匹配碰撞的新信息 m', 算法输出的是一个随机数 r', 使得 $Th(tpk,m,r)=Th(tpk,m',r')$。

本文采用的陷门哈希函数来自于文献[24]中所给出的基于离散对数难题构造的陷门哈希函数。

3.2 零知识证明算法——zk-SNARKs

零知识证明这一概念在 1985 年首次被提出[25], 他的定义被广泛使用至今, 零知识证明的 3 个特性为完备性、可靠性和零知识性, 在密码学中可以建立信任和隐私保护, 有效地提高了信息的安全性, 一种零知识证明算法 zk-SNARKs (零知识简洁非交互知识论证, Zero-Knowledge Succinct Non-Interactive Argument of Knowledge)成为目前区块链交易中最受欢迎的技术之一。在区块链交易平台中, 它被用于 Zcash[26]、ZETH[27]等加密货币中, 用于隐藏交易发送方和接收方的地址、交易金额等私人信息。在云服务提供商和数据所有者之间的数据共享中, 零知识证明结合智能合约技术可以实现双方数据之间的数据可用性验证, 在保护数据隐私的前提下确保交易的有效。

zk-SNARKs 在区块链与零知识证明算法结合生成零知识范围证明, 允许区块链网络在不公开信息的情况下验证信息是否存储在区块链上。这有助于在各种用例中达成共识, 有利于保护数据的隐私性。在区块链网络上, 基于 zk-SNARKs 算法将原本公开透明的上链数据信息通过电路生成 *Proof* , 验证者验证这个 *Proof* 程中不泄露任何有效的数据信息, 满足验证过程中对于隐私保护的要求。其原理可以转化为多项式问题 $t(x)h(x)=w(x)v(x)$。

定义 2: zk-SNARKs 算法由以下多项式时间算法所组成 $(Setup,ZKP\_KeyGen,Prover,Verify)$:

$Setup(r,C)$**:** 初始化算法, 选择公共参数, 一个素数 r, 两个 r 阶的循环群 G_1 和 G_2 分别通过生成元 $\mathcal{P}_1$、$\mathcal{P}_2$ 和一个配对 e: $G_1\times G_2\to G_T$ (其中 G_T 也是 r 阶循环群)。

$ZKP\_KeyGen\left(1^{\lambda},C\right)\to\left(EK_C,VK_C\right)$**:** 密钥生成算法, 密钥生成器通过输入电路 C 计算输出证明密钥 EK 和验证密钥 VK。

$Prover\left(EK,\vec{x},\vec{a}\right)\to(\pi)$**:** 证明生成算法, 证明方输入证明密钥 EK, 信息向量 $\vec{x}\in F_r^n$, $\vec{a}\in F_r^h$。经过 EK 将多项式转化为矩阵, 输出证据 π。

$Verify\left(VK,\vec{x},\pi\right)\to(0/1)$: 查询结果验证算法, 通过验证密钥 VK, 查询结果 $\vec{x}\in F_r^n$ 和证据 π, 将矩阵还原成多项式, 等式成立, 即通过验证, 输出 1, 否则输出 0, 验证失败, 判定查询结果错误。

4 解决方案

4.1 系统模型设计

C2BR-VDS 方案基于链上链下流混合存储架构设计流数据黑盒实时验证的系统模型, 如图 1 所示, 由以下 4 个部分组成。

数据拥有者: 流数据的所有者, 作为数据源不断产生流数据, 并将完整数据存储在云服务器中, 将验证信息通过零知识证明的电路存储在区块链上。

云服务器: 存储并维护流数据完整的可认证数据结构, 管理数据拥有着存储的数据文件, 响应用户的数据查询请求。

区块链: 是执行公平审计的第三方, 存储动态根节点, 其验证由去中心化的智能合约完成, 保证验证结果的公平性。

用户: 是流数据的合法访问者, 向云服务器发起数据访问请求, 云服务器经过身份验证之后, 进行响应, 然后返回查询结果及相关的完整性验证的证据, 用户通过查询结果和证据在本地计算出验证根节点, 并发送给区块链进行第三方数据完整性公平审计。

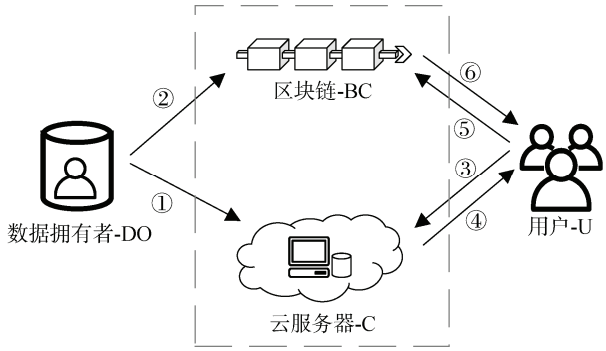

图 1 基于区块链的外包数据可验证数据结构方案模型

Figure 1 A verifiable data structure scheme model for outsourced data based on blockchain

C2BR-VDS 方案的流数据存储、查询和验证中主要分为 6 个步骤:

a: 数据存储阶段: ①数据拥有者为数据 m 在本地构造相应的陷门哈希树, 将完整的数据及其认证路径上传至云服务器; ②同时将产生的各层固定的子树根值上传至区块链。

b: 数据查询阶段: ③数据访问者获得数据访问权限之后, 向云服务器发起查询请求; ④云服务器通过范围查询返回查询结果 R 和相应的证据认证路径 $auth$ 。

c: 数据认证阶段: ⑤数据访问者在本地根据云服务器返回的信息计算出根值, 并发送给区块链; ⑥由区块链的智能合约执行审计, 并将审计结果 *True* / *False* 返回给数据访问者。

4.2 可认证数据结构

C2BR-VDS 方案基于陷门哈希函数, 构建了链上链下混合存储结构, 链上存储验证信息, 链下云服务器中存储完整的可认证数据结构。本方案的可认证数据结构不需要初始化认证树的深度, 可以根据数据流规模变化而自适应拓展, 在流数据外包服务中实现对大数据流的边读边写边验证, 能够更好地解决流数据规模不可预测、验证实时性高、延迟敏感等问题, 适用于云平台下大数据流场景。本方案底层的叶子节点在插入更新时, 认证树中总是存在唯一的一个陷门哈希节点与之对应。其中叶子节点都是以所处的最小子树的根节点作为验证信息, 记录在区块链上, 同时产生更短认证路径, 也避免了单点故障的问题。

链下完整可认证数据结构自适应拓展过程中, 每次当前的结构无法满足数据规模时, 都会新生成一个根节点, 作为验证信息, 存储在区块链上。原来的结构变为新生成根节点的左子树, 深度增加 1, 可以支持的数据量则扩展为原来的两倍。图 2 是 3 种不同规模下的认证树的结构, 不同规模下的认证树结构具有一致性, 规模较小的认证树为较大认证树的左子树。由于每次认证树进行拓展后, 支持的数据量都变为原来的两倍因此使得后续阶段扩展操作不会特别频繁。

如图 3 所示, 在插入每 2^i 个数据块之后, 哈希树结构会形成满二叉树结构, 为了继续插入新的数据块, 自适应陷门哈希树都会向上自适应扩展一层, 原来的结构变为新生成根节点的子树, 用作其右子树插入数据块 m 的验证根节点, 每产生一个新的子树根节点 $Root_i(0<i\leqslant n)$, 根节点集合重新通过零知识证明算法作为验证证据上传到区块链。

链上存储验证信息, 基于 zk-SNARKS 算法隐藏链上有效数据, 实现了链上智能合约的黑盒验证, 完整链上认证信息存储验证过程如图 4 所示, 链上验证信息为每层子树根节点 $\{C\}\rightarrow Root$, 数据拥有者用户 ID 和根节点上传时间戳经过电路设计生成验证证据 $Proof$ 上传给区块链。用户查询验证时, 在本地将查询的数据计算出该数据对应的验证根节点, 经过同样的电路运算, 交付给智能合约进行验证。此过程中, 用户不知道本地计算出的验证信息是否有效, 有效预防用户向云服务器抵赖。

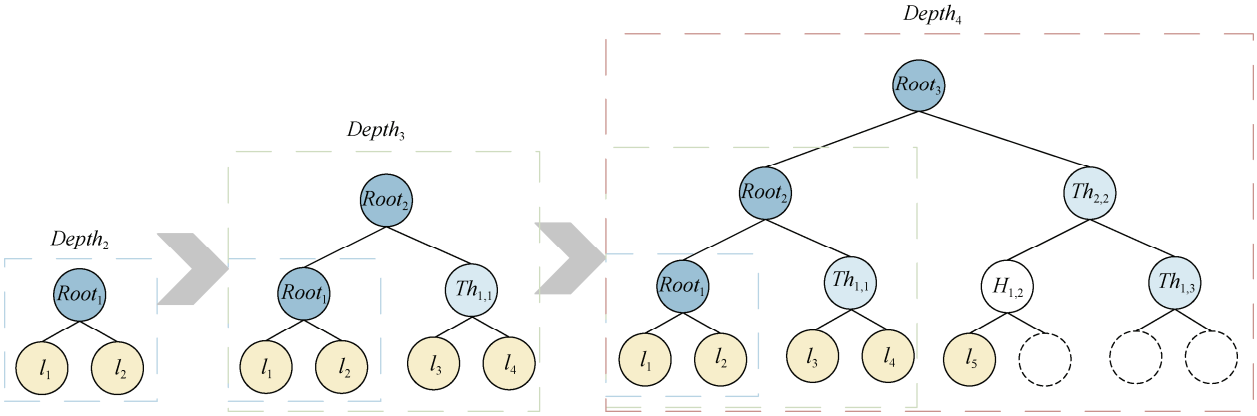

图 2　不同规模下自适应陷门哈希树结构

Figure 2　Adaptive trapdoor hash tree structure at different scales

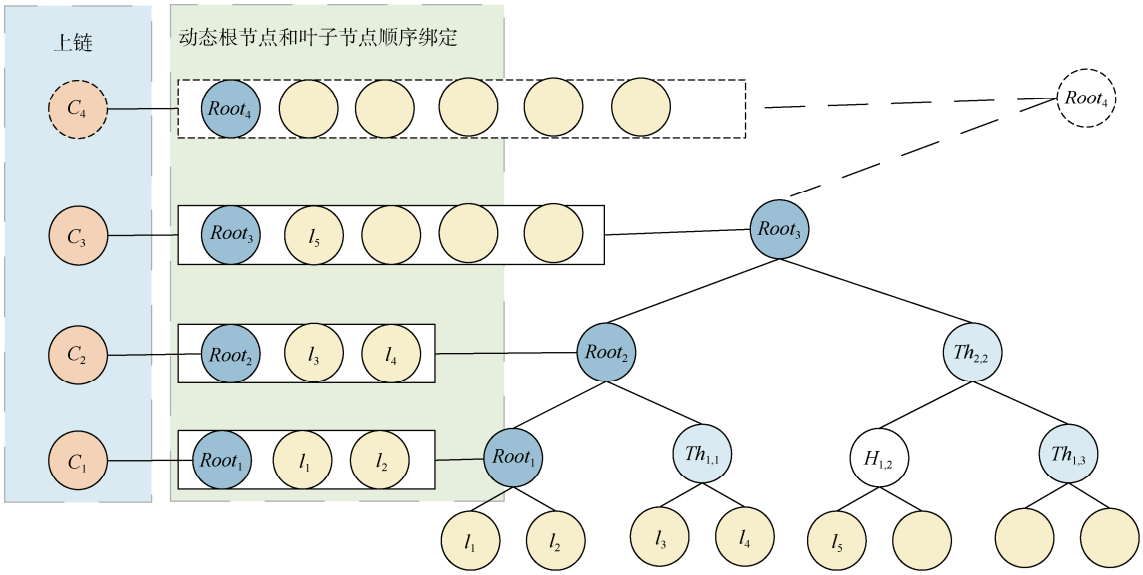

图 3　流数据陷门哈希树认证数据结构

Figure 3　Stream data trapdoor hash tree authentication data structure

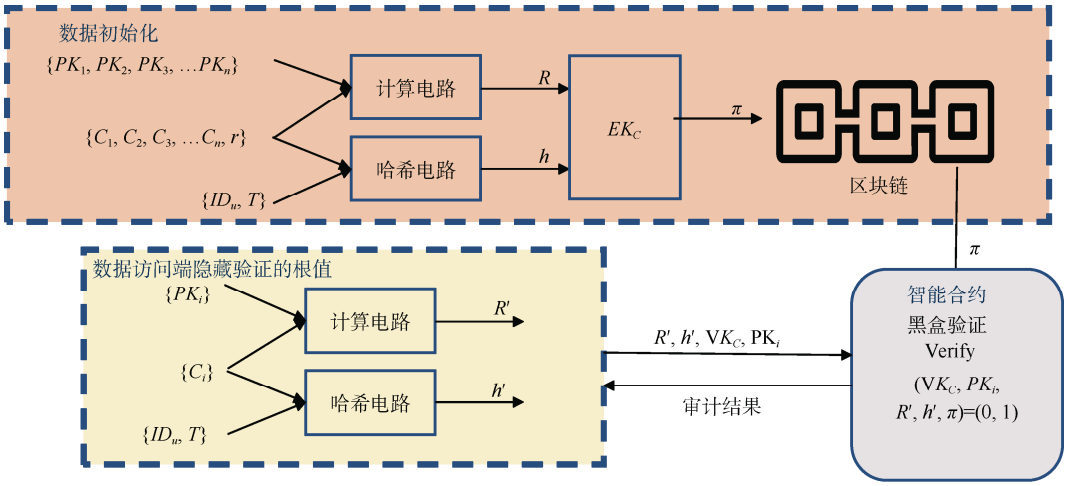

图 4　完整链上认证信息存储验证过程示意图

Figure 4　Schematic diagram of the complete on chain authentication information storage and verification process

设计 C2BR-VDS 方案的可认证数据结构具有实时性和黑盒性, 具体定义如下:

实时性: 基于自适应陷门哈希树构建了链上链下混合存储结构, 根据数据流规模变化可以实现自适应拓展, 在流数据外包服务中实现对大数据流的边读边写边验证, 能够更好地应用于云平台下大数据流场景。

黑盒性: 基于 zk-SNARKs 算法隐藏上链的验证信息, 有效维护链上认证数据的安全。使审计方在不泄露任何数据隐私的情况下进行黑盒验证, 实现数据在共享过程中的一致性和可用性, 保护数据各方的利益。

$$B-BRVSD=(Source_{Append};\ Server_{Append};\ BC_{Append};\ GenProof_{Server};\ GenProof_{User};\ Verify_{Contact})$$

4.3 链上链下混合存储架构的流数据黑盒实时验证方案的形式化表述

定义 3: 运行于链上链下混合存储架构的流数据黑盒实时验证方案由以下多项式时间算法的元组组成:

4.3.1 数据存储阶段

数据源数据添加算法

$$Source_{Append}(m,st)\rightarrow(auth_{send},st',c)$$

该算法由数据源, 即数据拥有者运行, 将每一条流数据 l 加密得到 l', 同该数据生成的时间戳 T 和数据拥有者 ID 生成数据块 $m=(l',T,ID_u)$, 并计算出对应的认证路径信息并更新状态向量 st, 后将加密数据块 m', 认证路径信息 $auth_{send}$, 插入位置信息 c, 数据生成时间 T 和数据拥有者 ID 上传给云存储服务器。

数据源在获取到数据后生成认证树的结构更新信息, 首先根据当前认证树的规模值(所能容纳的数据量由参数 $capacity$ 表示)和认证树已添加的叶子节点的数目 c 据认证树是否扩展可分为以下情况: 若当前树的容量已满 $(c=capacity)$, 则对认证树进行扩展, 深度值增加 $d\leftarrow d+1$, 认证树的容量变为原来的两倍 $capacity=capacity*2$, 产生新的根节点 $newroot\leftarrow R_d$, 原根节点为新根节点的左孩子 $R_{d.leftchild}\leftarrow R_{(d-1)}$, 将新生的根节点添加到 $Root$ 中, 随机选择 $x_{(d-1,1)},r_{(d-1,1)}$, 算新生成根节点的右孩子的陷门哈希值 $v_{(d-1,1)}\leftarrow Th\left(x_{(d-1,1)},r_{(d-1,1)}\right)$, 可以得到新生成根节点的哈希值 $R_d=H(R_{d-1}\|v_{(d-1,1)})$, 并改变 st 的状态: $st'\leftarrow st$, 计算出来的陷门哈希值添加到认证路径 $auth$ 中。接下来操作与树的容量未满时相同 $(c<capacity)$, 从 st 找最底层的傀儡节点(没有孩子的陷门哈希节点), 以该节点为根节点产生子树, 插入数据后自下而上更新沿叶子到对应陷门哈希节点的值, 将计算出来的陷门哈希值添加到数据的认证路径 $auth$ 具体过程参照算法 1:

算法 1: $Source_{Append}(m,st)$

//输入: 数据块 m' **和状态向量** st

//输出: 认证路径信息 $auth_{send}$ **, 插入位置信息** c

```
for h = 0 to st.dummySet.length do{
    min = MAX
    if (st.dummySet.get(h) < min)
        min = h
}
if (c = capacity)
    if (this.capacity == 0){
        this.capacity = 1;
    } else {
        this.capacity* = 2;
        Root.add = new root;
        Root.leftChild = this.root;
        this.root = Root;
    }
                Depth++;
}
v_{i,j} ← st.dummySet.get(min)
st.dummySet.delete(min)
for j = min−1 to 1 do{
    x_{2*i+1,j} = {0,1}^{2len}
    r_{2*i+1,j} = {0,1}^{k}
    v_{2*i+1,j} = Th(x_{d−1,1}, r_{d−1,1})
    st.dummySet.add(x_{2*i+1,j}, r_{2*i+1,j})
    auth.add(v_{2*i+1,j})
    i = 2*i
}
for h = 1 to min do{
        v_{c/2^h,h} = H(lefthash || righthash)
    else
        r'_{i,j} = col(tsk, x_{i,j}, r_{i,j}, v_{i,j})
auth.add(r'_{i,j})
```

$c = c + 1$
return auth

云服务器数据添加算法

$$Server_{Append}\left(auth, c\right) \rightarrow SNode_{new}$$

云服务器在收到认证路径信息 $auth_{send}$，数据信息以及数据的插入位置信息后，进行认证树 *SNode* 更新，找到对应的插入位置后自下而上更新对应的路径节点，根据当前数据的插入位置信息，即可以通过二叉查找即可实现完成数据的插入，并沿途计算缺少的陷门节点，具体过程参照算法 2。

算法 2: $Server_{Append}\left(auth, c\right)$

//输入：认证路径信息 $auth_{send}$**，插入位置信息** c
//输出：完整认证树 *SNode*

```
SNode = root
//找到最近插入节点
for h = D − 2 to 0
 if c/2^h is even
   SNode = SNode.leftchild
 else
        SNode = SNode.rightchild
//更新认证树节点
for h = 1 to log c
                SNode = SNode.parent;
              if (SNode is a HashNode){
       SNode.updateNodehash;
         SNode.brother.Hash = auth.get(h).hash;
     }
     else{
          SNode.r = auth.get(h).r;
        SNode.updateNode_x();
          SNode.brother.Hash
                              = auth.get(h).hash;
}
```

区块链数据添加算法

$$BC\_Server_{Append}\left(EK_C, Root\right)$$

智能合约发布阶段参照**定义 2,** 在计算任务中输入安全参数 λ 和电路 C，计算出密钥对 $\left(EK_C, VK_C\right)$，EK_C 用来生成零知识证明，VK_C 被用来验证零知识证明。

数据拥有者向区块链和云服务器发送数据初始化请求，指定创建默认零值 *Zero*，其中 *Zero* 用于计算存储数据为空时各层子树的根哈希值，区块链收到请求后初始化数据认证存储，首先在状态树中创建一个智能合约账户，首先在数据为空时，通过数据拥有者本地零知识证明计算出初始的认证证据，并将相关操作的合约代码存储在 *Code* 字段中，后续在公平审计阶段智能合约读取此字段信息执行公平公开审计操作。

区块链通过触发智能合约，将生成的子树根节点的集合作为认证信息通过共识，存储在区块链中，具体流程如下:

当数据所有者的流数据满足条件时，将流数据自适应陷门哈希树的每层子树根集合作为私有信息 *Root*，将自己的数字签名和本地时间附加到私有数据上，作为辅助信息 δ，提交给 zk-SNARKs，生成零知识证明:

$$Root \leftarrow \left\langle root_1,\ root_2, \ldots,\ root_n \right\rangle$$
$$\delta \leftarrow \left(Root, T, ID_u\right)$$

随机数 r,计算 r 和辅助信息 δ 的哈希值:

$$H\left(\delta, r\right)$$

数据拥有者通过私钥生成认证电子签名 Sign:

$$\sigma_a \leftarrow Authsign\left(PK_p, H\left(\delta\ r\right)\right)$$

数据所有者构建电路 C: $F^n \times F^h \rightarrow F^l$ 电路输入公共参数 $\left\langle PK_1, PK_2, \ldots, PK_n \right\rangle$ 和私有数据 *Root*，输出结果 R 和哈希值 h 验证数据的真实性和可用性。

$$\left(\left\langle root_1,\ root_2, \ldots,\ root_n, r\right\rangle\right)$$
$$\rightarrow \left(\left\langle Root, r\right\rangle\right) \rightarrow \left(R, h\right)$$

由零知识证明生成密钥 EK_C，数据所有者的辅助信息 δ，由电路 C 生成的输出结果 R 和哈希值 h，和数字签名一起生成零知识证明的认证信息 π。作为认证信息存储在区块链上。

$$BC_{Append}\left(EK_C, \delta, \sigma_a\right) \rightarrow \pi$$

4.3.2 数据查询阶段

数据质询算法

$$GenProof_{Server}\left(SNode, i\right) \rightarrow \left(m', auth'\right)$$

由云服务器在收到数据访问者质询信息 i 后，返回位置 i 的数据信息 m'，同时返回对应的认证信息 $auth'$，数据生成时间 T 和数据拥有者 *ID*。具体过程参照算法 3。

算法 3: $GenProof_{Server}\left(SNode, i\right)$

//输入：认证树 *SNode*，**质询信息** i

//输出：位置 i 的数据信息 m'，对应的认证信息 $auth'$

```
for i ≠ 0 do {
    if ⌊i/2⌋ is even then
        SNode = SNode.leftchild
        auth.add(SNode.brother.THash)
    else{
        SNode = SNode.rightchild
        auth.add(SNode.random, brother.hash)
        break;
    }
    i = i/2
return auth
```

4.3.3 数据认证阶段

数据完整性认证生成算法

$$GenProof_{User}(auth') \rightarrow (R', h')$$

当数据访问者收到云服务器的查询结果和认证的路径信息时, 在本地进行计算, 通过路径信息还原该数据所属的子树根 $Root_j$, 和对应数据的验证证据消息, 并发送给区块链, 触发智能合约进行验证。具体过程参照算法 4。

算法 4: $GenProof_{User}(auth')$

//输入：认证信息 $auth'$

//输出：对应的验证证据消息 R', h'

```
for h = 1 to log c do
{
  if c/2^h is odd
      x = x || auth.get(h).THash
      r = auth.get(h−1).random
      x = TH(x, r)
  if c/2^h is even
      x = auth.get(h).Hash || x
      x = H(x)
      }
(R', h') = {x, m.T, m.ID_u}
```

数据验证算法

$$Verify_{Contact}(VK_C, \pi, R', h', \sigma_a) \rightarrow (true / flase)$$

该算法由区块链智能合约执行, 查询算法返回的验证证据 δ', 验证第 i 个位置数据信息的完整性及序列信息的完整性。具体过程参照算法 5。

算法 5: $Verify_{Contact}(VK_C, \pi, R', h', \sigma_a)$

//输入：验证证据消息

//输出：验证结果 0/1

```
π' = Verify(VK_C, π, R', h', σ_a)
if π == π'
    return 1;
return 0;
```

智能合约首先使用数据所有者的公钥对其签名进行验证, 然后使用 zk-SNARKs 的验证密钥对零知识证明进行验证。验证通过后, 智能合约会自动比较数据所有者的零知识证明 π、计算结果 R、哈希值 h、云服务组织的零知识证明 π'、计算结果 R'、哈希值 h'。如果验证结果正确, 输出 1; 否则输出 0。

5 安全性分析

5.1 数据结构安全性分析

本节分析了 C2BR-VDS 方案中认证数据结构的安全性。首先认证数据结构的安全性可以规约成构建自适应陷门哈希树的安全性和区块链智能合约的数据存储安全性, 假设一个计算能力足够高效的对手 A, 从数据拥有者的角度出发, 希望云服务器和区块链可以安全维护数据, 另外从云服务器角度出发, 需要预防恶意用户为逃避支付数据存储使用过程中产生的维护费用, 将正确数据的验证结果伪造为错误, 既享受了正确数据带来的效益又逃避了数据使用费用的支付, 并且对云服务器自身信誉度也有所影响。

对于数据的安全存储问题, 数据拥有者产生 $q(\lambda)$(多项式级)个流式数据, 由于敌手 A 是足够高效的, 插入流式数据的数量是多项式有界的。将数据按照顺序插入陷门哈希认证树中。整个陷门哈希认证树的深度 $D = ploy(\lambda)$ 是多项式级的。对于数据拥有者来说, 整个陷门哈希认证树的结构状态向量 st 是作为私钥保密的, A 不能获得。A 要取得胜利, 就要输出一个元组 $(i^*, m^*, auth^*)$, 并且用将该元组发送给智能合约, 进行验证 $Verify(i^*, m^*, auth^*, pk)$。下面分两种情况 $1 \leqslant i^* \leqslant q$ 和 $i^* \geqslant q$ 对该方案的数据存储安全性进行讨论。

第一种情况, $1 \leqslant i^* \leqslant q$ 果敌手取得胜利, 即破坏陷门哈希认证树的结构保持的性质, 在这种情况下总能找到一对普通哈希函数的碰撞, 或者一对陷

门哈希函数的碰撞, 这与哈希函数抗碰撞的假设矛盾。第二种情况, $i^* \geqslant q$ 果敌手获胜, 即破坏陷门哈希认证树的单向性。可以证明敌手 A 要么能够构造陷门碰撞, 要么能够在陷门哈希函数中找到一对碰撞。这与假设矛盾。

对于恶意用户问题, 对手模拟恶意用户向云服务器发起质询 i 得到云服务器返回的一个结果元组 $\left(i^*, m^*, auth^*\right)$, 数据拥有者存储数据 i 的位置数据为 m, 如果 $i = i^*$, $m = m^*$ 且 $auth = auth^*$, $Verify\left(i^*, m^*, auth^*, pk\right) \to false$, 则对手取得胜利。由于该方案中用户其通过 $\left(i^*, m^*, auth^*\right)$ 计算出根值 $root^*$ 发送给区块链的智能合约, 由智能合约进行最终计算与判定, 由于云服务器返回的质询信息都是正确的, 对手通过正确计算得到的根值结果 $root^* \in Root$, 对手只有得到 $root^* \in Root$ 的结果, 才可以确定数据的正确性与完整性, 但是由于区块链上通过 zk_SNARKs 算法将自适应的陷门哈希函数的各层子树根进行了维护, 区块链上各级子树的根值对用户是保密的, 如果对手需要得到 $Verify\left(i^*, m^*, auth^*, pk\right) \to false$ 的结果, 需要伪造根值 $root'$ 且 $root* \neq root'$, 其伪造的根值无法通过认证, $Verify\left(i^*, m^*, auth^*, pk\right) \to false \to root' \notin Root$, 这与对手确定数据正确性相悖, 该方案具备验证结果的公平性。

5.2 C2BR-VDS 方案安全性分析

本节通过对 C2BR-VDS 方案的实时性、黑盒性、可追溯性和抗抵赖性等进行安全性分析, 证明本文方案的安全性。

实时性: 本文中外包流数据基于陷门哈希函数设计了完整存储的数据结构, 初始构建无需指定深度, 根据到达数据量的多少自适应扩展存储规模, 实现了流数据的即来即验证, 提高了实际应用中的外包流数据的验证效率。

黑盒性: 由于区块链链上所有节点公开透明, 链上验证过程中无法保证验证数据的隐私性, 本文方案中基于 zk-SNARKs 算法的安全性, 隐藏上链的验证信息, 使审计方在不泄露任何数据隐私的情况下进行黑盒验证, 有效维护链上黑盒认证阶段的数据隐私安全。

第三方审计公平性: 现有的外包流数据存储方案, 由于缺少可信的第三方审计, 存在数据被篡改的安全隐患。本文方案基于去中心化的区块链作为可信的第三方审计, 规约于智能合约的安全性, 由于区块链不可篡改的特性, 智能合约一经触发验证, 审计不受任何单一的私有节点控制, 因此区块链作为流数据外包场景中的第三方审计, 确保验证过程和结果不被篡改, 确保验证结果的公平性。

可追溯性: 由于该方案基于链上链下混合存储架构, 数据拥有者上传数据时, 向区块链发送验证信息, 用户查询数据向区块链发送待验证的根节点数据的行为, 在链上均会被记录, 因此数据完整性的验证过程是可追溯的, 有效保护共享中的数据安全。

抗抵赖性: 传统的流数据外包存储方案中, 查询结果的验证由用户方负责, 如果存在抵赖的恶意用户, 正确地查询数据, 恶意用户将正确的验证结果输出错误, 既享受了正确数据带来的效益, 又避免了正确数据查询的支付成本。本方案中将验证的结果输出交给区块链智能合约执行, 具有去中心化和不可篡改性, 并且基于智能合约和 zk-SNARKs 算法安全性, 用户不可以窥探验证的私有数据, 进而无法私自验证数据完整性, 并欺骗智能合约, 得到错误的验证结果。这种欺骗行为使用户既享受了正确数据带来的价值, 又向云服务器抵赖, 逃避了数据查询使用应支付的成本。本方案有效防止恶意用户对云服务器进行抵赖, 保证本方案的抗抵赖性。

6 性能分析

6.1 功能点比较

本方案有效提升了大数据流的实时验证效率, 降低了区块链的成本开销, 增强了流数据外包场景下验证结果的公正性。在保护验证数据隐私的前提下, 通过链上黑盒验证实现了流数据共享过程的有效性和一致性。基于区块链的不变性和可追溯性, 实现第三方公正审计中流数据的完整性验证, 基于 zk-SNARKs 算法实现链上黑盒认证, 有效预防了用户抵赖问题。通过对现有数据共享方案的比较分析, 本文分别从第三方审计、实时更新性、数据的可验证性, 以及验证数据隐私、抗抵赖和数据可追溯性 6 个方面与其他方案进行了对比, 如表 1 所示。

表 1 中 Papamanthou 等[5]提出的 GHT 结构, 仅能支持动态数据的查询验证, 且难以实现。Yu 等[6]提出的 FHMT 结构实现流数据的完整性验证, 具备可验证性。Sun 等[11]提出的支持数据动态操作与动态可扩展公共审计方案, 满足流数据实时更新的特性, 并通过布隆过滤器保证验证数据的隐私性。文献[12, 16, 19]方案中引入了区块链结构作为可信的第三方审计, 但由于区块链的公开透明性, 外包流数据验

表 1　方案功能特点比较

Table 1　Comparison of scheme functional characteristics

	第三方审计	实时更新性	可验证性	验证数据隐私	抗抵赖	可追溯性
Papamanthou 等[5]	×	×	×	×	×	×
Yu[6]	×	×	√	×	×	×
Sun 等[11]	×	√	√	√	×	×
Wang 等[12]	√	×	√	×	×	√
Zhang 等[16]	√	×	√	×	×	√
Zhang 等[19]	√	√	-	×	×	×
此方案	√	√	√	√	√	√

证中的数据隐私得不到保护, 我们的方案解决了这个问题并且实现了抗用户抵赖等功能。

6.2　性能分析

6.2.1　认证数据结构云存储开销分析

基于认证数据结构, 首先忽略文献[19]和我们方案中区块链存储访问的Gas花费, 对以下4种方案进行各种操作计算复杂度进行了比较, 如表2所示, 我们的方案在初始阶段具有最小的初始化开销, 同时插入, 查询和验证过程的计算复杂度都由原来的 $O(\log_2 n)$ 降低到了 $O(\log_2 i)$ $1 \le i \le n$, 就是说随着处理数据量的增大, 我们的方案所消耗的时间相对于其他方案增率更小。

表 2　方案不同操作计算复杂度比较

Table 2　Comparison of computational complexity for different schemes

	预处理	插入	查询	验证
Wang et al[3]	$O(2n-1)$	$O(\log_2 n)$	$O(\log_2 n)$	$O(\log_2 n)$
Schroder et al [9]	$\boldsymbol{O(\log_2 n)}$	$\boldsymbol{O(\log_2 n)}$	$O(\log_2 n)$	$O(\log_2 n)$
Zhang et al[19]	$O(1)$	$O(\log_2 i)$	$O(\log_2 n)$	$O(\log_2 i)$
此方案	$O(1)$	$O(\log_2 i)$	$O(\log_2 i)$	$O(\log_2 i)$

理论上对数据插入时认证树结构更新效率和数据访问时的验证效率进行详细推导与分析:

在初始阶段 Schroder 认证树[9]的深度值取为常量 D, 本方案基于陷门哈希函数的认证树的深度为 $d=0$, 后续自适应地随数据插入动态变化。在认证树进行实时插入更新时, 树中 1/2 的节点插入更新时只需向上一个陷门哈希节点的碰撞值, 1/4 节点需要向上计算两个节点, 以此类推可以得到插入数据时的更新路径, 在基于陷门哈希函数的认证树中, 所有叶子插入数据时, 路径平均更新长度为:

$$
\begin{aligned}
&Append\_len \\
&= 1\times\frac{1}{2}+2\times\frac{1}{4}+3\times\frac{1}{8}+\ldots+(d-1)\times\frac{1}{2^{d-1}}+d\times\frac{1}{2^d} \\
&= 2-\frac{1}{2^{d-1}}
\end{aligned}
$$

陷门哈希函数具有自适应扩展的特点, 在任一时刻认证树的数据量总是与当前深度相匹配, 所以无论树的规模多大, 数据插入时的平均更新长度稳定在 2 左右。图 5 为本方案中基于陷门哈希函数认证树(C2BR-VDS)结构和基于默克尔哈希认证树(MHT)结构平均插入更新路径长度随深度变化的对比图。

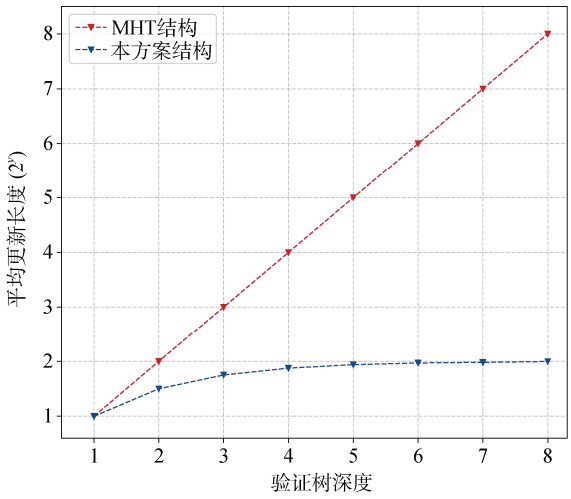

图 5　平均插入更新路径长度随深度变化对比图

Figure 5　Comparison chart of average insertion update path length with depth variation

6.2.2　区块链认证 Gas 开销分析

该方案通过将区块链作为第三方公平审计, 有效增强了数据云存储的安全性, 与传统的外包数据

库中主要关注查询效率的认证数据结构方案处理不同, 这里的一个关键挑战是将链上链下混合存储认证数据结构设计为具有更新效率, 智能合约需要有效的维护流数据外包存储中的计算成本和更新成本, 以太坊是一个开创性的智能合约平台, 它采用了一种独特的 Gas 模型来衡量智能合约的执行成本, 不同的操作产生了巨大的成本差异。 表 3 显示了以太坊中智能合约中常见操作的 Gas 成本。

表 3　区块链各种操作的 Gas 消耗

Table 3　Gas consumption of various blockchain operations

操作	Gas 消耗	注解
$G_{txcreate}$	32000	创建智能合约交易操作的 Gas 消耗
G_{store}	20000	存储操作的 Gas 消耗
$G_{selfdesttruct}$	5000	删除操作的 Gas 消耗
G_{load}	200	加载一个变量到内存操作的 Gas 消耗
$G_{txdatanonzero}$	16	为交易中每个非零字节的数据或代码操作的 Gas 消耗
G_{memory}	3	访问一次内存操作的 Gas 消耗
G_{tx}	$30+6x$	x 条信息 *hash* 操作的 Gas 消耗

在面向链上链下混合存储的流数据黑盒实时验证方案中, 基于零知识证明算法在智能合约中构建了流数据黑盒认证结构, 将基于陷门哈希函数的认证树的子树根值集合 $Root$ 、时间戳 T 和数据拥有者 ID 作为认证信息, 在数据本地, 基于 zk-SANARKs 算法生成认证证据 π , 通过智能合约上传到区块链。

因流数据是仅追加外包存储类型, zk-SNARKs 算法在零知识证明归类中代表基于二次算术程序的模型实现非交互, 设链上认证信息的个数为 C , 证明者的主要计算开销为 $O(C\log_2 C)$, 每 2^i 个流数据对应的认证结构会产生 1 个认证信息, 重新生成认证信息, 上传至区块链, 即 $C=\log_2 i\ \ (0\le i\le n)$, 这一操作过程中链上的 Gas 消耗为:

$$G_{Proofcreate}=32000+20000\times C,\ (0\le i\le n)$$

链上认证信息存储的开销:

$$O(Proof)=O(C\log_2 C)+O(C)$$

参照 Zcash 构造所使用的 Pinocchio 协议[28], 验证者的操作中群幂运算开销为 $O(|x|+|y|)$, 另外还有 11 次配对运算, 在我们方案的验证过程中, 验证方为智能合约, 其开销转换为 Gas 成本简单表示为:

$$G_{Verify}=32000+16\times O(|x|+|y|)+3\times 11$$

随着流数据无上限的线性追加, 链上存储的认证信息数量逐渐稳定, 即 $i\to+\infty$, 对应认证信息的生成速率为 $C'=(\log_2 i)'=0$, 链上验证的复杂性可以在对数时间内完成, 显著降低 Gas 消耗。

6.2.3　实验对比分析

本节通过实验进行开销等对比分析, 评估了本方案在实际使用中的效果, 由于本方案的可认证数据结构, 即陷门哈希函数, 本质上是一种特殊的默克尔哈希函数。因此本节也实现了基于默克尔哈希认证树(MHT)的外包数据查询验证方案, 通过对比两种算法在数据更新、查询和验证中的开销和在区块链上的维护成本, 最终验证了本方案的高效性。

本文通过 VMware 虚拟机与 Ubuntu 18.04.1 系统, 搭建 fabric 区块链模拟面向链上链下混合存储的实验环境。模拟流式数据即来即验证, 通过软件模拟了数据拥有者(物联网设备) 采集流式数据的过程, 总共模拟了 2^{18} 个数据块, 每个数据块的大小为 1 MB。

图 6 描述了将不同大小的数据集合, 流式追加写入链上链下混合存储结构的总 Gas 消耗。随着物联网设备数据采集数量的增加, 本方案与 MHT 方案的总 Gas 消耗都在增加, 但是随着数据集合的不断增大, 本方案的总 Gas 消耗远远低于 MHT, 实验结

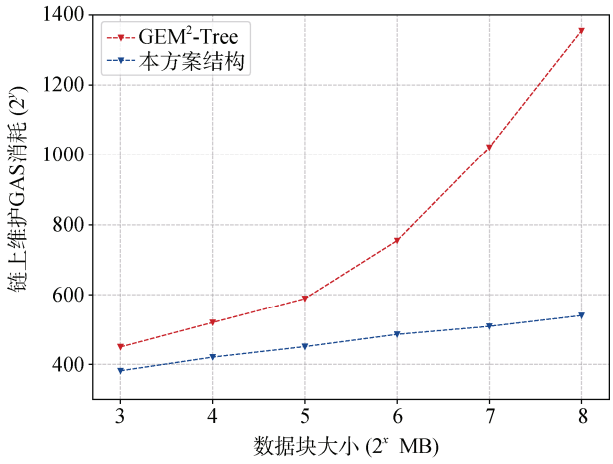

图 6　Gas 消耗随数据库大小变化对比图

Figure 6　Comparison of Gas consumption with database size

果表明在区块链上本方案具有更低的 Gas 消耗, 链上验证的复杂度可以在对数时间内完成。

图 7 描述了将不同大小的数据集合, 外包给链上链下混合存储结构之后, 对数据进行云服务器查询与链上验证花费的平均时间, 随着数据库大小的指数级的增加, MHT 方案的平均时间消耗都在增加, 但是随着数据集合的不断增长, 本方案查询验证的时间维持在 500 ms 左右, 随着数据库的增长, 本方案的时间花费远远低于 MHT 方案, 证明本方案中流式数据查询验证的效率显著提高。

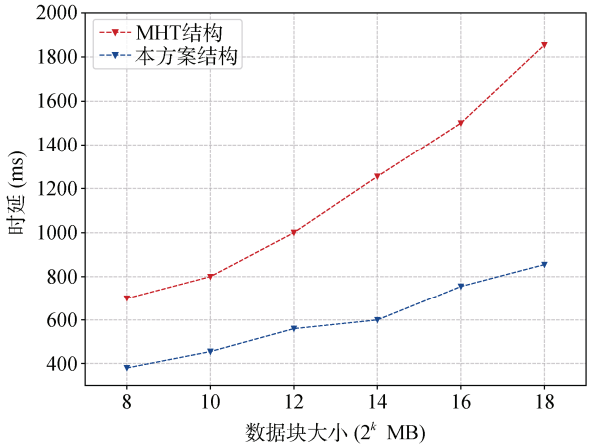

图 7 时间消耗随数据库大小变化对比图

Figure 7 Comparison of time consumption with database size

7 结束语

本文提出面向链上链下混合存储的流数据黑盒实时验证方案, 在流数据外包存储的应用场景下引入去中心化的区块链结构作为公正的第三方审计, 基于陷门哈希函数构建链上链下混合存储可认证数据结构, 随着大数据流的动态增长和变化进行实时构建, 不需要在初始构建时指定深度, 实现了数据流的即来即验证。验证由去中心化结构的智能合约执行, 提升了流数据外包场景下验证节点的可信度。同时在数据完整性认证过程中引入零知识证明算法, 将链上认证的复杂度控制在对数时间内, 隐匿相关认证信息, 有效预防了恶意用户的抵赖, 提高了链上链下混合存储结构的安全性, 并且基于链上链下混合存储结构中数据查询验证的效率显著提高。但是目前 C2BR-VDS 方案仅针对只进行追加的流数据类型, 针对于链上链下混合存储结构适用于更多数据类型的存储查询与验证, 还需要进一步研究。

参考文献

[1] Anthony Dewayne Hunt. Bitcoin: A Peer-to-Peer Electronic Cash System[Z]. https://api.semanticscholar.org/CorpusID:236214795. 2008.

[2] Sun Y, Yang F, Chen X Y, et al. Research progress of veri able technologies for outsourcing services[J]. *Scientia Sinica* (*Informationis*), 2024(3): 514-565.
(孙奕, 杨帆, 陈性元, 等. 面向外包服务的可验证技术研究进展[J]. *中国科学(信息科学)*, 2024(3): 514-565.)

[3] Wang Q, Wang C, Ren K, et al. Enabling public auditability and data dynamics for storage security in cloud computing[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2011, 22(5): 847-859.

[4] Wang Q, Wang C, Li J, et al. Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing[M]. Computer Security – ESORICS 2009. Berlin, Heidelberg: Springer, 2009: 355-370.

[5] Papamanthou C, Shi E, Tamassia R, et al. Streaming Authenticated Data Structures[C]. *Advances in Cryptology – EUROCRYPT 2013*, 2013: 353-370.

[6] Yu C M. POSTER: Lightweight Streaming Authenticated Data Structures[C]. *The 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015: 1693-1695.

[7] Xu J, Wei L W, Wu W, et al. Privacy-preserving data integrity verification by using lightweight streaming authenticated data structures for healthcare cyber–physical system[J]. *Future Generation Computer Systems*, 2020, 108: 1287-1296.

[8] Xu J, Wei L W, Zhang Y, et al. Dynamic fully homomorphic encryption-based merkle tree for lightweight streaming authenticated data structures[J]. *Journal of Network and Computer Applications*, 2018, 107: 113-124.

[9] Schroeder D, Schroeder H. Verifiable Data Streaming[C]. *The 2012 ACM Conference on Computer and Communications Security*, 2012: 953-964.

[10] Sun Y, Chen X Y, Du X H, et al. Dynamic authenticated method for outsourcing data stream with access control in cloud[J]. *Chinese Journal of Computers*, 2017, 40(2): 337-350.
(孙奕, 陈性元, 杜学绘, 等. 一种具有访问控制的云平台下外包数据流动态可验证方法[J]. *计算机学报*, 2017, 40(2): 337-350.)

[11] Sun Y, Liu Q, Chen X Y, et al. An adaptive authenticated data structure with privacy-preserving for big data stream in cloud[J]. *IEEE Transactions on Information Forensics and Security*, 2020, 15: 3295-3310.

[12] Wang M Y, Guo Y, Zhang C, et al. MedShare: A Privacy-Preserving Medical Data Sharing System by Using Blockchain[C]. *IEEE Transactions on Services Computing*, 2021: 438-451.

[13] Zhang X L, Xu Z C, Cheng H B, et al. Secure Collaborative Learning in Mining Pool *via* Robust and Efficient Verification[C]. *2023 IEEE 43rd International Conference on Distributed Computing Systems*, 2023: 794-805.

[14] Xu C, Zhang C, Xu J. vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases[C]. *International Conference on Management of Data*, 2019: 141-158.

[15] Wang H X, Xu C, Zhang C, et al. VChain+: Optimizing Verifiable

Blockchain Boolean Range Queries[C]. *2022 IEEE 38th International Conference on Data Engineering*, 2022: 1927-1940.

[16] Zhang C, Xu C, Xu J L, et al. GEM^2-Tree: A Gas-Efficient Structure for Authenticated Range Queries in Blockchain[C]. *2019 IEEE 35th International Conference on Data Engineering*, 2019: 842-853.

[17] Peng Z, Xu J, Hu H, et al. BlockShare: A Blockchain Empowered System for Privacy-preserving Verifiable Data Sharing[C]. *IEEE Compute Social Technology Communication Data Engineering*, 2022:14-24.

[18] Sun Z, Zhao P, Wang C P, et al. An Efficient and Secure Trading Framework for Shared Charging Service Based on Multiple Consortium Blockchains[C]. *IEEE Transactions on Services Computing*, 2022: 2437-2450.

[19] Zhang C, Xu C, Wang H X, et al. Authenticated Keyword Search in Scalable Hybrid-Storage Blockchains[C]. *2021 IEEE 37th International Conference on Data Engineering*, 2021: 996-1007.

[20] Cui N N, Wang D, Li J X, et al. Enabling Efficient, Verifiable, and Secure Conjunctive Keyword Search in Hybrid-Storage Blockchains[C]. *IEEE Transactions on Knowledge and Data Engineering*, 2023: 2445-2460.

[21] Sun Y S, Yang J C, Xia Q, et al. SMT: Efficient Authenticated Data Structure for Streaming Data on Blockchain[J]. *Journal of Software*, 2023, 34(11): 5312-5329.
(孙钰山, 杨靖聪, 夏琦, 等. SMT: 一种区块链上适用于流数据高效认证的数据结构[J]. *软件学报*, 2023, 34(11): 5312-5329.)

[22] Brassard G, Chaum D, Crépeau C. Minimum disclosure proofs of knowledge[J]. *Journal of Computer and System Sciences*, 1988, 37(2): 156-189.

[23] Krawczyk H, Rabin T, Chameleon Hashing and Signatures[C]. *the 7th Annual Network and Distributed System Security Symposium*, 2000: 143-l54.

[24] Tsiounis Y, Yung M. On the Security of ElGamal Based Encryption[C]. *Public Key Cryptography*, 1998: 117-134.

[25] Goldwasser S, Micali S, Rackoff C. The Knowledge Complexity of Interactive Proof-Systems[C]. *The Seventeenth Annual ACM Symposium on Theory of Computing*, 1985: 291-304.

[26] Ben Sasson E, Chiesa A, Garman C, et al. Zerocash: Decentralized Anonymous Payments from Bitcoin[C]. *2014 IEEE Symposium on Security and Privacy*, 2014: 459-474.

[27] A. Rondelet and M. Zajac. ZETH: On Integrating Zerocash on Ethereum[EB/OL]. 2019: ArXiv, abs/1904.00905.

[28] Parno B, Howell J, Gentry C, et al. Pinocchio: Nearly practical verifiable computation[C]. *IEEE Symposium on Security and Privacy* (*S&P 2013*). IEEE, 2013: 238-252.

林玮 于 2022 年在河南理工大学计算机科学与技术专业获得学士学位。现在信息工程大学网络空间安全专业攻读硕士学位。研究领域为网络与信息安全, 大数据安全。研究兴趣包括数据安全交换, 数据隐私保护。Email: windless124@163.com

孙奕 于 2015 年在北京交通大学信息安全专业获得博士学位。现任信息工程大学密码工程学院教授。研究领域为网络空间安全、数据安全交换。研究兴趣包括大数据安全、网络与信息安全。Email: 11112072@bjtu.edu.cn

杨佳硕 于 2020 年开始在信息工程大学网络信息防御专业攻读学士学位。研究领域为网络空间安全、网络与信息安全。研究兴趣包括大数据安全、数据隐私保护。Email: yljws20230609@163.com

李宇杰 于 2023 年在信息工程大学信息对抗专业获得学士学位。现在信息工程大学网络空间安全专业攻读博士学位。研究领域为联邦学习、数据安全交换。研究兴趣包括网络与信息安全。Email: evil20001120@ 163.com