计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(11): 2870−2887, 2025

面向多层级区块链架构的轻量级高效验证资产跨链转移方法

牛翔宇[1] 孔兰菊[1] 蒋亚丽[1] 钱 进[1] 崔立真[1] 李庆忠[1,2]
[1](山东大学软件学院 济南 250101)
[2](山大地纬软件股份有限公司 济南 250101)
(niuxiangyu@mail.sdu.edu.cn)

A Lightweight and Efficiently Verified Assets Cross-Chain Transfer Method for Multi-Level Blockchains Architecture

Niu Xiangyu[1], Kong Lanju[1], Jiang Yali[1], Qian Jin[1], Cui Lizhen[1], and Li Qingzhong[1,2]
[1](*School of Software*, *Shandong University*, *Jinan* 250101)
[2](*Dareway Software Co.*, *Ltd.*, *Jinan* 250101)

Abstract The multi-level blockchains architecture is an architecture that organizes multiple blockchains into a tree, where each blockchain can control and manage part of the functions and on-chain data of the next level of the blockchains to which it is connected by cross-chain technology. However, the cross-chain transfer of assets under this architecture is a multi-hop cross-chain problem, where the evidence of successful execution of a cross-chain transaction needs to be transmitted and verified in multiple hops on the path from the source chain to the target chain, resulting in longer execution latency of the cross-chain transaction, higher evidence transmission overhead and verification overhead. Therefore, we propose a lightweight and efficiently verified assets cross-chain transfer method for multi-level blockchains architecture, which introduces a top-level witness chain connecting each multi-level architecture and deploys a witness contract on each chain, so that the parent chains of the source and target chains in a cross-chain transaction act as witness chains to drive the completion of the cross-chain transaction. We also introduce a cross-chain transaction verification evidence based on Verkle tree, the method organizes the cross-chain transaction information to be processed in the same block in a Verkle tree employing KZG polynomial commitment, and adds the KZG commitment and the proof data into the evidence, and proves the execution state of the cross-chain transaction by verifying the evidence, so as to optimize the transmission and verification of the evidence. Theoretical analysis and experiments on the prototype of the method prove that the method reduces the execution latency of the cross-chain transaction and reduces the evidence verification overhead compared with the scheme using SPV without increasing the evidence transmission overhead, which is lightweight and efficiently verified.

Key words multi-level blockchains architecture; cross-chain; witness chain; Verkle tree; KZG commitment

摘 要 多层级区块链架构是一种将多个区块链组织成树形的架构,其中每层的区块链可以通过跨链技术控制和管理与其连接的下一层区块链的部分功能和链上数据.然而,这种架构下的资产跨链转移是一个多跳跨链问题,证明跨链交易成功执行的证据需要沿源链到目标链路径进行多跳的传递与验证,导致跨链事务的执行时延较长以及证据传输开销与验证开销较大的问题.为此,提出一种面向多层级区块链架构的轻量级高效验证资产跨链转移方法,该方法引入顶层见证链连接各多层级架构,并在各链上部署

收稿日期: 2024-08-01; **修回日期**: 2025-03-12
基金项目: 国家重点研发计划项目 (2021YFB2700100); 山东省重大科技创新项目 (2021CXGC010108)
This work was supported by the National Key Research and Development Program of China (2021YFB2700100) and the Major Science and Technology Innovation Program of Shandong Province (2021CXGC010108).
通信作者: 孔兰菊(klj@sdu.edu.cn)

见证合约，使跨链事务中的源链和目标链的父链作为见证链驱动跨链事务的完成. 此外，还提出一种基于Verkle树的跨链事务验证证据，将区块内需要处理的跨链事务信息组织在采用KZG（Kate-Zaverucha-Goldberg）多项式承诺的Verkle树中，并将KZG承诺和证明数据加入证据中，通过验证证据证明跨链事务的执行状态，从而优化证据的传输和验证. 理论分析和对方法原型的实验证明，所提方法降低了跨链事务的执行时延，并且相比采用简单支付验证的方案在不增加证据传输开销的基础上降低了证据验证开销，具有轻量级和高效验证的特点.

关键词 多层级区块链架构；跨链；见证链；Verkle树；KZG承诺

中图法分类号 TP391

DOI: 10.7544/issn1000-1239.202440649 **CSTR:** 32373.14.issn1000-1239.202440649

自比特币[1]诞生以来，区块链的发展十分迅速. 由于其具有去中心化、不可篡改、可追溯等特性，区块链被应用到商业[2-3]、政务[4-5]、医疗[6-7]、供应链[8-9]等领域. 随着区块链的应用范围变得广泛，人们对独立区块链之间相互协作、构建更广泛价值互联网的需求逐渐增强，但受限于区块链的“价值孤岛”效应[10]，这一需求长期无法得到满足. 为了解决这一问题，跨链技术开始得到关注和研究.

跨链技术，或链间互操作技术，是一种通过某种机制确认和验证跨链数据的真实性和有效性，在多个区块链之间实现信息和价值流通的技术. 利用跨链技术，一些行业中出现了如图1所示的多层级区块链架构，在这种架构中，多个许可链被组织成树形结构，通过信使节点进行跨链信息的传输. 其中的每层区块链可以通过跨链技术控制和管理与其连接的下一层区块链的部分功能和链上数据[11]，适用于区块链使用方具有不同作用域的场景，例如行政区（省-市-区）、产业链上下游等.

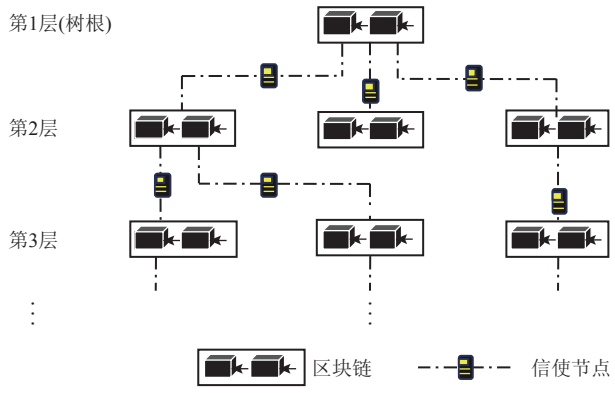


Fig. 1 Multi-level blockchains architecture

图1 多层级区块链架构

在这种架构下，不同区块链之间存在资产跨链转移，实现价值进一步流通的需求. 当源链和目标链间发生资产跨链转移时，源链上的原资产首先需要锁定，然后将证明原资产锁定交易成功执行的“证据”发送到目标链，目标链对“证据”验证成功后，铸造新资产并将证明新资产铸造交易成功执行的“证据”发送回源链进行验证，验证成功后，锁定的原资产将被冻结. “证据”是一个特殊的数据结构，包含能够证明交易在链上发生的证明数据.

然而，在多层级架构下，由于各链之间体现出按作用域隔离、层级化明显等特性，采用将源链与目标链直接相连的跨链方案会因较高的适配复杂度造成较差的可扩展性，而采用将各链连接至中继链的方案则会破坏原有的层级关系. 因此，多层级架构下的资产跨链转移是一个多跳跨链问题，即如果源链和目标链并未直接连接，那么证据就需要在源链和目标链之间的路径上进行多跳传递与验证. 在一般多跳跨链模式下，例如，在一个以国家-省-市为背景的多层级区块链架构下，假设市1链与市2链之间的路径为市1链—省1链—全国链—省2链—市2链，则当市1链的原资产（如户口、医保卡等）需要跨链转移到市2链时，原资产需要先在市1链上锁定，并由信使节点将原资产已锁定的证据 a 传递到省1链，省1链对证据 a 进行验证并生成证据 b，并由信使节点传递到全国链进行验证，以此类推，直到市2链收到证据 d 并进行验证；验证成功后，市2链铸造新资产，该交易的证据将以类似的方式返回至市1链，由市1链验证并将原资产冻结. 在上述过程中，市1链和市2链上交易的成功执行由路径上的所有区块链间接背书.

然而，上述的一般多跳跨链模式虽然可以实现多层级区块链架构下的资产跨链转移，但是却面临2个问题：

1）跨链事务的执行时延较长

在多层级架构资产跨链转移场景下，我们定义一个跨链事务由所有在源链、目标链和路径上的链发生的资产锁定以及铸造、证据验证等跨链交易组

成. 在一般多跳跨链模式下, 跨链事务中的各交易是在各链上串行执行的, 即证据需要从源链传递到目标链, 再从目标链传递回到源链. 假设源链与目标链之间相隔 *hop* 个区块链, 那么跨链事务的执行时延等于交易在 *hop*+2 个区块链上成功执行 2 次(目标链只需执行 1 次)的时间加上证据在链间传递 2(*hop*+1) 次的时间, 跨链事务的整体执行时延较长.

2)证据的传输开销和验证开销较大

现有方案对证据的验证多采用基于简单支付验证(simplified payment verification, SPV)的方式. 首先, 验证链上保存被验证链的验证规则(如被验证链背书节点的公钥、连续的区块头等). 当被验证链收到证据时, 验证链首先根据验证规则验证证据中 Merkle 根哈希的正确性, 再通过对跨链交易进行 Merkle 证明来验证跨链交易的有效性. 在采用 Merkle 树的 SPV 方案中, Merkle 证明的空间复杂度为 $O(\text{lb}\ n)$, 验证复杂度为 $O(\text{lb}\ n)$. 高空间复杂度和验证复杂度使得当跨链交易增多时, 链间传递的证据数量增多以及每个证据的体积增大, 带来较大的链间传输开销, 同时验证链上的 SPV 验证次数和哈希计算次数也增多, 需要消耗更多的链上计算资源, 带来较大的验证开销.

针对上述 2 个问题, 本文提出了一种面向多层级区块链的轻量级高效验证资产跨链转移方法. 具体而言, 本文提出的方法在架构方面引入了顶层见证链连接各多层级架构的树根区块链, 并充分利用树形结构的特点, 将上层父区块链作为下层子区块链的见证链. 各链上部署见证合约 SC_w 负责跨链交易的发起、见证和验证等操作. 见证链的设置使得跨链事务可从源链和目标链同时开始执行, 可以降低跨链事务的执行时延. 进一步地, 本文提出了一种基于采用 KZG(Kate-Zaverucha-Goldberg)多项式承诺[12] 的 Verkle 树[13] 的跨链事务验证证据, 对区块结构进行了改造, 将区块内需要处理的跨链事务信息组织为采用 KZG 多项式承诺的 Verkle 树, 并将 Verkle 树根的承诺保存到区块头. 利用 Verkle 树对跨链事务信息进行多项式承诺, 并将生成的证明数据放入证据中进行传递和验证. 证据的传输开销不会增大甚至更小, 体现了本文方法的轻量级特性, 证据验证所消耗的计算资源更少, 验证开销更小, 体现了本文方法的高效验证特性.

本文的主要贡献包括 3 个方面:

1)提出了一种面向多层级区块链的轻量级高效验证资产跨链转移方法. 方法在多层级区块链架构顶层引入顶层见证链, 并在各链上部署见证合约 SC_w, 使源链和目标链的父链作为见证链见证跨链事务的执行. 见证链的设置使得跨链事务可从源链和目标链同时开始执行, 降低跨链事务的执行时延.

2)提出了一种基于 Verkle 树的跨链事务验证证据. 将区块内需要处理的跨链事务信息组织为采用 KZG 承诺的 Verkle 树保存在区块内. 对跨链事务信息进行多项式承诺并将证明数据加入证据中, 通过证据的生成和验证, 实现在不增加证据传输开销的基础上降低证据验证所需的计算资源开销.

3)实现了方法的原型, 并通过理论分析和实验验证, 证明了本文方法相比于 SPV 方案在跨链事务的执行时延、证据传输开销和验证开销方面有较大优势, 具有轻量级和高效验证的特性.

1 相关工作

目前, 中继链是应用最为广泛的跨链方案. 中继链通过提供统一的跨链数据传递和验证协议连接各条区块链, 并由负责传递数据的节点收集跨链数据并在链间转发, 然后由各条链自行验证跨链数据的正确性, 具有更高的去中心化程度.

Cosmos[14] 是一个由称为 Hub 的中继链和许多独立并行的称为 Zone 的区块链组成的去中心化网络. Hub 与 Zone 之间通过区块链间通信协议 IBC 进行交互. IBC 协议要求需要直接跨链的 2 条链上互相运行对方的轻客户端, 轻客户端通过验证区块头和跨链数据的 Merkle 证明来验证跨链数据的合法性. 通过“Zone—Hub—Zone”实现 Zone 之间的间接跨链.

Polkadot[15] 是一个可扩展的异构多链跨链平台, 提供了一条中继链来连接独立的平行链, 用于平行链间的跨链通信. Polkadot 网络由收集人、验证人、提名人和渔民 4 种基本角色维护. 首先, 跨链交易被加入到源平行链中的出口队列并被收集人执行打包成候选区块交给验证人, 验证人负责验证平行链候选区块, 并在中继链上持久化候选区块, 然后, 跨链交易被加入到目的平行链的入口队列交由目的平行链执行. 其中, 候选区块的有效性证明由区块内交易状态读写集的 SPV 证明组成.

BitXHub[11] 是一个基于中继方案的许可链跨链技术平台, 由中继链通过跨链网关连接各应用链. 中继链主要用于跨链交易的可信验证与可靠路由, 应用链负责具体的业务逻辑. BitXHub 设计了链间消息传输协议 IBTP, 由跨链网关收集跨链数据并处理为特定的结构, 然后传递到目标链上进行验证. BitXHub

同构应用链将跨链交易组织成单独的 Merkle 树，并在 IBTP 中保存跨链交易的 Merkle 路径，由目标链对跨链交易进行 Merkle 证明.

PieBridge[16] 是一种按需可扩展的跨链架构，其基于“按需建域”的理念，将有交互需求的区块链划分在 1 个中继域内，各中继域基于域内中继链实现域内区块链的跨链互联. 域内的中继链可按需平行复制扩展.PieBridge 实现了基于 SPV 的跨链数据验证方法，通过区块头链的同步保证区块头的可信性，通过状态树验证保证指定状态数据在有效区块头中.

Cosmos, Polkadot, BitXHub, PieBridge 通过将具有跨链需求的区块链统一连接到中继链实现跨链. 在多层级区块链架构下，将所有存在资产跨链转移需求的区块链连接至同一条中继链会损害多层级架构的层级性，破坏原有的层级关系，导致资产违规跨链等问题. 在一些对隐私性、安全性和隔离性要求较高的情境下，这种模式的实现具有一定门槛. 此外，当区块链数量增多时，这种模式使得中继链成为整个系统的性能瓶颈. 虽然这 4 种方案可以通过构建中继链网络或中继链平行复制提高性能，但这无疑会带来新的中继链连接适配和维护成本.

见证链又称见证网络，来自文献 [17] 提出的 AC^{3}WN 协议，目的是解决基于哈希时间锁定的原子交换协议[18] 潜在的违反跨链事务原子性问题. AC^{3}WN 在需要跨链的区块链间引入额外的见证链，由跨链用户在见证链上部署见证合约 SC_w 注册跨链事务，并向 SC_w 提供连续的区块头和交易的 Merkle 证明作为跨链交易发生的证据，SC_w 通过验证证据改变跨链事务的状态，再由用户向源链和目标链提供证据证明 SC_w 中的跨链事务状态变动，进而驱动跨链事务完成或中止. 文献 [19] 在 AC^{3}WN 的基础上提出了 EAC^3 协议，将攻击者通过秘密操纵链分叉获取资产的可能性控制在安全水平. 本文利用多层级区块链架构树形结构的特点，将各链的父链作为见证链驱动跨链事务的执行，在保证不改变多层级特性的同时，无需引入额外的见证链维护成本. 本文还扩展了 SC_w 的部署范围和功能，同时修改了证据内容和验证方式，以降低证据的传输开销和验证开销.

2 技术背景

1）Verkle 树

Verkle 树[13] 是一种将 Merkle 树中的哈希函数替换为向量承诺的结构，用于减小 Merkle 证明的体积以节省带宽. 在实践中，Verkle 树多采用类似 MPT（Merkle Patricia trie）的结构和简单高效的多项式承诺充当向量承诺[20]. 多项式承诺可以对一个多项式进行运算产生一个承诺 C，并允许为该多项式上的任意一点产生一个证明 π，通过对 C 和 π 进行运算可以验证该点是否为多项式上的 1 个点.

如图 2 所示，1 棵内部节点宽度为 16 的 Verkle 树由叶子节点、内部节点和空节点 3 种节点组成. 其中，叶子节点保存了数据的 Key 和 Value；内部节点连接具有共同前缀 Prefix 的子节点，并保存子节点的共同前缀，其值为由各子节点通过拉格朗日插值法得到的多项式的承诺. 对 1 棵 Verkle 树进行承诺的过程为从最底层开始对内部节点承诺，再逐层向上对其他内部节点承诺，直至对根节点承诺，在图 2 中，各内部节点的多项式承诺生成顺序为 C_0—{C_1,C_2}—C_3. 根节点 Root 是 1 个特殊的内部节点，保存的承诺 C_3 为整棵 Verkle 树的承诺. 当需要证明叶子节点 Leaf2 在 Verkle 树中时，需要对从 Leaf2 到 Root 的路径上的所有节点进行多项式承诺的证明，即首先证明 Leaf2 是 C_0 承诺的多项式上的点，再证明 Inner0 是 C_1 承诺的多项式上的点，再证明 Inner1 是 C_3 承诺的多项式上的点. 有些多项式承诺方案还支持将路径上的所有证明进行合并，只需要 1 次验证就能证明叶子节点存在于 Verkle 树中，但本文并未实现这一特性.

2）KZG 承诺

KZG 承诺[12] 是一种多项式承诺方案，该方案对提供的多项式计算出一个承诺 C，然后允许在多项式

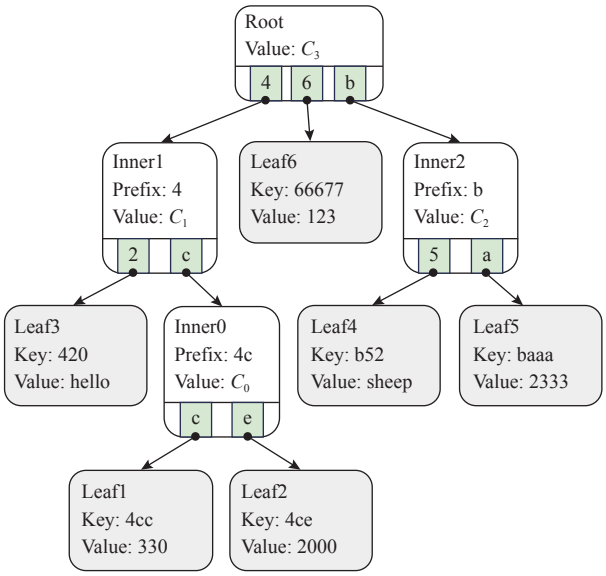


Fig. 2 An example of Verkle tree

图 2 Verkle 树示例

上的任意一点生成一个证明π, 其中C和π都是椭圆曲线上的点. 通过椭圆曲线双线性配对计算在不揭示多项式的情况下证明多项式在特定位置的值与指定的值一致. 假设G_1和G_2分别为2个阶为素数p的循环群, 其生成元分别为g_1和g_2, 且满足双线性配对e: $G_1\times G_2\to G_T$. 对于任意的$x\in F_p$定义$[x]_1=xg_1\in G_1$, $[x]_2=xg_2\in G_2$. 假设证明者拥有一个需要承诺的n阶多项式$p(x)$, 验证者需要验证点(z, y)是否满足$p(z)=y$, 则通过KZG承诺方案实现验证的步骤如下.

① 可信设置. 生成一个随机秘密值τ, 并计算$[\tau^i]_1$和$[\tau]_2$, 其中$i=0, 1, \cdots, n$, 将$[\tau^i]_1$和$[\tau]_2$作为公共参考字符串并丢弃τ. 任何人无法通过$[\tau^i]_1$和$[\tau]_2$破解出τ的值. 可信设置过程通常通过安全多方计算实现.

② 计算承诺. 证明者计算多项式$p(x)$的承诺C并公开, 如式(1)所示:

$$C=\left[p(\tau)\right]_1=\left[\sum_{i=0}^{n}p_i\tau^i\right]_1=\sum_{i=0}^{n}p_i\left[\tau^i\right]_1. \tag{1}$$

由式(1)可见, C是$p(x)$在τ处的值映射到G_1上的点.

③ 生成证明. 根据验证者提供的z, 证明者计算$p(z)$, 则多项式$p(x)-p(z)$在$x=z$达到0点, 于是$p(x)-p(z)$一定能被$x-z$整除. 证明者计算商多项式$q(x)=\dfrac{p(x)-p(z)}{x-z}$, 根据式(2)计算证明$\pi$并发送给验证者.

$$\pi=[q(\tau)]_1. \tag{2}$$

由式(2)可见, π是$q(x)$在τ处的值映射到G_1上的点.

④ 验证. 验证者得到C与π后, 验证式(3)是否成立.

$$e(\pi,[\tau-z]_2)=e(C-[y]_1,g_2), \tag{3}$$

其中$[\tau-z]_2=[\tau]_2-[z]_2=[\tau]_2-zg_2$, $[y]_1=yg_1$. 如果等式成立, 则说明$p(z)=y$, 验证成功.

3 方法概述

本文提出了一种面向多层级区块链的轻量级高效验证资产跨链转移方法. 本节将首先介绍方法的整体设计思路, 包括方法的整体架构和协议栈, 然后介绍方法的设计细节和资产跨链转移流程.

3.1 整体架构和协议栈

本文引入顶层见证链连接各多层级架构, 并充分利用树形结构的特点, 将上层父区块链作为下层子区块链的见证链, 如图3所示, 方法的区块链架构由顶层见证链、父链、子链和信使节点组成.

1)顶层见证链. 顶层见证链是一条额外引入的区块链, 连接了多个多层级架构的树根区块链, 作为各架构之间跨链的桥梁以及树根区块链的见证链. 顶层见证链上不设置跨链资产, 无法作为资产跨链转移的源链或目标链. 当需要连接的架构增多, 顶层见证链达到性能瓶颈时, 可以进行平行扩展, 增加顶层见证链的数量来组成顶层见证链网络, 并将一部分多层级架构重新连接到新顶层见证链上, 实现负载均衡.

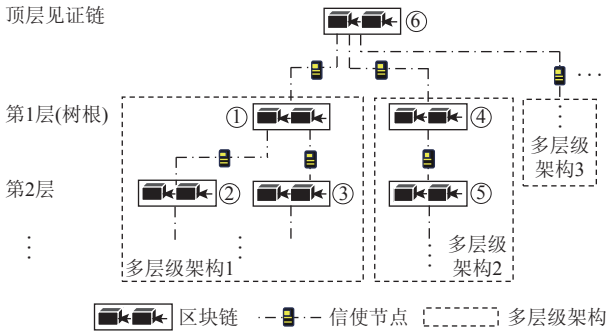


Fig. 3 Blockchains architecture in our method

图3 本文方法中的区块链架构

2)父链和子链. 父链和子链是一个相对的概念, 对于2个直接相连的区块链, 上层链是下层链的父链, 下层链是上层链的子链. 例如在图3中, 链①是链⑥的子链, 同时也是链②的父链. 父链将作为子链的见证链, 驱动跨链事务完成或中止.

3)信使节点. 连接父链和子链的信使节点负责在父链和子链之间收集跨链数据, 构造和传播证据. 根据需求不同, 信使节点可以设计为单独的节点, 也可以是运行共识算法和激励机制的节点集群, 关于信使节点的具体实施细节不在本文的讨论范围之内.

正如引言所述, 多层级架构下的资产跨链转移问题可以转化为多跳跨链问题. 具体而言, 源链和目标链可能处于同一层级(例如图3中的链②与链③), 也可能处于不同层级(例如图3中的链②与链④). 由于见证链的设置, 跨链事务在源链和目标链上的执行状态需要经过二者之间路径上的其他区块链路由到彼此的见证链, 这一过程与源链和目标链的层级关系无关, 只与源链和目标链之间路径上相隔区块链的个数(即跳数)有关, 根据跳数的不同, 跨链事务可以分为单跳跨链事务和多跳跨链事务, 详细内容将在3.2.2节介绍. 考虑到部分应用场景下对资产的跨链权限有特殊要求, 例如要求资产只能在同层级区块链间转移, 本文方法可以对跨链事务的源链和目标链进行约束, 默认资产可以跨层级转移.

本文方法的协议栈如图4所示, 一个跨链事务将

从源链和目标链同时开始执行直至本链的见证链(图 4 路径①)，再从本链见证链执行至对方链见证链(图 4 路径②)，最后从本链见证链开始执行直至本链结束(图 4 路径③)．在源链和目标链中，资产合约层主要负责控制资产合约 SC_a 的状态变化，目标链上的资产合约层还负责 SC_a 的铸造(合约部署)；见证合约层指在每条链上部署的 SC_w，负责由资产合约层调用发起跨链事务，同时对信使节点发来的证据进行验证，并调用 SC_a 进行状态变更；跨链账本层则负责将需要处理的跨链事务记录到 Verkle 树的叶子节点中，并为该跨链事务提供相应的多项式承诺和证明数据；SDK 负责为信使节点提供获取某个跨链事务的证明数据的接口．

信使节点会持续监听连接的父链和子链，当监听到相关的由 SC_w 触发的跨链事件时，信使节点会调用 SC_w 所在链的 SDK 获得跨链事务的证明数据，并将跨链事务信息及其证明构造成证据传递给下一条链的 SC_w 处理．每个信使节点都维护一个分布式哈希表 ID-DHT，保存了以其所连接的父链为根节点的子树中的所有链 ID．当监听到跨链事件时，信使节点将判断相关跨链事务的源链 ID 或目标链 ID 是否在 ID-DHT 内，以及是否需要由自己处理．当一条新的子链通过信使节点与父链建立连接时，信使节点会将链 ID 更新到 ID-DHT 中，然后将包括链 ID 在内的子链信息传输到父链 SC_w．之后，父链与更上层的父链之间的信使节点会将链 ID 更新到 ID-DHT 中，以此类推，直到顶层见证链与第 1 层区块链之间的信使节点将链 ID 更新到 ID-DHT 中．

见证链的见证合约层负责对跨链事务进行注册，同时根据收到的来自不同链的证据的类型和数量，对跨链事务状态进行相应的变更，从而驱动跨链事务完成或中止．图 4 中第 $L-1$ 层的源链见证链和目标链

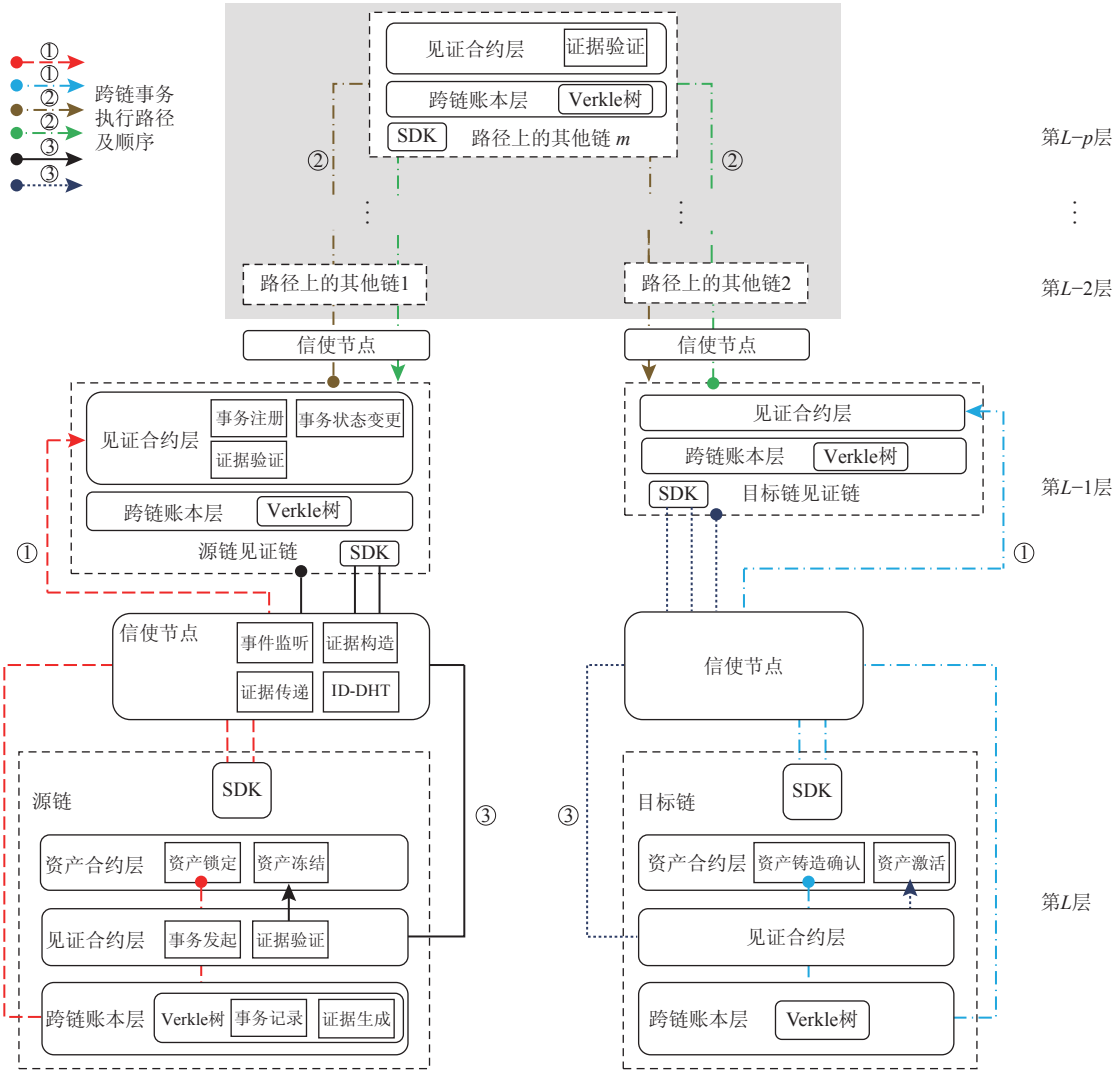


Fig. 4 Protocol stack of the assets cross-chain transfer method for multi-level blockchains architecture

图 4 面向多层级区块链架构的资产跨链转移方法协议栈

见证链之间的其他链(第$L-2$层到第$L-p$层的链)和信使节点将作为2条见证链之间的桥梁,将跨链事务在源链或目标链上的执行状态路由到对方见证链.

需要注意的是,图4中源链和目标链虽然同在第L层,但具有不同的父链作为见证链,若源链和目标链父链相同,则共同父链将作为见证链,具体内容将在3.3.1节介绍.

3.2 设计细节

3.2.1 用户与跨链资产 SC_a

用户分为普通用户、资产发行用户和管理员用户.其中,普通用户和资产发行用户在源链和目标链上均有账户.普通用户即资产的拥有者,有将自己拥有的资产转移至其他链的需求;资产发行用户即跨链资产的发行者,是资产合约的铸造者.管理员用户由权威机构设立,主要负责 SC_w 的审核和注册新资产发行用户及资产.

每个跨链资产 SC_a 都是一个合约,由资产发行用户发行,由普通用户所有,在 SC_w 中注册,并具备内部调用 SC_w 发起跨链事务的功能.只有当普通用户和资产发行用户都同意发起资产转移时(都正确调用了 SC_a 中的 *cross_chain_transfer*() 时),SC_a 才会调用 SC_w 发起跨链事务.SC_a 共有3种状态:激活(A)、未激活(NA)和锁定(L),其状态变化规则如图5所示,具体规则将在3.3节介绍.

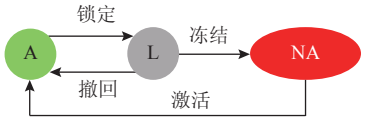


Fig. 5 The state change rule for a cross-chain asset

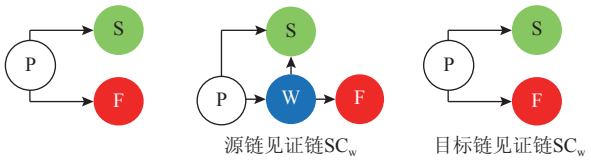
图5 跨链资产的状态变化规则

3.2.2 跨链事务

在本文方法中,跨链事务由所有在源链、目标链和路径上的链上发生的相关交易组成,例如资产锁定和铸造以及证据验证等.具体而言,跨链事务由源链上的资产锁定与冻结交易、目标链上的资产铸造与确认交易、见证链上的跨链事务注册和状态变更交易以及路径上其他链的证据验证交易组成.正如在3.1节中提到的多跳跨链问题,本文根据源链与目标链之间的跳数,将跨链事务分为单跳跨链事务和多跳跨链事务.例如,如图3所示,一个链②与链③之间的跨链事务中间只相隔链①1跳,因此是一个单跳跨链事务;一个链②与链⑤之间的跨链事务中间相隔链①—链⑥—链④3跳,因此是一个多跳跨链事务.对于源链与目标链分别为子链和父链的跨链事务,例如链②与链①之间的跨链事务,由于链②的见证链为链①,链①的见证链为链⑥,源链与目标链间相隔链①与链⑥2跳,因此该跨链事务是一个跳数为2的多跳跨链事务.

每个跨链事务都有唯一的ID,并需要在源链见证链和目标链见证链的 SC_w 中注册.如图3所示,链②与链③之间的单跳跨链事务将在二者的共同父链链①的 SC_w 中注册,而链②与链⑤之间的多跳跨链事务将在二者各自的父链链①和链④的 SC_w 中注册.此外,每个由 SC_w 验证过证据,需要交由下一个区块链处理的跨链事务信息将被单独保存在区块中并组织为一棵Verkle树,相关内容将在第4节介绍.

如图6所示,每个在 SC_w 中注册的跨链事务都有一个状态,对于单跳跨链事务,状态共有发布(P)、成功(S)和失败(F)三类,而多跳跨链事务的状态还增加了撤回中(W).2种跨链事务的状态变化规则不同,具体规则将在3.3节介绍.



(a) 单跳跨链事务 (b) 多跳跨链事务

Fig. 6 State change rules for two types of cross-chain transaction

图6 2种跨链事务的状态变化规则

为了驱动跨链事务状态的变更,跨链事务在源链和目标链上的执行状态(由证据中的跨链事务信息表示)需要路由到对方链的见证链,这一过程通过路径上所有区块链的 SC_w 和信使节点完成.信使节点监听到 SC_w 触发相关跨链事件时,判断链ID是否在ID-DHT内,若链ID在ID-DHT内,则进行证据的构造,并传递给相应链;否则直接忽略跨链事件.在一个跨链事务执行的过程中,上述路由过程将重复多次.

3.2.3 见证合约 SC_w

见证合约 SC_w 部署在架构中所有的区块链上,是整个方法的核心组件之一,具有新链注册、新资产发行用户和资产注册、信使节点注册、跨链事务发起、证据验证、跨链事务见证、跨链事务中止、资产状态变更等功能.

1)新链注册.一个新链若要加入架构中实现资产跨链转移,必须部署 SC_w 并由管理员用户审核,以生成唯一链ID供信使节点进行证据路由.同时由新

链和其父链的管理员用户将对方链的共识验证信息保存到 SC_w 中(例如共识节点的公钥).

2)新资产发行用户和资产注册. 一个资产发行用户若要获得发行跨链资产的权利, 必须先在 SC_w 中注册身份并由管理员用户审核. 同样, 资产发行用户发行的每一个新 SC_a 都必须向 SC_w 注册并由管理员用户审核, 以获得内部调用 SC_w 发起跨链事务的权利.

3)信使节点注册. 当首次部署以及后续有新的信使节点加入时, SC_w 必须记录各信使节点的基本信息, 例如共识验证规则、公钥证书等, 用于验证信使节点所提交的证据的合法性.

4)跨链事务发起. 跨链事务发起是指跨链资产内部调用本链上的 SC_w, 由 SC_w 生成跨链事务 ID 并记录源链 ID、目标链 ID、资产合约地址等信息, 然后通过触发事件 *InitTx* 将跨链事务 ID 和对方链 ID 提交给信使节点处理的过程.

5)证据验证. SC_w 接收信使节点发送的证据, 并验证证据的合法性. 若 SC_w 是跨链事务的见证链, 则进行跨链事务见证; 若 SC_w 是跨链事务的源链或目标链, 则进行资产状态变更; 若 SC_w 是跨链事务路由路径上的链, 则触发 *RouteTx* 或 *FailRouteTx* 事件, 促使信使节点将跨链事务路由到下一区块链处理.

6)跨链事务见证. 跨链事务见证是 SC_w 的核心功能, 包括跨链事务注册和跨链事务状态变更 2 个部分. 其中, 跨链事务注册是指 SC_w 第 1 次收到由子链作为源链或目标链的跨链事务的合法证据时, 将跨链事务信息记录下来的过程. 跨链事务状态变更是指 SC_w 通过验证证据, 改变记录的跨链事务状态, 决定完成或中止跨链事务, 触发 *FinishTx* 或 *FailTx* 事件的过程.

7)跨链事务中止. 每个跨链事务都有一个超时时间, 若跨链事务在超时时间内仍未完成, 用户可在源链调用 SC_w 发起跨链事务中止请求, 此时会触发 *AbortTx* 事件, 信使节点将构造中止跨链事务的证据, 并传递给见证链 SC_w 处理.

8)资产状态变更. 当 SC_w 收到跨链事务完成或中止的证据时, 调用跨链事务所涉及的 SC_a 进行资产状态的变更.

表 1 总结了 SC_w 可能触发的事件类型、触发功能、事件参数和含义. 这些事件将由相连的信使节点监听并处理. 其中, *txid* 表示跨链事务 ID, *cid* 表示链 ID.

Table 1 Types of Events in SC_w

表 1 SC_w 中的事件类型

事件类型	触发功能	事件参数	含义
InitTx	跨链事务发起	*txid*, *cid*	发起跨链事务 *txid* 且需要路由到链 *cid*.
FinishTx	跨链事务见证	*txid*	跨链事务 *txid* 已完成.
RouteTx	证据验证	*txid*, *cid*	跨链事务 *txid* 需要被路由到链 *cid*.
AbortTx	跨链事务中止	*txid*, *cid*	用户请求中止跨链事务 *txid* 且需要路由到链 *cid*.
FailTx	跨链事务见证	*txid*	跨链事务 *txid* 已中止.
FailRouteTx	证据验证	*txid*, *cid*	需要中止的跨链事务 *txid* 需要被路由到链 *cid*.

3.3 资产跨链转移

3.3.1 跨链事务的结构与状态变化规则

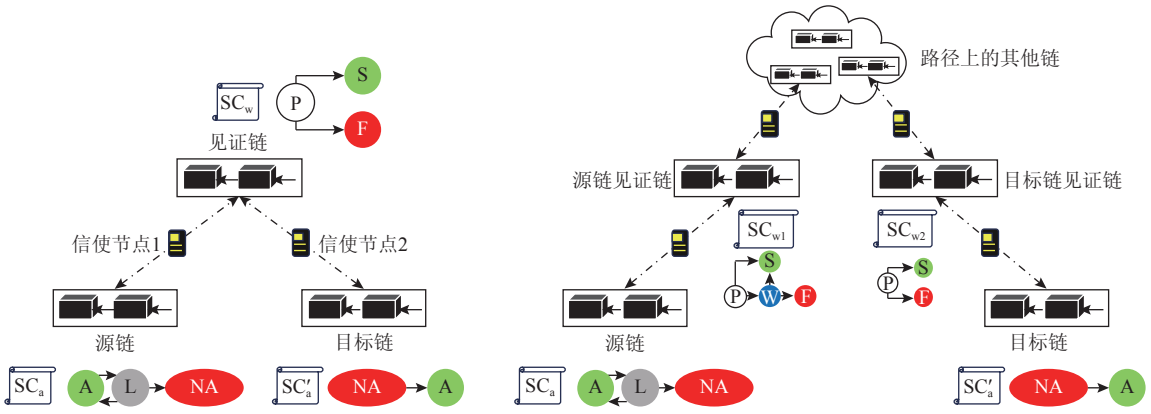
图 7 展示了 2 种跨链事务执行时的结构, 以及跨链资产和跨链事务的状态变化规则.

在单跳跨链事务中, 源链和目标链具有共同的父链作为共享见证链. 当跨链事务发起时, 源链上的原资产 SC_a 将被锁定, 状态将从 A 转变为 L, 此时 SC_a 将无法正常使用. 与此同时目标链上由资产发行方铸造新资产 SC'_a, 初始状态为 NA.

当跨链事务在见证链的 SC_w 注册时初始状态为 P, 注册后若 SC_w 收到第 2 个合法证据, 则跨链事务状态将由 P 转变为 S, 此后信使节点构造证据提交到源链 SC_w 或目标链 SC_w 证明跨链事务完成, 由源链 SC_w 将 SC_a 冻结, 状态由 L 改为 NA, 由目标链 SC_w 将 SC'_a 激活, 状态由 NA 改为 A.

若跨链事务在超时时间内仍未完成执行, 用户可在源链调用 SC_w 发起跨链事务中止请求, 此时信使节点将构建证据提交到见证链 SC_w, 若跨链事务状态为 P, 则被变更为 F, 否则将保留原状态. 若跨链事务状态被转变为 F, 则信使节点构造证据提交到源链 SC_w 或目标链 SC_w 证明跨链事务中止, 由源链 SC_w 将 SC_a 的锁定撤回, 状态由 L 改为 A, SC'_a 状态则仍为 NA.

多跳跨链事务流程与单跳跨链事务相似, 但有几处不同. 首先, 见证链为源链和目标链各自的父链, 此时跨链事务将在源链见证链 SC_{w1} 和目标链见证链



(a) 单跳跨链事务 (b) 多跳跨链事务

Fig. 7 Structure and state change rules for cross-chain transactions

图 7 跨链事务的结构与状态变化规则

SC_{w2} 中注册. 其次, 2 个见证链之间存在由源链到目标链间路径上的其他链(也可能不存在, 即见证链 1 和见证链 2 在顶层见证链网络中). 此外, SC_{w1} 中注册的跨链事务状态出现了 W, 该状态是用户发起跨链事务中止请求时的中间状态. 当用户发起跨链事务中止请求后, SC_{w1} 中的跨链事务状态将由 P 变为 W, 然后这一请求将被路由至 SC_{w2}, 若 SC_{w2} 收到中止请求时, 跨链事务状态仍为 P, 则将其变更为 F, 若 SC_{w2} 尚未注册跨链事务, 则直接注册跨链事务且将状态设置为 F. 同时这一变更的证据将被路由回 SC_{w1}, SC_{w1} 将变更跨链事务状态为 F. 若 SC_{w2} 收到中止请求时, 跨链事务状态为 S 或 F, 则跨链事务中止的请求将被忽略. 此后当 SC_{w1} 收到跨链事务完成或中止的证据时, 将跨链事务状态从 W 变更为 S 或 F.

3.3.2 跨链事务原子性分析

跨链事务执行中的某一步骤可能会出现异常, 导致违反跨链事务的原子性. 在本文场景下, 跨链事务的原子性被定义为在跨链事务执行结束后, SC_a 和 SC'_a不会同时处于 A 状态.

一方面, 恶意的攻击者可能会对 SC_a, SC'_a, SC_w 和信使节点进行攻击尝试使 SC_a 和 SC'_a同时处于 A 状态, 进而同时持有 2 份有效的资产而非法获益.

1)对 SC_a, SC'_a, SC_w 的攻击

在单跳跨链事务中, SC_a 和 SC'_a的状态变更由见证链 SC_w 中记录的跨链事务状态决定. 对于一个处于 L 状态的 SC_a 和处于 NA 状态的 SC'_a, 只有在 SC_w 接收到正确的证据后才会被置为状态 A, 而 SC_w 只能通过验证信使节点传递的证据改变跨链事务的状态, 因此攻击者无法通过直接操控 SC_a, SC'_a, SC_w 违反原子性. 此外, 假如 SC_w 在跨链事务 P 状态时同时收到了跨链事务完成和中止的证据, 由于跨链事务状态只能从 P 变为 S 或从 P 变为 F, 2 个改变跨链事务状态的交易不能出现在同一个见证链区块中, 见证链矿工必须从二者中选择 1 个. 因此, 这 2 个状态更改必须分别记录在 2 个区块中. 但在采用 PBFT 等确定性共识算法的许可链中, 由于不存在区块链分叉的情况, 2 个区块的出现必须有先后之分. 这说明跨链事务状态将从 S 转变为 F 或从 F 转变为 S, 然而, 这与跨链事务状态变化规则矛盾, 因此, 不存在跨链事务同时出现 S 状态和 F 状态的情况. 即便发生这种状况, 由于源链侧的信使节点和目标链侧的信使节点同时监听跨链事务状态变动的事件, 双方信使节点只会对第 1 个事件进行响应, 因此不会存在 SC_a 和 SC_a'的状态同时为 A 的情况.

在多跳跨链事务中, SC_a 和 SC'_a的状态分别由见证链 SC_{w1} 与 SC_{w2} 中的跨链事务状态决定. 而 SC_{w1} 与 SC_{w2} 中的跨链事务最终状态不一致的情况出现在用户发起跨链事务中止请求时. 在单跳跨链事务原子性分析的基础上, 由于 SC_{w1} 中的跨链事务存在中间状态 W, 因此当用户请求跨链事务中止时, SC_{w1} 中的跨链事务状态将与 SC_{w2} 中的跨链事务保持一致, 因此不会出现跨链事务状态在不同的 SC_w 中分别为 F 和 S 的情况. 此外, 在路径上的其他链上, SC_w 只会对信使节点传递的证据进行验证, 进而触发相应事件, 促使信使节点完成跨链事务的路由. 因此攻击者无法通过直接操控路径链上的 SC_w 违反原子性.

因此, 攻击者只能通过在链上部署存在漏洞的 SC_w 操纵证据的验证结果, 使跨链事务在源链和目标链上的状态矛盾, 或发行存在漏洞的 SC_a 进而直接改变资产状态, 使 SC_a 和 SC_a'同时处于 A 状态. 然而, 存

在漏洞的 SC_w 将无法通过管理员用户的审核，因此无法成功应用到架构中参与资产跨链转移，进而无法发行存在漏洞的 SC_a，跨链事务的原子性得以保证.

2)对信使节点的攻击

首先，攻击者可以攻击信使节点进而伪造虚假的证据使跨链事务在源链见证链中的状态为 F，在目标链见证链中的状态为 S，导致 SC_a 从 L 状态解锁为 A 状态，SC'_a 从 NA 状态激活为 A 状态. 然而，由于证据中包含证据来源链的共识节点对相关区块的签名列表，而恶意的信使节点无法伪造这些签名，因此虚假证据无法通过 SC_w 的签名验证. 因此恶意的信使节点无法通过伪造证据的方式违反跨链事务原子性.

其次，恶意的信使节点有可能拒绝在链间传递证据，这可能会影响跨链事务的执行效率，但不会违反跨链事务的原子性. 在实际应用过程中，可以通过设置集群，采用安全的共识算法等方式提高信使节点的可用性，降低被攻击的风险. 具体实施细节不在本文的讨论范围之内.

最后，当出现网络阻塞、区块链吞吐量瓶颈等情况时，跨链事务子交易可能无法及时执行，跨链事务的执行效率将会降低，整体执行时延增大，但跨链事务的执行过程不受影响，因此原子性不会受到影响.

综上所述，跨链事务在 SC_w 中有着严格的状态变化规则，且只能通过验证信使节点传递的证据自动改变状态，同时 SC_a 和 SC'_a 也只能根据跨链事务的状态进行状态变化，因此攻击者无法通过直接操纵 SC_a, SC'_a, SC_w 而违反原子性. 每条链上管理员用户的设置则有效防止了攻击者通过部署存在漏洞的合约违反原子性的情况. 而证据中的签名列表则使得攻击者无法通过攻击信使节点伪造证据的途径违反原子性. 而网络阻塞、区块链吞吐量瓶颈等异常情况仅仅会影响跨链事务的执行效率而不会影响原子性. 通过以上分析，本文方法可以有效保证跨链事务的原子性.

3.3.3 资产跨链转移流程

本节以资产转移单跳跨链事务为例介绍方法实现资产跨链转移的流程，具体过程如图 8 所示，包括跨链请求发起、跨链请求路由、跨链见证、跨链结果路由和资产状态变更 5 个阶段.

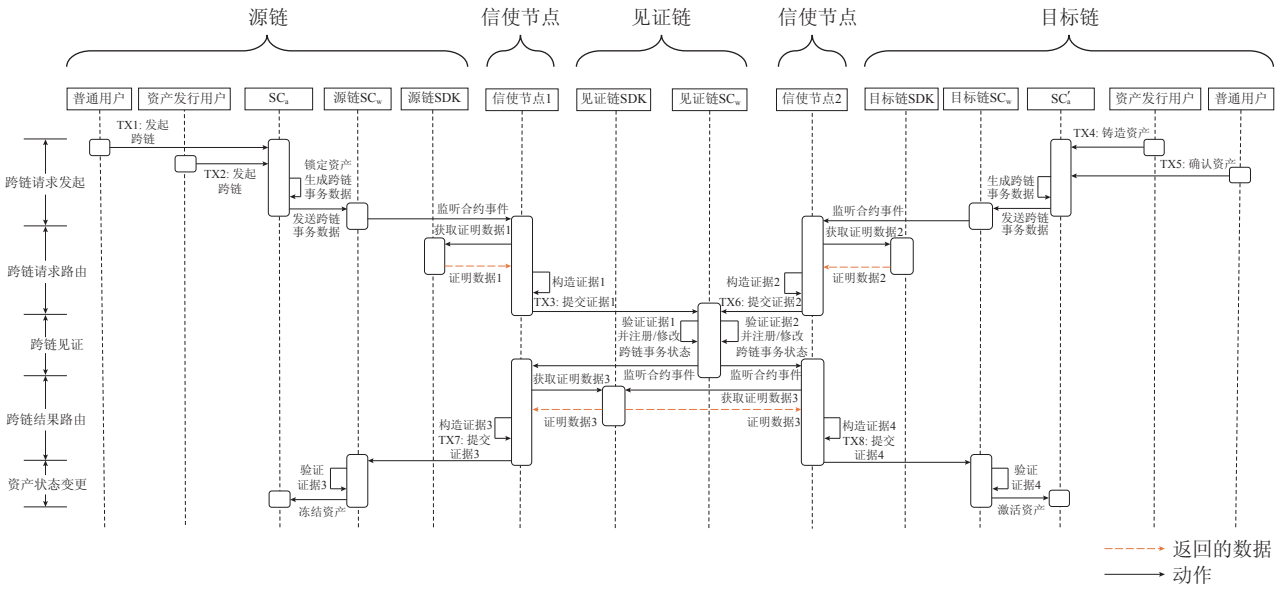


Fig. 8 Process of a single-hop cross-chain transaction for asset cross-chain transfer

图 8 资产跨链转移单跳跨链事务流程

1)跨链请求发起阶段. 资产拥有用户和资产发行用户在源链上调用 SC_a 触发跨链请求，SC_a 设置为 L 状态，同时生成跨链事务信息发送给源链 SC_w，由源链 SC_w 触发 *InitTx* 事件. 与此同时，在目标链上，资产发行用户铸造新资产 SC'_a（初始状态为 NA）并由资产拥有用户检查确认，SC'_a 生成跨链事务数据发送给目标链 SC_w，由目标链 SC_w 触发 *InitTx* 事件.

2)跨链请求路由阶段. 信使节点 1 通过监听源链 *InitTx* 事件获取跨链事务数据，并调用源链 SDK 得到跨链事务在源链执行发起的证明数据 1，构造证据 1 以交易的形式提交给见证链 SC_w. 同样，信使节点 2 将构造证据 2 提交给见证链 SC_w.

3)跨链见证阶段. 见证链 SC_w 在收到证据后对证据进行验证. 当收到第 1 个合法证据后，见证链 SC_w 将跨链事务信息记录下来，设置状态为 P. 当收到第 2 个合法证据后，若跨链事务状态为 P，则见证链

SC_w 将跨链事务状态变更为 S, 并触发 *FinishTx* 事件表示跨链事务完成.

4)跨链结果路由阶段. 信使节点 1 和信使节点 2 通过监听见证链 *FinishTx* 事件获取已完成的跨链事务数据, 并调用见证链 SDK 得到跨链事务在见证链完成的证明数据 3, 分别构造证据 3 和证据 4 以交易的形式提交给源链 SC_w 和目标链 SC_w.

5)资产状态变更阶段. 源链 SC_w 和目标链 SC_w 在收到信使节点发送的跨链事务完成的证据后, 对证据进行验证, 然后调用对应的资产合约进行状态变更. 在源链上, SC_w 将 SC_a 的状态由 L 设置为 NA; 在目标链上, SC_w 将 SC'_a 的状态由 NA 设置为 A.

多跳跨链事务的流程与单跳跨链事务类似, 不同之处在于需要在 2 个见证链上进行见证, 跨链事务在一条链上的执行信息需要路由到另一条链, 因此在跨链请求路由阶段和跨链结果路由阶段, 需要多次信使节点与路径上的其他链进行交互.

SC_w 通过证据的验证得知跨链事务在上一条链的执行状态, 有关证据的内容将在第 4 节中介绍.

4 基于 Verkle 树的跨链事务验证证据

本节将对基于 Verkle 树的跨链事务验证证据进行介绍. 首先介绍证据的结构, 然后介绍区块中由跨链事务信息组成的 Verkle 树, 最后介绍证据的构建和验证过程.

4.1 证据的结构

如表 2 所示, 一个证据包含了 10 个字段, 保存了证据的 ID、摘要、区块信息、跨链事务相关信息和证明, 以及 2 个签名列表.

每个信使节点发送的证据都有 1 个单调递增的唯一 ID, 便于 SC_w 实现证据的过滤, 也可以根据需求实现证据的有序处理. *digest* 为证据内容的摘要. 区块信息主要包括跨链事务在上一条区块链中被处理时所在区块头的哈希值 *bHash* 和区块头中其他字段的聚合哈希值 *subBHash*. 为了便于证据的验证, 本文方法修改了区块结构, 将由跨链事务信息构成的 Verkle 树根的承诺字符串 C_{Root} 保存到区块头中, 并按式(4)计算区块哈希值.

$$bHash = hash(subBHash, C_{\text{Root}}), \tag{4}$$

即 *bHash* 由 *subBHash* 与 C_{Root} 聚合后再进行 1 次哈希计算得到.

跨链事务相关信息包括跨链事务的 ID(*transactionId*)、处理类型(*type*)和信息(*param*). 其中, *transactionId* 是一个哈希值, *type* 告知 SC_w 应如何处理跨链事务, 根据 *type* 的不同, *param* 具有不同的值, 表 3 展示了 *type* 与 *param* 的类型, 其中 *cid* 表示区块链 ID.

Table 2 Structure of Evidence

表 2 证据结构

证据字段	含义
id	证据的 ID
digest	证据内容摘要哈希值
bHash	区块头哈希值
subBHash	区块头中其他字段的聚合哈希值
transactionId	跨链事务 ID
type	跨链事务处理类型
param	跨链事务信息
proof	跨链事务的证明数据
sigList	区块链共识节点签名列表
mSigList	信使节点的签名列表

Table 3 The Types of *type* and *param* in Evidence

表 3 证据中的 *type* 和 *param* 类型

类型	*type*	*param*	含义
Init	0	*cid*	发起跨链事务且需要路由到链 *cid*.
Finish	1	无	跨链事务已完成.
Route	2	*cid*	跨链事务需要被路由到链 *cid*.
Abort	−1	*cid*	用户请求中止跨链事务且需要路由到链 *cid*.
Fail	−2	无	跨链事务已中止.
FailRoute	−3	*cid*	需要中止的跨链事务需要被路由到链 *cid*.

如表 3 所示, *type* 与 *param* 的类型与 SC_w 可能触发的事件类型与参数十分对应. 当 SC_w 收到 Init 类型的证据时, 表示当前区块链的子链发起了跨链事务, 因此需要作为见证链注册跨链事务, 同时对 *cid* 进行检查. 若 *cid* 链是当前区块链的子链, 则等待另一方链证据的到来; 若 *cid* 链不是当前区块链的子链, 则需要触发 *RouteTx* 事件. 当 SC_w 收到 Finish 类型的证据时, 则需要对跨链事务所对应的资产合约进行状态变更. 当 SC_w 收到 Route 类型的证据时, 对 *cid* 进行检查, 若 *cid* 链是当前区块链的子链, 则对跨链事务的状态进行变更; 若 *cid* 链不是当前区块链的子链, 则需要触发 *RouteTx* 事件. 需要注意的是, 若一方见证链在收到子链的 Init 类型证据之前收到了来自另一方的 Route 类型的证据, 且 *cid* 是子链, 则进行跨链事务的注册, 在收到子链的 Init 类型证据时进行跨链事务的状态变更. Abort, Fail, FailRoute 类型的证据的处理过程与 Init, Finish, Route 类型的证据类似, 此处不再赘述.

proof 字段是跨链事务在前一区块链中被正确处理的证明数据，具体组成将在 4.2 节介绍，*proof* 字段中包含了 C_{Root}.*sigList* 和 *mSigList* 是 2 个签名列表，其中 *sigList* 保存了前一区块链共识节点的签名列表，供 SC_w 验证区块的有效性，保证了 C_{Root} 的正确性；*mSigList* 是连接当前区块链和前一区块链的信使节点的签名列表，供 SC_w 验证证据的合法性.

4.2 跨链事务信息 Verkle 树

本区块内所有被 SC_w 处理的需要被下一区块链继续处理的跨链事务信息将被组织为 1 棵 Verkle 树，同时 Verkle 树根节点的承诺字符串 C_{Root} 也将被保存到区块头中，图 9 展示了一个区块内由跨链事务构成的 Verkle 树的示例.

如图 9(a)表格所示，假设本区块内共有 5 个跨链事务被 SC_w 处理. 每个跨链事务都将被图 9(b)的 Verkle 树中的 1 个叶子节点保存，其中 Key 为跨链事务 ID，它是一个哈希值，Value 由证据的 *type* 与 *param* 按顺序拼接而成. 内部节点连接了所有具有共同前缀的叶子节点和内部节点，其中，Prefix 保存了共同前缀，Value 则保存了由所有子节点的 Key/Prefix-Value 通过拉格朗日插值法构成的多项式的 KZG 承诺，该多项式的参数数组 *coeffs* 将被保存在 Poly 字段，该字段不会公开. 出块矿工构建 Verkle 树的算法如算法 1 所示.

算法 1. Verkle 树的构建.

输入：跨链事务列表 *txList*，Verkle 树节点宽度 *w*；

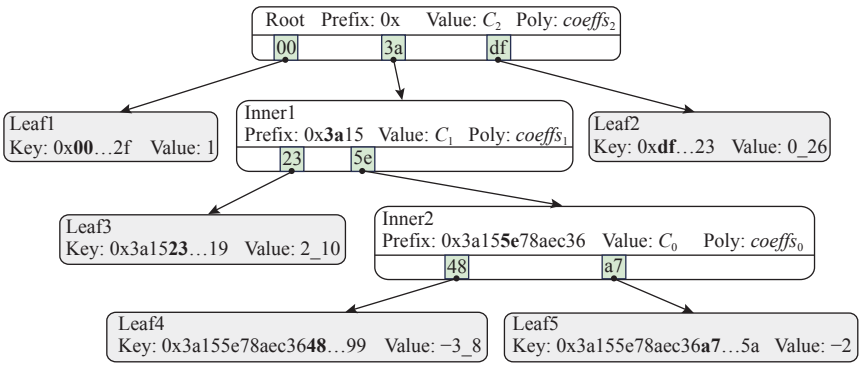
输出：跨链事务 Verkle 树 *t*.

① function *construct*(PatriciaTree *pt,* InnerNode *n*)
/*定义将 1 棵 Patricia 树构建为 Verkle 树的函数*/
② for each *cn* in *n.childNode*
③ if *cn.isInstanceOf* (InnerNode) and *cn.Value* == NULL
④ *pt* ← *construct*(*pt*, *cn*);
⑤ end if
⑥ end for
⑦ *points* ← *List*((1, 1));
⑧ for each *cn* in *n.childNode*
⑨ if *cn.isInstanceOf* (InnerNode)
⑩ *x* ← *cn.Prefix*;
⑪ else
⑫ *x* ← *cn.Key*;
⑬ end if
⑭ *y* ← *x* + *cn.Value*;
⑮ *points.append*((*x*, *y*));
⑯ end for
⑰ *coeffs* ← *interpolate*(*points*);
⑱ *n.Poly* ← *coeffs*;
⑲ *C* ← *commit*(*coeffs*);
⑳ *n.Value* ← *C*;
㉑ *pt.update*(*n*);
㉒ return *pt*;
㉓ end function
㉔ *t* ← *constructPatriciaTree*(*w*); /*首先构建 1 棵 Patricia 树*/
㉕ for each *tx* in *txList*
㉖ *t.insert*(*tx.transactionId*, *tx.type*, *tx.param*);
㉗ end for
㉘ *t* ← *construct*(*t*, *t.Root*).

算法 1 将输入中的跨链事务列表 *txList* 构建为一棵宽度为 *w* 的 Verkle 树 *t*. 首先定义了一个函数 *construct*()，该函数接受一棵 Patricia 树 *pt* 和一个内部节点 *n* 为参数，对 *pt* 内以 *n* 为根节点的子树进行承诺. 该函数首先对 *n* 的所有子节点 *cn* 进行遍历，若 *cn* 是

序号	*transactionId*	*type_param*
①	0x00…2f	1
②	0xdf…23	0_26
③	0x3a1523…19	2_10
④	0x3a155e78aec3648…99	−3_8
⑤	0x3a155e78aec36a7…5a	−2

(a) SC_w处理的跨链事务信息表



(b) 跨链事务信息组成的Verkle树

Fig. 9 An example of a cross-chain transaction Verkle tree

图 9 跨链事务 Verkle 树示例

InnerNode 且承诺值为 NULL, 则需要递归调用函数先对 *pt* 内以 *n* 为根节点的子树进行承诺. 之后, 构建一个初始包含点 (1, 1) 的点列表 *points*, 将 *n* 的所有子节点信息按规则组合作为多项式的点加入 *points* 中, 然后对 *points* 中的所有点通过拉格朗日插值法得到多项式系数数组 *coeffs*. 最后, 对多项式进行 KZG 承诺得到承诺值 *C*, 将 *coeffs* 和 *C* 保存到 *n* 中, 并将 *n* 更新到 *pt* 中并返回 *pt*. 当需要构建一棵 Verkle 树时, 首先需要构建一棵 Patricia 树, 然后将 *txList* 中的每个跨链事务插入树中, 最后调用 *construct*() 构建 Verkle 树.

4.3 证据的构建和验证

当监听到来自 SC_w 的事件时, 信使节点需要构建证据并将其传递到连接的另一条链的 SC_w 进行验证. 信使节点构建证据的过程和 SC_w 验证证据的过程如算法 2 和算法 3 所示.

算法 2. 证据的构建.

输入: 日志事件 *event*, 向 SC_w 提交过的证据数 *nonce*, 私钥 *priKey*;

输出: 证据 *e*.

① function *chain.getProof* (Hash *blockHash,* Hash *txHash*) /*定义区块链 SDK 中的获取跨链事务证明的函数*/
② *block* ← *getBlock*(*blockHash*);
③ *vt* ← *block.verkleTree*;
④ *node* ← *vt.getNode*(*txHash*);
⑤ *proofList* ← *List*();
⑥ while *node.pNode* ≠ NULL do
⑦ *pNode* ← *node.pNode*;
⑧ if *node.isInstanceOf* (InnerNode)
⑨ π, *C* ← *genProof* (*pNode.Poly*, *node.Prefix*);
⑩ else
⑪ π, *C* ← *genProof* (*pNode.Poly*, *node.Key*);
⑫ end if
⑬ *proofList.append*((π, *C*));
⑭ *node* ← *pNode*;
⑮ end while
⑯ return *proofList*;
⑰ end function
⑱ *e* ← *constructEvidence*();
⑲ *e.id* ← *nonce* + 1;
⑳ *e.bHash* ← *event.blockHash*;
㉑ *block* ← *chain.getBlockByHash*(*e.bHash*);
㉒ *e* ← *setOtherFields*(*e*, *event*, *block*);
㉓ *e.proof* ← *chain.getProof* (*e.bHash*, *e.transactionID*);
㉔ *e.digest* ← *e.hash*();
㉕ *e.mSigList* ← *sign*(*priKey*, *e.digest*).

在算法 2 中, 首先定义了区块链 SDK 中的 *chain.getProof* 函数, 该函数通过接收区块哈希 *blockHash* 和跨链事务 ID *txHash* 作为参数, 获取跨链事务在 Verkle 树中的证明数据 *proofList*. 具体而言, 函数首先通过 *blockHash* 得到 Verkle 树 *vt*, 并根据 *txHash* 获得 *vt* 中对应的叶子节点 *node*. 之后从 *node* 开始遍历直至根节点, 将路径上的节点的父内部节点对自身 Key/Prefix 和 Value 的证明 π 和父内部节点的承诺 C 加入到 *proofList* 中, 最后返回 *proofList*. 当构建证据 *e* 时, 信使节点根据事件 *event* 的信息和向 SC_w 提交过的证据数 *nonce* 对 *e* 的各字段进行填充, 并调用 SDK 获取 *chain.getProof* 函数证明数据, 最后计算 *e* 的摘要并签名.

算法 3. 证据的验证.

输入: 证据 *e*;

输出: 证据验证结果 *result*.

① require *verifySig*(*e.mSigList*, *mPubKey*);
② require *e.bHash* == *hash*(*e.subBHash*, *e.proof* [*e.proof.length* − 1].*C*);
③ require *verifySig*(*e.sigList*, *rPubKey*);
④ *x* ← *e.transactionID*;
⑤ *y* ← *e.transactionID* + *e.type_e.param*;
⑥ for each *p* in *e.proof*
⑦ if !*verify*(*p.C*, *p.proofData*, *x*, *y*)
⑧ *result* ← false;
⑨ return *result*;
⑩ else
⑪ *x* ← *p.Prefix*;
⑫ *y* ← *x* + *p.C*;
⑬ end if
⑭ end for
⑮ *result* ← true;
⑯ return *result*.

在算法 3 中, SC_w 首先对证据 *e* 和区块的签名进行验证, 然后对 *e* 中的 *proof* 字段中的每个证明数据 *p* 进行验证. 第 1 次验证时, 验证的多项式点为 (*e.transactionID*, *e.transactionID* + *e.type_e.param*), 之后每次验证的点为 (*p.Prefix*, *p.Prefix* +*C*). 只有当 *proof* 字段中的每个 *p* 都验证成功时, *e* 才被视为证明成功. SC_w 将根据 *e* 的类型对跨链事务做出相应处理.

5 理论分析与实验

本节首先通过理论分析比较多层级区块链架构在一般多跳跨链模式下和见证链模式下进行资产跨链转移时的跨链事务执行时延. 然后通过实验比较基于 Verkle 树的证据和基于 SPV 的证据的传输开销和验证开销. 实验在 2 台 Intel® Xeon® Gold 6240C CPU @ 2.60 GHz, 256 GB 运行内存的服务器上进行, 2 台服务器各自被拟化为 5 台虚拟机, 每台虚拟机上运行 1 条单节点兼容 EVM 的私有链. 本文修改了区块结构, 使其额外维护一棵由本区块内处理的跨链事务信息组成的 Verkle 树, KZG 承诺选取“永续 Tau(τ)计算”项目①作为可信设置, 使用 BN256[21] 椭圆曲线. 各链上部署 SC_w 原型, 并由链间的信使节点原型负责证据的构建和传递, 其中 SC_w 原型使用 Solidity 语言实现, 信使节点原型使用 Python 语言实现. 通过理论分析与实验验证本文方法在多层级区块链架构下实现资产跨链转移的轻量级和高效验证的特性.

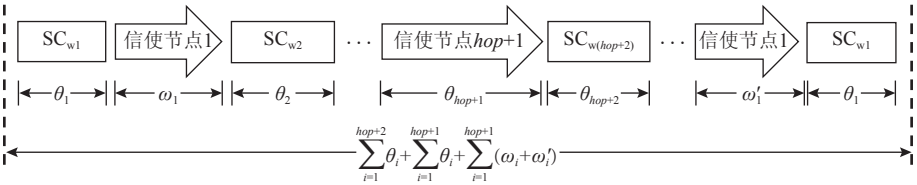
5.1 跨链事务执行时延

在资产跨链转移场景下本文将跨链事务的执行时延定义为从源链资产成功锁定到目标链资产成功铸造后源链资产冻结的间隔时间. 本文假设区块链的序号从源链到目标链沿路径单调递增, 定义一个区块链 i 成功执行 1 次 SC_w 验证证据并处理跨链事务所需的时间为 θ_i. 考虑到由于区块链向不同方向跨链频次的不同, 信使节点向不同方向构造和传递证据的速度可能不同, 因此本文定义连接区块链 i 和区块链 i+1 之间的信使节点 i 以完成证据的构建并定义将证据从区块链 i 传递到区块链 i+1 所需时间为 ω_i, 从区块链 i+1 传递到区块链 i 所需时间为 ω'_i. 跨链事务执行时延在一般多跳跨链模式下和见证链模式下的示意图如图 10 所示.

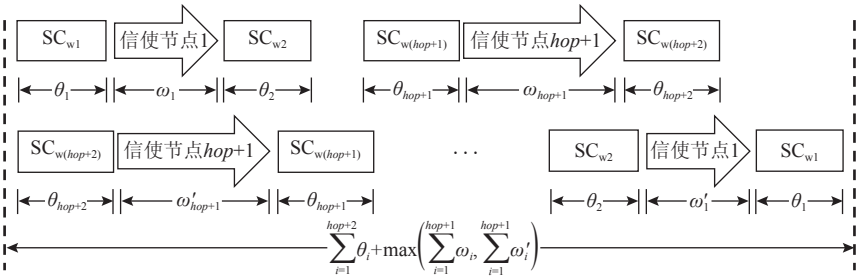
在一般多跳跨链模式下, 跨链事务从源链发起跨链请求开始, 到目标链的跨链回执原路返回到源链, 源链资产冻结时结束. 这需要经过跨链事务信息从源链到目标链再从目标链回到源链的过程. 图 10(a)是一个串行过程, 若跨链事务源链与目标链之间相隔 hop 跳, 则跨链事务从源链执行到目标链必须经过 hop+2 个区块链及 hop+1 个信使节点, 执行时延为 $\sum_{i=1}^{hop+2}\theta_i+\sum_{i=1}^{hop+1}\omega_i$, 再从目标链执行到源链仍必须经过 hop+1 个区块链和信使节点, 执行时延为 $\sum_{i=1}^{hop+1}(\theta_i+\omega'_i)$, 整体的执行时延 T_c 如式(5)所示.

$$T_c=\sum_{i=1}^{hop+2}\theta_i+\sum_{i=1}^{hop+1}\theta_i+\sum_{i=1}^{hop+1}(\omega_i+\omega'_i). \tag{5}$$

在见证链模式下, 跨链事务从源链和目标链同时开始执行. 如图 10(b)所示, 当跨链事务从源链向目标链执行时, 跨链事务首先执行至目标链的见证链, 然后再由目标链的见证链向目标链执行; 当跨链事务从目标链向源链执行时, 跨链事务首先执行至



(a) 一般多跳跨链模式



(b) 见证链模式

Fig. 10 Illustration of execution latency of cross-chain transaction

图 10 跨链事务执行时延示意图

① https://github.com/privacy-scaling-explorations/perpetualpowersoftau

源链的见证链, 然后再由源链的见证链向源链执行. 因此, 跨链事务的整体执行时延 T_w 取决于 2 个方向执行时延的最大值, 如式(6)所示.

$$T_w = \max\left(\sum_{i=1}^{hop+1}\theta_i+\theta_{hop+2}+\sum_{i=1}^{hop+1}\omega_i, \sum_{i=1}^{hop+1}\theta_i+\theta_{hop+2}+\sum_{i=1}^{hop+1}\omega'_i\right)=\sum_{i=1}^{hop+2}\theta_i+\max\left(\sum_{i=1}^{hop+1}\omega_i, \sum_{i=1}^{hop+1}\omega'_i\right). \tag{6}$$

那么, 2 种模式下跨链事务执行的时延差 ΔT 和优化率 η 分别如式(7)和式(8)所示.

$$\Delta T = T_c - T_w = \sum_{i=1}^{hop+1}\theta_i+\min\left(\sum_{i=1}^{hop+1}\omega_i, \sum_{i=1}^{hop+1}\omega'_i\right), \tag{7}$$

$$\eta = \frac{\Delta T}{T_c} = \frac{\sum_{i=1}^{hop+1}\theta_i+\min\left(\sum_{i=1}^{hop+1}\omega_i, \sum_{i=1}^{hop+1}\omega'_i\right)}{\sum_{i=1}^{hop+2}\theta_i+\sum_{i=1}^{hop+1}\theta_i+\sum_{i=1}^{hop+1}(\omega_i+\omega'_i)}. \tag{8}$$

分析可得, 当 $\sum_{i=1}^{hop+1}\omega_i=\sum_{i=1}^{hop+1}\omega'_i$ 时, η 达到最大值, 该最大值接近 50%, 证明本文方法最高可将跨链事务执行时延降低为一般多跳跨链模式下的近 50%; 当 $\sum_{i=1}^{hop+1}\omega_i$ 与 $\sum_{i=1}^{hop+1}\omega'_i$ 差值的绝对值增大时, η 减小但 η 仍大于 0, 且考虑到现实应用场景下 ω_i 和 ω'_i 小于 θ_i, η 仍接近 50%, 这证明了本文方法可以有效降低跨链事务的执行时延.

5.2 证据的传输开销和验证开销

本节通过实验比较本文方法所采用的基于 Verkle 树的验证证据和 BitXHub 等采用的基于 SPV 的证据在传输开销和验证开销方面的表现, 其中, 传输开销采用证据的大小作为指标, 验证开销采用 SC_w 验证证据所使用的 Gas 作为指标. 本文的 Verkle 树多项式承诺验证方案实现了内部节点宽度为 16 和 256 的方案(Verkle 树-16 和 Verkle 树-256); SPV 验证方案除了实现基于 Merkle 树(SPV-Merkle 树)的方案外, 还实现了内部节点宽度为 16 的 MPT 的方案(SPV-MPT-16). 在 SPV 方案中, 每个区块额外维护一棵由跨链事务信息组成的 Merkle 树或 MPT.

5.2.1 2 条链之间的证据的传输开销和验证开销

本节对跨链事务执行时 2 条链之间的证据的传输开销和验证开销进行了实验, 实验测试 4 种方案在 2 条链之间单个区块内处理的跨链事务数增加时, 所有跨链事务的证据平均大小和验证平均使用的 Gas. 实验使用 2 条私有链, 其中一条链负责触发跨链事件供信使节点监听, 另一条链负责对证据进行验证. 实验预先随机生成一定数量的跨链事务, 并在 4 种方案下执行, 每个实验进行 3 次并将实验结果取平均值.

1)证据的传输开销

本节测试了当 1 个区块内需要处理的跨链事务数量为 10~100 和 100~1 000 时, 信使节点所构造的证据的平均大小. 实验结果如图 11 和图 12 所示.

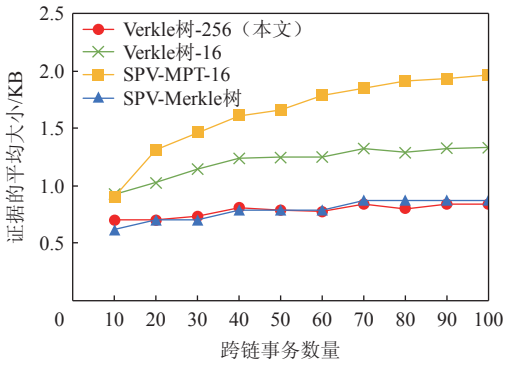


Fig. 11 Average size of evidence

图 11 证据的平均大小

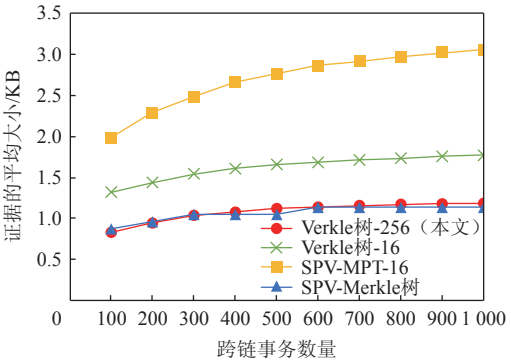


Fig. 12 Average size of evidence

图 12 证据的平均大小

如图 11 和图 12 所示, 当 1 个区块内需要处理的跨链事务数量增加时, 4 种方案的证据平均大小都呈增加态势. 其中, 当跨链事务数量等于 10 时, SPV-MPT-16 方案与 Verkle 树-16 方案的证据平均大小相当, 这是因为跨链事务数量较少, Verkle 树与 MPT 的根节点未满, Verkle 树方案所需的承诺和证明数据大小与 MPT 方案所需的哈希值大小相近. 而随着跨链事务数量继续增多, SPV-MPT-16 方案证据的平均大小增长比其他方案更快, 几乎达到 Verkle 树-16 方案的 2 倍以及 Verkle 树-256 和 SPV-Merkle 树方案的 3 倍, 这是因为随着跨链事务数量的增加以及 MPT 每个内部节点的子节点数量增加, 验证所需要的哈希值数量也增加. 而 Verkle 树-16 方案与 Verkle 树-256 方案随着跨链事务数量增加以及 Verkle 树的层数增加, 证据的平均大小也随之增加, Verkle 树-16 方案

与 Verkle 树-256 方案相比具有更小的宽度，因此树的深度更大，证据的平均大小也更大. Verkle 树-256 方案的证据平均大小与 SPV-Merkle 树方案的相当，且由于 Merkle 树是二叉树，每当跨链事务数量达到 2 的整数幂时，Merkle 树的深度会增加 1，因此证据的平均大小也会增加 1 个哈希值的大小. 从 2 条链间的证据传输开销来看，本文方法所采用的 Verkle 树-256 方案与 SPV-Merkle 树方案相当，且相比于 SPV-MPT-16 方案降低了 22%~61%，这说明本文方法不会比 SPV 方法更容易增加证据的大小，具有轻量级的特点.

2）证据的验证开销

本节测试了当 1 个区块内需要处理的跨链事务数量为 10~100 和 100~1 000 时，SC_w 验证证据平均使用的 Gas. 实验结果如图 13 和图 14 所示.

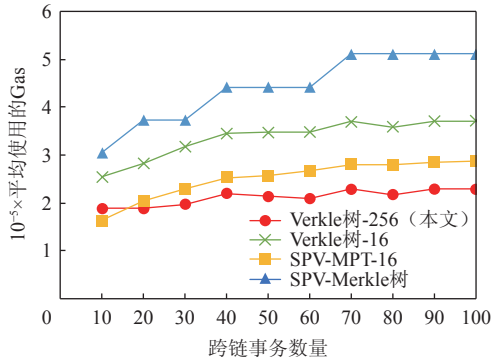


Fig. 13 Average Gas used of evidence validation

图 13 证据验证平均使用的 Gas

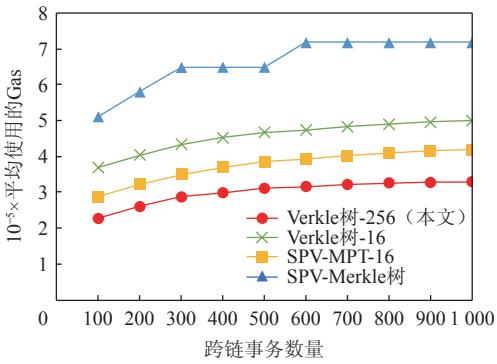


Fig. 14 Average Gas used of evidence validation

图 14 证据验证平均使用的 Gas

如图 13 和图 14 所示，当 1 个区块内需要处理的跨链事务数量增加时，4 种方案的证据验证平均使用的 Gas 呈增加态势. 其中，当跨链事务数量等于 10 时，SPV-MPT-16 方案相比 Verkle 树-256 方案验证证据平均使用的 Gas 更少，但随着跨链事务数量的增加，SPV-MPT-16 方案使用的 Gas 逐渐超过 Verkle 树-256 方案，但比 Verkle 树-16 方案少，而 SPV-Merkle 树方案则平均需要使用更多 Gas，是 Verkle 树-256 方案的 2 倍多，Verkle 树-16 方案的 1.5 倍，且随着 Merkle 树深度的增长，所使用的 Gas 呈阶梯状增长. 而 Verkle 树-16 方案稍差于 SPV-MPT-16 方案，主要是由于进行 KZG 验证比 MPT 树进行 Merkle 验证需要使用更多的 Gas，但由于在证据传输方面具有优势，因此在带宽比计算能力更昂贵的情境下，采用 Verkle 树-16 方案是一种更好的选择[13]. 而 Verkle 树-256 方案仍具有最小的证据验证平均使用 Gas. 从 2 条链间的证据验证所使用的 Gas 来看，本文方法所采用的 Verkle 树-256 方案相较于 BitXHub 等方案采用的 SPV 验证方案具有很大的优势，当 1 个区块内需要处理的跨链事务数量大于 30 时，相比于 SPV-MPT-16 方案降低了 13%~21%，相比于 SPV-Merkle 树方案降低了 47%~56%，证据的验证的计算资源开销更低，具有高效验证的特点.

5.2.2 多层级架构内的证据的传输开销和验证开销

本节对跨链事务执行过程中在多层级架构所有的证据的总传输开销和总验证开销进行了实验，实验测试 4 种方案在相同的源链和目标链间同时（同一区块内）发生 100 个跨链事务的条件下，当跨链事务跳数增加时，所有证据的总大小和验证使用的总 Gas. 实验使用 10 条私有链，将其中 2 条链设置为源链和目标链，逐步增加跨链事务的跳数从 1 到 8，每个实验预先生成 100 个跨链事务，并在 4 种方案下执行，每个实验进行 3 次并将实验结果取平均值.

1）证据的传输开销

本节测试了当 100 个跨链事务跳数从 1 增加到 8 时，所有信使节点所构造的证据的总大小，实验结果如图 15 所示. 在跨链事务数量不变的条件下，当跨链事务的跳数增加时，4 种方案在架构内生成的所有

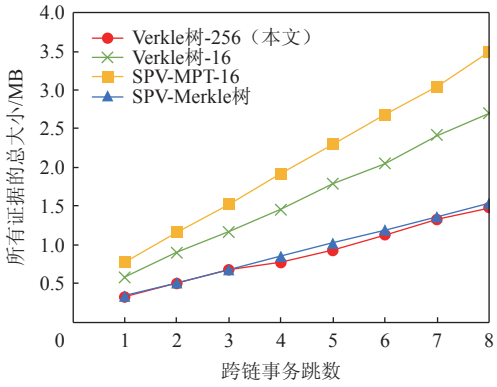


Fig. 15 The total size of all evidence (the number of cross-chain transactions is 100)

图 15 所有证据的总大小 (跨链事务数量为 100)

证据总大小呈近乎线性增加的态势. 其中, Verkle 树-256 方案和 SPV-Merkle 树方案的证据总大小近乎相同; 而随着跨链事务跳数增多, SPV-MPT-16 方案证据的平均大小增长比其他方案更快, 达到了 Verkle 树-256 和 SPV-Merkle 树方案的 2 倍多; Verkle 树-256 方案的证据平均大小与 SPV-Merkle 树方案的相当, 上述实验结果的特点与 5.2.1 节 1)中的一致. 从整个架构内的证据传输开销来看, 本文方法的轻量级特性依然存在, 所采用的 Verkle 树-256 方案与 SPV-Merkle 树方案的传输开销相当, 且相比于 SPV-MPT-16 方案降低了 56%~60%.

2)证据的验证开销

本节测试了当 100 个跨链事务跳数从 1 增加到 8 时, 所有证据验证使用的总 Gas, 实验结果如图 16 所示.

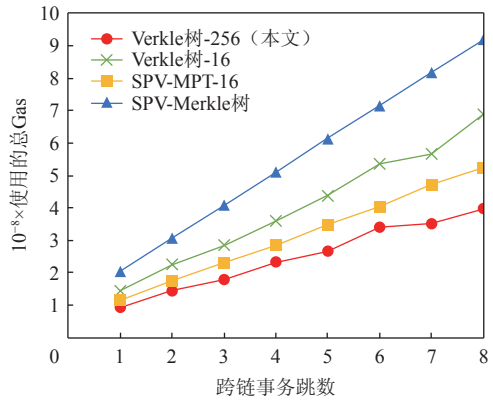


Fig. 16 The total Gas used of all evidence validation (the number of cross-chain transactions is 100)

图 16 所有证据验证使用的总 Gas (跨链事务数量为 100)

在跨链事务数量不变的条件下, 当跨链事务的跳数增加时, 4 种方案在架构内生成的所有证据验证使用的总 Gas 呈近乎线性增加的态势. 与 5.2.1 节 2)的实验结果一致, SPV-Merkle 树方案需要使用的总 Gas 最多, Verkle 树-256 方案需要使用的总 Gas 最少, Verkle 树-16 方案仍然稍差于 SPV-MPT-16 方案. 从整个架构内的证据验证开销来看, 本文方法所采用的 Verkle 树-256 方案比 SPV 验证方案具有很大的优势, 当 1 个区块内需要处理的跨链事务数量大于 30 时, 相比于 SPV-MPT-16 方案降低了 16%~28%, 相比于 SPV-Merkle 树方案总 Gas 降低了 52%~58%, 体现了本文方法高效验证的特性.

综合 5.2.1 节和 5.2.2 节的实验来看, 无论是在 2 条链之间还是在整个多层级架构中, Verkle 树-256 的基于 Verkle 树的证据在跨链事务执行过程中具有与采用 Merkle 树的 SPV 方案的证据相当的传输开销, 比采用 MPT 的 SPV 方案的证据具有更低的传输开销, 且大大降低了验证开销, 相比于采用 SPV 验证的方案具备更大优势, 体现了 Verkle 树-256 在多层级区块链架构下实现资产跨链转移的轻量级和高效验证的特性.

6 结 论

本文提出了一种面向多层级区块链的轻量级高效验证资产跨链转移方法. 方法在多层级区块链架构顶层引入顶层见证链, 并利用多层级区块链的特点, 使源链和目标链的父链作为见证链见证跨链事务的完成, 从而降低跨链事务的执行时延; 通过在各链上部署见证合约实现资产跨链转移的各项功能, 还提出了一种基于 Verkle 树的跨链事务验证证据, 将区块内需要处理的跨链事务信息组织为采用 KZG 承诺的 Verkle 树保存在区块内. 对跨链事务信息进行多项式承诺并将证明数据加入证据中, 通过 KZG 承诺和证明验证跨链事务的执行状态, 从而在不增加证据传输开销的基础上降低证据验证所需的计算资源开销, 具有轻量级和高效验证的特性.

未来的工作包括 2 个方面:

1)优化证据的证明数据生成和验证方式, 将 Verkle 树路径上的多个 KZG 证明合并为 1 个, 从而进一步降低证据的传输开销和验证开销.

2)扩展方法的功能, 实现多层级区块链架构存在的其他跨链需求(如跨链协作等).

作者贡献声明: 牛翔宇提出方法思路和实验方案, 完成实验并撰写论文; 孔兰菊负责方法设计, 提出指导意见并修改论文; 蒋亚丽、钱进负责实验指导; 崔立真、李庆忠提出指导意见并修改论文.

参 考 文 献

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[EB/OL]. [2024-03-18]. https://bitcoin.org/bitcoin.pdf

[2] Taherdoost H, Madanchian M. Blockchain-based new business models: A systematic review[J]. Electronics, 2023, 12(6): 1479

[3] Frizzo-Barker J, Chow-White P A, Adams P R, et al. Blockchain as a disruptive technology for business: A systematic review[J]. International Journal of Information Management, 2020, 51: 102029

[4] Luthra S, Janssen M, Rana N P, et al. Categorizing and relating

implementation challenges for realizing blockchain applications in government[J]. Information Technology & People, 2023, 36(4): 1580−1602

[5] Elisa N, Yang Longzhi, Chao Fei, et al. A framework of blockchain-based secure and privacy-preserving E-government system[J]. Wireless Networks, 2023, 29(3): 1005−1015

[6] Andrew J, Isravel D P, Sagayam K M, et al. Blockchain for healthcare systems: Architecture, security challenges, trends and future directions[J]. Journal of Network and Computer Applications, 2023, 215: 103633

[7] Ghosh P K, Chakraborty A, Hasan M, et al. Blockchain application in healthcare systems: A review[J]. Systems, 2023, 11(1): 38

[8] Amiri M J, Loo B T, Agrawal D, et al. Qanaat: A scalable multi-enterprise permissioned blockchain system with confidentiality guarantees[J]. Proceedings of the VLDB Endowment, 2022, 15(11): 2839−2852

[9] Liu Jiaguo, Zhang Huimin, Zhen Lu. Blockchain technology in maritime supply chains: Applications, architecture and challenges[J]. International Journal of Production Research, 2023, 61(11): 3547−3563

[10] Buterin V. Ethereum 2.0 mauve paper[EB/OL]. [2024-03-18]. https://cdn.hackaday.io/files/10879465447136/Mauve%20Paper%20Vitalik.pdf

[11] Xu Caichao, Wang Xiaoyi, Xia Liwei, et al. BitXHub whitepaper [EB/OL]. [2024-03-18]. https://upload.hyperchain.cn/BitXHub%20Whitepaper.pdf

[12] Kate A, Zaverucha G M, Goldberg I. Constant-size commitments to polynomials and their applications[C] //Proc of the 16th Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2010: 177−194

[13] Kuszmaul J. Verkle trees[EB/OL]. [2024-03-18]. https://klein.mit.edu/research/highschool/primes/materials/2018/Kuszmaul.pdf

[14] Kwon J, Buchman E. Cosmos: A network of distributed ledgers [EB/OL]. [2024-03-18]. https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md

[15] Wood G. Polkadot: Vision for a heterogeneous multi-chain framework[EB/OL].[2024-03-18].https://github.com/polkadot-io/polkadot-white-paper/blob/master/PolkaDotPaper.pdf

[16] Duan Tiantian, Guo Yi, Li Bo, et al. PieBridge: An on-demand scalable cross-chain architecture[J]. Journal of Computer Research and Development, 2023, 60(11): 2520−2533 (in Chinese)
(段田田, 郭仪, 李博, 等. PieBridge: 一种按需可扩展的跨链架构[J]. 计算机研究与发展, 2023, 60(11): 2520−2533)

[17] Zakhary V, Agrawal D, Abbadi A E. Atomic commitment across blockchains[J]. Proceedings of the VLDB Endowment, 2020, 13(9): 1319−1331

[18] Herlihy M. Atomic cross-chain swaps[C] //Proc of the 37th SIGACT-SIGOPS ACM Symp on Principles of Distributed Computing. New York: ACM, 2018: 245−254

[19] Jin Fuqi, Li Wenquan, Kong Lanju, et al. An efficient atomic cross-chain commitment resisting fork fraud[J]. Frontiers of Computer Science, 2023, 17(5): 175614

[20] Buterin V. Verkle trees[EB/OL]. [2024-03-18]. https://vitalik.eth.limo/general/2021/06/18/verkle.html

[21] Barreto P S L M, Naehrig M. Pairing-friendly elliptic curves of prime order[C] //Proc of the 12th Int Workshop on Selected Areas in Cryptography. Berlin: Springer, 2005: 319−331



Niu Xiangyu, born in 2000. Master. Student member of CCF. His main research interest includes cross-chain technologies.

牛翔宇, 2000 年生. 硕士. CCF 学生会员. 主要研究方向为跨链技术.



Kong Lanju, born in 1978. PhD, professor, PhD supervisor. Senior member of CCF. Her main research interests include blockchain and cloud computing.

孔兰菊, 1978 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为区块链和云计算.



Jiang Yali, born in 1976. PhD, lecturer. Member of CCF. Her main research interest includes data privacy protection.

蒋亚丽, 1976 年生. 博士, 讲师. CCF 会员. 主要研究方向为数据隐私保护.



Qian Jin, born in 1980. PhD, engineer. Member of CCF. His main research interests include database optimization and software engineering.

钱　进, 1980 年生. 博士, 工程师. CCF 会员. 主要研究方向为数据库优化、软件工程.



Cui Lizhen, born in 1976. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include software and data engineering and AI.

崔立真, 1976 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为软件与数据工程和人工智能.



Li Qingzhong, born in 1965. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include blockchain and privacy protection.

李庆忠, 1965 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为区块链、隐私保护.