软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(6):2477−2509 [doi: 10.13328/j.cnki.jos.007479] [CSTR: 32375.14.jos.007479]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

XCMP 协议的形式化验证与改进: 提升跨链交互安全性*

吕永阳[1], 冯睿韬[2], 王子墨[1], 刘俊超[1], 吴汉炜[1,3], 李晓红[1]

[1](天津大学 智能与计算学部, 天津 300354)

[2](Faculty of Science and Engineering, Southern Cross University, Gold Coast QLD 4225, Australia)

[3](海南大学 网络空间安全学院, 海南 海口 570228)

通信作者: 李晓红, E-mail: Xiaohongli@tju.edu.cn

摘　要: 随着区块链技术及应用的不断发展, 人们对区块链之间的交互需求日益增加. 然而, 不同区块链系统之间缺乏有效的互操作性, 限制了区块链技术的进一步发展. 为解决区块链异构互联互通问题, 跨链技术应运而生, 并迅速成为新的研究热点. 其中, 跨链消息传递 (cross-chain massage passing, XCMP) 协议作为最流行的跨链通信协议之一, 不仅提供了一个安全高效的跨链通信机制, 还为未来的区块链创新和应用提供了广阔的平台. 然而, XCMP 协议仍然处于不断发展和完善的阶段, 面临着重放攻击、拒绝服务攻击、延迟攻击等安全问题. 对 XCMP 协议进行了形式化验证与改进, 旨在为在其基础上构建更安全、功能更丰富的去中心化应用提供坚实支撑. 首先, 利用一种以经典集合论和一阶谓词逻辑为基础的形式化描述语言——Z 语言, 对 XCMP 协议的 10 条关键安全目标、协议内容进行总结提炼与形式化建模, 并借助支持 Z 语言的自动化验证工具 Z/EVES 验证 XCMP 协议是否满足安全目标. 验证结果表明 XCMP 协议未满足 3 条安全目标. 其次, 通过对验证结果进行全面分析, 针对 XCMP 协议未满足的安全目标, 引入承诺机制、监督机制和轮询机制, 提出了 E-XCMP (enhanced cross-chain message passing) 协议. 最后, 将 E-XCMP 协议形式化建模, 并借助安全协议分析工具 Scyther 和自动化验证工具 Z/EVES 对其安全性和可靠性进行评估, 评估结果表明 E-XCMP 协议不仅满足上述未满足要求的 3 条安全目标, 并且能够有效解决重放攻击、拒绝服务攻击、延迟攻击等安全问题, 具有较好的安全性和可靠性.

关键词: 跨链协议; 跨链消息传递 (XCMP); 形式化分析; Z 语言; Scyther; 模型检测; Pedersen 承诺

中图法分类号: TP311

中文引用格式: 吕永阳, 冯睿韬, 王子墨, 刘俊超, 吴汉炜, 李晓红. XCMP协议的形式化验证与改进: 提升跨链交互安全性. 软件学报, 2026, 37(6): 2477–2509. http://www.jos.org.cn/1000-9825/7479.htm

英文引用格式: Lyu Y, Feng RT, Wang ZM, Liu JC, Wu HW, Li XH. Formal Verification and Improvement of XCMP Protocol: Improving Security of Cross-chain Interaction. Ruan Jian Xue Bao/Journal of Software, 2026, 37(6): 2477–2509 (in Chinese). http://www.jos.org.cn/1000-9825/7479.htm

Formal Verification and Improvement of XCMP Protocol: Improving Security of Cross-chain Interaction

LYU Yong-Yang[1], FENG Rui-Tao[2], WANG Zi-Mo[1], LIU Jun-Chao[1], WU Han-Wei[1,3], LI Xiao-Hong[1]

[1](College of Intelligence and Computing, Tianjin University, Tianjin 300354, China)

[2](Faculty of Science and Engineering, Southern Cross University, Gold Coast QLD 4225, Australia)

[3](School of Cyberspace Security, Hainan University, Haikou 570228, China)

Abstract: With the continuous development of blockchain technology and applications, the demand for interaction between blockchains is increasing. However, the lack of effective interoperability between different blockchain systems limits the further development of blockchain technology. To address the problem of heterogeneous interconnection between blockchains, cross-chain technology has emerged

* 基金项目: 国家重点研发计划 (2021YFF1201102)
收稿时间: 2024-10-23; 修改时间: 2025-01-26; 采用时间: 2025-05-29; jos 在线出版时间: 2025-12-03
CNKI 网络首发时间: 2025-12-04

and quickly become a prominent research topic. Specifically, the XCMP protocol, one of the most popular cross-chain communication protocols, not only provides a secure and efficient communication mechanism but also offers a broad platform for future blockchain innovation and applications. However, the cross-chain massage passing (XCMP) protocol is still in a phase of continuous development and improvement, facing security challenges such as replay attacks, denial of service attacks, and delay attacks. This study formally verifies and improves the XCMP protocol, aiming to provide solid support for the development of more secure and feature-rich decentralized applications based on it. First, Z language, a formal description language based on classical set theory and first-order predicate logic, is used to summarize, refine, and formally model the 10 key security goals and protocol contents of the XCMP protocol. The security goals are then verified using Z/EVES, an automated verification tool supporting the Z language. The verification results show that the XCMP protocol does not meet three of the security goals. Second, after a comprehensive analysis of the verification results, the study introduces a commitment mechanism, a supervision mechanism, and a polling mechanism to address unmet security goals of the XCMP protocol, proposing an enhanced cross-chain message passing (E-XCMP) protocol. Finally, the E-XCMP protocol is formally modeled, and its security and reliability are evaluated using the security protocol analysis tool Scyther and the automatic verification tool Z/EVES. The evaluation results show that the E-XCMP protocol not only meets the three previously unmet security goals but also effectively solves security issues such as replay attacks, denial of service attacks, and delay attacks, demonstrating strong security and reliability.

Key words: cross-chain protocol; cross-chain message passing (XCMP); formal analysis; Z language; Scyther; model checking; Pedersen commitment

区块链 (blockchain, BC) 是一种分布式账本技术[1], 它通过密码学方法[2]将各个“区块”相互连接, 形成一个安全可信的网络. 自 2008 年 10 月 31 日 Nakamoto 发表开创性论文以来[3], 这一技术以其独特的去中心化特性和加密安全性, 迅速吸引了全球的关注. 在区块链技术的快速发展过程中, 以比特币[3]和以太坊[4]为代表的各种不同的区块链系统如雨后春笋般涌现, 它们在设计和应用上各具特色, 满足了多样化的业务需求, 截至 2023 年 12 月, 全球共有区块链企业 10291 家[5]. 然而, 由于这些区块链系统针对不同的目的和应用程序采用了不同的技术框架和设计理念, 它们往往各自独立运作, 形成了一座座“数据孤岛”. 不同区块链之间的数据和资产无法直接互通, 这限制了区块链技术的整体效能和广泛应用. 例如, 以太坊支持智能合约并使用权益证明 (proof of stake, PoS) 机制来达成共识[6], 而比特币主要是被设计为基于工作量证明 (proof of work, PoW)[7]的加密货币. 在异构和隔离的区块链[8]之间缺乏有效的通信方法. 为了打破这些隔阂, 同时为满足区块链技术发展中的关键需求: 扩展性[9]和互操作性[10], 实现不同区块链网络间的互联互通, 跨链技术应运而生. 跨链技术的核心在于允许不同的区块链平台进行通信和数据交换, 甚至实现资产的转移和交易[11]. 它就像是一座桥梁, 将不同的区块链网络连接起来, 构建了一个更加广阔和协同的区块链生态系统.

跨链技术作为连接不同区块链世界的桥梁, 实现了数据和信息交换的功能. 它不仅促进了公共区块链之间的资产的交换, 还推动了多链结构的发展, 为去中心化应用 (DApp) 的互操作性和扩展性提供了新的可能性. 公共区块链之间的资产交换是跨链技术最初的驱动力之一. 然而, 由于这些区块链的异构性和独立性, 直接的资产转移面临着技术和安全上的挑战. 多链结构的发展进一步扩展了跨链技术的视野, 为用户提供了更高的灵活性和吞吐量[12]. 它不仅关注单一的资产交换, 还着眼于构建一个由多个区块链组成的生态系统, 每个链都可以根据其特定的需求和优势进行优化. 例如, Cosmos 通过其 Hub 和 Zone 的架构, 实现了不同区块链之间的互操作性[13]. Polkadot 则通过共享安全性和无需信任的跨链交互, 允许其平行链进行高效的通信和数据交换[13,14]. 在构建多链框架时, 除了资产交换之外, 还需要一种更根本且具有普遍性的手段, 以便在不同的区块链之间实现互通. 这种手段被称为跨链通信协议. Polkadot 网络中的跨链消息传递 (cross-chain message passing, XCMP) 协议作为最流行的跨链通信协议之一, 使得 Polkadot 生态中的不同区块链 (称为平行链) 之间可以直接交换信息和价值[14,15], 目前, Web3 基金会已经完成了 XCMP 的技术原型设计[16], Polkadot 生态中的多个项目 (如 Acala、Moonbeam 等) 已经为 XCMP 的落地做好了准备[17], 具有广泛的应用前景.

与其他跨链通信协议相比, XCMP 协议能够高效、安全地实现跨链通信[18,19], 解决了区块链孤岛问题, 使得多个独立的区块链能够互通并协同工作. 它通过去中心化的结构提升了系统的可靠性, 确保数据和价值能够在不同的平行链之间流动而不依赖中心化的中介[18]. 随着 DeFi、NFT 等多种去中心化应用的崛起, 跨链技术的需求日益增长, XCMP 协议作为 Polkadot 生态的核心协议之一, 将成为支持未来区块链生态繁荣的关键技术之一[20–22]. 此

© 中国科学院软件研究所 http://www.jos.org.cn

外, XCMP 具有极强的可扩展性, 能够支持未来更多不同类型的区块链互联互通[14,15]. 因此, XCMP 在跨链技术领域的影响力不断扩大, 推动了区块链技术的进一步发展.

通过一个实际案例可以说明 XCMP 在跨链场景中的应用. 例如, 在一个去中心化金融平台中, 假设用户 A 需要将资产从平行链 A 转移到平行链 B, 以参与一个高收益的流动性挖掘项目. 那么用户 A 在平行链 A 上发起一笔资产转移请求, 将 100 单位的代币发送到平行链 B. 该请求通过 XCMP 协议发送, 并包含必要的交易信息和用户签名. 然而, 历史上由于协议设计缺陷、实施不当或部署错误, 导致了多起重大安全事件, 造成了巨大的经济损失. 据统计, 价值超过 10 亿美元的资产被窃取或锁定在区块链系统中[20]. 例如, 由于智能合约的漏洞, Poly Network[23]损失了约 6 亿美元, 而 Wormhole[24]也遭受了超过 3 亿美元的盗窃. 这些事件不仅给用户带来了巨大的损失, 也动摇了公众对跨链技术的信任.

对于 Polkadot 平台, XCMP 协议的安全同样至关重要, 因为它确保了跨链消息的传输是可信的、有序的, 并且不会被篡改或丢失. 目前, XCMP 协议处于不断发展和完善的阶段, 面临着重放攻击、拒绝服务攻击、延迟攻击等问题. XCMP 的安全性将直接影响到整个生态系统的稳定性和吸引力. 因此, 完善 XCMP 协议和保证其安全性, 在保障 Polkadot 平台的互操作性、增强用户体验、提升去中心化应用的可组合性等方面具有至关重要的意义, 并能有效推动区块链向创新型应用发展.

形式化方法是借助数学方法建立数学模型并进行模型验证的方法, 可以对模型满足的性质提供严格的保证[25]. 利用形式化方法, 开发人员发现并改进协议漏洞, 进而保证跨链通信协议正确性和安全性. 在利用形式化方法研究和实现 XCMP 协议时, 主要面临以下 3 个挑战.

(1) XCMP 协议的核心在于跨链消息传递, 这需要对多个平行链的状态进行精确协调. 每个平行链有独立的状态和数据结构, 确保消息在不同链之间的一致性与可靠性至关重要. 例如, 当一个平行链发送消息到另一个链时, 必须准确同步其状态, 以防止因时序错误而导致的数据丢失或冲突. 这种状态管理不仅增加了 XCMP 协议的复杂性, 还要求其具备高效的错误处理机制, 以应对实际操作中可能出现的异常情况.

(2) 对 XCMP 协议进行形式化建模是一项复杂的任务, 主要由于其涉及的多层次和多维度特性. 研究者需要考虑多个平行链的独立性及其之间的交互, 而这通常涉及复杂的状态空间和操作模式. 此外, XCMP 协议的动态性和依赖性增加了建模的挑战, 如何准确捕捉协议的行为特征和安全属性, 要求研究者在抽象与具体之间找到合适的平衡. 因此, 构建一个全面、准确的模型以验证 XCMP 协议的性质和功能是一项不小的挑战.

(3) 跨链通信引入了多种安全风险, 例如重放攻击等. 由于多个平行链之间的相互依赖, 任何一链的安全漏洞都有可能影响整个网络的安全性. 因此, 对 XCMP 协议的安全性进行全面分析, 要求研究者深入了解潜在的攻击模式和防御机制. 然而, 现有文献中往往缺乏足够的实例和测试数据, 导致难以验证协议在不同环境下的安全性, 从而增加了研究的难度.

基于这些挑战, 本文使用 Z 语言的形式化方法来检测和提升跨链通信协议 XCMP 的安全性和可靠性. 对于处于不断发展和完善阶段的 XCMP 协议, 首先, 总结提炼 XCMP 协议流程与安全目标, 利用 Z 语言对该协议进行形式化建模, 利用支持 Z 语言的自动化验证工具 Z/EVES 来检测潜在错误并发现 3 个未满足要求的安全目标. 其次, 针对 Z/EVES 所检测出的安全漏洞, 对 XCMP 协议的实体功能进行扩展, 并引入承诺机制、监督机制和轮询机制, 提出了 E-XCMP (enhanced cross-chain message passing) 协议. 最后, 为了评估该协议的可行性和有效性, 利用 Scyther 和 Z/EVES 对其进行形式化建模和安全性分析. 验证结果表明, 我们成功解决了 3 个安全目标存在的问题, 并证明出 E-XCMP 满足多个安全属性. 通过对 XCMP 协议的不断发展和完善, Polkadot 网络的跨链能力将进一步提升, 为构建更加复杂和功能丰富的去中心化应用提供强大支持. 本文的研究工作及贡献主要包含以下方面.

(1) 对 XCMP 的安全目标进行形式化建模. 本文总结提炼并使用 Z 语言形式化解释 XCMP 协议所期望满足的安全目标[26], 同时利用 Z/EVES 工具[27]进行严格的验证. 通过构建 XCMP 协议的数学模型, 本文明确地表达了其在平行链、消息队列等方面的 10 条安全目标, 并使用 Z/EVES 进行逻辑推理和证明, 以确保这些目标在协议的实现中得到满足. 安全目标的形式化解释能够帮助开发人员和用户更好地理解 XCMP 协议.

(2) 对 XCMP 协议进行形式化建模和安全性分析. 基于 Polkadot 平台的架构, 本文利用 Z 语言在 Z/EVES 中对 XCMP 所涉及实体和跨链消息传递过程进行建模, 着重研究分析 XCMP 协议实体的不诚实行为所引发的安全

© 中国科学院软件研究所 http://www.jos.org.cn

问题. 在建模实现的过程中, 本文定义平行链、*DOT* (Polkadot 网络的原生代币)、字符串等基本类型, 并重点设计出平行链状态模式、中继链状态模式和通道状态模式 3 个基本实体模式和 6 个操作模式, 还原 XCMP 协议的跨链消息传递与资产转移等功能, 并支持模型扩展和改进. Z/EVES 模型检测的结果显示, 在 10 条安全目标中存在 3 条安全目标未满足要求, 该协议存在亟待解决的安全漏洞.

(3) 基于未满足的安全目标提出 E-XCMP 协议. 通过分析 XCMP 协议存在的问题, 本文扩展 XCMP 各个实体的功能, 引入承诺机制、监督机制和轮询机制, 提出了 E-XCMP 协议, 该协议的功能实现覆盖以上 10 条安全目标并且具有良好的安全性和可靠性. 其中, 引入承诺 Pedersen 机制, 用以监控同一区块消息的完备性; 引入轮询机制并设计轮询检测智能合约, 以保证接收平行链公平地接收各个发件方的消息; 引入了一个高效的监督机制, 即利用钓鱼者 (Fishermen) 作为消息有效性的智能验证者, 并加入 Pedersen 承诺机制和轮询机制, 使得整个跨链生态系统更加健壮与可信.

(4) 对 E-XCMP 协议进行形式化建模和安全性分析. 基于所提出的 E-XCMP 协议, 本文利用 Scyther 工具和 Z/EVES 工具对其进行安全性分析以保证其正确性与可靠性. 经 Scyther 工具验证, E-XCMP 协议能解决重放攻击、拒绝服务攻击、延迟攻击等问题, 满足存活性、弱协议性、非单调一致性和非单射同步性. 形式化建模验证表明, E-XCMP 协议可解决原协议存在的安全漏洞, 并满足机密性、认证性、完整性和非否认性等安全属性.

本文第 1 节介绍本文用到的一些基础知识, 包括 Polkadot 平台的相关架构知识和 Pedersen 承诺的核心知识, 以及 Z 语言、Z/EVES 工具和 Scyther 工具的基本语法和符号. 第 2 节对 XCMP 协议的系统模型进行概述, 并简述该协议的工作流. 第 3 节基于 XCMP 协议的参与者和实体提出了安全假设, 定义了协议的安全目标, 并使用一阶逻辑对 10 个安全目标进行形式化描述. 第 4 节采用 Z 语言对安全目标和协议模式进行形式化建模与安全性分析, 并基于 Z/EVES 的分析结果检验 XCMP 协议对各安全目标的满足情况, 最后对未满足的 3 个安全目标进行深入分析. 第 5 节主要介绍 E-XCMP 协议的系统模型, 并详细阐述该协议的具体内容. 第 6 节采用 Z/EVES 和 Scyther 工具对 E-XCMP 进行形式化建模和安全性验证, 并分析最终的检测结果, 确保其有效性与可靠性. 第 7 节讨论国内外的相关工作. 第 8 节总结全文.

1 预备知识

1.1 Polkadot 平台及 XCMP 协议

自 2008 年比特币诞生以来, 区块链技术在许多领域都产生了重大影响. 然而, 不同的独立区块链之间缺乏有效的沟通, 限制了区块链产业的促进和生态发展. 在此背景下, 跨链技术迅速发展, 成为一个新的研究热点. Polkadot 就是一种允许独立区块链互相交换信息的系统[14,15], 目标是实现各个链之间资产与数据的互相流通, 它的创建可以追溯到 2016 年, 由以太坊共同创始人之一的 Wood[15]提出, 并由 Web3 基金会推动开发. Polkadot 项目在 2017 年进行了一次成功的 ICO, 并于 2020 年 5 月 26 日正式启动其主网 (relay chain). 在 Polkadot 平台上的跨链满足数据跨链、资产跨链和互操作性的功能, Polkadot 系统由平行链、中继链和转接桥组成, 中继链为主链, 加入进来的区块链称为平行链. Polkadot 平台的基本结构如图 1 所示.

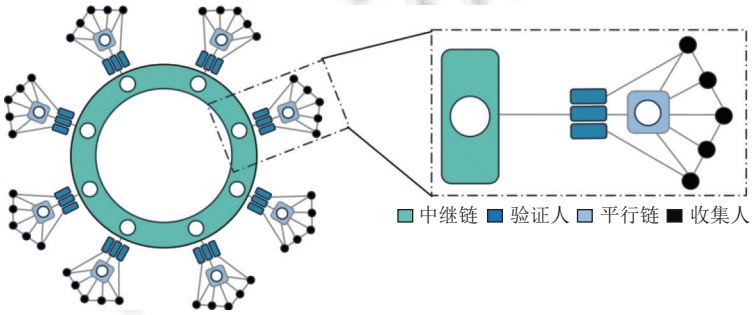

图 1 Polkadot 平台基本结构

© 中国科学院软件研究所 http://www.jos.org.cn

跨链消息传递 (XCMP) 是 Polkadot 协议的一个子集. 它定义了在除了共享中继链的安全之外没有其他的信任假设的情况下, 消息如何得以在平行链之间传递, 具有跨链消息传递、去中心化、可扩展性等优点, 是 Polkadot 跨链系统的核心.

1.2 Pedersen 承诺

Pedersen 承诺[28]是构建在椭圆曲线密码学基础之上, 利用椭圆曲线上的点和群运算来实现. Pedersen 承诺的核心思想是允许一个用户 (承诺者) 对一个数值进行承诺, 而这个承诺在某个未来时间之前是不可打开的, 即外界无法知道承诺的具体内容. 同时, 承诺者也无法在不被发现的情况下改变承诺的内容. Pedersen 承诺的核心公式为:

$$C = r \times G + v \times H,$$

其中, C 是生成的承诺值, G 和 H 是特定椭圆曲线上的生成点, r 是盲因子, 即一个随机选择的数, 用于提供隐藏性. v 是原始信息, 即承诺者想要隐藏的数据.

Pedersen 承诺具有同态性、隐藏性和绑定性这 3 个核心性质. 其中, Pedersen 承诺的同态性使得可以在不解密的情况下对加密数据进行操作. 而隐藏性和绑定性基于离散对数问题的困难性假设. 隐藏性确保承诺值不会泄露任何关于消息 v 的信息, 绑定性则确保承诺者无法在不被发现的情况下更改消息 v. 这些特性使得 Pedersen 承诺在区块链和数字货币中被广泛应用.

1.3 Z 语言

在本文中, 主要使用 Z 语言进行形式化建模. Z 语言[29−33]是一种形式化描述语言, 它以经典集合论和一阶谓词逻辑为基础, 提供了一种称为模式的结构, 以此描述一个规格说明的状态空间和操作. Z 语言采用严格的数学理论, 从而产生简明、精确、无歧义且可证明的规格说明. 该语言的关键思想是把软件开发中的需求规格说明阶段和软件设计阶段分开, 采用忽略过程而强调功能描述的操作抽象, 旨在帮助开发人员和用户找出规格说明的不一致、不完整之处, 更安全地设计和实现软件和协议等技术. Z 语言的核心构造是 Z 模式, 包括状态模式和操作模式两种类型. 状态模式用于定义系统某部分的状态空间及其约束, 而操作模式则用于描述系统某部分的行为特征, 通过对比操作前后的状态值来定义操作的特性.

Z 规格说明是 Z 语言的形式化表达方式, 用于精确描述和建模软件系统的行为与结构. 它由一系列模式 (称为 “scheme”) 组成, 每个模式定义一个抽象对象或操作, 并用谓词判定描述给出新的对象或操作的语义约束. 在 Z 语言的模式中, “?”和“!”分别表示输入和输出. 由于 Z 语言是基于集合论的, 其支持的数据类型均为集合, 如 $\mathbb{Z}$ 表示整数集合, $\mathbb{P}$ 表示其后跟随的集合的幂集, “↔”用于描述集合之间的关系, “→”表示从一个集合映射到另一个集合的函数, seq 表示特殊函数类型序列. Z 语言的模式可以被视为抽象数据集合并通过类似于逻辑运算符的各种运算符进行操作, 从而组合成新的 Z 模式, 新的 Z 模式继承原模式的一切属性和约束. 通过 Include、and 等可以实现不同模式之间的关联和新模式的产生, 如 Include S 表明该模式包含模式 S, 即包含模式 S 的声明和谓词约束, 用于从简单的模式组合出更为复杂的模式; S and T 表明该模式是模式 S 和模式 T 的交集, 即包含模式 S 和模式 T 的声明且同时满足二者的谓词约束.

1.4 Z/EVES 工具

在本文中, 主要使用 Z/EVES 辅助工具编写 Z 语言并进行编写、验证和分析. 目前存在的支持 Z 语言的工具和解析器包括 Z/EVES[27]、ProofPower[34]等, 用于帮助开发人员编写、分析和验证 Z 语言规格. Z/EVES 是一款专为 Z 语言开发设计的辅助工具, 它提供了一个集成开发环境 (IDE), 支持语法高亮, 方便用户编写和修改 Z 语言, 同时内置自动推理机制, 检查规格的自洽性和完整性, 通过模型检查技术, 验证规格是否满足指定的性质, 找出潜在的错误. Z/EVES 具有强大的交互式定理证明功能和广泛的应用场景. 它能够处理复杂的逻辑推导, 并支持用户通过命令手动完成复杂证明. 此外, Z/EVES 的自动证明能力也可以加快验证过程, 对于简单目标的验证非常有效.

Z/EVES 提供了一个结构化界面, 用于验证形式化规范的正确性与完备性. 每个段落被赋予两个状态. 第 1 列显示语法和语义检查结果, “?”表示暂未检查, “Y”表示没有错误, “N”表示存在错误. 第 2 列反映证明状态, “?”表示没有成功检查, “N”表示存在未证明的目标, “Y”表示所有关联目标都已证明.

© 中国科学院软件研究所 http://www.jos.org.cn

1.5 Scyther

本文借助 Scyther[35,36]这种形式化的安全协议分析工具, 来全面评估所提出的方案的安全性. Scyther 工具运用 Athena 自动检查算法[37], 它可以自动验证安全协议的正确性, 检测协议是否满足机密性、完整性、认证性等安全属性, Scyther 采用攻击者模型来模拟敌手的能力, 通过搜索协议的执行轨迹来发现潜在的攻击, 该工具支持多种协议模型, 包括认证协议、密钥交换协议和组成员协议等, 它使用一种通用的协议表示语言, 可以处理各种加密原语和协议构造, Scyther 输出的结果包括协议是否安全、发现的攻击轨迹以及安全隐患, 为协议设计者和分析者提供了一种有效的自动化验证方法, 有助于提高安全协议的可靠性和鲁棒性.

2 协议概况

本节将全面概述 XCMP 协议的系统模型, 并详细描述其工作流程.

2.1 系统模型

在构建系统模型时, 各实体的功能至关重要, 以确保跨链通信的有效性和安全性. 本节根据 XCMP 协议的原理和要求, 总结出协议模型, 模型如图 2 所示, 并由以下实体构成.

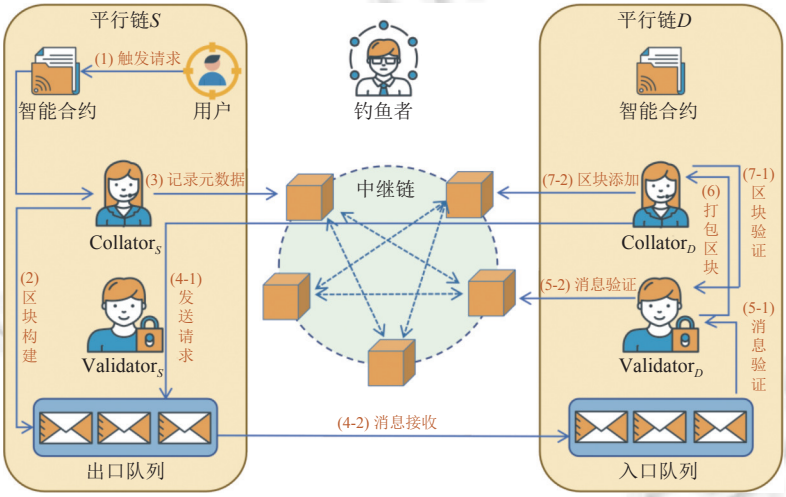

图 2 XCMP 协议模型

(1) 用户 (User): 发起和接收跨链消息. 他们通过在平行链上触发操作, 如智能合约调用或资产转移, 生成跨链数据. 这些数据经过平行链的收集者 (Collator) 处理, 并通过中继链传递到目标平行链, 实现跨链数据交互和操作的完成.

(2) 收集者 (Collator): 生成平行链区块并处理跨链消息. 他们将待发送的跨链消息收集和打包至区块中, 再提交给验证者. 同时, 收集者会将跨链消息的元数据记录在中继链上, 以便验证者根据中继链记录的元数据进行进一步验证和确认.

(3) 验证者 (Validator): 负责确保跨链消息的有效性和顺序. 他们通过中继链提供的元数据来验证消息的真实性, 并确保消息在发送链上已被最终确认. 经过验证的跨链消息会被纳入中继链中, 从而维护系统的数据一致性和安全性.

(4) 钓鱼者 (Fisherman): 负责监控和验证跨链消息的正确性. 他们通过独立的机制检测平行链或验证者在处理消息时是否有不正当行为或错误. 钓鱼者提供有关不符合协议的消息或区块的证据, 从而保障网络的安全性和一致性, 防止恶意操作或错误传播.

© 中国科学院软件研究所 http://www.jos.org.cn

2.2 协议工作流

本节分析现有 XCMP 协议的工作流, 帮助读者清晰地理解其跨链消息发送与接收过程, 步骤可对应图 2.

(1) 触发请求: 平行链 S 中的用户调用智能合约, 触发跨链数据的发送请求.

(2) 区块构建: Collator_S 接收到跨链数据后, 将其打包成跨链消息, 并将该消息放入平行链 S 的出口队列, 待平行链 S 的出口队列中积累了一定数量的跨链消息, Collator_S 会构建新的平行链区块.

(3) 记录元数据: 随后, Collator_S 将消息的元数据 (metadata) 记录在中继链 (relay chain) 上, 包括消息的哈希值、发送信息及相关 Merkle 证明, Merkle 证明是一种验证数据完整性和来源的技术, 通过提供该证明, 接收链的节点可以确认接收到的消息是有效的, 并且确实来自特定发送链的指定区块.

(4) 发送请求和消息接收: Collator_D 节点定期向所有其他收集者节点发送请求, 查询是否有新的跨链消息. 当 Collator_D 节点在请求中发现平行链 S 发送跨链消息时, 它会通过已建立的单向通道将该消息添加到平行链 D 的入口队列.

(5) 消息验证: Validator_D 接收到跨链消息调入队列的消息后, 通过中继链提供的元数据来验证消息的真实性, 确保消息在发送链上已被最终确认.

(6) 打包区块: 验证通过后, Collator_D 将队列中的跨链消息打包成区块, 并执行相应的操作.

(7) 区块验证和区块添加: 执行完毕后, Collator_D 将队列中的跨链消息打包成区块提交给 Validator_D 验证, 验证通过后, 该区块被添加到中继链尾部, 完成跨链消息的传递和处理.

3 安全假设和安全目标

在对 XCMP 协议的分析中, 本文主要关注在跨链数据传递过程中的平行链的收集者和验证者所带来的安全问题, 目标是尽可能多地研究收集者和验证者的行为, 以更全面地分析用户可能面临的安全风险.

3.1 安全假设

本文的分析是基于对 XCMP 协议的参与者和实体的以下安全假设.

(1) 区块链的假设: XCMP 协议为独立区块链提供了一种相互通信的机制, 负责确保链间的可靠性, 不考虑区块链内的拜占庭错误, 如共识机制的失败. 此外, XCMP 协议适用于不同共识机制的异构区块链, 包括概率最终共识算法. 因此, 本文假设所涉及的区块链是安全的.

(2) 诚实和攻击的假设: XCMP 协议中验证者和收集者被认为是半诚实的, 他们遵守协议规则, 但会尝试从执行协议过程中获取额外信息. 他们不会主动去破坏协议的执行, 但会尽可能地利用协议的漏洞来获取他们不应该访问的信息. 而中继链和钓鱼者是诚实的, 他们严格遵守协议的每一步, 不会尝试获取任何额外的信息, 也不会尝试干扰协议的正常执行, 完全按照协议的设计意图行事. 由于平行链共享中继链的安全和钓鱼者的存在, 本文认为跨链消息传递过程是安全的. 因此, 本文重点研究分析 XCMP 协议实体的不诚实行为所引发的安全问题.

3.2 安全目标

XCMP 协议为实现跨链消息传递, 需要满足以下安全目标[14]. 表 1 详细列出了这些安全目标的形式化结果, 而表中各符号的具体含义将在第 4.1 节中进行详细阐述.

目标 1. 平行链的跨链消息按顺序到达另一平行链. 如果消息不是按照发送顺序到达, 可能会导致接收链的状态与发送链的状态不一致. 此外, 在资产转移的场景中, 如果消息乱序到达, 攻击者可能会利用这个漏洞进行双花攻击[38]; 同时, 许多区块链应用都有一定的业务逻辑, 这些逻辑可能依赖于事件的顺序.

目标 2. 平行链按顺序接收传入的跨链消息. 内部平行链可能根据它们自己的逻辑对消息进行延迟或重新排序. 但是, 它们必须按照中继链给出的一致历史记录所确定的顺序接收消息. 平行链总是接收由头位于更早的中继链块中的平行链块发送的消息.

目标 3. 平行链接收另一平行链发送的同一区块中所有的消息, 要么全部成功接收, 要么全部接收失败. 跨链消息传递的原子性要求一系列操作要么全部执行, 要么全部不执行, 不会出现中间状态. 它在一致性维护、错误处

© 中国科学院软件研究所 http://www.jos.org.cn

理和安全性增强等方面具有重要的作用.

表 1 安全目标的形式化

序号	形式化安全目标
目标1	$\forall Mi, Mj \in Validator, \forall S \in Parachains, Mi \in S.egressQueue \wedge Mj \in S.egressQueue \wedge (Mi.sendtime < Mj.sendtime) \Rightarrow Mi.arrivetime < Mj.arrivetime$
目标2	$\forall Mi, Mj \in Validator, \forall D \in Parachains, Mi \in D.ingressQueue \wedge Mj \in D.ingressQueue \wedge (Mi.sendtime < Mj.sendtime) \Rightarrow Mi.arrivetime < Mj.arrivetime$
目标3	$\forall S \in Parachains, \forall B \in Block, \forall Mi, Mj \in Validator, (Mi, Mj \in B.messages \wedge Mi \in S.ingreesQueue \wedge Mi \in S.ingressQueue) \Rightarrow (sucess(Mi) \wedge sucess(Mj)) \vee (fail(Mi) \wedge fail(Mj))$
目标4	$\forall Mi \in Validator, \forall S \in Parachains, (Mi \in S.ingressQueue) \Rightarrow Mi \notin RelayChain.messages$
目标5	$\forall Mi \in Validator, \lvert Mi \rvert \leqslant MaxSize$
目标6	$\forall S, D \in Parachains, (S \neq D) \Rightarrow (mkChannel(S, D) \in openChannels \Rightarrow \exists Mi \in Validator, Mi.source = S \wedge Mi.dest = D)$
目标7	$\forall Mi, Mj \in Validator, D \in Parachains, (Mi.source \neq Mj.source) \Rightarrow (WaitTime(Mi) > WaitTime(Mj) \Rightarrow QueuePosition(D, Mi) < QueuePosition(D, Mj))$
目标8	$\forall Mi \in Validator, S, D \in Parachains, (Mi.source = S \wedge Mi.dest = D \wedge Mi \in D.ingressQueue) \Rightarrow Mi \in S.egressQueue$
目标9	$\forall Mi \in Validator, S, D \in Parachains, (Mi.sounce = S \wedge Mi, dest = D) \Rightarrow ((Mi \in S.egressQueue \wedge Mi \notin D.ingressQueue) \vee (Mi \in D.ingressQueue \Rightarrow Mi \in S.egressQueue))$
目标10	$\forall Mi \in Validator, S \in Parachains, (Mi.source = S) \wedge Mi \in S.egressQueue \Rightarrow \exists t. (Mi.timestamp = t) \wedge (currentTime - t > 24\ hours)$

目标 4. 跨链消息不会传递到中继链. 每个平行链有能力直接与其他平行链通信, 而无需通过一个中心化的中继链. 这表明 XCMP 协议需要实现发送链和接收链之间的状态同步等细节.

目标 5. 跨链消息的大小将受到字节数限制. 限制消息大小有助于防止网络拥塞, 同时保护网络中的节点资源, 避免单个节点因处理过大的消息而耗尽内存或计算资源. 此外, 较小的消息可以更快地在网络中传播, 有助于提高跨链操作的同步效率.

目标 6. 创建通道是启动跨链消息传递队列的前提. 通道具有单向性和唯一性, 它由发送方和接收方的平行链辨识, 这意味着它是一个单向通道. 一对平行链之间最多可以有两个通道, 一个用于发送消息, 另一个用于接收消息.

目标 7. 收件人将在各个发件人之间公平地接收消息. 收件人在处理跨链消息时, 不会偏好任何特定的发件人. 这表明所有发件人的消息都有平等的机会被接收和处理. XCMP 协议通常会采用队列系统来管理跨链消息, 进而实现公平性.

目标 8. 到达的消息是在发送链的最终确定的记录中发送的. 确保消息在发送链的最终确定记录中发送, 可以防止双重花费或其他类型的攻击, 并保证了接收链收到的消息与发送链的最终状态一致, 维护了跨链操作的一致性. 同时, 由于消息是作为最终确定记录的一部分发送的, 因此可以在发送链上进行追溯和验证.

目标 9. 消息的发送和接收的状态是一致的. 消息发送后无法撤回, 不存在一条消息同时存在未发送, 但是已被接收两种状态. 无论消息是否发送系统都可以达到最终状态, 因为任何中继链都提供了一致的历史记录.

目标 10. 任何区块的任何传出消息需要保证可用 1 天左右. 为确保消息能够最终到达目的地并被正确执行, 系统需要保持这些消息可用, 以便在需要时重新发送或验证. 同时, 发送平行链的所有节点都会存储这些传出消息, 直到它们确认这些消息已被中继链或其他目标平行链接收并执行.

4 协议建模及安全分析

为全面、精确地分析 XCMP 协议的安全性, 本节选择 Z 语言进行协议建模. Z 语言的优势在于它是一种数学化的形式化语言, 能够用精确、无歧义的方式描述系统的行为、结构以及约束条件, 特别适用于复杂系统的建模与验证. 采用 Z 语言进行建模, 确保能够全面、严谨地分析 XCMP 协议的功能、数据流及其安全属性. 基于这种形式化分析方法, 本节明确了 XCMP 协议的形式化安全目标, 以确保协议设计和实现过程中符合既定的安全目标.

© 中国科学院软件研究所 http://www.jos.org.cn

接着, 在上述工具和安全目标的指导下, 对 XCMP 协议进行了系统的建模, 从而为后续的安全分析提供了坚实依据.

4.1 协议建模

在本节中, 将首先进行模式定义, 然后基于此进行操作定义, 以系统性地完成对 XCMP 协议的建模.

4.1.1 模式定义

(1) 基本类型

在建模中, 首先定义了 XCMP 协议中涉及的基本类型, 包括 *Parachain* (平行链)、*DOT* (数字货币)、*string* (字符串类型) 以及 *Block* (区块).

$$[Parachain, DOT, string, Block]$$

这些基本类型构成了消息和状态的基础. 在此基础上, 消息类型被进一步定义, 包括以下关键组成部分. 具体见代码 1 和代码 2.

- *S*, *D*: *Parachain* 源平行链 *S* 和目标平行链 *D*, 明确消息的起点和目的地.
- *CCdata*: *string* 跨链数据完整记录和传递跨链消息中的所有必要操作信息.
- *timestamp*: $\mathbb{Z}$ 时间戳记录了消息的生成时间.
- *size*: $\mathbb{Z}$ 消息大小.
- *metadata*: *Metadata* 元数据中包括消息的哈希值 (*messageHash*)、源平行链和目标平行链的标识.

代码 1. *Metadata*.

```
messageHash: string
S, D: Parachain
```

代码 2. *Message*.

```
S: Parachain
D: Parachain
CCdata: string
timestamp: ℤ
size: ℤ
metadata: Metadata
```

$size \leqslant maxMessageSize$

$metadata.messageHash = Hash(S, D, CCdata, timestamp)$

$Metadata.S = S$

$Metadata.D = D$

(2) 平行链状态模式

平行链状态通过 *ParachainState* 模式进行描述, 具体如代码 3, 包括以下两个部分.

- *ingressQueue*: seq *Message* 入口队列用于存储接收到的消息, 等待处理.
- *egressQueue*: seq *Message* 出口队列存储待发送的消息, 准备发送至目标平行链.

代码 3. *ParachainState*.

ingressQueue: seq *Message*

egressQueue: seq *Message*

$\forall\, m{:}Message \mid m \in \mathrm{ran}\ ingressQueue \vee m \in \mathrm{ran}\ egressQueue$

$\cdot\ m.size \leqslant maxMessageSize$

© 中国科学院软件研究所 http://www.jos.org.cn

states 映射定义了平行链到其状态 *ParachainState* 的双射关系, 用于管理每个平行链的当前状态信息.

$$states : Parachain \leftrightarrow ParachainState$$

(3) 中继链状态模式

中继链状态通过 *RelaychainState* 模式进行建模, 描述了中继链中关键的数据结构, 具体见代码 4.

- *metadatas*: seq *Metadata* 记录了所有已收集的消息元数据, 为跨链消息的验证提供依据.
- *validatedBlocks*: seq *Block* 该序列存储了已被中继链验证的区块, 以确保区块链的一致性和安全性.

代码 4. *RelaychainState*.

metadatas: seq *Metadata*
validatedBlocks: seq *Block*

(4) 通道模式

通道管理部分定义了平行链之间消息传递所需的通道结构, 通过 *Channel* 模式 (见代码 5) 进行描述. 通道的关键属性为源平行链和目标平行链, 用以明确通道的起点和终点.

代码 5. *Channel*.

S: *Parachain*
D: *Parachain*

4.1.2 操作模式

(1) 打开通道模式 (*OpenChannel*)

在 *OpenChannel* 操作 (代码 6) 模式中, 处理了平行链间通道的开启. 该操作首先检查发送者和接收者平行链是否不同, 并且都属于已定义的平行链集合 (*Parachains*), 同时通过 *mkChannel* 函数确认通道尚未开启. 满足这些条件后, 操作会将新的通道添加到开放通道集合中, 并在通道存款映射中记录相应的存款金额.

代码 6. *OpenChannel*.

$openChannels, openChannels' : \mathbb{P}\ Channel$
$channelDeposits, channelDeposits' : Channel \leftrightarrow DOT$
$S, D : Parachain$
$deposit : DOT$

$S \neq D \wedge \{S, D\} \subseteq Parachains \wedge \neg mkChannel(S, D) \in openChannels$
$\Rightarrow openChannels' = openChannels \cup \{mkChannel(S, D)\}$
$\quad \wedge\ channelDeposits' = channelDeposits \oplus \{(mkChannel(S, D) \mapsto deposit)\}$

(2) 消息收集模式 (*CollectingMessage*)

在消息收集模式中, 平行链 *S* 中的用户调用智能合约后, 平行链 *S* 的收集者接收到跨链数据 (*ccdata*). 收集者将发送方平行链 *S*、接收方平行链 *D*、时间戳、消息大小及元数据打包生成跨链消息 *M*, 并将该消息放入平行链 *S* 的出口队列. 随后, 收集者将消息的元数据插入中继链的元数据序列 *metadatas* 中, 以便后续验证跨链消息的真实性. 具体见代码 7.

代码 7. *CollectingMessage*.

ccdata?: *string*
S?, *D*?: *Parachain*

© 中国科学院软件研究所 http://www.jos.org.cn

$timestamp?: \mathbb{Z}$
$size?: \mathbb{Z}$
$metadata?: Metadata$
$M: Message$
$states, states': Parachain \leftrightarrow ParachainState$
$\Delta RelaychainState$

$M.S=S?$
$M.D=D?$
$M.CCdata=ccdata?$
$M.timestamp=timestamp?$
$M.size=size?$
$M.metadata=metadata?$
$states'$
$\quad =states$
$\quad\quad \oplus \{(S?$
$\quad\quad\quad \mapsto mkParachainState((states\ S?).ingressQueue,$
$\quad\quad\quad\quad ((states\ S?).egressQueue \wedge \langle M \rangle))\}$
$metadatas'=metadatas \wedge \langle M.metadata\&$

---

(3) 消息接收模式 (*ReceiveMessage*)

在消息接收模式中, 平行链 D 的收集者节点在向所有其他收集节点发送请求时, 发现平行链 S 的出口队列中有以平行链 D 为目的地的跨链消息, 并且消息 M 在 D 的出口队列的队头. 收集者随后检查平行链 D 与平行链 S 之间是否已经建立了单向通道. 如果通道存在, 将消息 M 添加到平行链 D 的入口队列. 具体见代码 8.

代码 8. *ReceiveMessage*.

$states, states': Parachain \leftrightarrow ParachainState$
$S, D: Parachain$
$M?: Message$

$M?.D=D$
$M?=head(states\ S).egressQueue$
$mkChannel(S, D) \in openChannels$
$\Rightarrow states'$
$\quad =states$
$\quad\quad \oplus \{(D$
$\quad\quad\quad \mapsto mkParachainState(((states\ D).ingressQueue \wedge \langle M? \rangle),$
$\quad\quad\quad\quad (states\ D).egressQueue))\}$

---

(4) 消息验证模式 (*ValidateMessage*)

在消息验证模式中 (见代码 9), 平行链 D 的验证者接收到跨链消息进入入口队列的消息后, 需在中继链的元数据序列 *metadatas* 中查找是否存在满足以下条件的元数据.

- 元数据中的发送者与跨链消息 M 的发送链 S 一致.
- 元数据中的接收者与跨链消息 M 的接收链 D 一致.

© 中国科学院软件研究所 http://www.jos.org.cn

● 元数据中的消息哈希值 (*messageHash*) 与根据跨链消息 M 计算出的哈希值一致.

如果存在满足上述条件的元数据, 则验证通过. Collator_D 将队列中的消息打包成区块并执行相应的操作.

代码 9. *ValidateMessage*.

$\Xi RelaychainState$
$M?: Message$
$Block, Block': \text{seq}\ Message$

$\exists metadata: Metadata$
$\cdot\ metadata \in \text{ran}\ metadatas$
$\wedge\ metadata.S = M?.S$
$\wedge\ metadata.D = M?.D$
$\wedge\ metadata.messageHash$
$= Hash(M?.S, M?.D, M?.CCdata, M?.timestamp)$
$\Rightarrow Block' = Block \wedge \langle M? \rangle$

(5) 区块验证模式 (*ValidateBlock*)

在区块验证模式中, 执行完相应的操作后, 平行链 D 的收集者将区块提交给验证者验证, 验证通过后, 该区块被添加到中继链尾部, 完成跨链消息的传递和处理. 具体如代码 10 所示.

代码 10. *ValidateBlock*.

$\Delta RelaychainState$
$Block?: \text{seq}\ Message$

$Blocktest(Block?)$
$\Rightarrow validatedBlocks' = validatedBlocks \wedge \langle Block? \rangle$

4.2 安全分析

在对现有 XCMP 协议进行形式化建模后, 本文定义了一系列形式化安全目标, 以确保跨链消息传递的正确性和可靠性. 这些目标旨在规范协议行为, 确保消息的顺序、完整性、公平性和持久性, 为协议的设计与验证提供明确指导, 并帮助识别潜在问题. 本文依据这 10 个安全目标进行了详尽的验证, 并通过 Z 语言建模检验协议在关键安全目标上的符合性. 表 2 列出了 XCMP 协议在这些目标上的满足情况, √表示满足, ×表示不满足.

表 2 安全目标的形式化结果

序号	安全目标	形式化结果
目标1	平行链的跨链消息按顺序到达另一平行链	√
目标2	平行链按顺序接收传入的跨链消息	√
目标3	平行链接收另一平行链发送的同一区块中所有的消息, 要么全部成功接收, 要么全部接收失败	×
目标4	跨链消息不会传递到中继链	√
目标5	跨链消息的大小将受到字节数限制	√
目标6	创建通道是启动跨链消息传递队列的前提	√
目标7	收件人将在各个发件人之间公平地接收消息	×
目标8	到达的消息是在发送链的最终确定的记录中发送的	√
目标9	消息的发送和接收的状态是一致的	√
目标10	任何区块的任何传出消息需要保证可用1天左右	×

© 中国科学院软件研究所 http://www.jos.org.cn

根据表 2 可知, 现有协议不满足目标 3、目标 7 和目标 10 这 3 条形式化安全目标. 下面展示了针对这 3 个安全目标所编写的定理, 并对其进行了形式化验证. 这些定理的验证结果进一步佐证了表 2 的结论, 即现有 XCMP 协议未能满足这些安全目标.

(1) 目标 3: 完整接收消息

针对目标 3 编写的定理在 Z/EVES 中的验证结果如图 3 所示. 根据验证结果显示, 现有的 XCMP 协议无法满足目标 3. 违反安全目标 3 表明, 由于缺乏完备周到的异常消息检测和处理机制, 而导致接收消息形成平行链块的完整性和安全性无法保证. 以一次消息发送为例, 平行链 S 为发送链, 平行链 D 为接收链, 平行链 S 的收集者收集整理交易数据, 打包成候选区块 M 并放入出口队列. 候选区块中的交易消息 Mi 按顺序从出口队列发送, 经过平行链 S 和平行链 D 的单向通道, 由平行链 D 以中继链给出的一致历史记录所确定的顺序接收消息, 并放入自己的入口队列中, 以待被打包进区块. 若一切顺利, 平行链 D 的收集者将队列中的跨链消息 Mi 打包进区块, 同时每个跨链消息 Mi 会执行平行链 D 上相应的智能合约, 完成跨链数据共享传递. 然而在非理想情况下, 跨链消息 Mi 可能会在发送过程中遭遇恶意篡改、丢失或重放等问题, 导致平行链 D 接收到的消息与平行链 S 发送的消息不一致. 这种情况违反了协议的安全性和完整性要求, 可能导致系统状态不一致, 进而引发跨链操作错误. 例如, 如果平行链 D 未能接收到所有相关消息, 可能会出现资产转移失败、重复转移或丢失等问题, 从而影响系统的可靠性和数据完整性. 确保消息的完整接收是维护系统一致性和完整性的关键, 缺乏有效的完整接收机制将使得重放攻击等威胁得以实施, 造成系统状态混乱, 并可能导致操作错误或重复.

Syntax Proof

Y N **theorem** *CompleteBlockReception*

$\forall S$: *Parachain*; B: *Block*; Mi, Mj: *Message*

| $Mi \in B.Messages$

$\wedge\ Mj \in B.Messages$

$\wedge\ Mi \in \mathrm{ran}\ (states\ S).ingressQueue$

$\wedge\ Mj \in \mathrm{ran}\ (states\ S).ingressQueue$

• $sucess\ Mi = 1 \wedge sucess\ Mj = 1 \vee fail\ Mi = 1 \wedge faili\ Mj = 1$

图 3 目标 3 定理验证结果

(2) 目标 7: 公平消息接收

针对目标 7 编写的定理在 Z/EVES 中的验证结果如图 4 所示. 根据验证结果, 现有的 XCMP 协议无法满足目标 7. 为了保证发送平行链永远不会无限期地等待他们的消息被看到, 接收平行链需要在各个发送平行链之间公平地接收消息. 然而, 现有的轮询并未说明具体的选择机制, 且缺乏严密的检测机制以保证平行链遵循轮询机制, 这可能会导致某一发送平行链的消息长期无法得到响应, 跨链消息将永远堆积在入口队列, 跨链消息传递无法进行. 除此之外, 攻击者可能会实施拒绝服务攻击, 通过发送大量垃圾消息来使有效消息的处理延迟或被阻塞.

Syntax Proof

Y N **theorem** *FairMessageReception*

$\forall Mi, Mj$: *Message*; D: *Parachain*

| $Mi.S \neq Mj.S$

$\wedge\ Mi \in \mathrm{ran}\ (states\ D).ingressQueue$

$\wedge\ Mj \in \mathrm{ran}\ (states\ D).ingressQueue$

$\wedge\ waittime\ Mi > waittime\ Mj$

• $QueuePosition\ (D, Mi) < QueuePosition\ (D, Mj)$

图 4 目标 7 定理验证结果

(3) 目标 10: 消息持久化

针对目标 10 编写的定理在 Z/EVES 中的验证结果如图 5 所示. 验证结果显示, 现有的 XCMP 协议无法满足

© 中国科学院软件研究所 http://www.jos.org.cn

目标 10. 由于需要平行链最终对所有消息都做出处理, 任何一条消息在传递中的丢失都会导致跨链消息无法打包成块, 因此需要在跨链消息传递系统中留出足够的冗余, 即任何验证区块的验证者都应该保持来自该块的任何传出消息可用 24 h (本文将其称为有效期). 然而, 违反安全目标 10 表明开发人员并未考虑对消息是否可用进行可靠检测, 同时未针对超时情况设计合适的处理机制, 这样会导致接收平行链等待时间过长, 降低跨链消息传递效率, 且有构建出不正确区块的可能性, 这不满足协议的认证性和完整性等性质. 这使得攻击者可能实施延迟攻击故意延缓消息的传递, 使接收方无法在有效期内接收到消息, 导致交易超时失效. 此类攻击不仅威胁单个交易的成功, 还可能对整个系统的可靠性和用户信任造成负面影响.

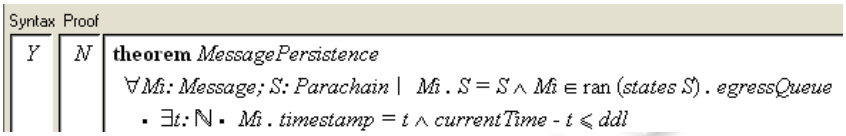

图 5 目标 10 定理验证结果

以上未实现的安全目标暴露了现有 XCMP 协议在消息完整性、公平性和持久性方面的不足. 接下来, 本文将基于 XCMP 协议中的不足, 在 Polkadot 平台的框架结构下提出 E-XCMP 协议.

5 E-XCMP 协议

针对在建模中发现的问题与不足, 本节基于 Polkadot 平台的框架结构, 通过扩展收集者、验证者、钓鱼者等实体的功能, 并加入监督机制、轮询机制和 Pedersen 承诺机制, 提出了 E-XCMP 协议, 以加强 XCMP 协议的安全性. 本节将对 E-XCMP 协议进行详细介绍.

5.1 系统模型

E-XCMP 协议模型如图 6 所示, E-XCMP 协议对各个实体的功能做出了扩展, 实体扩展功能如下.

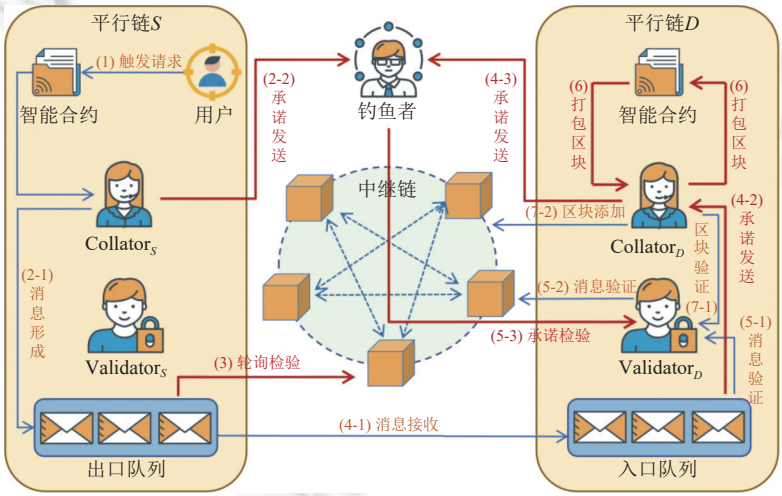

图 6 E-XCMP 协议模型

(1) 验证者: 记录跨链消息进入出口队列时开始等待的时间. 当一条消息从出口队列发送出去后, 记录该消息的消息标识 *IDi*、承诺 *PMi* 等信息. 当消息的总承诺不相等时, 响应钓鱼者的调取信息的请求.

(2) 收集者: 在收到跨链数据后生成承诺 *PMi*. 当消息从出口队列发送后, 将 *PMi* 合并为总承诺 *PMN*, 并将 *PMN* 发送给钓鱼者等待验证. 接收平行链的收集者待入口队列的消息数达到 *N* 时, 将其 *PMi* 合并为总承诺

© 中国科学院软件研究所 http://www.jos.org.cn

PMN′.

(3) 钓鱼者: 接收并记录收集者发送的 *PMN*. 消息超时或不可用时, 对平行链进行判决和处罚. 验证每一个区块发送前后的总承诺 *PMN* 是否相等. 总承诺不相等时, 调取平行链验证者中记录的消息标识 *IDi*、承诺 *PMi* 等信息并进行对比验证, 作出处罚.

5.2 协议工作流

本节介绍 E-XCMP 协议的工作流程, 详细步骤可对应图 6, 其中红色标记的流程为扩展部分.

(1) 触发请求: 平行链 *S* 中的用户调用智能合约, 触发跨链数据发送给平行链 *S* 的 Collator_S.

(2) 消息形成和承诺发送: Collator_S 在收到跨链数据后, 形成以平行链 *D* 为目的地的跨链消息、消息标识和承诺, 放入出口队列, 并计算出总承诺发送给钓鱼者 *C*.

(3) 轮询检验: 平行链 *D* 的 Collator_D 通过轮询机制选择以自己为目的地的消息, 并接收智能合约的检验, 检验通过后, 该消息经由单向通道开始向平行链 *D* 传递.

(4) 消息接收和承诺发送: 平行链 *D* 收到跨链消息后, 将其放入平行链 *D* 的入口队列中, 同时 Collator_D 计算出总承诺发送给钓鱼者 *C*.

(5) 消息验证和承诺检验: 平行链 *D* 的 Validator_D 对接收到的跨链消息进行验证, 随后钓鱼者 *C* 对比发送前后的总承诺进行检验.

(6) 打包区块: 验证通过后, Collator_D 将队列中的跨链消息打包成区块, 并执行智能合约, 完成资产转移.

(7) 区块验证和区块添加: Collator_D 将新生成的区块提交给 Validator_D 验证, 验证通过后, 该区块被添加到中继链尾部.

5.3 安全方案描述

针对每个 XCMP 协议不满足的 3 个安全目标, 本节将重点解释 E-XCMP 协议中的解决方法和相应机制.

(1) 目标 3 完整接收消息: 一个平行链要么接收另一个平行链发送的同一个平行链块中的所有消息, 要么都不接收. 然而, 违反安全目标 3 表明, 由于缺乏完备的异常消息检测和处理机制, 将导致跨链操作的失败或结果不一致, 严重影响系统的整体功能和用户信任.

为解决上述问题, 本文在方案中引用一种 Pedersen 承诺[28]机制. 该机制在收集者整合交易数据形成候选区块 *M* 时, 为每一条跨链消息 *Mi* 生成一个承诺 *PMi*. 当出口队列中每当发出一条消息时, 收集者将同一区块的消息的承诺 *PMi* 进行合并, 待发送的消息达到总数 *N*, 形成总承诺 *PMN*. 随后, 总承诺 *PMN* 被发送给钓鱼者 *C* 等待验证 (该步骤对应第 5.4 节 (2) 消息发送准备). 待到区块 *M* 中的所有跨链消息 *Mi* 到达平行链 *D* 的入口队列, 平行链 *D* 的收集者将跨链消息 *Mi* 的所有承诺 *PMi* 以相同的方式进行合并得到 *PMN′*. *PMN′*发送给钓鱼者 *C* 验证二者是否相等, 若相等则可以认为在消息发送和接收过程中没有发生篡改和丢失等恶意操作, 若不相等则表明区块 *M* 的跨链消息 *Mi* 不再完整或完全, 钓鱼者 *C* 则从平行链 *S* 和平行链 *D* 中检索并对比每条消息的 *IDi* 和 *PMi* 等信息, 并没收失败方部分 *DOT* 代币作为惩罚 (该步骤对应第 5.4 节 (5) 承诺验证).

Pedersen 承诺机制在跨链消息传递中起到了确保消息完整性和防止篡改的作用. 通过为每条跨链消息生成承诺 *PMi*, 并将所有消息的承诺合并形成总承诺 *PMN*, 该机制使得消息发送方与接收方之间的通信可以被验证. 当承诺不匹配时, 说明消息不完整或遭遇篡改, 钓鱼者 *C* 将介入并进行验证, 最终通过没收失败方的部分 *DOT* 代币作为惩罚, 保障了跨链操作的安全性和一致性, 提升了系统的可信度.

(2) 目标 7 公平消息接收: 为防止发送平行链无限期地等待消息响应, 接收平行链需要在各个发送平行链之间公平地接收消息. 然而, 初始协议缺乏严密的检测机制以保证平行链遵循轮询机制, 如果消息接收存在不公平现象, 这将导致攻击者有可能实施拒绝服务攻击, 不仅破坏了系统的公平性和可靠性, 还削弱了用户之间的信任. 为优化跨链消息传递的公平性, 本协议提出一种轮询机制和相应的轮询检测机制.

轮询机制如下, 当存在多个平行链有意向平行链 *D* 发送跨链消息时, 跨链消息被加入各个平行链的出口队列, 同时消息进入队列的确切时间 *ti* 被精确记录, 并作为重要信息嵌入平行链的区块头中 (该步骤对应第 5.4 节 (2) 消

© 中国科学院软件研究所 http://www.jos.org.cn

息发送准备). 此时, 区块头作为记录平行链状态变迁和交易详情的关键部分, 承载跨链消息等待时间等信息. 在平行链 D 进行轮询操作时, 中继链参与维护各平行链区块头的记录. 基于这些记录, 中继链能够识别出当前等待时间最长的平行链 (假设为平行链 S). 随后, 中继链会优先引导平行链 D 对平行链 S 做出响应, 启动跨链消息 Mi 的传递流程 (该步骤对应第 5.4 节 (3) 消息转移发送).

由于无法保证接收平行链 D 是诚实的, 在假设其为半诚实的前提下, 本文提出一种轮询检测机制, 该机制利用智能合约检测平行链 D 是否公平地接收消息. 智能合约通过自动化执行代码来确保系统规则得到遵守, 智能合约算法伪代码如算法 1 所示. 在该轮询检测算法中, 假设消息进入队列等待的时间 tw、消息从发送平行链的出口队列发送的时间 ts 是公开可见的, 智能合约的核心目的是验证平行链 D 是否公平地处理消息. 智能合约依赖于公开可见的消息进入队列时间和从出口队列发送的时间, 来计算消息的等待时间差 $tw1$ 和 $tw2$. 该合约会实时监控并对比这两个等待时间, 若发现平行链 D 未按照轮询规则处理消息, 合约会自动触发相应的惩罚机制, 如没收 DOT 代币 (该步骤对应第 5.4 节 (3) 消息转移发送).

算法 1. 轮询检测算法.

输入: {$Addi$, Ti, N, Xi, $Data$, PMi}, $tw1$, $ts1$, $tw2$, $ts2$; /*tw 为消息进入队列的时间、ts 为消息从平行链 S 的出口队列发送的时间*/

输出: True/False.

```
1. begin
/*读取 tw1, ts1, tw2, ts2*/
2.   Tw1←Subtract(ts1, tw1)
3.   Tw2←Subtract(ts2, tw2)
/*Tw1 和 Tw2 分别为消息 1 和消息 2 在出口队列的等待时长*/
4.   if Tw1≥Tw2 then
5.       return True
/*平行链 D 将消息 2 放入入口队列*/
6.   else if Tw1<Tw2 then
7.       return False
/*平行链 S 未遵循轮询机制, 没收其部分 DOT 代币*/
8.   end
9. end
```

E-XCMP 协议的轮询机制不仅确保了跨链消息传递过程中的公平性, 还通过考虑消息在出口队列中的等待时间, 有效避免了某些平行链因频繁发送消息而长期占据传输资源, 从而保障了所有平行链在跨链通信中的平等机会. 通过将等待时间信息公开记录在区块头上, 也增加了系统的透明度和可信度. 同时, 通过智能合约执行这些规则, 无需依赖中介机构, 从而确保过程的透明性和公平性. 此外智能合约的不可篡改性也确保了算法的执行不受任何一方的干扰, 进一步提高了系统的可信度和安全性.

(3) 目标 10 消息持久化: 消息持久化在确保跨链消息在网络中有效处理上至关重要. 然而, 初始协议未能保证消息的持久化存储, 意味着跨链消息在传递过程中可能面临丢失或重复处理的风险. 恶意攻击者可能利用这一漏洞实施延迟攻击, 这不仅会影响单个交易的成功, 还可能削弱整个系统的可靠性. 缺乏有效的消息持久化机制, 系统将无法确保跨链操作的可靠性和一致性, 从而影响整体的健壮性和安全性.

因此, E-XCMP 协议引入一个高效的监督机制, 即利用钓鱼者 C 作为消息有效性的智能验证者. 具体而言, 钓鱼者 C 对来自平行链 S 的消息进行实时检测, 评估其在当前时间窗口内的有效性. 若钓鱼者 C 通过其算法判定平行链 S 的消息已超出有效期限或存在其他不可用因素, 该机制将自动触发惩罚措施, 从平行链 S 的账户中扣除一

© 中国科学院软件研究所 http://www.jos.org.cn

定量的 *DOT* 代币作为违规成本. 同时, 钓鱼者 *C* 将获得相应数量的代币作为奖励 (该步骤对应第 5.4 节 (4) 消息转移接收和第 5.4 节 (5) 承诺验证). 反之, 若钓鱼者 *C* 验证结果显示平行链 *S* 的消息处于有效状态且可正常使用, 则消息将顺利流转至平行链 *S*, 并在其内部等待合适的时机被打包进新的区块中, 从而完成跨链通信的完整流程 (该步骤对应第 5.4 节 (6) 打包消息成区块). E-XCMP 协议的监督机制不仅保障了跨链消息处理的公平性与高效性, 加强了消息的时效性与可用性, 还通过经济激励的手段促进了验证者的积极参与与自我监督, 使得整个跨链生态系统更加健壮与可信.

此外, 针对超时情况, 本文在 E-XCMP 协议的监督机制中加入超时重传机制[39]. 该机制旨在确保在钓鱼者 C 检测到消息未按时到达或处理失败时, 能够自动触发消息重传请求. 通过对超时事件的监控和重传机制的执行, 可以显著降低消息丢失的风险, 提升系统的消息处理能力 (该步骤对应第 5.4 节 (4) 消息转移接收).

5.4 协议具体内容

基于扩展的机制, 本节将详细阐述 E-XCMP 协议的实现细节. 图 7 以时序图的形式描述了 E-XCMP 协议跨链消息的发送与接收过程.

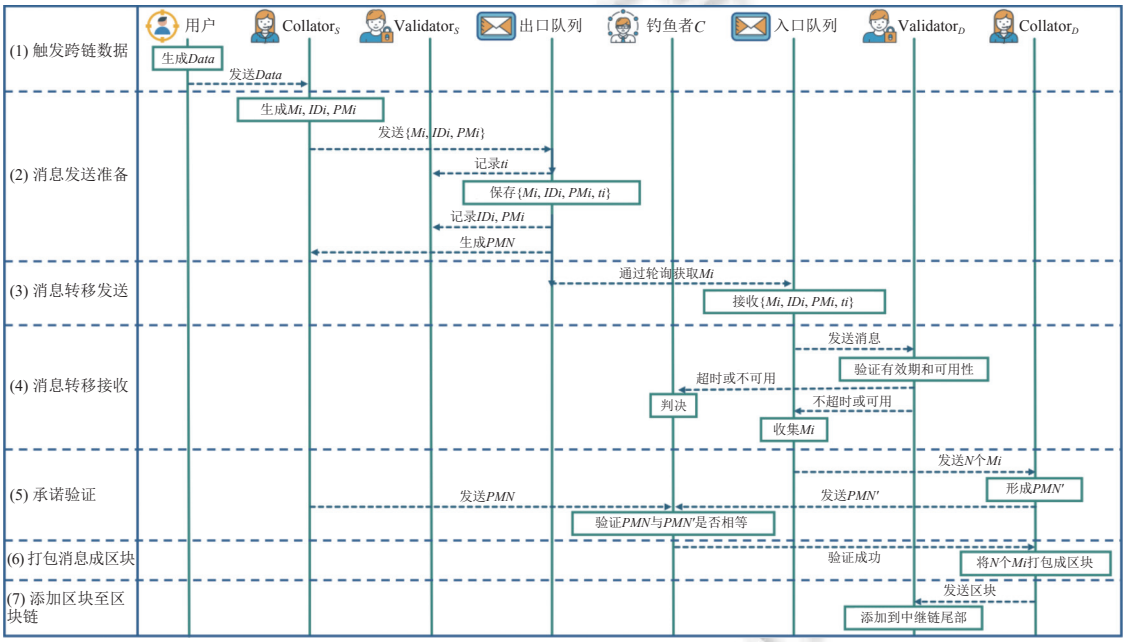

图 7 E-XCMP 协议时序图

(1) 触发跨链数据

平行链 *S* 中的用户调用已部署的智能合约, 触发一条发往平行链 *D* 的跨链数据 *Data*, 将 *Data* 发送给平行链 *S* 的 Collator$_S$.

(2) 消息发送准备

Collator$_S$ 在收到 *Data* 后, 连同目的地址 (*Addi*)、时间戳 (*Ti*)、总消息数 (*N*)、块标识 (*Xi*) 等信息形成消息 $Mi=\{Addi, Ti, N, Xi, Data\}$, 并生成消息标识 *IDi*, 同时使用 Pedersen 承诺[28]机制按照如下公式生成承诺 *PMi*:

$$PMi = Data*G + seedi*H \tag{1}$$

其中, *G* 和 *H* 为有限域椭圆曲线上两个位置不同的固定点; *Mi* 为隐私跨链数据; *表示有限域上离散对数运算的二元关系; *seedi* 为平行链 *D* 产生的随机数种子, $i \in \{1,2,3,\ldots,N\}$.

随后, Collator$_S$ 将 *PMi*、*IDi* 和 *Mi* 放入平行链 *S* 的出口队列中. 此时, 平行链 *S* 的 Validator$_S$ 记录进入队列等

© 中国科学院软件研究所 http://www.jos.org.cn

待的时间 Ti 并写入平行链 S 的区块头 (区块头包含平行链的状态变化和交易信息). 每当从出口队列发送出区块 Xi 内的一条消息后, Validator_S 便记录其 IDi、PMi 等信息, Collator_S 则将 PMi 按照如下的公式进行合并, 待发送的消息到达总数 N 时形成总承诺 PMN. Collator_S 将总承诺 PMN 发送给钓鱼者 C, 等待被验证.

$$PMN = \sum_{i=1}^{N} PMi \tag{2}$$

其中, Pedersen 承诺机制的安全性属性可通过密码学中的归约证明方法证明, 即将攻击者破解 Pedersen 承诺的能力归约到解决离散对数问题的能力. 如果存在一个能够有效破解 Pedersen 承诺的攻击者, 那么同样可以构造出一个解决离散对数问题的算法, 这与离散对数问题的困难性假设相矛盾. 因此, Pedersen 承诺机制是安全的.

(3) 消息转移发送

平行链 D 的 Collator_D 通过轮询方式 (轮询根据消息发送链的出口队列等待时间的长短, 由长到短进行响应) 选择到一条从平行链 S 发来的跨链消息 Mi. 平行链 D 轮询时, 中继链根据对与自己相连的平行链的区块头的记录, 协调各平行链之间的消息传递, 令平行链 D 对目前等待时间 ti 最长的平行链 (假设为平行链 S) 做出响应, 本文使用一种智能合约保证接收平行链在选择发送平行链响应时是否遵循轮询作为检测机制, 智能合约伪代码如算法 1 所示. 跨链消息借由二者之间的单向通道, 开始进行跨链消息传递.

(4) 消息转移接收

当跨链消息 Mi 被平行链 D 收到后, Validator_D 验证 IDi、Xi、Ti、N 等信息, 并根据时间戳检验该消息是否在有效期 (24 h) 内, 若在有效期内, 则检验是否可用, 若可用将其放入自己的入口队列中, 以待被打包进区块. 若已经超时或消息不可用, 则丢弃 Mi, 同时将该情况向钓鱼者 C 报告, C 进行判决, 没收失败方的部分 DOT 代币.

(5) 承诺验证

待入口队列中待被打包的消息 Mi 到达总数 N 时. Collator_D 将所有消息的承诺 PMi 按照公式 (2) 形成 PMN', 并发送给 Validator_D, Validator_D 将其转发给钓鱼者 C 进行验证, 钓鱼者 C 收到消息后则比较 PMN 和 PMN'是否相等. 若不相等则丢弃与跨链消息 Mi 块标识相同的所有消息, 同时钓鱼者 C 调取平行链 S 和平行链 D 中记录的 IDi、PMi 等信息进行比对验证, 并没收失败方的部分 DOT 代币.

(6) 打包消息成区块

如果 PMN 和 PMN'相等, 则 Collator_D 将队列中的跨链 N 个消息 Mi 打包进区块, 同时消息 Mi 会执行平行链 D 上相应的智能合约, 完成资产转移操作.

(7) 添加区块至区块链

Collator_D 将新生成的区块 (包含消息 Mi) 提交给 Validator_D 进行验证, 当确认无误后该区块将添加到中继链的尾部, 跨链操作得到确认.

6 协议分析

本节同时采用 Z/EVES 和 Scyther 工具来验证 E-XCMP 协议的功能正确性和信息安全性. Z 语言用于验证跨链协议的功能正确性, 同时关注协议的规范性和一致性保证. 针对 E-XCMP 协议的扩展功能 (承诺机制、监督机制、轮询机制), Scyther 补充了其在具体实现阶段的信息安全性验证, 它能够通过模拟实际运行中的协议交互, 检查是否存在如身份冒充、信息泄露等实际的安全问题. 通过结合两者, 可以最大程度地保证 E-XCMP 协议的正确性、完整性和安全性, 从理论到实际执行的各个层面都得到充分的验证.

6.1 采用 Z/EVES 验证 E-XCMP 协议

本节使用 Z/EVES 工具用于对 E-XCMP 协议进行严格验证, 以判断其是否满足预设的安全目标. 首先, 基于 Z 语言对协议进行了形式化建模, 将协议的状态和操作以数学方式精确描述. 随后利用 Z/EVES 工具, 构建了 3 个定理, 并通过自动化的定理证明功能, 严格验证了 E-XCMP 协议对 3 条安全目标的符合性, 确保每条安全目标在形式化模型中都得到了充分证明.

© 中国科学院软件研究所 http://www.jos.org.cn

6.1.1 协议建模

在本节中, 首先进行模式定义, 随后基于模式定义进行操作定义, 实现对 E-XCMP 协议的系统性建模.

(1) 模式定义

(a) 消息模式

- *Add*: *Parachain* 消息的接收方.
- *T*: $\mathbb{Z}$ 消息的时间戳.
- *N*: $\mathbb{N}$ 消息所属区块中消息的个数, 用于确保消息在区块内的完整性.
- *X*: *string* 消息所属区块的唯一标识符.
- *m*: *string* 表示跨链数据的具体内容.
- *size*: $\mathbb{Z}$ 消息大小.

约束 $size \leqslant maxMessageSize$ 确保每条消息的大小在预定限制内, 以保证消息传递在资源限制范围内进行. 具体见代码 11.

代码 11. *Message*.

```
Add: Parachain
T: ℤ
N: ℕ
X: string
m: string
size: ℤ
size ≤ maxMessageSize
```

(b) 区块模式

- *Xi*: *string* 表示该区块的唯一标识符, 用于标识每个区块.
- *messages*: seq *Message* 是一个消息序列, 表示该区块中包含的所有消息.

约束条件 $\forall m : Message | m \in \mathrm{ran}\ messages @ m.X = Xi$ 确保区块中所有消息的标识符 $m.X$ 与区块的标识符 Xi 一致. 这意味着每条消息都必须与所属区块匹配, 确保消息的正确关联和完整性. 具体见代码 12.

代码 12. *Block*.

```
Xi: string
message: seq Message
∀ m: Message | m ∈ ran messages · m.X=Xi
```

(c) 平行链状态模式

- *ingressQueue*: seq *Message* 平行链的入口队列.
- *egressQueue*: seq *Message* 平行链的出口队列.
- *dot*: *DOT* 表示平行链的存储资源, 用于管理通道的存款和交易费用.

约束条件 $\forall m : Message | m \in \mathrm{ran}\ ingressQueue \vee m \in \mathrm{ran}\ egressQueue @ m.size \leqslant maxMessageSize$ 确保所有消息, 无论是在入口队列还是出口队列中, 其大小都不超过最大消息大小 *maxMessageSize*. 具体见代码 13.

代码 13. *ParachainState*.

```
ingressQueue: seq Message
```

© 中国科学院软件研究所 http://www.jos.org.cn

egressQueue: seq *Message*
dot: *DOT*
$\forall$ *m*: *Message* | *m* $\in$ ran *ingressQueue* $\vee$ *m* $\in$ ran *egressQueue*
· *m.size*$\leqslant$*maxMessageSize*

(d) 中继链状态模式

● *validatedBlock*: seq *Block* 表示已验证的区块序列. 这个序列记录了中继链上所有经过验证的区块, 确保系统的一致性和完整性.

● *validatedXi*: $\mathbb{P}$ *string* 表示已验证的区块的标识符序列.

● *BlockHeader* 表示区块头的状态, 用于管理消息的时效性.

● *maxwaittime* : *Message* $\rightarrow \mathbb{N}$ 该映射用于判断当前时刻某个消息的等待时间是否为最大. 如果消息的等待时间是当前所有消息中最长的, 那么该消息对应的 *maxwaittime* 值为 1; 否则, 为 0. 此机制帮助中继链在消息轮询过程中优先处理等待时间最长的消息, 从而优化消息传递的时效性. 具体见代码 14.

代码 14. *RelaychainState*.

validatedBlock: seq *Block*
validatedXi: $\mathbb{P}$ *string*
BlockHeader
maxwaittime: *Message*$\rightarrow \mathbb{N}$

在区块头模式 *BlockHeader* 的定义中, *waittime* : *Message* $\rightarrow \mathbb{N}$ 记录了每条消息 *Message* 的等待时间. 这一等待时间用于跟踪和管理消息从生成到被处理的延迟. 具体见代码 15.

代码 15. *BlockHeader*.

waittime: *Message*$\rightarrow \mathbb{N}$

在平行链 *D* 进行轮询时, 中继链利用 *BlockHeader* 中记录的等待时间信息来协调平行链之间的消息传递. 具体来说, 中继链会根据与平行链 *D* 相连的其他平行链的区块头记录, 来确定各平行链的消息等待时间. 中继链会优先处理那些等待时间最长的平行链, 以保证消息在系统中的公平和及时传递. 这种机制有助于防止消息在某些平行链中被无限期地延迟, 从而提高整个系统的消息处理效率和响应速度.

(e) 钓鱼者状态模式

● *fishmanPMN* : *string* $\leftrightarrow \mathbb{N}$ 通过将块标识符 *X* 映射到其对应的 *PMN* (区块内消息的 *PMi* 的总和), 储存每个需要发送的区块的 *PMN* 值. 这种映射使得钓鱼者能够在后续的验证或处理过程中快速检索和使用这些信息. 具体见代码 16.

代码 16. *Fishman*.

fishmanPMN: *string*$\leftrightarrow \mathbb{N}$

(2) 操作定义

(a) 消息收集模式 (*CollectingMessage*)

在 *CollectingMessage* 操作中, 消息收集时的流程如图 8 所示, 代码详见代码 17. 主要步骤如下.

① 使用 *mkMessage* 函数生成跨链消息 *Mi*, 通过 *mkPromise* 函数计算消息的承诺值 *PMi*.

② 将消息 *Mi* 的标识符 *IDi* 和承诺值 *PMi* 更新到 *MessageID* 和 *MessagePM* 映射中, 形成新的 *MessageID'*和 *MessagePM'*. 随后将消息 *Mi* 添加到平行链 *S* 的出口队列中, 然后更新平行链 *S* 的状态, 最终形成新的 *states'*.

© 中国科学院软件研究所 http://www.jos.org.cn

③ 将消息 *Mi* 的等待时间 *ti*?记录到 *waittime* 中.

④ 如果当前区块标识 *Xi*?对应的消息总数 *SBlockcont Xi*?已经达到预定数量 *N*?, 则不更新 *SBlockPMN* 和 *SBlockcont*, 直接将当前的 *SBlockPMN* 记录到 *fishmanPMN* 中. 否则更新 *SBlockPMN*, 将当前 *Xi*?的承诺值 *PMi* 加入其中, 并更新 *SBlockcont*, 增加对应的计数, *fishmanPMN* 保持不变.

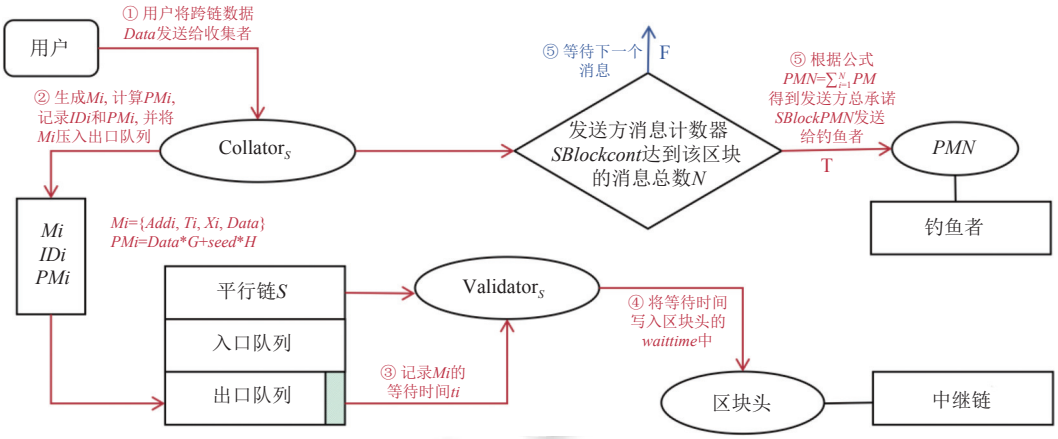

图 8 消息收集流程图

代码 17. *CollectingMessage*.

```
Addi?: Parachain
Ti?, N?, seedi, PMi, ti?: ℕ
Xi?, Data?, IDi: string
Mi: Message
S, S': Parachain
states, state': Parachain↔ParachainState
MessageID, MessageID': Message→string
MessagePM, MessagePM': Messsage→ ℕ
SBlockPMN, SBlockPMN': string→ ℕ
SBlockcont, SBlockcont': string→ ℕ
ΔFishman
ΔRelaychainState
PMi=mkPromise(Data?, seedi)
Mi=mkMessage(Addi?, Ti?, N?, Xi?, Data?)
MessageID'=MessageID ⊕ {(Mi→IDi)}
MessagePM'=MessagePM ⊕ {(Mi→PMI)}
states'
  =states
    ⊕ {(S
        ↦ mkParachainState((states S).ingressQueue,
          ((states S).egressQueue ⌢ ⟨M⟩))))}
waittime Mi=ti?
```

© 中国科学院软件研究所 http://www.jos.org.cn

if *SBlockcont Xi*?=*N*?
then *SBlockPMN*′=*SBlockPMN*
∧ *SBlockcont*′=*SBlockcont*
∧ *fishmanPMN*′=*fishmanPMN* ⊕ {(*Xi*?→*SBlockPMN Xi*?)}
else *SBlockPMN*′=*SBlockPMN* ⊕ {(*Xi*?→*SBlockcont Xi*?+*PMi*)}
∧ *SBlockcont*′=*SBlockcont* ⊕ {(*Xi*?→*SBlockcont Xi*?+*l*)}
∧ *fishmanPMN*′=*fishmanPMN*

(b) 消息轮询模式 (*PollingMessage*)

在 *PollingMessage* 模式中, 消息轮询流程如图 9 所示, 平行链 *D* 处理从平行链 *S* 发来的 *Mi* 时执行以下步骤.

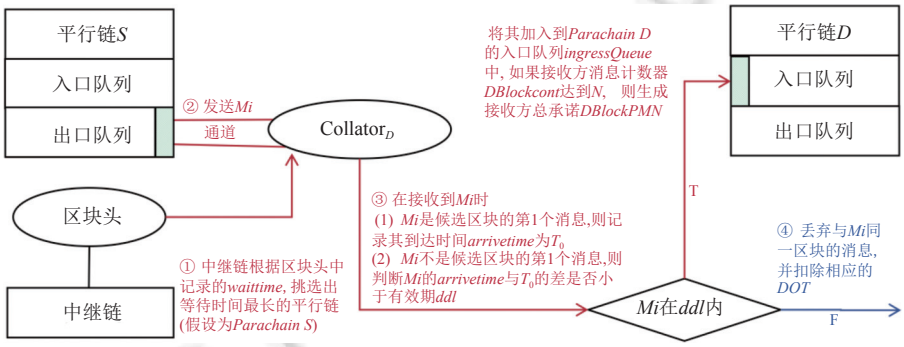

图 9 消息轮询流程图

① 首先, 确认 *Mi* 的目标平行链为平行链 *D*, 并确保平行链 *S* 与平行链 *D* 之间已建立通道. 此外, *Mi* 为所有消息中等待时间最长的消息, 即 *maxwaittime Mi*=1.

② 判断候选区块 *Mi.X* 是否在 *DBlockcont* 的记录中. 如果 *Mi.X* 不在记录中, 即 *Mi.X* ∉ dom *DBlockcont*, 平行链 D 将此时该消息的到达时间 *arrivetime*?及块标识 *Mi.X* 更新到 T_0 映射中. 若 *Mi.X* 已在记录中, 则保持 T_0 不变.

③ Collator$_D$ 接收到候选区块 *Mi.X* 的消息时, 如果消息 *Mi* 在有效期内, 即该消息的到达时间 *arrivetime*?与第 1 个消息到达时间之间的差值小于有效期 *ddl*, 它会将消息 *Mi* 放入平行链 *D* 的入口队列中等待打包. 如果消息超出了有效期, 则丢弃与超时消息相同区块的所有消息 (*DeleteMessage Mi.X*=1), 并扣除相应的 *DOT* 代币 (*deductDot* (*S*, *D*)=1).

④ 在处理消息时, 如果候选区块 *Mi.X* 的消息数量 *DBlockcont* 达到预期的总数 *Mi.N*, 则保持 *DBlockPMN* 和 *DBlockcont* 不变. 否则, 将消息的承诺值 *MessagePM Mi* 添加到 *DBlockPMN* 中, 并更新 *DBlockcont* 记录. 具体见代码 18.

代码 18. *PollingMessage*.

states, *states*′: *Parachain*↔*ParachainState*
S, *D*: *Parachain*
DBlockPMN, *DBlockPMN*′: *string*→ ℕ
DBlockcont, *DBlockcont*′: *string*→ ℕ
Ξ*RelaychainState*
Mi: *Message*

© 中国科学院软件研究所 http://www.jos.org.cn

T_0, T_0': *string*→ ℕ
arrivetime?: ℕ
Mi.Add=*D*
maxwaittime Mi=1
mkChannel(*S*, *D*) ∈ *openChannels*
⇒ (**if** *Mi.X* ∉ dom *DBlockcont*
  then $T_0' = T_0 \oplus \{(Mi.X \mapsto arrivetime?)\}$
  else $T_0' = T_0$)
if *arrivetime*?− T_0' *Mi.X*≤*ddl*
then *states′*
  =*states*
  ⊕ {(*D*
    ↦ *mkParachainState* (((*states D*).*ingressQueue* ∧ ⟨*Mi*⟩),
      (*states D*).*egressQueue*))}
  ∧(**if** *DBlockcont Mi.X*=*Mi.N*
    then (*DBlockPMN′*=*DBlockPMN* ∧ *DBlockcont′*=*DBlockcont*)
    else (*DBlockPMN′*
      = *DBlockPMN* ⊕ {(*Mi.X* ↦ *DBlockPMN Mi.X*+*MessagePM Mi*)}
      ∧ *DBlockcont′*
      =*DBlockcont* ⊕ {(*Mi.X* ↦ *DBlockcont Mi.X*+1)}))
else *DeleteMessage Mi.X*=1 ∧ *deductDot*(*S*, *D*)=1

(c) 承诺验证模式 (*PromiseValidate*)

在 *PromiseValidate* 模式中, 承诺验证的流程如图 10 所示, 详细代码如代码 19 所示. 进行跨链消息承诺的验证时, 主要步骤如下.

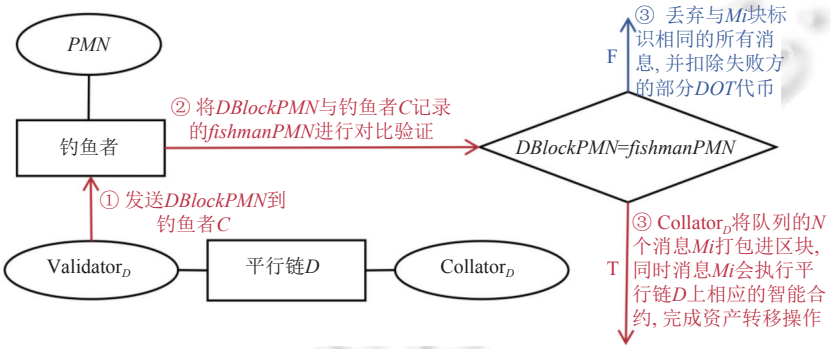

图 10 承诺验证流程图

① 首先检查平行链 *D* 的 *DBlockcont* 中标识为 *Xi*?的区块的消息数量是否等于预期总数 *N*?. 确保接收到的消息数量与预期一致.

② 确认块标识 *Xi*?存在于 *fishmanPMN* 中, 并且 *DBlockPMN* 中对应 *Xi*?的承诺值与 *fishmanPMN* 中存储的值相匹配.

③ 如果上述验证通过, 则根据块标识 *Xi*?生成新块 *newBlock*, 并确认该块通过 *AssetTransfer* 函数验证为有效

© 中国科学院软件研究所 http://www.jos.org.cn

(返回值为 1), 即资产转移操作成功. 如果验证失败, 则删除块标识 *Xi*?相关的消息 (*DeleteMessage Xi*?=1), 并扣除过错方的 *DOT* 代币 (*deductDot*(*S*, *D*)=1).

代码 19. *PromiseValidate*.

$\Xi Fishman$
$Xi?: string$
$N?: \mathbb{N}$
$S, D: Parachain$
$\Delta RelaychainState$
$DBlockPMN: string \rightarrow \mathbb{N}$
$DBlockcont: string \rightarrow \mathbb{N}$
$newBlock: Block$

$DBlockcont\ Xi?=N?$
$Xi? \in \mathrm{dom}\ fishmaxPMN$
if $DBlockPMN\ Xi?=fishmanPMN\ Xi?$
then $newBlock=mkBlockXi? \wedge AssetTransfer\ newBlock = 1$
else $DeleteMessage\ Xi?=1 \wedge deductDot(S, D)=1 \wedge \neg Xi? \in validatedXi$

(d) 区块验证模式 (*ValidateBlock*)

在 *ValidateBlock* 操作中, 区块验证的流程如图 11 所示, 详细代码如代码 20 所示. 主要目标是验证新块 *newBlock* 并更新中继链状态, 主要步骤如下.

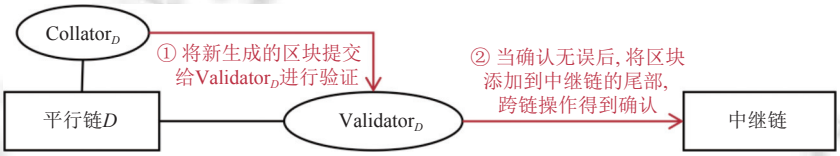

图 11 区块验证流程图

① 首先, 将新生成的区块交给 Validator$_D$, 通过 *validateBlock* 函数对新块 *newBlock*?进行验证. 如果验证成功, *validateBlock newBlock*?的结果为 1.

② 若验证成功, 则将新块 *newBlock*?添加到中继链的已验证区块序列 *validatedBlock* 中. 同时, 将新块的块标识 *Xi* 添加到已验证块标识的集合 *validatedXi* 中, 形成新的 *validatedXi'*.

代码 20. *ValidateBlock*.

$\Delta RelaychainState$
$newBlock?: Block$

$validateBlock\ newBlock?=1$
$\Rightarrow validateBlock'=validatedBlock \wedge \langle newBlock? \rangle$
$\wedge\ validatedXi'=validatedXi \cup \{newBlock?.Xi\}$

6.1.2 形式化验证

在此建模的基础上, 本节使用 Z/EVES 工具对 E-XCMP 协议进行形式化验证. 通过构建 3 个定理, 对应协议的 3 条安全目标, 利用 Z/EVES 的自动化定理证明功能, 严格验证了这些目标的符合性, 确保每条安全目标都在形式化模型中得到充分证明.

© 中国科学院软件研究所 http://www.jos.org.cn

(1) 目标 3: 完整接收消息

CompleteBlockReception 定理在 Z/EVES 中的验证结果如图 12 所示. 定理证明了当平行链 D 的区块标识 $Xi?$ 在钓鱼者 *fishmanPMN* 中存在但与 *DBlockPMN* 不匹配时 ($DBlockPMN\ Xi? \neq fishmanPMN\ Xi?$), 则 $Xi?$ 必未被标记为已验证的区块. 换句话说, 如果钓鱼者记录的承诺值与平行链 D 中记录的承诺值不一致, 则该区块标识 $Xi?$ 没有经过验证, 这确保了只有那些承诺值匹配的区块标识才会被标记为已验证, 从而保持了消息接收的完整性和准确性. 因此提出的 E-XCMP 协议通过其加强的消息验证机制, 能够有效防止重放攻击, 来进一步保障跨链操作的安全性.

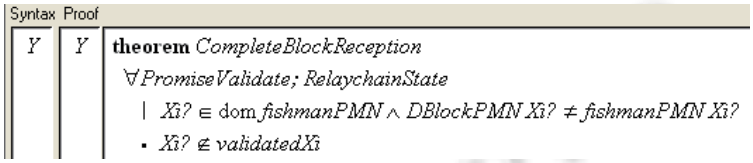

图 12 *CompleteBlockReception* 定理验证结果

(2) 目标 7: 公平消息接收

FairMessageReception 定理在 Z/EVES 中的验证结果如图 13 所示. 定理确保了在平行链 D 接收到消息 Mi 时, 消息的接收是公平的. 如果消息的目的地为 D ($Mi.Add=D$), 且该消息为目前等待时间最长的消息 ($maxwaittime\ Mi = 1$), 且发送链和接收链之间的通道已经打开 ($mkChannel(S,D) \in openChannels$), 并且消息到达时间与记录时间的差值不超过规定的延迟 ($arrivetime? - T_0\ Mi.X \leqslant ddl$), 则消息 Mi 会被正确地添加到平行链 D 的入口队列中. 这保证了消息的及时处理和公平接收, 即在规定的时间内, 所有符合条件的消息都将被平等地接收和处理. 因此, E-XCMP 协议通过其设计有效抵御拒绝服务攻击, 维护系统的整体稳定性和公正性.

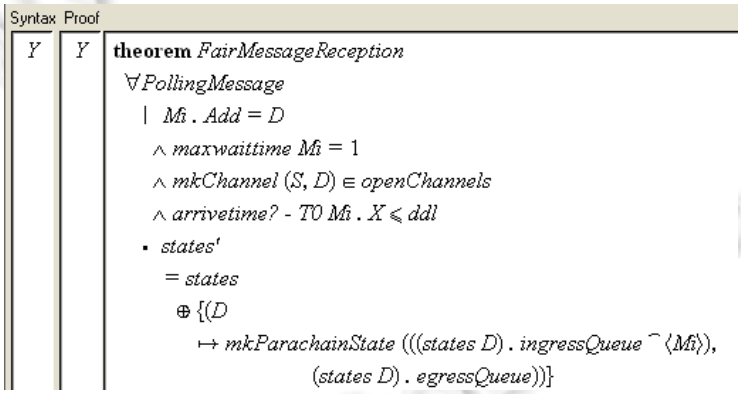

图 13 *FairMessageReception* 定理验证结果

(3) 目标 10: 消息持久化

FishmanTimeout 定理在 Z/EVES 中的验证结果如图 14 所示. 定理处理了消息超时的情况. 如果消息 Mi 的目的地为 D ($Mi.Add = D$), 该消息为目前等待时间最长的消息 ($maxwaittime\ Mi = 1$), 并且发送链和接收链之间的通道已经打开 ($mkChannel(S,D) \in openChannels$), 同时消息的到达时间与记录时间的差值超过了规定的延迟 ($arrivetime? - T_0\ Mi.X > ddl$), 则该消息 $Mi.X$ 将被删除 ($DeleteMessage\ Mi.X$=1), 并且会扣除过错方的 *DOT* 代币 ($deductDot(S,D) = 1$). 这确保了超时消息被正确删除, 并对发送链和接收链实施了惩罚措施, 从而维持了系统的消息持久性和及时性. 因此 E-XCMP 协议通过此设计有效防止延迟攻击, 确保消息在规定时间内被正确处理, 提升系统的整体可靠性.

© 中国科学院软件研究所 http://www.jos.org.cn

```
Syntax Proof
Y   Y   theorem FishmanTimeout
          ∀PollingMessage
            | Mi . Add = D
              ∧ maxwaittime Mi = 1
              ∧ mkChannel (S, D) ∈ openChannels
              ∧ arrivetime? - T0 Mi . X > ddl
            ▪ DeleteMessage Mi . X = 1 ∧ deductDot (S, D) = 1
```

图 14 *FishmanTimeout* 定理验证结果

6.2 基于 Scyther 工具的形式化验证

为进一步验证 E-XCMP 的安全性和可靠性, 证明其能够抵御重放攻击, 拒绝服务攻击和延迟攻击. 本节基于 Scyther 工具对 E-XCMP 进行形式化建模和模型检测.

Scyther 作为目前流行的自动形式验证工具之一, 被安全与隐私领域的研究人员广泛用来对认证方案的可靠性和安全性进行分析、证明和验证[36]. 它的工作原理是在用于设计安全协议的所有密码操作的假设下都是完美的, 常被用于检测从使用加密操作开发协议的方式中可能发生的问题或攻击.

Scyther 工具与安全协议描述语言 (SPDL) 相结合, 用于定义和验证协议设计的安全性. 它能够根据定义的假设识别协议中的潜在安全问题. Scyther 提供了 4 种声明方式: 存活性 (Alive)、弱协议性 (Weakagree)、非单调一致性 (Niagree) 和非单射同步性 (Nisynch), 这些声明与 SPDL 语言结合使用, 为协议安全性分析提供了全面的视角, 确保了验证过程的全面性和准确性. 基于 Scyther 工具进行形式化验证的流程如图 15 所示.

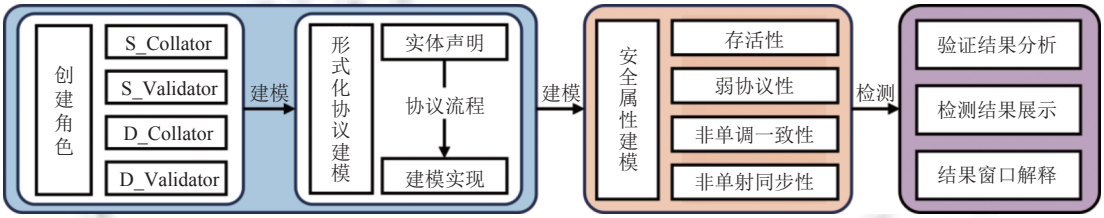

图 15 基于 Scyther 工具进行形式化验证的流程

6.2.1 形式化协议建模

本模型创建了 4 个不同角色: 平行链 S 的收集者 S_Collator, 平行链 S 的验证者 S_Validator, 平行链 D 的收集者 D_Collator, 平行链 D 的验证者 D_Validator.

(1) 实体声明

```
1. usertype text;
2. usertype Timestamp;
3. hashfunction H;
```

其中, hashfunction 为内置哈希函数, usertype 为用户自定义类型.

(2) 建模实现

本文的 E-XCMP 协议关键流程形式化建模源码如后文表 3 所示.

6.2.2 形式化安全属性建模

Scyther 形式化验证工具没有对认证性的直接形式化验证, 需要通过机密性、存活性、弱协议性、非单调一致性以及非单射同步性等验证其保密性, 表 4 对协议形式化验证中关于安全属性建模进行举例分析.

表 3 Scyther 建模实现

协议步骤	建模实现

S_Collator生成*Mi*, *PMi*, *IDi*	fresh *T*1: Timestamp;
	fresh *Data*, *N*: Nonce;
	fresh *Addi*, *Ti*, *Xi*, *PMi*, *IDi*, *Mi*: text;
	match(*PMi*, H(*Data*));
	match(*Mi*, H(*Addi*, *N*, *Xi*, *mi*));

S_Collator向S_Validator发消息	send_1(S_Collator, S_Validator, {S_Collator, S_Validator, *IDi*, *PMi*, *T*1}pk(S_Validator));
S_Validator接收S_Collator的消息	recv_1(S_Collator, S_Validator, {S_Collator, S_Validator, *IDi*, *PMi*, *T*1}pk(S_Validator));
S_Validator进行验证操作	fresh *T*2: Timestamp; var *T*1: Timestamp; var *ACK*1, *IDi*, *PMi*: text; match(*ACK*1, H(*IDi*, *PMi*));
S_Collator向D_Collator发消息	send_2(S_Collator, D_Collator, {S_Collator, D_Collator, *PMi*, *Mi*, *IDi*, *T*1}pk(D_Collator));
D_Collator接收S_Collator的消息	recv_2(S_Collator, D_Collator, {S_Collator, D_Collator, *PMi*, *Mi*, *IDi*, *T*1}pk(D_Collator));
S_Collator打包生成区块	fresh *T*3: Timestamp; var *BLOCK*, *IDi*, *Mi*, *PMi*: text; var *T*1: Timestamp; match(*BLOCK*, H(*PMi*, *Mi*, *IDi*, *T*1));
D_Collator向D_Validator发送区块	send_3(D_Collator, D_Validator, {D_Collator, D_Validator, *BLOCK*, *T*1}pk(D_Validator));
D_Validator接收D_Collator的区块	recv_3(D_Collator, D_Validator, {D_Collator, D_Validator, *BLOCK*, *T*1}pk(D_Validator));
D_Validator进行验证操作	fresh *T*4: Timestamp; var *T*1: Timestamp; var *ACK*2, *BLOCK*: text; match(*ACK*2, H(*BLOCK*, *T*1));

表 4 安全属性建模举例

安全属性	描述定义	建模
机密性	Secret	claim(A, Secret, MAC)
存活性	Aliveness	claim(A, Alive);
弱协议性	Weak agreement	claim(A, Weakness);
非单调一致性	Non-injective agreement	claim(A, Niagree);
非单射同步性	Non-injective synchronisation	claim(A, Nisynch);

表 4 中, 存活性、弱协议性、非单调一致性以及非单射同步性的认证性强度依次增大. Alive 即存活性认证, 是一种基本的认证, 确保了预期的通信方 A 是存在的; Weakagree 即弱协议认证, 要求在协议执行期间表示参与方之间的某些状态或值应保持一致; Niagree 即非单调一致性认证, 用于描述在协议执行期间, 参与方之间的通信或协商结果不能被否认. Nisynch 即非单射同步性认证, 表示在攻击者获取代理 A 的私钥的情况下, 声明事件 (claim event) 之前的所有发送事件或接收事件 (send/recv) 都能被正确的代理 A 以正确的顺序和内容执行, 此性质保证接收者收到的信息都满足完整性, 同样用于描述在协议中参与方之间的通信或协商结果不能被否认, 并且保持一致性. Nisynch 与 Niagree 的定义十分相似, 但不同点在于 Nisynch 增加了对于预期次序的要求, 因此有更强的认证性.

6.2.3 形式化验证结果分析

在 Scyther 工具中, 本文创建 S_Collator、S_Validator、D_Collator 和 D_Validator 这 4 个不同角色, 在其中的任意二者进行信息传输时, 选取时间戳保证了消息的时效性. 用 SPDL 描述本文的协议, 首先 S_Collator 收到用户调用智能合约触发的跨链数据 Data 并形成消息 *Mi*, 同时产生消息标识 *IDi* 和承诺 *PMi*, 将 *PMi*、*Mi*、*IDi* 一起发送给 D_Collator, 并将 *IDi*、*PMi* 发送给 S_Validator. S_Validator 收到后将 *IDi* 和 *PMi* 形成 *ACK*1, 对消息进行验证确认. D_Collator 收到 S_Collator 发送的消息后, 利用 *PMi*、*Mi*、*IDi* 和时间戳打包形成区块 *BLOCK*, 并发送给 D_Validator. D_Validator 收到后对 *BLOCK* 进行验证, 确认无误后形成 *ACK*2. 运行本协议的 SPDL 模型, 并且测

© 中国科学院软件研究所 http://www.jos.org.cn

试 4 个角色的安全声明, 可以说明本文的协议实现了预期的安全属性, 包括消息的机密性, 以及对重放攻击、拒绝服务攻击和延迟攻击等. 验证结果如图 16 所示.

• Scyther results:verify

Claim				Status	Comments
E_XCMP	S_Collator	E_XCMP,S_Collator1	Secert T1	OK	No attacks.
		E_XCMP,S_Collator2	Alive	OK	No attacks.
		E_XCMP,S_Collator3	Weakagree	OK	No attacks.
		E_XCMP,S_Collator4	Niagree	OK	No attacks.
		E_XCMP,S_Collator5	Nisynch	OK	No attacks.
	S_Validator	E_XCMP,S_Validator1	Secert T2	OK	No attacks.
		E_XCMP,S_Validator2	Alive	OK	No attacks.
		E_XCMP,S_Validator3	Weakagree	OK	No attacks.
		E_XCMP,S_Validator4	Niagree	OK	No attacks.
		E_XCMP,S_Validator5	Nisynch	OK	No attacks.
	D_Collator	E_XCMP,D_Collator1	Secert T3	OK	No attacks.
		E_XCMP,D_Collator2	Alive	OK	No attacks.
		E_XCMP,D_Collator3	Weakagree	OK	No attacks.
		E_XCMP,D_Collator4	Niagree	OK	No attacks.
		E_XCMP,D_Collator5	Nisynch	OK	No attacks.
	D_Validator	E_XCMP,D_Validator1	Secert T4	OK	No attacks.
		E_XCMP,D_Validator2	Alive	OK	No attacks.
		E_XCMP,D_Validator3	Weakagree	OK	No attacks.
		E_XCMP,D_Validator4	Niagree	OK	No attacks.
		E_XCMP,D_Validator5	Nisynch	OK	No attacks.

Done.

图 16 Scyther 对 E-XCMP 协议的验证结果

结果窗口解释如下: 协议 SDSQ 的所有角色 S_Collator、S_Validator、D_Collator 和 D_Validator 对于无限次数的运行都是正确的. 在结果窗口中, Scyther 将为每个声明输出一行, 每一行被分为几列. 第 1 列显示各声明所在的协议, 第 2 列显示的是协议的各个角色, 第 3 列显示唯一的声明标识符. 第 3 列的形式为“X, Y”, 其中 X 代表协议, Y 代表声明标识符. 第 4 列显示声明类型和声明参数. 在标题“Status”下存在两列, 其中第 5 列给出了验证过程的实际结果. 第 6 列提炼了前面的陈述. 而第 7 列的“Comments”旨在进一步解释研究结果的状况. 综上所述, 在 Scyther 工具证明下, 可以表明 E-XCMP 协议能够解决重放攻击、拒绝服务攻击、延迟攻击等安全问题, 且具有良好的安全性和可靠性.

7 相关工作

7.1 跨链协议

跨链协议[1,18,19]已成为解决跨链问题的主要方法. 在跨链通信协议上, Xu 等人[40]提出一种新的基于中继网络的跨链数据交互协议, 该协议使用一个可信的执行环境 (TEE) 来执行记录事务状态的服务节点. Lv 等人[41]提出了一种在异构区块链之间共享个人健康记录 (PHR) 的方案, 通过一种增强的代理重加密 (PRE) 算法实现了在区块链之间共享 PHR 的操作. 在跨链资产交易协议上, Pillai 等人[42]提出一种 BURN-TO-CLAIM 协议通过将资产从一个

© 中国科学院软件研究所 http://www.jos.org.cn

区块链系统转移到另一个区块链系统, 使资产从源区块链中烧掉, 并在目标区块链上进行声明来在网络之间无缝交换资产. Ding 等人[43]提出的 Lilac 支持并行资产锁定. 它用所有参与用户生成的多个子凭据替换了 HTLC 中唯一的资产解锁凭据, 并且该子凭据的序列被用作完整的资产解锁凭据. 用户只有在所有资产都被锁定时才能获得完整的凭证, 且凭证构建过程与资产被锁定的顺序无关, 因此当用户并行锁定其资产时, 可以保证原子性. Vishwakarma 等人[44]提出一种资产转移技术 CrossLedger, 他们使用了一种新的资产支持者选择 (AFS)、信任建立 (TE) 和资产转移与确认 (ATC) 算法, 且满足不可否定性、不可链接性、保密性、原子性和互操作性. 这些跨链协议都是通过测试和经验主义来证明其有效性, 缺乏严格、强力的保证. 本文利用形式化方法验证与分析 XCMP 跨链协议, 利用 Z 语言构建协议模型并利用基于严格数学证明的模型检测工具进行安全性分析, 针对发现的问题在 Polkadot 平台的基础上提出扩展方案, 并再次通过形式化建模与分析为扩展方案的正确性和安全性提供严格的保证.

7.2 区块链形式化

采用形式化技术对区块链系统或其内部机制进行精确建模和全面的安全性分析, 已成为确保系统安全性的关键策略[45−50]. 形式化技术用于验证基于区块链提出的方案或协议的安全性与可靠性. 例如, Dong 等人[51]提出了一种基于区块链的工业互联网无证书跨域认证机制, 并利用自动形式化验证工具 Scyther 证明了该方案的安全性. Afzaal 等人[52]提出了一种新的基于信任的区块链众包共识协议, 并利用模型检验技术对其进行形式化验证, 检查属性是否满足或违反要求, 以确保协议的属性符合预定的安全标准. Wang 等人[53]提出一种基于区块链 (BC-based) 的可信传输系统, 并引入了一种结合区块链可信属性的 BC-based DH (Diffie-Hellman) 协议, 接着, 他们利用自动验证形式化分析工具, 验证了传统 DH 协议所缺乏的保密性和完全前向保密性 (PFS). Wang 等人[54]提出了一种用于生产资源形式化分析的模型数据驱动 (DD) 框架, 该方法将形式建模阶段的时间成本降低了至少 75%.

此外, 形式化技术可用于对现有的区块链协议进行安全性分析, 为开发人员发现并改进安全漏洞提供帮助. Feng 等人[55]通过形式化 FIDO UAF 协议的安全假设和目标并在 ProVerif 中进行验证, 发现了两个对于安卓 FIDO 应用的实际攻击. Wei 等人[22]利用时间逻辑规范语言 TLA+形式化 IBC 跨链协议的 3 个主要组件, 并使用模型检测工具 TLC 验证其重要的要求, 发现了其中的问题. Neupane 等人[56]对发布索赔、批准或拒绝索赔的过程中涉及的链码进行形式化验证, 确保区块链中交易的安全可靠处理. Afzaal 等人[57]使用过程分析工具包 (PAT) 对以太坊 2.0 核心的信标链的状态初始化、证明和最终确定检查点的过程进行了形式化验证, 提高人们对信标链的正确性和可靠性的信心. 与上述工作相似, 本文利用 Z 语言提出 XCMP 协议的形式化模型, 通过描述其应满足的安全目标并利用 Z/EVES 辅助工具分析其安全性. 在分析出协议存在的安全漏洞后, 本文针对发现的安全漏洞提出 E-XCMP 协议, 同时, 本文利用 Z 语言和 Scyther 工具对提出的协议进行安全性分析, 验证了 E-XCMP 的正确性.

8 总 结

为提升 XCMP 协议的安全性, 增强 Polkadot 网络的跨链能力, 本文采用 Z 语言的形式化方法, 对跨链通信协议 XCMP 的安全性及可靠性进行验证和改进. 基于正在持续完善的 XCMP 协议, 本文首先梳理了相关实体和协议流程, 并总结提炼出 XCMP 的安全目标. 基于 Polkadot 平台, 本文对 XCMP 协议及其安全目标进行了 Z 语言的形式化建模, 并使用支持 Z 语言的自动化验证工具 Z/EVES 来探测可能存在的错误, 最终识别出 3 个未满足的安全目标. 接着, 针对 Z/EVES 发现的这些安全缺陷, 本文对 XCMP 协议的实体功能进行了扩展, 并引入了承诺机制、监督机制和轮询机制, 提出了 E-XCMP 协议. 为了验证 E-XCMP 协议的实用性和有效性, 本文使用 Scyther 和 Z/EVES 对其进行了形式化建模和模型检测, 并进行安全分析. 结果表明, 本文提出的 E-XCMP 协议解决了 3 个未满足的安全目标, 添加了轮询检测、智能验证、超时重传等功能, 并证明了 E-XCMP 协议满足机密性、认证性、完整性和非否认性等多个安全属性, 能有效解决重放攻击、拒绝服务攻击、延迟攻击等问题. 通过本文提出的 E-XCMP 协议, Polkadot 网络的跨链功能将得到增强, 从而为构建更复杂、功能更丰富的去中心化应用提供了坚实的支撑. 本文基于 Z 语言的分析也证明了形式化方法在像跨链协议等复杂场景中的有效性. 在分析和建模过程中, 本文所

© 中国科学院软件研究所 http://www.jos.org.cn

做的引入承诺机制、轮询机制等安全增强措施, 可以为未来的研究提供参考. 接下来, 我们计划将研究范围扩展到 XCMP 协议的应用层面, 以进一步推动 Polkadot 网络的发展. 此外, 我们还将探索分析如 IBC 等其他跨链协议, 旨在以本文的研究成果为基础帮助优化跨链协议设计.

References

[1] Si BR, Xiao J, Liu CY, Dai XH, Jin H. Survey on blockchain network. Ruan Jian Xue Bao/Journal of Software, 2024, 35(2): 773–799 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6985.htm [doi: 10.13328/j.cnki.jos.006985]

[2] Zheng ZB, Xie SA, Dai HN, Chen XP, Wang HM. Blockchain challenges and opportunities: A survey. Int'l Journal of Web and Grid Services, 2018, 14(4): 352–375. [doi: 10.1504/IJWGS.2018.095647]

[3] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. Satoshi Nakamoto Institute. 2008. https://bitcoin.org/en/bitcoin-paper

[4] Wood G. Ethereum: A secure decentralised generalised transaction ledger. 2014. https://the-blockchain.com/docs/Dr.%20Gavin%20Wood%20-%20Ethereum%20-%20A%20Secure%20Decentralised%20Generalised%20Transaction%20Ledger.pdf

[5] China Academy of Information and Communications Technology. Blockchain White Paper (2023). Beijing: China Academy of Information and Communications Technology, 2023 (in Chinese).

[6] Nguyen CT, Hoang DT, Nguyen DN, Niyato D, Nguyen HT, Dutkiewicz E. Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities. IEEE Access, 2019, 7: 85727–85745. [doi: 10.1109/ACCESS.2019.2925010]

[7] Gervais A, Karame GO, Wüst K, Glykantzis V, Ritzdorf H, Capkun S. On the security and performance of proof of work blockchains. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 3–16. [doi: 10.1145/2976749.2978341]

[8] Ou W, Huang SY, Zheng JJ, Zhang QL, Zeng G, Han WB. An overview on cross-chain: Mechanism, platforms, challenges and advances. Computer Networks, 2022, 218: 109378. [doi: 10.1016/j.comnet.2022.109378]

[9] Wang F, Zhang Q, Liu Y, Liu LL, Lu Y. Research progress of blockchain from perspective of scalability. Application Research of Computers, 2023, 40(10): 2896–2907 (in Chinese with English abstract). [doi: 10.19734/j.issn.1001-3695.2023.02.0075]

[10] Duan TT, Zhang HW, Li B, Song ZX, Li ZC, Zhang J, Sun Y. Survey on blockchain interoperability. Ruan Jian Xue Bao/Journal of Software, 2024, 35(2): 800–827 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6950.htm [doi: 10.13328/j.cnki.jos.006950]

[11] Zhu H, Wu S. Overview of blockchain cross-chain technology and its security. Application Research of Computers, 2024, 41(12): 3543–3552 (in Chinese with English abstract). [doi: 10.19734/j.issn.1001-3695.2024.04.0116]

[12] Liu ZY, Li ZP. A blockchain-based framework of cross-border e-commerce supply chain. Int'l Journal of Information Management, 2020, 52: 102059. [doi: 10.1016/j.ijinfomgt.2019.102059]

[13] Goes C. The interblockchain communication protocol: An overview. arXiv:2006.15918, 2020.

[14] Burdges J, Cevallos A, Czaban P, Habermeier R, Hosseini S, Lama F, Alper HK, Luo XM, Shirazi F, Stewart A, Wood G. Overview of Polkadot and its design considerations. arXiv:2005.13456, 2020.

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

[16] PolkaWorld. The design and implementation of XCMP to further enhance Polkadot's cross-chain capabilities? 2024 (in Chinese). https://m.jinse.cn/blockchain/3681944.html.

[17] PolkaWorld. A review of XCM integration in the Moonbeam network. 2023 (in Chinese). https://foresightnews.pro/article/detail/24540?utm_source=chatgpt.com.

[18] Chen WL, Zhen ZB. Blockchain data analysis: A review of status, trends and challenges. Journal of Computer Research and Development, 2018, 55(9): 1853–1870 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2018.20180127]

[19] Meng B, Wang YB, Zhao C, Wang DJ, Ma BH. Survey on cross-chain protocols of blockchain. Journal of Frontiers of Computer Science & Technology, 2022, 16(10): 2177–2192 (in Chinese with English abstract). [doi: 10.3778/j.issn.1673-9418.2203032]

[20] Robinson P. Survey of crosschain communications protocols. Computer Networks, 2021, 200: 108488. [doi: 10.1016/j.comnet.2021.108488]

[21] Ren KP, Ho NM, Loghin D, Nguyen TT, Ooi BC, Ta QT, Zhu FD. Interoperability in blockchain: A survey. IEEE Trans. on Knowledge and Data Engineering, 2023, 35(12): 12750–12769. [doi: 10.1109/TKDE.2023.3275220]

[22] Wei QY, Zhao XF, Zhu XY, Zhang WH. Formal analysis of IBC protocol. In: Proc. of the 31st IEEE Int'l Conf. on Network Protocols (ICNP). Reykjavik: IEEE, 2023. 1–11. [doi: 10.1109/ICNP59255.2023.10355573]

[23] Poly Network. Honour, exploit, and code: How we lost 610M dollar and got it back. 2021. https://medium.com/poly-network/honour-

© 中国科学院软件研究所 http://www.jos.org.cn

exploit-and-code-how-we-lost-610m-dollar-and-got-it-back-c4a7d0606267

[24] Wormhole. Wormhole incident report. 2022. https://wormholecrypto.medium.com/wormhole-incident-report-02-02-22-ad9b8f21eec6

[25] Miao XL, Chang R, Pna SP, Zhao YW, Jiang LH. Modeling and security analysis of access control in trusted execution environment. Ruan Jian Xue Bao/Journal of Software, 2023, 34(8): 3637–3658 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6612.htm [doi: 10.13328/j.cnki.jos.006612]

[26] Zou SR, Zheng GL. Comparison of Z and VDM with B. Computer Science, 2002, 29(10): 136–138 (in Chinese with English abstract). [doi: 10.3969/j.issn.1002-137X.2002.10.041]

[27] Saaltink M. The Z/EVES system. In: Proc. of the 10th Int'l Conf. of Z Users. Reading: Springer, 1997. 72–85. [doi: 10.1007/BFb0027284]

[28] Liu F, Yang J, Li ZB, Qi JY. A secure multi-party computation protocol for universal data privacy protection based on blockchain. Journal of Computer Research and Development, 2021, 58(2): 281–290 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2021.20200751]

[29] Bowen JP. The Z notation: Whence the cause and whither the course? In: Proc. of the 1st Int'l School on Engineering Trustworthy Software Systems. Chongqing: Springer, 2014. 103–151. [doi: 10.1007/978-3-319-29628-9_3]

[30] Cao Z, Wang H. Extending interface automata with Z notation. In: Proc. of the 4th Int'l IPM Conf. on Fundamentals of Software Engineering. Tehran: Springer, 2011. 359–367. [doi: 10.1007/978-3-642-29320-7_25]

[31] Dimou C, Tzima F, Symeonidis AL, Mitkas PA. Performance evaluation of agents and multi-agent systems using formal specifications in Z notation. In: Proc. of the 10th Int'l Workshop on Agents and Data Mining Interaction. Paris: Springer, 2014. 64–78. [doi: 10.1007/978-3-319-20230-3_6]

[32] Klein MJ, Sawicki S, Roos-Frantz F, Frantz RZ. On the formalisation of an application integration language using Z notation. In: Proc. of the 16th Int'l Conf. on Enterprise Information Systems. Lisbon: SCITEPRESS, 2014. 314–319. [doi: 10.5220/0004967003140319]

[33] Jamil A, Murtza Z, Nazir MK, Waseem M, Ghulam Z, Farooq RU. A generic formal specification of an infinite runner games for handheld devices using Z-notation. In: Proc. of the 4th Int'l Conf. on Computer and Communication Systems (ICCCS). Singapore: IEEE, 2019. 409–413. [doi: 10.1109/CCOMS.2019.8821750]

[34] Oliveira M, Cavalcanti A, Woodcock J. Unifying theories in ProofPower-Z. In: Proc. of the 1st Int'l Symp. on Unifying Theories of Programming. Walworth Castle: Springer, 2006. 123–140. [doi: 10.1007/11768173_8]

[35] Lv YY, Feng RT, Ma MD, Zhu MQ, Wu HW, Li XH. Reinventing multi-user authentication security from cross-chain perspective. IEEE Trans. on Information Forensics and Security, 2024, 19: 8908–8923. [doi: 10.1109/TIFS.2024.3463533]

[36] Cremers CJF. The scyther tool: Verification, falsification, and analysis of security protocols. In: Proc. of the 20th Int'l Conf. on Computer Aided Verification. Princeton: Springer, 2008. 414–418. [doi: 10.1007/978-3-540-70545-1_38]

[37] Song DX. Athena: A new efficient automatic checker for security protocol analysis. In: Proc. of the 12th IEEE Computer Security Foundations Workshop. Mordano: IEEE, 1999. 192–202. [doi: 10.1109/CSFW.1999.779773]

[38] Zhang SJ, Lee JH. Double-spending with a Sybil attack in the Bitcoin decentralized network. IEEE Trans. on Industrial Informatics, 2019, 15(10): 5715–5722. [doi: 10.1109/TII.2019.2921566]

[39] Psaras I, Tsaoussidis V. Why TCP timers (still) don't work well. Computer Networks, 2007, 51(8): 2033–2048. [doi: 10.1016/j.comnet.2006.10.006]

[40] Xu SJ, Zhang LL, Wang LH, Mihaljević MJ, Zhang SH, Shao W, Wang QZ. Relay network-based cross-chain data interaction protocol with integrity audit. Computers and Electrical Engineering, 2024, 117: 109262. [doi: 10.1016/j.compeleceng.2024.109262]

[41] Lv YY, Li XH, Wang YWB, Chen K, Hou Z, Feng RT. Cross-chain sharing of personal health records: Heterogeneous and interoperable blockchains. In: Proc. of the 2024 IEEE Int'l Conf. on Bioinformatics and Biomedicine (BIBM). Lisbon: IEEE, 2024. 3588–3591. [doi: 10.1109/BIBM62325.2024.10822679]

[42] Pillai B, Biswas K, Hóu Z, Muthukkumarasamy V. The burn-to-claim cross-blockchain asset transfer protocol. In: Proc. of the 25th Int'l Conf. on Engineering of Complex Computer Systems (ICECCS). IEEE, 2020. 119–124. [doi: 10.1109/ICECCS51672.2020.00021]

[43] Ding DH, Long B, Zhuo F, Li ZC, Zhang HW, Tian C. Lilac: Parallelizing atomic cross-chain swaps. In: Proc. of the 2022 IEEE Symp. on Computers and Communications (ISCC). Rhodes: IEEE, 2022. 1–8. [doi: 10.1109/ISCC55528.2022.9912942]

[44] Vishwakarma L, Kumar A, Das D. CrossLedger: A pioneer cross-chain asset transfer protocol. In: Proc. of the 23rd IEEE/ACM Int'l Symp. on Cluster, Cloud and Internet Computing (CCGrid). Bangalore: IEEE, 2023. 568–578. [doi: 10.1109/CCGrid57682.2023.00059]

[45] Wang F, Wu JL, Nan YH, Aafer Y, Zhang XY, Xu DY, Payer M. ProFactory: Improving IoT security via formalized protocol customization. In: Proc. of the 31st USENIX Security Symp. Boston: USENIX Association, 2022. 3879–3896.

[46] Yin JQ, Fei Y. FVF-BIoT: A formal verification framework for blockchain-based IoT authentication. Software Quality Journal, 2024,

© 中国科学院软件研究所 http://www.jos.org.cn

32(4): 1457–1480. [doi: 10.1007/s11219-024-09691-3]

[47] Chen JF, Feng QW, Cai SH, Shi DZ, Sosu RNA. Vulnerability detection model for blockchain systems based on formal method. Ruan Jian Xue Bao/Journal of Software, 2024, 35(9): 4193–4217 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7133.htm [doi: 10.13328/j.cnki.jos.007133]

[48] Ge N, He YK, Zhai SM, Li XZ, Zhang L. Formal verification of consensus protocols: Survey and perspective. Ruan Jian Xue Bao/Journal of Software, 2023, 34(11): 4989–5007 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6684.htm [doi: 10.13328/j.cnki.jos.006684]

[49] Alam Q, Malik SUR, Akhunzada A, Choo KKR, Tabbasum S, Alam M. A cross tenant access control (CTAC) model for cloud computing: Formal specification and verification. IEEE Trans. on Information Forensics and Security, 2017, 12(6): 1259–1268. [doi: 10.1109/TIFS.2016.2646639]

[50] Zhang YD, Song F. Model-checking for heterogeneous multi-agent systems. Ruan Jian Xue Bao/Journal of Software, 2018, 29(6): 1582–1594 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5462.htm [doi: 10.13328/j.cnki.jos.005462]

[51] Dong JN, Xu GX, Ma C, Liu J, Cliff UGO. Blockchain-based certificate-free cross-domain authentication mechanism for industrial Internet. IEEE Internet of Things Journal, 2024, 11(2): 3316–3330. [doi: 10.1109/JIOT.2023.3296506]

[52] Afzaal H, Imran M, Janjua MU. Formal verification of persistence and liveness in the trust-based blockchain crowdsourcing consensus protocol. Computer Communications, 2022, 192: 384–401. [doi: 10.1016/j.comcom.2022.06.014]

[53] Wang DY. Formal analysis of key exchange protocol in a blockchain-based trustworthy transmission system. In: Proc. of the 4th Int'l Conf. on Information Communication and Software Engineering (ICICSE). Beijing: IEEE, 2024. 153–158. [doi: 10.1109/ICICSE61805.2024.10625692]

[54] Wang G, Li D, Song HB. A formal analytical framework for IoT-based plug-and play manufacturing system considering product life-cycle design cost. IEEE Trans. on Industrial Informatics, 2023, 19(2): 1647–1654. [doi: 10.1109/TII.2022.3192681]

[55] Feng HN, Guan JJ, Li H, Pan XS, Zhao ZM. FIDO gets verified: A formal analysis of the universal authentication framework protocol. IEEE Trans. on Dependable and Secure Computing, 2023, 20(5): 4291–4310. [doi: 10.1109/TDSC.2022.3217259]

[56] Neupane RL, Bonnah E, Bhusal B, Neupane K, Hoque KA, Calyam P. Formal verification for blockchain-based insurance claims processing. In: Proc. of the 2024 IEEE Network Operations and Management Symp. Seoul: IEEE, 2024. 1–5. [doi: 10.1109/NOMS59830.2024.10575641]

[57] Afzaal H, Zafar NA, Tehseen A, Kousar S, Imran M. Formal verification of justification and finalization in beacon chain. IEEE Access, 2024, 12: 55077–55102. [doi: 10.1109/ACCESS.2024.3389551]

附中文参考文献

[1] 司冰茹, 肖江, 刘存扬, 戴小海, 金海. 区块链网络综述. 软件学报, 2024, 35(2): 773–799. http://www.jos.org.cn/1000-9825/6985.htm [doi: 10.13328/j.cnki.jos.006985]

[5] 中国信息通信研究院. 区块链白皮书 (2023 年). 北京: 中国信息通信研究院, 2023.

[9] 王锋, 张强, 刘扬, 刘琳琳, 路阳. 从扩展性角度看区块链. 计算机应用研究, 2023, 40(10): 2896–2907. [doi: 10.19734/j.issn.1001-3695.2023.02.0075]

[10] 段田田, 张瀚文, 李博, 宋兆雄, 李忠诚, 张珺, 孙毅. 区块链互操作技术综述. 软件学报, 2024, 35(2): 800–827. http://www.jos.org.cn/1000-9825/6950.htm [doi: 10.13328/j.cnki.jos.006950]

[11] 朱涵, 吴胜. 区块链跨链技术及其安全性综述. 计算机应用研究, 2024, 41(12): 3543–3552. [doi: 10.19734/j.issn.1001-3695.2024.04.0116]

[16] PolkaWorld. 让波卡跨链能力更进一步的 XCMP 是将如何设计和实现? 2024. https://m.jinse.cn/blockchain/3681944.html

[17] PolkaWorld. 盘点 Moonbeam 网络中的 XCM 集成. 2023. https://foresightnews.pro/article/detail/24540?utm_source=chatgpt.com

[18] 陈伟利, 郑子彬. 区块链数据分析: 现状、趋势与挑战. 计算机研究与发展, 2018, 55(9): 1853–1870. [doi: 10.7544/issn1000-1239.2018.20180127]

[19] 孟博, 王乙丙, 赵璨, 王德军, 麻斌豪. 区块链跨链协议综述. 计算机科学与探索, 2022, 16(10): 2177–2192. [doi: 10.3778/j.issn.1673-9418.2203032]

[25] 苗新亮, 常瑞, 潘少平, 赵永望, 蒋烈辉. 可信执行环境访问控制建模与安全性分析. 软件学报, 2023, 34(8): 3637–3658. http://www.jos.org.cn/1000-9825/6612.htm [doi: 10.13328/j.cnki.jos.006612]

[26] 邹盛荣, 郑国梁. B 语言和方法与 Z、VDM 的比较. 计算机科学, 2002, 29(10): 136–138. [doi: 10.3969/j.issn.1002-137X.2002.10.041]

[28] 刘峰, 杨杰, 李志斌, 齐佳音. 一种基于区块链的泛用型数据隐私保护的安全多方计算协议. 计算机研究与发展, 2021, 58(2):

281–290. [doi: 10.7544/issn1000-1239.2021.20200751]
[47] 陈锦富, 冯乔伟, 蔡赛华, 施登洲, Sosu RNA. 基于形式化方法的区块链系统漏洞检测模型. 软件学报, 2024, 35(9): 4193–4217. http://www.jos.org.cn/1000-9825/7133.htm [doi: 10.13328/j.cnki.jos.007133]
[48] 葛宁, 贺俞凯, 翟树茂, 李晓洲, 张莉. 共识协议的形式化验证研究现状与展望. 软件学报, 2023, 34(11): 4989–5007. http://www.jos.org.cn/1000-9825/6684.htm [doi: 10.13328/j.cnki.jos.006684]
[50] 张业迪, 宋富. 异构多智能体系统模型检查. 软件学报, 2018, 29(6): 1582–1594. http://www.jos.org.cn/1000-9825/5462.htm [doi: 10.13328/j.cnki.jos.005462]

作者简介

吕永阳, 博士生, 主要研究领域为形式化方法, 区块链, 网络安全, 数据安全.
冯睿韬, 博士, 副研究员, 主要研究领域为机器学习, 软件安全, 区块链安全, 形式化方法.
王子墨, 本科生, 主要研究领域为形式化方法, 区块链.
刘俊超, 本科生, 主要研究领域为形式化方法, 区块链.
吴汉炜, 博士生, 主要研究领域为形式化方法, 安全协议验证, 区块链安全.
李晓红, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为形式化方法, 软件安全, 区块链安全.

www.jos.org.cn
www.jos.org.cn