软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007641] [CSTR: 32375.14.jos.007641]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

OpenSwap: 基于先锁后绑模式的跨链资产交换协议*

卓 凤[1,2], 段田田[1], 贾林鹏[1], 李忠诚[1,2], 孙 毅[1,2,3,4]

[1](中国科学院 计算技术研究所, 北京 100190)

[2](中国科学院大学 计算机科学与技术学院, 北京 100049)

[3](未来区块链与隐私计算高精尖创新中心 (北京航空航天大学), 北京 100083)

[4](山东省区块链金融重点实验室, 山东 济南 250014)

通信作者: 贾林鹏, E-mail: jialinpeng@ict.ac.cn

摘 要: 跨链资产交换是资产跨区块链流通的典型模式之一. 以哈希时间锁 (hashed timelock contract, HTLC) 协议为代表的现有跨链资产交换方案普遍采用“先绑定跟随方、跟随方再锁定资产”的先绑后锁模式. 在该模式下, 一旦已绑定的跟随方退出, 发起方将无法更换交换对象, 只能等待时间锁超时后回收资产并重新发起交换, 从而显著延长交换周期并带来高额的链上开销. 为此, 提出一种先锁后绑的跨链资产交换新模式, 即交换开始前不在合约中预设跟随方, 而是由符合条件的响应方先行锁定资产, 再将其绑定为跟随方. 该模式可避免绑定对象退出带来的多次重试与资产长期锁定问题. 在该模式下, 提出一种基于地址签名锁的跨链资产交换协议——OpenSwap. OpenSwap 设计了一种地址签名锁, 通过将跟随方身份信息嵌入锁结构, 使其随着资产的锁定与解锁过程在两条链之间同步, 从而确保两条链对跟随方绑定的一致性. 此外, OpenSwap 还通过挑战期、协助解锁等设计, 提升协议安全性和执行效率. 理论分析与实验结果表明, OpenSwap 在保证原子性的同时, 显著降低了交换延迟, 并在低用户响应场景下降低了链上开销, 为跨链资产交换提供了更灵活与高效的解决方案.

关键词: 区块链; 跨链资产交换; 哈希时间锁

中图法分类号: TP311

中文引用格式: 卓凤, 段田田, 贾林鹏, 李忠诚, 孙毅. OpenSwap: 基于先锁后绑模式的跨链资产交换协议. 软件学报. http://www.jos.org.cn/1000-9825/7641.htm

英文引用格式: Zhuo F, Duan TT, Jia LP, Li ZC, Sun Y. OpenSwap: Cross-chain Asset Swap Protocol Based on Lock-then-bind Mode. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7641.htm

OpenSwap: Cross-chain Asset Swap Protocol Based on Lock-then-bind Mode

ZHUO Feng[1,2], DUAN Tian-Tian[1], JIA Lin-Peng[1], LI Zhong-Cheng[1,2], SUN Yi[1,2,3,4]

[1](Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)

[2](School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China)

[3](Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing (Beihang University), Beijing 100083, China)

[4](Shandong Key Laboratory of Blockchain Finance, Jinan 250014, China)

Abstract: Cross-chain asset swaps are a typical mode of asset circulation across blockchains. Existing cross-chain swap solutions, represented by the hashed timelock contract (HTLC) protocol, typically adopt a bind-then-lock mode, in which the follower is bound first and then locks assets. Under this mode, once the bound follower withdraws, the initiator cannot replace the swap counterparty and must wait for the timelock to expire before reclaiming the assets and restarting the swap process, which significantly prolongs the swap duration and incurs high on-chain costs. To address this issue, this study proposes a lock-then-bind mode for cross-chain asset swaps. In this paradigm, the follower is not pre-specified in the contract before the swap starts. Instead, an eligible responder first locks assets and is

* 基金项目: 国家重点研发计划 (2023YFB2704803); 国家自然科学基金 (U22B2032)
收稿时间: 2025-09-02; 修改时间: 2025-11-25; 采用时间: 2026-01-23; jos 在线出版时间: 2026-04-29

then bound as the follower. This paradigm avoids repeated retries and long-term asset locking caused by follower withdrawal. Under the new paradigm, this study proposes a cross-chain asset swap protocol based on an address signature lock, namely OpenSwap. OpenSwap designs an address signature lock by embedding the follower's identity information into the lock structure, enabling it to be synchronized between two blockchains during the asset locking and unlocking process, thus ensuring consistency in follower binding across both chains. In addition, OpenSwap enhances protocol security and execution efficiency through the introduction of a challenge period and an assisted unlocking mechanism. Theoretical analysis and experimental results demonstrate that OpenSwap ensures atomicity while significantly reducing swap latency and lowering on-chain costs in low user response scenarios, providing a more flexible and efficient solution for cross-chain swaps.

Key words: blockchain; cross-chain asset swap; hashed timelock contract (HTLC)

区块链技术[1]自问世以来, 凭借其去中心化与不可篡改的特性, 迅速成为全球技术创新的重要方向. 经过十余年的发展, 区块链已在数字金融、政务服务、供应链管理等多个领域得到应用, 形成了大量异构区块链并存的多链生态[2]. 由于不同区块链系统彼此拥有独立的共识过程, 链与链之间难以直接互操作, 用户难以在不同区块链之间进行点对点资产互换. 早期通常通过中心化交易所实现这种交换, 但此类方式要求用户将资产托管于第三方, 存在严重的安全风险. 随着中心化交易所安全事件频繁发生[3−5], 在无需引入可信第三方的前提下实现安全、可靠的跨链资产交换[6,7], 已成为区块链互操作[8−10]研究中一个至关重要的问题.

现有跨链资产交换方案根据是否依赖跨链基础设施, 可分为系统级方案和用户级方案两类. 系统级方案[11−13]通常借助公证人、轻客户端、中继链等机制构建跨链基础设施, 通过链间信息传输与验证使参与交换的区块链获取彼此的链上状态, 并据此执行交换中对应的资产转移操作, 从而完成跨链资产交换. 用户级方案[6,7,14,15]则不依赖跨链基础设施, 而是由参与交换的用户自行监听交换链的状态, 并及时执行相应的资产转移来完成交换. 哈希时间锁 (hashed timelock contract, HTLC) 协议[6,7]是最具代表性的用户级跨链资产交换方案, 凭借其实现灵活、易于部署等优势, 已被广泛应用于众多跨链平台与去中心化交易所中[16−19].

在跨链资产交换的基础场景中, 用户 Alice (发起方) 持有一定数量的币安币 (binance coin, BNB), 希望找到持有一定数量以太币 (ether, ETH) 的用户 Bob (跟随方), 与其交换资产. 该交换涉及两个需要原子性执行的资产转移: 在币安链上, Alice 将待交换的 BNB 资产发送给 Bob; 在以太坊链上, Bob 将待交换的 ETH 资产发送给 Alice. 基于 HTLC 协议完成该交换时, 发起方 Alice 先将待交换的 BNB 资产锁定至币安链的 HTLC 合约 (主合约), 并预先指定交换对象 Bob, 完成跟随方的绑定操作. 随后, 跟随方 Bob 再将待交换 ETH 资产锁定至以太坊链的 HTLC 合约 (从合约) 中, 完成锁定操作. 一旦双方都锁定资产, 解锁凭证将被释放, 两合约中的资产均会被解锁给相应的资产接收方, 否则已锁资产将在时间锁超时后被退还给资产原有方.

然而, HTLC 协议采用的这种先绑后锁模式存在跟随方可随意退出、导致发起方被动等待与多次重试的机制性缺陷. 如图 1, 在这种模式下, 由于发起方在锁定资产时已将跟随方绑定至主合约, 一旦跟随方退出, 发起方将无法更换交换对象, 只能被动等待时间锁超时后回收资产并重新发起交换. 考虑到时间锁往往持续数小时甚至数天[20,21], 多次重试将显著延长交换周期. 此外, 每次尝试需要在链上锁定与解锁资产, 多次重试将增大交易开销. 同时, 恶意对手可以通过反复响应再退出的方式, 使发起方陷入多次锁定与长时间等待.

近些年, 已有多项研究在通用性、安全性以及长期锁定问题等方面对 HTLC 协议进行了优化. 其中, 针对长期锁定问题的一类工作[22−26], 通过引入惩罚机制来降低理性参与方退出交换的概率. 针对协议容错性的一类工作[27]通过同时与多个候选跟随方并行地进行交换, 使得即使部分候选方退出, 交换也有可能完成. 这两类工作虽然关注点不同, 但均能提升单次尝试的成功概率, 因此能够减少重试次数, 从而带来执行时间的改善效果. 然而, 这些优化工作仍属于先绑后锁模式, 在跟随方退出时无法避免 (或无法完全避免) 交换失败与重试, 因而也无法避免随重试次数线性增长的交换完成时间与交易开销. 此外, 当前容错类的方案[27]由于需要多个候选跟随方并行锁定资产, 还会造成大量资产长期锁定, 并带来显著的链上交易开销.

为此, 本文提出了一种先锁后绑的跨链资产交换新模式. 如图 2, 在该模式中, 发起方锁定资产时不预设具体的跟随方, 而是允许任何满足条件的用户作为候选方响应. 所有候选方均可尝试在链上发起锁定操作参与响应, 仅一位候选方能够成功锁定资产, 成为实际跟随方. 其身份信息随后将被传递至主合约, 由主合约完成绑定. 由于跟

随方的绑定发生在锁定资产之后, 且锁定资产后无法随意退出, 该模式有效避免了跟随方退出带来的交换失败以及发起方资产长期锁定等问题. 由于在跨链资产交换场景中, 主从合约分别位于两条区块链上, 主合约无法确认从合约上的跟随方, 因此如何在主合约上绑定与从合约一致的跟随方, 从而保障两合约中资产的一致解锁, 成为该模式下协议设计的关键难点.

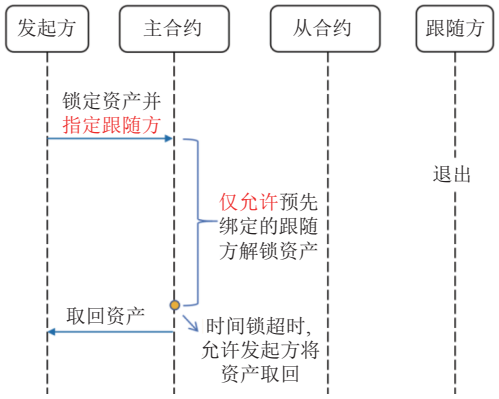


图 1 先绑后锁模式下跟随方退出导致交换失败及资产长期锁定

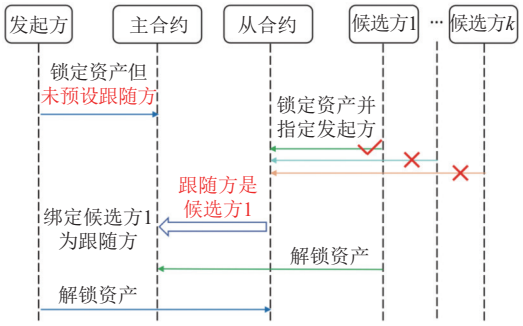


图 2 先锁后绑的跨链资产交换模式

为实现上述新模式, 本文提出一种跨链资产交换协议 OpenSwap. 该协议设计了一种地址签名锁来锁定资产, 通过将跟随方身份信息嵌入锁结构, 使其随着资产的锁定与解锁过程在两条链之间同步, 实现了主从合约对跟随方身份的一致绑定. 在此基础上, 为防止发起方利用多个签名进行双重解锁攻击, OpenSwap 引入挑战期的设计, 通过延迟资产释放保障了两合约的一致解锁, 从而提升交换的安全性. 此外, OpenSwap 还通过协助解锁的设计, 激励参与者主动触发解锁, 进一步提升交换的执行效率.

综上所述, 本文的主要贡献如下.

1) 模式创新: 提出先锁后绑的跨链资产交换模式, 打破传统先绑后锁的结构限制, 有效避免了失败等待和多次重试等问题.

2) 协议设计: 设计基于先锁后绑模式的 OpenSwap 协议, 提出可携带信息的地址签名锁, 并通过挑战期与协助解锁等设计, 实现了安全、高效的两方跨链资产交换.

3) 安全性证明: 通过定理推导、博弈论建模和 TLA+ (temporal logic of action)[28]验证, 证明 OpenSwap 能够保障两方跨链资产交换的原子性.

4) 性能评估: 实现了 OpenSwap 完整协议, 并与 HTLC 协议进行对比实验. 实验结果表明, OpenSwap 显著缩短了交换的平均完成时间, 并有效避免多次尝试所带来的链上交易开销.

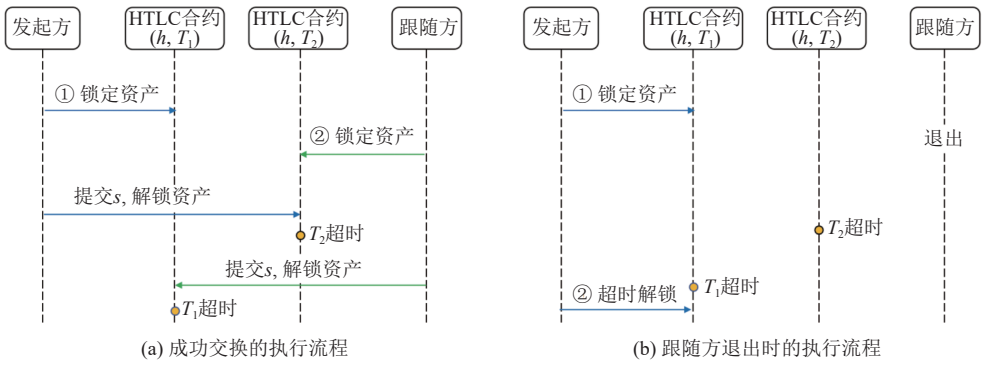
1 背景与相关工作

1.1 哈希时间锁协议

在基于哈希时间锁协议[6]的两方跨链资产交换中, 首先, 发起方需要通过某些交易撮合平台[29]找到一个符合条件的跟随方, 并与之达成交换约定. 随后, 双方在链上经过资产锁定与解锁两个阶段, 完成原子的资产转移.

哈希时间锁协议通过 HTLC 合约实现交换用户资产的锁定与解锁. 一个 HTLC 合约中包含一个哈希锁 h 和一个时间锁 T, h 是某个秘密值 s (即解锁凭证) 进行哈希计算 (例如, $h = SHA\text{-}256(s)$) 得到的结果, T 是一段时间或一定数量的区块间隔. 当合约中锁定了一份待交换资产时, 如果资产的接收方在 T 超时前向合约提交了 s, 哈希锁会打开, 合约会将资产解锁给该接收方; 否则, 资产的发送方在 T 超时后可以取回锁定的资产.

如图 3 所示, 基于 HTLC 协议的两方资产交换流程具体如下.



(a) 成功交换的执行流程　　(b) 跟随方退出时的执行流程

图 3　哈希时间锁协议的执行流程

1) 发起方首先在其待交换资产所在链上创建主合约 (h,T_1) 并将其待交换资产锁定至该合约.

2) 跟随方关注到发起方已锁定资产后, 在其待交换资产所在链上创建与主合约具有同样哈希锁的从合约 (h,T_2), 并将其待交换资产锁定至该合约.

3) 如图 3(a), 当发起方确认跟随方已锁定资产后, 将在 T_2 超时前向从合约提交解锁凭证 s, 使从合约的哈希锁打开, 将跟随方锁定的资产释放给发起方, 完成跟随方到发起方的资产转移.

4) 当跟随方看到解锁凭证 s 后, 将在 T_1 超时前将其提交至主合约, 使主合约的哈希锁打开, 将发起方锁定的资产释放给跟随方, 完成发起方到跟随方的资产转移.

在交换过程中, 若发起方锁定资产后跟随方未如约锁定资产 (如图 3(b)), 或跟随方锁定资产后发起方拒绝公开解锁凭证, 交换将失败. 在失败的情况下, 如发起方已锁定资产, 其资产可在时间锁 T_1 到期后取回; 如跟随方已锁定资产, 其资产可在时间锁 T_2 到期后取回.

本文聚焦于发起方有较强烈的交换意愿, 但跟随方交换意愿没有那么强烈的场景. 在该场景下, 图 3(b) 中跟随方拒绝锁定资产的情况发生概率较高, 发起方资产长期锁定及重新发起交换较频繁.

1.2　相关工作

2013 年, TierNolan[6]在比特币论坛首次提出原子跨链交换 (atomic cross-chain swap) 的概念, 并提出用哈希时间锁 (HTLC) 实现两方交换协议. 随后, Decred 项目[19]首次将该协议付诸实现, 用于 Decred 和 LiteCoin 间的资产互换. 近年来, 研究者针对 HTLC 协议的实际应用需求, 围绕通用性、锁定时间与安全性等方面提出了大量扩展和优化工作.

● 提升协议通用性: 传统 HTLC 协议要求交换链均支持脚本语言和时间锁. 为拓展适用范围, 研究者提出多种可运行于不支持脚本或时间锁的区块链上的方案. 例如, LightSwap 等方案[30,31]将时间锁逻辑集中至一条链上实现, 使另一条链无需支持时间锁. Manevich 等人[32]提出一种密码学原语 AVTC 实现时间锁逻辑. JugglingSwap[33]通过分段加密实现了无需脚本的部分公平交换. Thyagarajan 等人[15]提出的 universal atomic swaps 则利用可验证时间签名 (verifiable timed signature, VTS) 实现了无需脚本的多方多资产交换.

● 扩展协议使用场景: 传统 HTLC 协议仅支持两方参与. 为扩展协议的使用场景, Herlihy 等人[7]提出支持多参与方的交换协议, 并进一步设计跨链协作 (deal) 协议[14]以支持跨链拍卖、跨链闪电贷等复杂跨链应用. Shadab 等人[34]构建了一种支持包含链下步骤的跨链交易通用协议. Lilac 等方案[35,36]则在性能、计算开销方面对多方交换协议进行优化. Chan 等人[37]提出一种支持用户偏好设定的交换协议.

● 缓解长期锁定: 已有研究[38−40]表明, 在价格波动等因素影响下, 用户可能理性地选择退出交换, 从而导致 HTLC 协议频繁失败并造成资产长期锁定风险[41]. 为此, Han 等人[22]提出对退出交换方进行惩罚、对资产长期锁定方进行补偿, 后续多项研究[23−26]围绕惩罚对象、金额设定等方面进行优化, 提升协议公平性. Zhuo 等人[42]提出的 FS-Swap 通过设计快速退款机制有效避免了因交换失败导致的资产长期锁定.

● 防御原子性破坏: 部分工作专注于防御 HTLC 协议中由临时审查攻击[43]引发的原子性破坏风险. Winzer 等人[43]提出 3 种审查攻击方式; Nadahalli 等人[44]对攻击收益边界进行了分析; Tsabary 等人[45]提出的 MAD-HTLC 借助理性矿工的监督来防止攻击. Wadhwa 等人[46]则提出 He-HTLC, 解决了 MAD-HTLC 中存在的逆向贿赂问题.

● 隐私保护: 传统 HTLC 协议会将交易细节和参与者暴露在链上, 缺乏隐私保障. Deshpande 等人[47]和 Cao 等人[48]分别提出基于适配器签名与零知识证明的私密 HTLC 方案, 用于隐藏参与者身份与交易内容.

● 容错性: 传统 HTLC 协议中, 跟随方退出交换即导致交换失败. Xue 等人[27]提出使发起方同时与多个候选跟随方并行地进行交换, 使得协议具备容错性, 即使部分候选方退出, 交换也有可能完成.

尽管上述工作从不同角度对 HTLC 协议进行扩展与优化, 但这些方案均属于“先绑再锁”模式, 用户退出均会导致交换失败与重试, 无法避免线性增长的执行时间与交易开销.

除了无需跨链基础设施的 HTLC 类方案外, 一些系统级跨链协议[11−13]通过公证人、轻客户端、中继链等机制构建跨链基础设施, 通过链间信息传输与验证来支持更一般化的跨链场景. 近些年, 围绕这些协议也出现了诸多优化工作. 例如, 聂鹏等人[49]提出一种面向异构区块链系统的中继链跨链架构, 提升了系统的可扩展性与兼容性. 张佩云等人[50]设计了两阶段跨链访问控制机制, 以增强访问过程的安全性与灵活性. 魏秋阳等人[51]和吕永阳等人[52]对 IBC[12]等协议进行形式化建模与验证, 发现重放攻击、延迟攻击等潜在风险, 并提出修复建议以提升协议安全性. 这类研究多聚焦协议机制与安全性保障, 侧重于跨链通信基础设施增强, 与 HTLC 类方案形成互补的研究趋势.

本文提出的 OpenSwap 协议作为 HTLC 类的一种优化工作, 通过先锁后绑的交换模式, 有效解决了交换失败与多次重试带来的效率与开销问题, 为跨链资产交换提供了一种更高效、低开销的解决方案.

2 系统模型

2.1 基础模型定义

OpenSwap 运行于无需可信第三方与跨链基础设施的多区块链环境中, 参与方通过链上合约调用完成资产交换操作. 系统模型包括以下要素.

● 参与方. 发起方在区块链 BC_1 上持有一定数量的资产 $Coin_1$, 希望拿其中的 x_1Coin_1 换得区块链 BC_2 上 x_2Coin_2 的资产. 协议所需押金为 $yCoin_2$. 区块链 BC_2 上存在 n 个持有 $Coin_2$ 资产数目大于 x_2+y 的候选跟随方可与发起方进行交换. 任一交换方均在链 BC_1 和链 BC_2 上都有账户.

● 区块链. 交换涉及的两条区块链 BC_1、BC_2 应具备主流区块链平台的能力, 满足以下条件.

1) 支持部署智能合约 (或脚本), 能够实现基于时间锁的条件执行.

2) 支持 ECDSA 等常用的数字签名算法.

3) 链上交易一旦执行即不可撤销.

4) 支持链上交易执行和事件记录的查询和验证 (以便参与方获取链上信息).

5) 所有节点对交易执行列表最终达成一致.

● 通信模型. 我们将区块链系统中, 从用户发起一笔交易到该交易被打包进区块并被系统中其他用户看到所需的时间, 称为交易完成时间. 与 HTLC 协议类似, OpenSwap 要求每条链的交易完成时间都存在一个确定的时间上界[7,23]. 该时间上界可通过理论分析或交易历史数据获得, 用于在协议中合理设置时间锁的取值. 例如, 假设在链 BC_1 中交易完成时间上界为 Δ_1, 在 BC_2 中交易完成时间上界为 Δ_2. 在 HTLC 协议中, 链 BC_1 上的时间锁 T_1 需要覆盖 1 笔在链 BC_1 的交易 (跟随方的解锁交易)、2 笔在链 BC_2 的交易 (跟随方的锁定交易及发起方的解锁交易) 的完整执行时间, 因此其值应不小于 $\Delta_1+2\Delta_2$. 同理链 BC_2 上时间锁 T_2 应不小于 Δ_2.

2.2 跨链资产交换原子性定义

原子性是跨链资产交换的基础目标. 现有研究[7,14,23,34,36,53]普遍认为, 用户级跨链资产交换方案这类通过用户监听链上状态, 在链上发布相应交易来完成交换的方案, 无法保障传统意义上的绝对原子性 (即资产转移要么全部执行, 要么全部不执行). 它们保障的是一种博弈论意义下的原子性: 当所有参与方都理性时, 没有参与方有动力偏

离协议, 交换将顺利完成; 即使存在不理性的参与者, 遵守协议的理性参与者也不会丢失资产 (即未获得目标资产的前提下失去待交换资产).

例如, 在 HTLC 协议中, 如果发起方在跟随方锁定资产前就泄露了解锁凭证, 可能导致其资产被对方取走, 却无法获得对方资产. 从传统原子性的角度看, 这种行为破坏了协议的原子性. 但这种行为违背发起方自身利益, 因此在理性假设下不会发生. 而如果发起方确实出现这样非理性的行为, 该行为也不会使理性的跟随方丢失资产. 因此, 该行为并不会使 HTLC 协议违背博弈论意义下的原子性.

本文采用现有研究普遍认可的跨链资产交换原子性[7]定义, 具体如下.

定义 1. 一致性 (Uniform). 一个跨链资产交换协议是一致的, 如果它满足以下两个条件.

1) 如果所有参与方均遵守协议, 交换将成功完成.

2) 任意参与方联合偏离协议都不会使遵守协议的参与方丢失资产.

定义 2. 原子性 (Atomic). 一个跨链资产交换协议是原子的, 如果它满足以下两个条件.

1) 它是一致的.

2) 它所诱导出的行为策略构成一个强纳什均衡 (strong Nash equilibrium)[54], 即不存在任何参与者子集可以通过联合偏离协议使所有偏离成员的收益同时提升.

2.3 先锁后绑模式设计难点

由于区块链交易具有不可回撤的特性, 若将跨链资产交换中的资产转移操作作为链上普通交易执行, 可能会出现一条链上的资产转移交易未执行而另一条链上的资产转移交易已执行且不可回滚的情况, 破坏原子性. 因此, 现有跨链资产交换方案通常要求交换双方先将资产锁定至各自链上的托管合约, 再通过协调机制使两托管合约进行一致解锁: 要么资产均释放给接收方, 要么资产均退还给原有方.

协调机制的关键在于: 一条链上的托管合约必须能够获知另一条链上托管合约的执行情况, 以据此决定本合约中资产的处置方式. 例如, 当确认对端已将资产释放给接收方时, 本合约才能安全地将资产释放给接收方. 然而, 在缺乏链间通信机制时, 合约无法直接访问对端状态, 只能通过参与者提交的信息间接推断对端执行情况. 由于参与者可能不诚实, 合约必须在不直接信任该信息的前提下完成判断.

传统协议基于先绑后锁模式, 即在锁定资产前, 交换双方的身份信息已写入托管合约. 在该模式下, 合约只需要判断资产应释放给接收方, 还是退还给发送方, 即两合约仅需要对解锁方向达成一致. 以 HTLC 协议为例, 其合约部署形式如下.

- 主合约 (发起方锁定资产): $C_1 = \langle addr_I, addr_F, h, T_1 \rangle$
- 从合约 (跟随方锁定资产): $C_2 = \langle addr_F, addr_I, h, T_2 \rangle$

其中, $addr_I$ 为发起方地址, $addr_F$ 为接收方地址 (某个参与方在两条链上的地址实际是不同的, 这里简化表示为同一符号). 哈希锁 h 的原像 s 仅由发起方持有. 在该协议中, 仅在发起方锁定资产后跟随方才会锁定资产, 仅在跟随方锁定资产后发起方才会公开 s, 因此, s 的公开意味着双方均锁定资产. 合约可据此判断解锁方向: 若收到 s, 则意味着两合约均锁定资产并均可释放给接收方, 于是将资产释放给接收方; 若未收到, 则意味着要么存在参与方未能锁定资产, 要么发起方在双方锁定资产后反悔, 合约即可决定将资产退还给发起方.

本文提出的先锁后绑模式不同之处在于, 发起方在锁定资产时尚未确定跟随方, 主合约中并未记录跟随方的地址. 具体而言, 合约部署形式如下.

- 主合约 (发起方锁定资产): $C_1 = \langle addr_I, ?, Condition\_List \rangle$
- 从合约 (跟随方锁定资产): $C_2 = \langle ?, addr_I, Condition\_List \rangle$

其中, $Condition\_List$ 指两合约中定义的资产解锁条件. 在发起方锁定资产后, 任何符合条件的用户都可以进行响应, 最终成功锁定资产并被从合约绑定为跟随方的用户可能是任意一个. 这种情况下, 协议需要解决两个问题.

1) 两合约如何确认一致的跟随方. 在考虑资产的一致解锁前, 首先需要保障两合约对接收方的认定是一致的. 当有候选方成功响应时, 从合约将认定唯一的跟随方, 需要解决的问题是将跟随方的信息同步至主合约. 直观的想法是直接将跟随方地址发送至主合约, 但问题的关键在于主合约难以验证被提交的地址是否确为跟随方地址. 任

何参与者或外部观察者都有动力伪造身份以认领接收方资格, 主合约在缺乏链间通信的前提下无法判断其真实性, 因此需要额外的身份认证机制. 需要注意的是, 现有哈希锁机制并不绑定身份, 凭证一旦公开, 任何观察者都可利用其向合约发起解锁请求, 因此通过该机制合约无法确认提交者是否为真实参与者.

2) 两合约如何进行一致的资产解锁. 先锁后绑模式在解锁前增加了身份信息同步的步骤, 解锁逻辑必须考虑信息同步是否成功. 这是因为, 假设从合约在信息同步完成前便能够先行将资产释放给发起方, 那么当跟随方的身份信息同步失败时, 跟随方将无法获得主合约中的资产, 从而丢失资产. 为确保在信息同步成功或失败时均能保障两合约进行一致的资产解锁, 一个直接的想法是仍采用哈希锁机制, 利用某个秘密凭证 s' 的释放与否标识信息同步成功与否, 使两合约均据此进行解锁. 但这种方法在如下新模式下实际是不可用的.

- 如果由发起方控制 s', 无论信息同步成功与否, 发起方都有动力释放 s' 以获取跟随方的资产. 如果跟随方身份信息的同步实际未成功, 跟随方就会丢失资产.
- 如果由跟随方控制 s', 跟随方可在未锁资产的情况下直接释放 s', 获取发起方的资产, 使发起方丢失资产.

综上, 先锁后绑模式在跨链身份信息同步和一致解锁方面引入了新的挑战, 而现有的哈希锁机制无法直接应对这些挑战. 第 3 节将介绍本文提出的 OpenSwap 协议如何应对这些挑战, 以保障跨链资产交换的原子性.

3 OpenSwap 协议设计

针对第 2 节提到的两个挑战, OpenSwap 分别采用地址签名锁与挑战期的设计, 具体如下.

1) 设计地址签名锁, 以发起方对跟随方地址的签名作为解锁凭证, 使凭证同时具备身份认证功能, 从而在解锁过程中实现跟随方身份信息由从合约向主合约的同步.

2) 鉴于信息同步可能失败, OpenSwap 设置了一个挑战期, 将从合约中资产的实际解锁延后至信息同步结果确认之后, 以保障两条链间解锁操作的一致性.

在此基础上, OpenSwap 还通过协助解锁的设计进一步提升协议的执行效率.

3.1 地址签名锁

OpenSwap 将跟随方身份信息嵌入锁结构, 在资产解锁过程中实现跟随方信息随解锁凭证由从合约到主合约的同步. 具体而言, 地址签名锁的解锁凭证是发起方签名的跟随方收款地址, 即:

$$key = Sign_{sk_I}(r, addr_F) \tag{1}$$

其中, sk_I 是发起方的私钥, 在主从合约创建时对应的公钥 pk_I 即被写入合约中. $\mathrm{Sign}_{sk_I}(\cdot,\cdot)$ 是用私钥 sk_I 对消息进行数字签名的函数. r 是为避免重放攻击选取的随机数, 在每次交换中 r 的取值不同. $addr_F$ 是跟随方在主合约所在区块链上用于接收发起方资产的地址.

如后文图 4, 发起方发起交换后, 跟随方在响应交换时提交地址 $addr_F$ 给从合约, 在从合约建立起地址签名锁. 为打开地址签名锁, 获取跟随方在从合约中锁定的资产, 发起方需要对地址 $addr_F$ 进行签名, 生成正确的解锁凭证 key, 并将 key 提交至从合约. 从合约收到 key 后会利用预存的发起方公钥 pk_I 来验证 key 的正确性, 判断 key 是否确为发起方的签名以及被签名的地址是否确为跟随方上传的地址 $addr_F$. 假如发起方确实公开了正确的解锁凭证 key, 跟随方仅需将 key 提交至主合约即可完成地址 $addr_F$ 由从合约向主合约的同步. 主合约收到 key 后将通过与从合约中一致的发起方公钥 pk_I 验证 key 是否确实为发起方的签名, 验证通过后将绑定 key 中被签名的地址 $addr_F$ 作为跟随方地址.

3.2 通过挑战期实现一致解锁

由于解锁凭证由发起方生成, 发起方可能签署多个地址, 生成若干解锁凭证. 这些凭证在从合约中是无效的, 但主合约无法判别, 仍会将收到的凭证中的地址绑定作为跟随方地址. 假如从合约收到正确的 key 后直接将资产解锁给发起方, 但主合约绑定了其他地址作为跟随方地址, 跟随方就会丢失资产. 具体如图 5 所示, 发起方可同时签名跟随方的地址 $addr_F$ 和他自己的地址 $addr_I$, 生成双方共同的解锁凭证 key 和仅发起方自己可使用的解锁凭证 key^*, 然后发动双重解锁攻击: 向从合约提交 key, 同时向主合约提交 key^*. 在这种情况下, 当跟随方试图提交 key 至主合约时就会同步失败 (主合约已绑定了地址 $addr_I$). 假如从合约已将资产释放给发起方, 跟随方就会丢失资产.

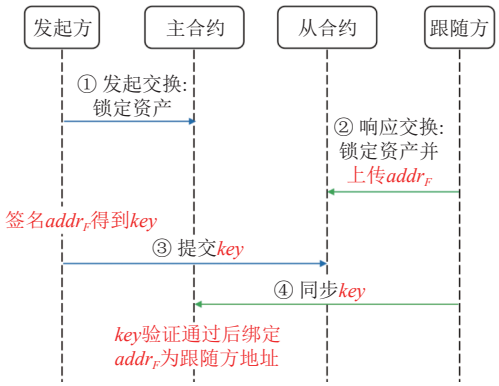


图 4 跟随方身份信息随解锁凭证的跨链同步

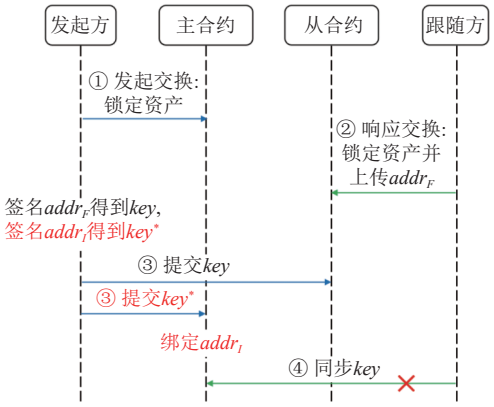
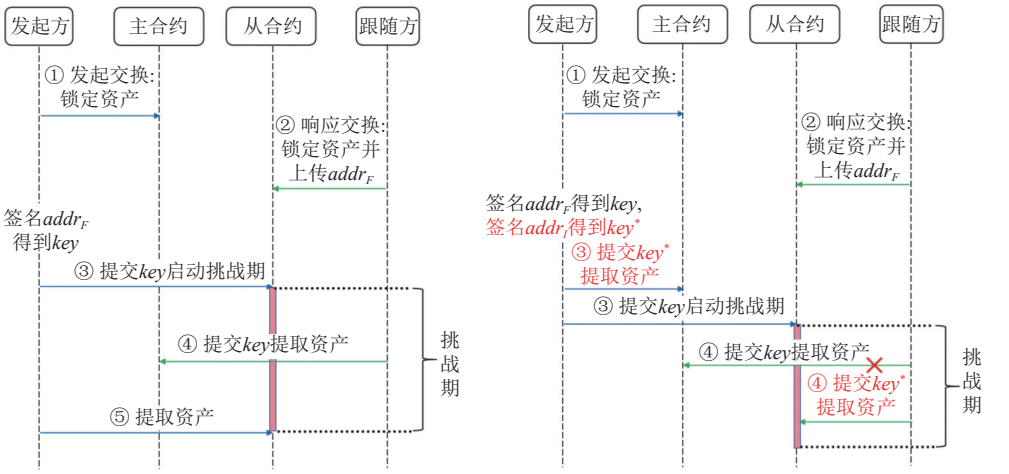


图 5 发起方签署多个地址进行双重解锁攻击

为避免上述情况发生, 进而保障两合约的一致解锁, OpenSwap 引入挑战期延后从合约资产的实际释放时间, 使其在信息同步结果确认后 (主合约资产解锁完成后) 再发生. 如果信息同步成功, 主合约资产释放给正确的跟随方, 那么挑战期结束后从合约资产也将释放给发起方. 反之, 如果信息同步失败, 主合约资产未能释放给正确的跟随方, 那么跟随方能够通过在挑战期内举报发起方的恶意行为保护自己的资产, 使其迅速退还给跟随方.

如图 6(b), 当发起方提交用于解锁从合约资产的 key 后, 从合约不会立即释放资产, 而是进入一个持续时间为 T 的挑战期. 在此期间, 跟随方若发现主合约已绑定其他地址 (通过发起方对该地址的签名 key^*), 可向从合约提交 key^*, 其可以作为发起方的作恶证据, 从合约受到该证据后会立即将资产退还给跟随方 (如图 6(b)). 只有当 T 时间内未收到发起方的作恶证据, T 时间后从合约才会将资产释放给发起方 (如图 6(a)).



(a) 挑战期机制下的正常执行路径 (b) 发起方签署多个地址时跟随方通过举报保护自己的资产

图 6 通过挑战期实现跨链合约一致解锁

3.3 协助解锁

通过解锁凭证与挑战期的设计, OpenSwap 已经能够保障跨链合约的一致解锁, 但在当前设计下发起方即使不作恶也必须等待 T 时间才能提取跟随方资产, 协议的执行效率受损. 针对该问题, OpenSwap 借鉴已有研究[42]中的协助解锁思想, 设计了协助解锁机制: 在发起方不作恶时, 使跟随方主动触发从合约提前释放资产. 如图 7 所示, 如果跟随方通过 key 成功提取发起方的资产, 跟随方接下来可以发一笔交易主动触发从合约将资产提前解锁给发起方, 而不必等待挑战期结束.

为激励跟随方配合完成此流程, 协议要求跟随方在锁定资产时同时锁定一笔押金. 若其协助释放成功, 则该押

金自动退还; 若其不配合, 则发起方需等待挑战期结束后提取资产, 同时获取押金作为补偿.

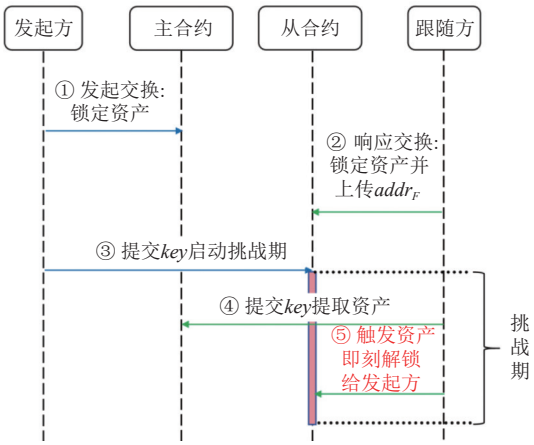


图 7 协助解锁提升协议执行效率

4 OpenSwap 完整协议流程

4.1 启 动

在协议开始前, 发起方需要在区块链 BC_1 和 BC_2 上分别部署主合约 C_1 与从合约 C_2. 表 1 中列出了合约初始化时的关键变量及其关键函数, 具体包括如下.

表 1 OpenSwap 中合约初始化时的关键变量与关键函数说明

	变量/函数	说明
主合约 C_1	pk_I	发起方公钥, 用于验证发起方签名
	r	本次交换的随机数, 用于防止签名重放
	x_1	发起方用于交换的资产金额
	Lock	用于锁定发起方资产
	Unlock	用于解锁发起方资产
从合约 C_2	pk_I	发起方公钥, 用于验证发起方签名
	r	本次交换的随机数, 用于防止签名重放
	x_2	发起方希望换得的资产金额
	y	跟随方需要锁定的押金金额
	Lock	用于锁定跟随方资产和押金
	Start_Unlock	用于开启挑战期
	Unlock2InitiatorByFollower	跟随方在挑战期内主动释放资产给发起方, 并取回押金
	Unlock2InitiatorByTimeout	发起方在挑战期结束后超时提取资产, 并获得押金
	Unlock2FollowerByFollower	跟随方举报发起方作恶行为取回资产, 并取回押金
	Unlock2FollowerByTimeout	跟随方在发起方退出交换时超时取回资产, 并取回押金

- 发起方公钥 pk_I: 用于在两条链上验证解锁凭证 (即发起方对跟随方收款地址的签名).
- 交换随机数 r: 由发起方生成, 参与签名计算, 防止签名重放攻击.
- 资产和押金金额: 发起方用于交换的资产金额 x_1, 跟随方用于交换的资产金额 x_2 和押金金额 y.
- 合约函数: 主合约中包含 Lock 和 Unlock; 从合约中包含 Lock、Start_Unlock、Unlock2InitiatorByFollower、Unlock2FollowerByTimeout 等函数, 用于控制资产的锁定与解锁.

4.2 交换发起与响应

在协议开始后, 发起方首先发起交换 (如图 8 步骤 ①): 发起方调用主合约 C_1 的 Lock 函数, 将其用于交换的

资产 x_1Coin_1 锁定至 C_1. 注意, 该锁定步骤和主合约的创建可由单笔交易完成.

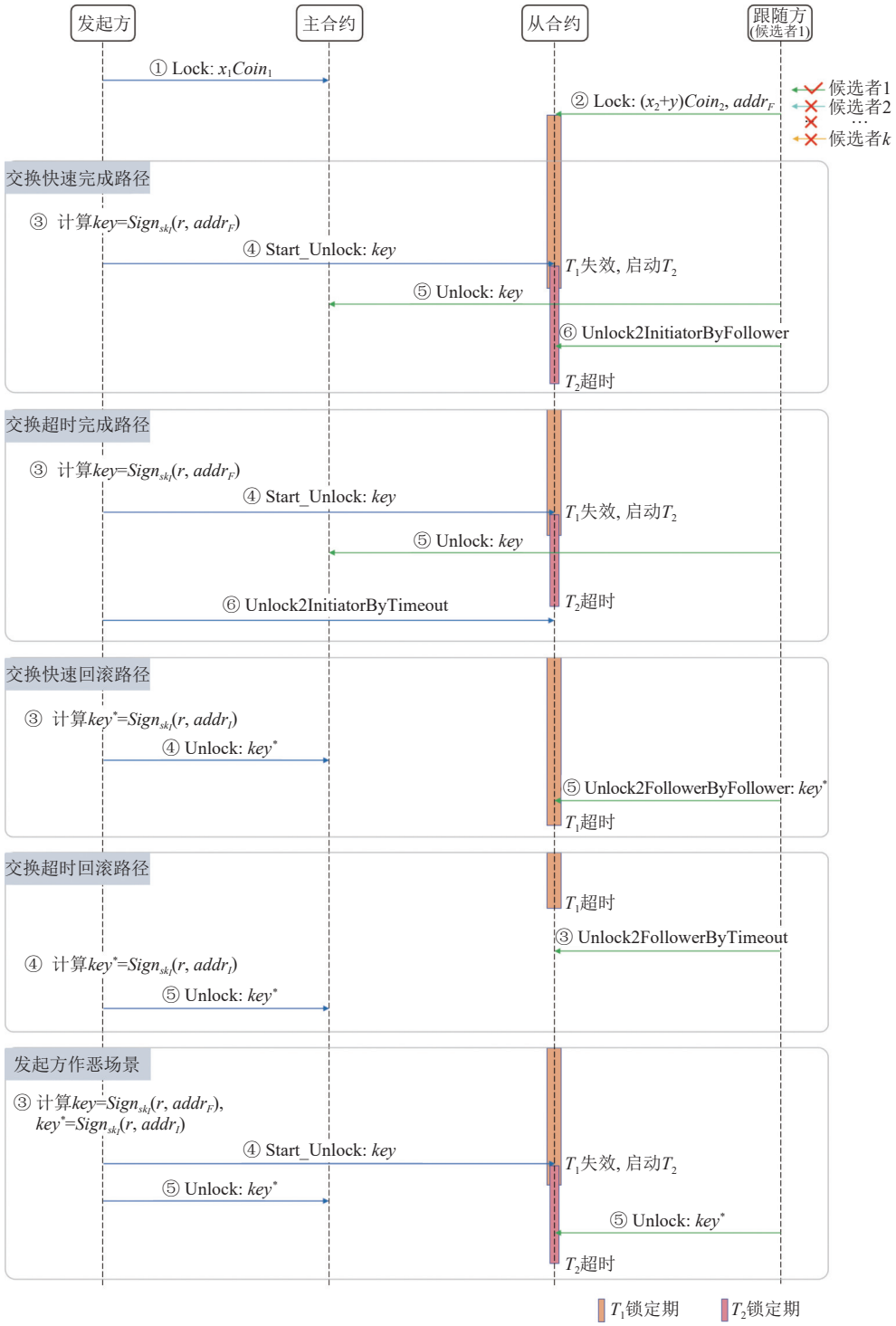


图 8 OpenSwap 完整协议流程

然后, 任何关注到该交换请求且持有足够目标资产的用户均可响应这次交换 (如图 8 步骤 ②). 响应通过调用从合约 C_2 的 Lock 函数进行. 调用方需发送 x_2+y 数量的 $Coin_2$ (其中 x_2 为用于交换的资产, y 为押金), 并提交其在链 BC_1 接收资产的地址 $addr_F$. 若多个候选者尝试响应, 最终仅有一个能成功完成调用, 成为实际的跟随方.

此外, 该调用还会启动一个时间为 T_1 的时间锁, 用于跟随方超时撤回资产. T_1 需要覆盖后续发起方在链 BC_2 调用 Start_Unlock 函数需要的时间, 因此 $T_1 \geqslant \Delta_2$, Δ_2 为链 BC_2 交易完成时间上限.

4.3 交换完成或回滚

(1) 交换完成路径

在跟随方锁定资产后, 发起方需要在时间锁 T_1 超时前完成以下步骤.

1) 计算解锁凭证 $key = Sign_{sk_I}(r, addr_F)$.

2) 调用从合约的 Start_Unlock 函数, 将 key 提交至链上 (如图 8 中交换快速完成路径步骤 ④).

发起方调用 Start_Unlock 函数时, 从合约将根据合约初始化时记录的 pk_I、r 以及跟随方锁定资产时提交的 $addr_F$ 对签名 key 的正确性进行验证. 如验证通过, 合约中的时间锁 T_1 将立即失效, 然后启动一个新的时间锁 T_2, 用于实现第 3.2 节提到的挑战期. T_2 需要覆盖后续跟随方在链 BC_1 发起解锁交易 (调用主合约 Unlock 函数) 以及在链 BC_2 发起举报交易 (调用从合约 Unlock2FollowerByFollower 函数) 的完整执行时间, 因此 $T_2 \geqslant \Delta_1 + \Delta_2$.

跟随方在观察到 key 后, 将调用主合约的 Unlock 函数并提交 key. 主合约将验证 key 是否为发起方签名, 如验证通过, 则将合约中锁定的 x_1Coin_1 释放至被签名的地址 $addr_F$.

随后, 如图 8 交换快速完成路径, 跟随方将调用从合约的 Unlock2InitiatorByFollower. 从合约验证调用方为跟随方后, 即无条件将资产 x_2Coin_2 释放给发起方, 并将 $yCoin_2$ 退还给跟随方. 至此, 双方均获得彼此资产, 交换完成.

若跟随方在获得发起方资产后未协助发起方解锁资产, 则发起方将按照图 8 中交换超时完成路径解锁资产. 具体而言, 发起方将在 T_2 超时后调用从合约的 Unlock2InitiatorByTimeout. 从合约验证在当前时间 T_2 已超时后, 即无条件将锁定的资产 x_2Coin_2 及押金 $yCoin_2$ 释放给发起方.

(2) 交换回滚路径

若发起方在跟随方锁定资产后不愿继续交换, 将签名自身地址 $addr_I$, 生成 key^*, 通过调用主合约的 Unlock 函数并提交 key^* 取回资产. 此时, 跟随方有如下两种方式取回其锁定的资产与押金.

1) 如图 8 中的快速回滚路径, 在任意时间, 跟随方均可调用从合约的 Unlock2FollowerByFollower 并提交 key^* 取回资产. 该调用执行时, 合约将对 key^* 及其签名的地址进行验证, 若验证表明 key^* 确实为发起方在此次交换中的签名, 且被签名的地址不是 $addr_F$, 合约将立即退还资产与押金给跟随方.

2) 如图 8 中的超时回滚路径, 若发起方在 T_1 超时前未公开任何签名, 跟随方也可以在 T_1 超时后调用从合约的 Unlock2FollowerByTimeout 取回资产.

(3) 发起方作恶的场景

如图 8 中发起方作恶场景, 若发起方在调用 Start_Unlock 的同时, 又另行签名自身地址生成 key^* 以试图从主合约取回资产, 跟随方将调用从合约的 Unlock2FollowerByFollower 并提交 key^*, 从而立即取回其锁定的资产和押金.

5 安全性分析

本节对 OpenSwap 进行博弈论建模, 证明协议满足原子性.

5.1 协议建模

5.1.1 参与方动作全集

定义协议的参与者集合 $P = \{I, CF_1, CF_2, \ldots, CF_n\}$, 其中 I 为发起方, $CF_1, CF_2, \ldots, CF_n$ 为 n 个候选跟随方.

发起方的动作全集为 $A_I = \{lock^{C_1}, submit^{C_2}_{key_F}, unlock^{C_1}_{key}, unlock^{C_2}_{T_2}\}$, 各动作具体的含义如下.

- $lock^{C_1}$: 锁定资产至 C_1.
- $submit^{C_2}_{key_F}$: 将地址签名锁的正确凭证 key_F, 即对 C_2 中绑定的跟随方地址的签名, 提交至 C_2.

● $unlock_{key}^{C_1}$: 将对某地址的签名 key 提交至 C_1, 触发资产解锁, 使资产释放至该地址. 该地址可能是发起方在链 BC_1 上的地址 (对应凭证表示为 key_I), C_2 中绑定的跟随方的地址 (对应凭证表示为 key_F), 或其他无关地址 (对应凭证表示为 key_O).

● $unlock_{T_2}^{C_2}$: 在 C_2 中时间锁 T_2 超时后触发 C_2 将锁定的资产和押金释放给发起方.

任一候选跟随方的动作全集为 $A_{CF}=\{lock_{addr}^{C_2}, unlock_{key_F}^{C_1}, unlock^{C_2}, unlock_{key}^{C_2}, unlock_{T_1}^{C_2}\}$, 各动作具体含义如下.

● $lock_{addr}^{C_2}$: 锁定资产与押金至 C_2, 并将 $addr$ 绑定为跟随方在链 BC_1 的收款地址. 候选跟随方上传的 $addr$ 可能是其自身在链 BC_1 上的地址, 或其他错误地址. 仅有一个候选跟随方 CF_t 能够成功完成此动作, CF_t 将成为实际跟随方. 后文用 F 或 CF_t 表示跟随方.

● $unlock_{key_F}^{C_1}$: 用发起方提交至 C_2 的 key_F 解锁 C_1 中资产.

● $unlock^{C_2}$: 触发 C_2 中资产释放给发起方, 押金返还给跟随方.

● $unlock_{key}^{C_2}$: 提交某个非绑定地址的签名 key 至 C_2, 触发资产和押金返还给跟随方. 该非绑定地址可能是发起方在链 BC_1 上的地址, 或其他地址.

● $unlock_{T_1}^{C_2}$: 在 C_2 中时间锁 T_1 超时后触发 C_2 将锁定的资产和押金返还给跟随方.

5.1.2 参与方策略和收益

发起方策略集合 $S_I=2^{A_I}$, 候选跟随方 CF_i 策略集合 $S_{CF_i}=2^{A_{CF}}$, 所有参与方策略组合的集合 $S=S_I\times S_{CF_1}\times\ldots\times S_{CF_n}$.

对于一个策略组合 $s=(s_I, s_{CF_1}, s_{CF_2},\ldots, s_{CF_n})$, 在策略组合 s 下参与者 $p\in P$ 的收益函数为 $u_p(s)$: $S\to R$, R 为实数域. $u_p(s)$ 是 s 中各动作带来的资产变动的累计结果. 我们假设在整个交换过程中交换资产的价值固定为 v, 跟随方锁定押金的价值固定为 d, 每个参与方的初始收益为 0, 当参与方锁定资产时其收益 $-v$, 锁定押金时收益 $-d$, 成功解锁资产时收益 $+v$, 成功解锁押金时收益 $+d$.

OpenSwap 定义的发起方策略为 $s_I^0=\{lock^{C_1}, submit_{key_F}^{C_2}\}$, 跟随方 CF_t 的策略为 $s_{CF_t}^0=\{lock_{addr\_CF_t}^{C_2}, unlock_{key_F}^{C_1}, unlock^{C_2}\}$, 其中, key_F 为发起方对 CF_t 在链 BC_1 的地址 $addr\_CF_t$ 签名生成的凭证. OpenSwap 定义的其他任意候选跟随方 CF_i 的策略为 $s_{CF_i}^0=\varnothing$.

在协议定义的策略 $s^0=(s_I^0, s_{CF_1}^0, s_{CF_2}^0,\ldots, s_{CF_n}^0)$ 之下, 发起方锁定资产后获得跟随方 CF_t 的资产, 跟随方 CF_t 锁定资产与押金后获得发起方的资产、取回押金, 其他候选跟随方未锁资产也不获得资产, 因此 $u_I(s^0)=u_{CF_i}(s^0)=0$, $i\in\{1,2,\ldots,n\}$.

5.1.3 偏离策略与收益

发起方偏离协议时存在以下 5 种策略.

● s_I^1: 在跟随方锁定资产和押金后无响应.

● s_I^2: 锁定资产后又自行取回.

● s_I^3: 在未解锁跟随方资产的前提下将 C_1 中锁定的资产释放至 C_2 中绑定的跟随方的地址.

● s_I^4: 在未解锁跟随方资产的前提下将 C_1 中锁定的资产释放至其他无关地址.

● s_I^5: 启动解锁阶段后又取回自己的资产.

跟随方 CF_t 偏离时存在以下 6 种策略.

● s_F^1: 在发起方启动解锁阶段后无响应.

● s_F^2: 在 C_2 中绑定错误的地址, 而后正常执行后续步骤.

● s_F^3: 在 C_2 中绑定错误的地址, 而后正常到 C_1 解锁资产, 但拒绝协助发起方解锁资产.

● s_F^4: 在 C_2 中绑定错误的地址, 而后放弃到 C_1 解锁资产, 但正常协助发起方解锁资产.

● s_F^5: 在 C_2 中绑定错误的地址, 而后放弃执行后续步骤.

● s_F^6: 在正常取得发起方资产后拒绝协助发起方解锁资产.

其他候选参与方不存在偏离策略.

当某参与方偏离协议时, 其他参与方将会调整策略以争取自己的利益最大化. 为简化第 5.2 节的原子性证明过程, 我们将各种偏离情况下的偏离方策略、另一方的应对策略以及两方最终的收益整理成表 2.

表 2　偏离情况下的偏离方策略、另一方的应对策略以及两方最终的收益

偏离策略编号	偏离策略	对方应对策略	u_I	u_F
s_I^1	$\{lock^{C_1}\}$	$\{lock^{C_2}_{addr\_CF_t}, unlock^{C_2}_{T_1}\}$	0	0
s_I^2	$\{lock^{C_1}, unlock^{C_1}_{key_I}\}$	$\{lock^{C_2}_{addr\_CF_t}, unlock^{C_2}_{key_I}\}$	0	0
s_I^3	$\{lock^{C_1}, unlock^{C_1}_{key_F}\}$	$\{lock^{C_2}_{addr\_CF_t}, unlock^{C_2}_{T_1}\}$	$-v$	v
s_I^4	$\{lock^{C_1}, unlock^{C_1}_{key_O}\}$	$\{lock^{C_2}_{addr\_CF_t}, unlock^{C_2}_{key_O}\}$	$-v$	0
s_I^5	$\{lock^{C_1}, submit^{C_2}_{key_F}, unlock^{C_1}_{key_I}\}$	$\{lock^{C_2}_{addr\_CF_t}, unlock^{C_1}_{key_F}, unlock^{C_2}_{key_I}\}$	$-v/0$	$v/0$
s_F^1	$\{lock^{C_2}_{addr\_CF_t}\}$	$\{lock^{C_1}, submit^{C_2}_{key_F}, unlock^{C_2}_{T_2}, unlock^{C_2}_{key_I}\}$	$v+d$	$-v-d$
s_F^2	$\{lock^{C_2}_{addr\_O}, unlock^{C_1}_{key_F}, unlock^{C_2}\}$	$\{lock^{C_1}, submit^{C_2}_{key_F}\}$	0	$-v$
s_F^3	$\{lock^{C_2}_{addr\_O}, unlock^{C_1}_{key_F}\}$	$\{lock^{C_1}, submit^{C_2}_{key_F}, unlock^{C_2}_{T_2}\}$	d	$-v-d$
s_F^4	$\{lock^{C_2}_{addr\_O}, unlock^{C_2}\}$	$\{lock^{C_1}, submit^{C_2}_{key_F}, unlock^{C_1}_{key_I}\}$	v	$-v$
s_F^5	$\{lock^{C_2}_{addr\_O}\}$	$\{lock^{C_1}, submit^{C_2}_{key_F}, unlock^{C_2}_{T_2}, unlock^{C_1}_{key_I}\}$	$v+d$	$-v-d$
s_F^6	$\{lock^{C_2}_{addr\_CF_t}, unlock^{C_1}_{key_F}\}$	$\{lock^{C_1}, submit^{C_2}_{key_F}, unlock^{C_2}_{T_2}\}$	d	$-d$

表 2 中包含以下 11 种情况.

情况 s_I^1: 在发起方在跟随方锁定资产和押金后无响应的情况下, C_2 中启动了时间锁 T_1, 跟随方将在 T_1 超时后执行 $unlock^{C_2}_{T_1}$ 取回资产和押金, 因此跟随方的收益为 0. 发起方虽然不响应, 但是随时可通过 $unlock^{C_1}_{key_I}$ 取回资产, 所以认为发起方的收益也为 0.

情况 s_I^2: 在发起方锁定资产后未启动解锁阶段即自行取回资产的情况, 跟随方将立即执行 $unlock^{C_2}_{key_I}$ 取回资产和押金, 所以跟随方的收益为 0. 发起方锁定资产又取回, 其收益为 0.

情况 s_I^3: 在发起方未解锁跟随方资产的前提下将 C_1 中锁定的资产释放至 C_2 中绑定的跟随方地址的情况下, 跟随方将获得发起方资产, 同时在时间锁 T_1 超时后取回自己的资产和押金, 其收益为 v. 发起方失去资产, 收益为 $-v$.

情况 s_I^4: 在发起方未解锁跟随方资产的前提下将 C_1 中锁定的资产释放至其他无关地址的情况下, 发起方失去资产, 收益为 $-v$, 跟随方将在时间锁 T_1 超时后取回己方资产和押金, 其收益为 0.

情况 s_I^5: 在发起方锁定资产后既启动解锁阶段又取回自己资产的情况下, key_F 和 key_I 均被释放. 跟随方将立即执行 $unlock^{C_2}_{key_I}$ 取回锁定于 C_2 中的资产和押金. 此外, 跟随方还将执行 $unlock^{C_1}_{key_F}$, 试图获得 C_1 中的资产. 此时, 发起方执行的 $unlock^{C_1}_{key_I}$ 将与跟随方执行的 $unlock^{C_1}_{key_F}$ 产生竞争. 如果发起方执行成功, 发起方将会取回资产, 发起方和跟随方的收益均为 0. 否则, 跟随方获得发起方的资产, 发起方收益为 $-v$, 跟随方收益为 v.

情况 s_F^1: 在跟随方在发起方启动解锁阶段后无响应的情况下, C_2 中启动了时间锁 T_2, 发起方将先在 T_2 超时后执行 $unlock^{C_2}_{T_2}$ 取得跟随方锁定的资产和押金, 而后执行 $unlock^{C_1}_{key_I}$ 取回自己锁定的资产, 因此发起方收益为 $v+d$. 跟随方失去资产和押金, 收益为 $-v-d$.

情况 s_F^2: 在跟随方绑定错误地址的情况下, 若跟随方仍正常执行交换的后续步骤, 则交换能够完成, 发起方能够获得跟随方的资产, 但发起方的资产被释放到错误的地址, 跟随方仅能取回押金. 因此, 发起方的收益为 0, 跟随方的收益为 $-v$.

情况 s_F^3: 在跟随方绑定错误地址的情况下, 若跟随方仍正常执行后续的交换步骤, 仅拒绝协助发起方解锁资产, 则交换依然可以完成. 发起方将在时间锁 T_2 超时后执行 $unlock^{C_2}_{T_2}$, 取得跟随方锁定的资产和押金, 但发起方的

资产被释放到错误的地址. 因此, 发起方的收益为 d, 跟随方的收益为 $-v-d$.

情况 s_F^4: 在跟随方绑定错误地址的情况下, 若跟随方放弃执行后续的交换步骤, 但仍然协助发起方解锁资产, 那么跟随方能够取回押金, 发起方将获得跟随方资产, 同时发起方也将通过 $unlock_{key_I}^{C_2}$ 取回自己的资产. 因此, 发起方的收益为 v, 跟随方的收益为 $-v$.

情况 s_F^5: 在跟随方绑定错误地址的情况下, 若跟随方放弃执行后续所有的交换步骤, 那么发起方将在时间锁 T_2 超时后执行 $unlock_{T_2}^{C_2}$ 取得跟随方锁定的资产和押金, 同时执行 $unlock_{key_I}^{C_2}$ 取回自己的资产. 因此, 发起方的收益为 $v+d$, 跟随方的收益为 $-v-d$.

情况 s_F^6: 在跟随方取得发起方资产后拒绝协助发起方解锁资产的情况, 发起方将在 T_2 超时后执行 $unlock_{T_2}^{C_2}$ 取得跟随方锁定的资产和押金, 因此发起方收益为 d. 跟随方锁定资产和押金后获得发起方的资产, 收益为 $-d$.

5.2 原子性证明

定理 1. 当至少存在一个候选跟随方响应时, OpenSwap 协议满足一致性.

证明: 本文从一致性的两个条件分别验证.

1) 若所有参与方均遵守协议, 交换将成功完成. 当参与方按照协议定义的策略进行交换时, 交换将按照如下步骤执行: 发起方执行 $lock^{C_1}$ 将资产锁定至 C_1. 跟随方 CF_t 执行 $lock_{addr\_CF_t}^{C_2}$ 将资产锁定至 C_2, 并绑定地址 $addr\_CF_t$. 发起方签名 $addr\_CF_t$ 形成 key_F, 并执行 $submit_{key_F}^{C_2}$ 将其提交至 C_2. 跟随方执行 $unlock_{key_F}^{C_1}$, 触发 C_1 将发起方锁定的资产释放至地址 $addr\_CF_t$, 然后执行 $unlock^{C_2}$, 触发 C_2 将跟随方锁定的资产释放给发起方. 按照上述步骤, 交换完整执行, 发起方与跟随方均获得其目标资产, 条件 1 成立.

2) 任意参与方联合偏离协议都不会使遵守协议的参与方丢失资产. 未响应或响应失败的候选跟随方不涉及资产锁定, 不会丢失资产. 若发起方偏离协议, 根据表 2 前 5 行, 跟随方的收益均不小于 0. 若跟随方偏离协议, 根据表 2 后 6 行, 发起方的收益均不小于 0. 因此, 任意参与方联合偏离协议都不会使遵守协议的参与方丢失资产, 条件 2 得证.

综上, OpenSwap 协议满足一致性.

定理 2. 策略组合 $s^0=(s_I^0, s_{CF_1}^0, s_{CF_2}^0, \ldots, s_{CF_n}^0)$ 构成一个强纳什均衡.

证明: 当参与方均遵守协议时, 所有参与方的收益均为 0. 根据表 2, 发起方偏离协议时其收益均不大于 0, 跟随方偏离协议时其收益均小于 0, 因此发起方和跟随方均没有动力偏离协议. 其他候选参与方无法偏离协议. 故定理 2 成立.

定理 3. 当至少存在一个候选跟随方进行响应时, OpenSwap 协议满足原子性.

证明: 由定理 1, 协议满足一致性. 由定理 2, 协议诱导的行为策略构成强纳什均衡. 根据原子性的定义, 若一个协议满足一致性, 且其诱导出的策略为强纳什均衡, 则它是原子的. 故定理 3 成立.

5.3 TLA+验证

本文通过 TLA+[28]实现了 OpenSwap 协议, 验证了图 9 中列出的属性. 图 9 中各属性的含义如下.

- *Correct_Completion*: 当交换双方均遵守协议时, 交换最终成功完成, 双方均获得对方资产.
- *NoLoss_INITIATOR*: 当发起方遵守协议时, 交换结束后发起方的资产总量 (包括资产和押金) 不会低于其初始资产总量.
- *NoLoss_FOLLOWER*: 当跟随方遵守协议时, 交换结束后跟随方的资产总量不会低于其初始资产总量.
- *NoDeviation_INITIATOR*: 当发起方偏离协议时, 交换结束后发起方的资产总量不会高于其初始资产总量.
- *NoDeviation_FOLLOWER*: 当跟随方偏离协议时, 交换结束后跟随方的资产总量不会高于其初始资产总量.

其中, *Correct_Completion* 和 *NoLoss_INITIATOR*、*NoLoss_FOLLOWER* 表明 OpenSwap 满足一致性. *NoDeviation_INITIATOR* 和 *NoDeviation_FOLLOWER* 表示参与方没有动力偏离协议, 即协议定义的策略构成一个强纳什均衡. 上述属性均验证通过, 进一步证明 OpenSwap 满足原子性.

$$
\begin{aligned}
&Completion \triangleq \\
&\quad \wedge\, init\_contract.state = TRANSFERED \\
&\quad \wedge\, follow\_contract.state = TRANSFERED \\
&\quad \wedge\, wallet[\text{“INITIATOR”}].balance = wallet[\text{“INITIATOR”}].init \\
&\quad \wedge\, wallet[\text{“FOLLOWER”}].balance = wallet[\text{“FOLLOWER”}].init \\
&Correct\_Completion \triangleq (AllDone \wedge AllConfirming) \Rightarrow Completion \\
&NoLoss\_INITIATOR \triangleq \\
&\quad (AllDone \wedge conforming[INITIATOR]) \\
&\quad \Rightarrow (wallet[\text{“INITIATOR”}].balance \geqslant wallet[\text{“INITIATOR”}].init) \\
&NoLoss\_FOLLOWER \triangleq \\
&\quad (AllDone \wedge conforming[FOLLOWER]) \\
&\quad \Rightarrow (wallet[\text{“FOLLOWER”}].balance \geqslant wallet[\text{“FOLLOWER”}].init) \\
&NoDeviation\_INITIATOR \triangleq \\
&\quad (AllDone \wedge \neg conforming[INITIATOR] \wedge conforming[FOLLOWER]) \\
&\quad \Rightarrow (wallet[\text{“INITIATOR”}].balance \leqslant wallet[\text{“INITIATOR”}].init) \\
&NoDeviation\_FOLLOWER \triangleq \\
&\quad (AllDone \wedge \neg conforming[FOLLOWER] \wedge conforming[INITIATOR]) \\
&\quad \Rightarrow (wallet[\text{“FOLLOWER”}].balance \leqslant wallet[\text{“FOLLOWER”}].init)
\end{aligned}
$$

图 9　OpenSwap 通过验证的属性

6　性能评估

本节通过性能分析和实验验证评估 OpenSwap 在不同场景下的性能表现和链上开销.

6.1　理论性能分析

假设系统中共 N 个符合成为跟随方条件的用户 (即候选方), 每个候选方响应的概率为 p. 若 N 个候选方均不响应, 则无论 OpenSwap 还是 HTLC 类协议均无法完成交换. 由于本文关注的是协议完成一个交换的时间, 因此在本节分析中假设 N 足够大, 能够确保至少存在一个候选方响应交换, 使两协议最终均能完成交换.

尽管 OpenSwap 一次尝试即可完成交换, 其完成交换的时间及开销均与 p 无关, 但 HTLC 类协议的尝试次数 k 受 p 影响, 进而影响完成一个交换的时间与开销. 因此, 本节通过分析 HTLC 类协议完成交换的时间及链上开销与 k 及 p 的关系, 评估 OpenSwap 相较 HTLC 类协议在执行时间和链上开销两方面的表现.

6.1.1　执行时间

(1) 与尝试次数 k 的关系

在系统模型定义中, 将区块链系统中从用户发起一笔交易到该交易被打包进区块并被系统中其他用户看到所需要的时间称作交易完成时间. 假设在链 BC_1 中交易完成时间的平均值为 t_1, 最大值为 Δ_1, 在 BC_2 中平均值为 t_2, 最大值为 Δ_2. 根据现有经验和数据[20,21,44]可知, $t_1 \ll \Delta_1$, $t_2 \ll \Delta_2$.

对于 HTLC 类协议, 串行尝试 k 次完成一个交换的时间为:

$$T_{\text{htlc}}(k) = (k-1)T_{\text{fail}} + T_{\text{success}} \tag{2}$$

其中, T_{fail} 为单次失败尝试需要的执行时间, T_{success} 为单次成功尝试需要的执行时间. 不同协议之间的差异主要体现在部分协议能够提高单次尝试的用户响应概率 p (如通过惩罚机制抑制退出或并行地与多个候选方尝试交换), 从而减少尝试次数 k.

对于每一次尝试的执行时间, HTLC 协议的优化方案一般不低于 HTLC 协议. 因此本节以 HTLC 协议为例来分析单次尝试时间 T_{fail} 与 T_{success}. 假设每次交换开始前发起方与跟随方链下协商的时间为 γ. 当一次交换成功执行时, 其过程为: 双方链下协商, 发起方在链 BC_1 创建合约并锁定资产, 跟随方在链 BC_2 创建合约并锁定资产, 发起方在链 BC_2 解锁资产, 跟随方在链 BC_1 解锁资产. 因此, 执行时间 $T_{\text{success}} = \gamma + 2t_1 + 2t_2$. 当一次交换由于跟随方退出而执行失败时, 其过程为: 双方链下协商, 发起方创建合约并锁定资产, 链 BC_1 时间锁超时后发起方取回资产. 因此, 执行时间 $T_{\text{fail}} = \gamma + \Delta_1 + 2\Delta_2 + 2t_1$. 因此, HTLC 协议完成一个交换的时间为:

$$T^*_{\text{htlc}}(k) = (k-1)(\gamma + \Delta_1 + 2\Delta_2 + 2t_1) + (\gamma + 2t_1 + 2t_2) \tag{3}$$

对于 OpenSwap, 在多数情况下, 跟随方愿意进行协助解锁, 交换通过快速完成路径完成, 具体过程如下: 发起

方在链 BC_1 创建合约并锁定资产、同时在链 BC_2 创建合约, 跟随方在链 BC_2 锁定资产, 发起方在链 BC_2 启动挑战期, 跟随方在链 BC_1 解锁资产, 跟随方在链 BC_2 协助解锁资产. 因此, 执行时间为:

$$T_{\text{fast}} = \max(t_1, t_2) + t_1 + 3t_2 \tag{4}$$

在少数情况下, 跟随方拒绝协助解锁, 交换通过超时完成路径完成, 具体完成过程为: 发起方在链 BC_1 创建合约并锁定资产、同时在链 BC_2 创建合约, 跟随方在链 BC_2 锁定资产, 发起方在链 BC_2 启动挑战期, 跟随方在链 BC_1 解锁资产, 发起方等待挑战期结束后在链 BC_2 解锁资产. 因此, 执行时间为:

$$T_{\text{timeout}} = \max(t_1, t_2) + \Delta_1 + \Delta_2 + 3t_2 \tag{5}$$

根据 T_{htlc}、T_{fast} 和 T_{timeout} 可知, HTLC 类协议的执行时间是随 k 的增加线性增长的, 而 OpenSwap 的执行时间与 k 无关. 以 HTLC 协议为例, 与 OpenSwap 比较如下.

1) 当 $k = 1$ 时, HTLC 协议的执行时间为 $\gamma + 2t_1 + 2t_2$, OpenSwap 在多数情况下执行时间为 $\max(t_1, t_2) + t_1 + 3t_2$, 少数情况下执行时间为 $\max(t_1, t_2) + \Delta_1 + \Delta_2 + 3t_2$, 在 OpenSwap 中跟随方拒绝协助解锁或 $\gamma < \max(t_1, t_2) + t_2 - t_1$ 时, HTLC 协议较快, 否则 OpenSwap 较快.

2) 当 $k > 1$ 时, HTLC 协议的执行时间在 $(k-1)(\Delta_1 + 2\Delta_2)$ 量级, 而 OpenSwap 的执行时间不变, OpenSwap 始终快于 HTLC 协议, 在跟随方协助解锁的情况下快约 $(k-1)(\Delta_1 + 2\Delta_2)$, 在跟随方拒绝协助解锁的情况下快约 $(k-2)\Delta_1 + (2k-3)\Delta_2$.

(2) 与响应概率 p 的关系

HTLC 类协议每次尝试交换成功的概率是 p, 成功即止, 失败则重试, 直至成功. 据此可知, 当 N 足够大时, 成功时的尝试次数 k 服从几何分布, 其数学期望 $E(k) = \dfrac{1}{p}$. 进而可算得完成交换的执行时间的数学期望:

$$E(T_{\text{htlc}}) = \left(\frac{1}{p} - 1\right) T_{\text{fail}} + T_{\text{success}} \tag{6}$$

由此可见, HTLC 类协议的期望执行时间与响应概率 p 为反比例关系. p 越小, 执行时间越长, p 越大, 执行时间越短. 由于 OpenSwap 一次即执行成功, 所以其完成交换的执行时间的数学期望恒为单次执行时间.

由于同一用户空间中采用不同协议时用户的响应概率 p 是不同的, 所以无法分析比较不同协议的执行时间. 但 HTLC 协议与 OpenSwap 协议均无提升用户响应概率的措施, 因此相同用户空间中用户的响应概率是相同的. 因此, 我们选取几个特殊的 p 值对两协议进行比较.

1) $p \to 0$ 时, $E(T_{\text{htlc}}) \to \infty$.这种情况下 OpenSwap 执行时间远小于 HTLC 协议.

2) $p = 0.5$ 时, $E(T_{\text{htlc}}) = T_{\text{fail}} + T_{\text{success}}$. OpenSwap 协议通过快速完成路径执行时的执行时间, 与之相比约快了 $\Delta_1 + 2\Delta_2$, 通过超时完成路径执行时的执行时间与之相比约快了 Δ_2.

3) $p = 1$ 时, $E(T_{\text{htlc}}) = T_{\text{success}}$. OpenSwap 协议通过快速完成路径执行时的执行时间与之近似.

6.1.2 链上开销

(1) 与尝试次数 k 的关系

HTLC 类协议串行尝试 k 次完成一个交换的链上开销为:

$$C_{\text{htlc}}(k) = (k-1)C_{\text{fail}} + C_{\text{success}} \tag{7}$$

其中, C_{fail} 为单次失败尝试需要的链上开销, C_{success} 为单次成功尝试需要的链上开销. 同样地, 不同协议之间的差异主要体现在部分协议能够提高单次响应成功概率 p, 从而减少尝试次数 k. 对于每一次尝试的链上开销, HTLC 协议的优化方案一般不低于 HTLC 协议. 因此本节同样以 HTLC 协议为例来分析单次尝试开销 C_{fail} 与 C_{success}.

假设 HTLC 协议和 OpenSwap 协议的单合约单次部署开销分别为 c^D_{htlc}、c^D_{OS}, 单笔锁定或解锁交易的平均开销分别为 $c^{\text{TX}}_{\text{htlc}}$、$c^{\text{TX}}_{\text{OS}}$. 由于 OpenSwap 协议的合约逻辑较为复杂, 有 $c^D_{\text{htlc}} < c^D_{\text{OS}}$. 由于 OpenSwap 协议的解锁交易涉及数字签名的验证, 这部分计算增加了交易开销, 因此有 $c^{\text{TX}}_{\text{htlc}} < c^{\text{TX}}_{\text{OS}}$.

对于 HTLC 协议, 当一次交换成功执行时, 链上开销 $C_{\text{success}} = 2c^D_{\text{htlc}} + 4c^{\text{TX}}_{\text{htlc}}$; 当一次交换由于跟随方退出而执行失败时, 链上开销 $C_{\text{fail}} = c^D_{\text{htlc}} + 2c^{\text{TX}}_{\text{htlc}}$. 串行尝试 k 次完成一个交换需要付出的链上开销为:

$$C^{*}_{\text{htlc}} = (k+1)c^{D}_{\text{htlc}} + 2(k+1)c^{\text{TX}}_{\text{htlc}} \tag{8}$$

对于 OpenSwap, 完成交换需要的链上开销为:

$$C_{\text{OS}} = 2c^{D}_{\text{OS}} + 5c^{\text{TX}}_{\text{OS}} \tag{9}$$

根据 C_{htlc}、C_{OS}, HTLC 协议完成一个交换需要付出的链上开销随 k 增加, 而 OpenSwap 完成一个交换需要付出的链上开销为常量. 由于 OpenSwap 单个交易执行开销较大, 当 k 较小时 OpenSwap 的链上开销较大, 但随着 k 增大, OpenSwap 的链上开销终将小于 HTLC 协议.

(2) 与响应概率 p 的关系

与执行时间的数学期望相同的计算方式可得, HTLC 类协议完成一个交换的交易开销的数学期望为:

$$E(C_{\text{htlc}}) = \left(\frac{1}{p} - 1\right)C_{\text{fail}} + C_{\text{success}} \tag{10}$$

OpenSwap 完成一次交换的交易开销的数学期望恒为单次执行开销. 由于两者单次执行开销难以进行比较, 因此此处不再进行分析.

6.2 实验验证

为了运行协议进行实验, 本文首先通过 Solidity 实现了 OpenSwap 主从合约和 HTLC 合约; 然后在内核为 Intel(R) Xeon(R) Gold 6230, 内存为 256 GB 的服务器上使用 Geth 运行了 2 条出块间隔为 5 s 的以太坊私有链; 最后用 Python 分别实现了 OpenSwap 的发起方、跟随方客户端, 以及 HTLC 协议的发起方、跟随方客户端, 每个客户端都可以连接两条链, 依据协议逻辑在两条链上读取信息, 发布合约部署、资产锁定与解锁等交易.

针对执行时间, 本文测试了 OpenSwap 成功完成一次交换的执行时间, 并与不同尝试次数 k 与候选参与方的响应概率 p 下 HTLC 协议完成一次交换的执行时间进行了对比. 针对链上开销, 本文测量了 OpenSwap 各函数的执行时需要消耗的链上 Gas, 计算了 OpenSwap 的执行总开销, 并与 HTLC 协议进行了对比.

值得注意的是, 现有 HTLC 优化方案与 HTLC 协议同属于先绑后锁模式, 都存在参与方退出交换失败的问题, 完成一个交换也都需要经历多次重试的过程, 因此总执行时间 (或总链上开销) 都可视作 $k-1$ 次失败尝试的执行时间 (或链上开销) 与 1 次成功尝试的执行时间 (或链上开销) 的加和. HTLC 协议的优化方案每一次尝试的执行时间和链上开销一般不低于 HTLC 协议, 因此 HTLC 协议单次尝试的性能表现可以代表 HTLC 类协议单次尝试的最好情况. 此外, HTLC 类方案不同协议的主要差异体现在部分协议能够提高单次尝试的用户响应概率 p, 从而减少尝试次数 k. 基于这一观察, 本文在实验验证中给出的 HTLC 协议在不同响应概率和尝试次数下的时间与开销结果, 已经完整覆盖了所有 HTLC 类协议的性能表现. 因此, 文中没有额外选取其他 HTLC 类协议与 OpenSwap 进行比较.

6.2.1 执行时间

针对两种协议, 实验均设置 1 个发起方和 10 个候选跟随方, 具体实验内容如下.

1) 运行 OpenSwap 协议, 使交换通过快速完成路径执行, 测量执行时间.

2) 运行 OpenSwap 协议, 使交换通过超时完成路径执行, 测量执行时间.

3) 在候选方的响应概率为 0.3–1.0 时, 每个概率下运行 HTLC 协议 100 次, 统计如下.

- 每个概率下 HTLC 协议的平均执行时间.
- 记录每次运行时为完成一个交换发起方的尝试次数, 统计每个尝试次数下 HTLC 协议的运行时间.

在实验过程中, 假设交换双方持续在线且不延迟执行, 具体而言: HTLC 协议的链下协商过程中一方收到另一方消息后会立即进行响应, OpenSwap 和 HTLC 协议链上执行过程中一方看到前一步骤的交易完成后, 会立即发起下一步骤的交易. 两协议中时间锁的值根据需要覆盖的交易数目来计算. 参考现有项目[20,21], 每覆盖一个交易, 时间锁的值增加 144 个出块间隔.

不同响应概率下两种协议的执行时间如图 10. 根据图 10, HTLC 协议的执行时间随响应概率的变化符合公式 (6) 描述的反比例关系. 当响应概率较低 (如 0.3) 时, HTLC 协议往往需要多轮尝试才能成功完成一次交换, 导致其平

均执行时间高达 4 000 s 以上. 随着响应概率提升至接近 1.0, HTLC 协议的执行时间才逐渐下降至与 OpenSwap 相近的水平. 根据部分链上数据[55]以及现有一些建模分析工作[22,40,53], HTLC 协议的成功率往往小于 0.8, 因此, 多数情况下 HTLC 协议的执行时间都远高于 OpenSwap. 相比之下, OpenSwap 协议在两种执行路径下均表现出稳定且显著更优的执行效率.

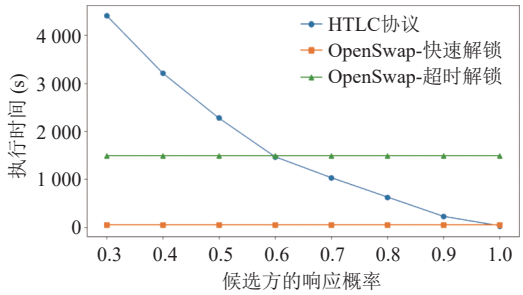


图 10 不同响应概率下 OpenSwap 和 HTLC 协议完成一个交换的平均执行时间比较

● “OpenSwap-快速解锁”表示交换通过快速完成路径执行的时间, 多数情况下协议通过该路径执行. 该执行路径下 OpenSwap 完成一个交换的时间始终保持在约 44 s, 远低于 HTLC 协议.

● “OpenSwap-超时解锁”表示交换通过超时完成路径执行的时间, 仅少数情况下协议通过该路径执行. 尽管该路径执行时间略高, 但也仅约为 1 485 s, 明显优于 HTLC 协议在低响应概率下的表现.

不同尝试次数完成交换时两协议的平均执行时间如图 11. 根据图 11, HTLC 协议的执行时间随尝试次数的变化符合公式 (3) 描述的线性增长关系. 原因在于, 每增加一次失败尝试, 协议都需要重复经历一次完整的失败执行流程, 从而使总执行时间按尝试次数累积增长. 需要指出的是, 每次失败尝试的执行时间主要由 3 部分构成: 链下的交换磋商时间、用户在链上发布交易的交易完成时间、时间锁的持续时长. 其中, 链下磋商时间与链上交易确认时间虽然是变量, 但其取值通常较小. 相比之下, 时间锁的持续时间由协议参数设定, 数值远大于前两部分且在整个交换过程中保持不变. 因此, 每新增一次失败尝试所带来的额外时间开销近似等于时间锁的持续时间, 从而在图中表现为随尝试次数等幅增加, 但实际增幅是存在微小波动的. 在这种线性增长趋势下, 当尝试次数较多时, HTLC 协议的总执行时间迅速攀升, 达到近 20 000 s, 严重影响实际使用效率. 相比之下, OpenSwap 协议的执行时间始终保持稳定.

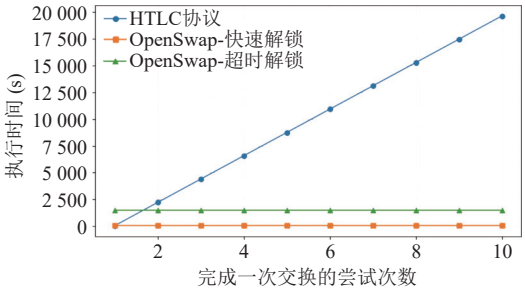


图 11 不同尝试次数完成交换时 HTLC 协议和 OpenSwap 协议的平均执行时间比较

从另一个角度来看图 10 和图 11 的结果, HTLC 类协议只有将用户的响应概率提升至接近 1.0, 将尝试次数降低至 1, 其执行时间才能达到与 OpenSwap 相近的水平.

6.2.2 链上开销

验证开销受数字签名算法的影响. 不同区块链支持的不同签名方法不同, 因此协议的开销也不同. 本文实验采用以太坊测试链, 其采用的数字签名算法是 ECDSA 算法.

本文统计了 OpenSwap 和 HTLC 协议各合约函数的 Gas 开销, 如表 3 和表 4 所示. 注意, 尽管 OpenSwap 和

HTLC 协议中某些合约部署 (Constructor) 和锁定 (Lock) 操作可由单笔交易完成, 但是本文还是分开统计了两种操作的开销. 此外, 表 3 中标记了 OpenSwap 协议不同执行路径需要调用的合约函数, 表 4 中标记了 HTLC 协议不同执行路径需要调用的合约函数. 对于每种执行路径, 表中√表示该执行路径下需要调用该函数, 留空表示该执行路径不涉及该函数. 通过将每种执行路径所调用函数的 Gas 消耗加和, 可以得到每种执行路径单次执行的 Gas 开销.

表 3　OpenSwap 各合约函数执行的 Gas 开销及不同执行路径涉及的函数调用

合约类型	函数	Gas消耗	快速完成路径	超时完成路径
主合约	Constructor	585 453	√	√
	Lock	62 950	√	√
	Unlock	48 862	√	√
从合约	Constructor	1 177 946	√	√
	Lock	85 854	√	√
	Start_Unlock	60 463	√	√
	Unlock2InitiatorByFollower	38 553	√	—
	Unlock2InitiatorByTimeout	30 920	—	√
	Unlock2FollowerByFollower	42 821	—	—
	Unlock2FollowerByTimeout	30 884	—	—

表 4　HTLC 协议各合约函数执行的 Gas 开销及不同执行路径涉及的函数调用

合约类型	函数	Gas消耗	交换完成路径	交换回滚路径
主合约	Constructor	356 748	√	√
	Lock	62 944	√	√
	Hash_Unlock	36 779	√	—
	Time_Unlock	35 766	—	√
从合约	Constructor	356 748	√	—
	Lock	62 944	√	—
	Hash_Unlock	36 779	√	—
	Time_Unlock	35 766	—	—

根据表 3, OpenSwap 单次执行的 Gas 开销为 2 060 081 (快速完成路径) 和 2 052 448 (超时完成路径). 在两种路径下, 协议均单次执行即可完成交换, 因此, 完成交换的链上总开销即为协议单次执行的 Gas 开销. 具体而言, OpenSwap 在协助解锁情况下交换的链上总开销为 2 060 081, 在超时解锁情况下交换的链上总开销为 2 052 448.

HTLC 协议完成一个交换需要 k 次尝试, 涉及 $k-1$ 次交换回滚路径的执行, 1 次交换完成路径的执行, 根据表 4, HTLC 协议单次成功执行的 Gas 开销为 912 942 (交换完成路径), 单次失败执行 (因跟随方退出) 的 Gas 开销为 455 458 (交换回滚路径). 因此, HTLC 协议完成一个交换的链上总开销为 $(k-1)\times 455458+912942$.

将两种方案的链上总开销进行比较可知, 当 $k\leqslant 3$ 时, HTLC 协议的 Gas 开销较小; 当 $k>3$ 时, OpenSwap 的 Gas 开销较小.

根据公式 (8), HTLC 协议完成一个交换的交易开销的数学期望为 $912942+\dfrac{455458(1-p)}{p}$. 据此可知, 当 $p\leqslant 0.28$ 时, OpenSwap 的 Gas 开销较小; 当 $p>0.28$ 时, HTLC 协议的 Gas 开销较小.

就两协议的 Gas 消耗差额而言, 在 p 较大, k 较小时, HTLC 协议开销较小, 但其最小值也仅比 OpenSwap 省约 55.7% 的 Gas. 当 p 较小时, 随着 k 值的增大, HTLC 协议比 OpenSwap 多消耗的 Gas 是无上限的.

6.3　评估结果总结

根据性能分析和实验结果, OpenSwap 能够避免多次重试, 保持稳定的交换完成时间和链上开销. 在绝大多数情况下, OpenSwap 相比 HTLC 协议的交换完成时间均有显著降低. 在候选方响应概率较低的情况下, OpenSwap 相比 HTLC 协议的链上开销也有所降低. 仅在候选方响应概率较高的情况下, OpenSwap 相比 HTLC 协议的链上开销有幅度有限的增加.

7 讨 论

7.1 协议适用性讨论

OpenSwap 的适用场景与原始 HTLC 协议是一致的, 主要面向在无可信第三方、无跨链基础设施条件下的两方跨链资产交换场景. 与 HTLC 协议相同, OpenSwap 的协议运行仅依赖两条区块链自身的脚本或智能合约能力, 并要求两条链支持相同的哈希函数. 与 HTLC 协议不同的是, OpenSwap 额外要求两条链支持一致的签名算法, 以便验证参与方提交的地址签名锁的解锁凭证. 该要求在当前主流区块链中普遍满足, 即使对于未原生提供一致签名算法的区块链, 也可通过智能合约实现对应签名验证逻辑, 因此不会对协议的部署范围造成实质性限制. 需要强调的是, OpenSwap 协议在当前形式下是一种面向两方交换的基础协议, 定位上与原始 HTLC 协议相当. 正如 HTLC 协议在提出之后逐步被扩展到多方交换[7,36]、跨链拍卖[14]等更复杂的跨链协同应用, OpenSwap 同样具备向更复杂场景演化的潜力, 可作为更复杂跨链应用的构建基础.

7.2 资产流动性损失讨论

本文以资产数量的变化来衡量参与者的损失, 以理性参与方在交换中不丢失资产作为安全目标之一, 这与现有跨链原子交换研究的通行做法[7]一致. 然而, 需要指出的是, 在跨链交换的实际场景中, 除了资产数量本身, 不同程度的时间消耗也可能带来资产流动性下降、错失交易机会等间接损失, 因此时间成本在经济意义上也可被视为一种“弱损失”. 本文提出的先锁后绑模式和 OpenSwap 协议, 可以从机制层面缓解因跟随方退出带来的资产长期锁定问题, 从而减少时间成本造成的流动性损失. 本文在性能分析中对执行时间的理论推导, 以及在实验部分对执行时延的测量, 均能体现 OpenSwap 在降低流动性损失方面的实际效果.

8 结 论

本文针对传统 HTLC 协议的执行时间长、链上开销大等问题, 提出了一种支持开放响应的跨链资产交换协议 OpenSwap. 该协议突破了现有先绑后锁模式, 允许任意符合条件的用户响应交换请求, 待响应成功后再将身份信息绑定至链上, 从而有效避免因预先绑定的对手退出而导致的资产长期锁定与重复执行开销. 通过协议分析与形式化验证, 我们证明 OpenSwap 能够保障交换的原子性. 同时, 性能分析与实验结果表明, 在多次重试场景下, OpenSwap 的平均执行时间和链上交易成本显著优于现有 HTLC 方案. 在候选用户响应概率较低的场景中, OpenSwap 相较于 HTLC 协议优势更加突出.

OpenSwap 拓展了跨链资产交换的设计空间, 为无需预匹配、动态参与的开放式交换提供了基础框架, 具有良好的通用性与实用价值. 该协议为降低用户门槛、提升多链交互效率提供了新的设计范式. 后续工作可从以下两个方面进一步拓展: 一是在先锁后绑模式下探索新的开放式交换实现方法, 进一步降低交易执行开销; 二是扩展模型以支持多方、多资产等更复杂的交换场景.

References

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf

[2] Invest like an icon. 2025. https://www.blockchain.com/

[3] The New York Times. Apparent theft at Mt.Gox shakes bitcoin world. 2014. https://www.nytimes.com/2014/02/25/business/apparent-theft-at-mt-gox-shakes-bitcoin-world.html

[4] The New York Times. Binance blockchain hit by $570 million hack. 2022. https://www.nytimes.com/2022/10/07/business/binance-hack.html

[5] BBC NEWS. BitMart: Crypto-exchange loses $150m to hackers. 2021. https://www.bbc.co.uk/news/technology-59549606

[6] TierNolan. 2013. Alt chains and atomic transfers. https://bitcointalk.org/index.php?topic=193281.0

[7] Herlihy M. Atomic cross-chain swaps. In: Proc. of the 2018 ACM Symp. on Principles of Distributed Computing. Egham: ACM, 2018. 245–254. [doi: 10.1145/3212734.3212736]

[8] Buterin V. Chain interoperability. 2016. https://r3.com/wp-content/uploads/2016/09/Chain_Interoperability_R3.pdf

[9] 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]

[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] Chainlink Foundation. Chainlink CCIP—Cross-chain interoperability protocol. 2025. https://docs.chain.link/ccip

[12] Powering the interchain. 2025. https://ibcprotocol.dev/

[13] Zamyatin A, Harz D, Lind J, Panayiotou P, Gervais A, Knottenbelt W. XCLAIM: Trustless, interoperable, cryptocurrency-backed assets. In: Proc. of the 2019 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2019. 193–210. [doi: 10.1109/SP.2019.00085]

[14] Herlihy M, Liskov B, Shrira L. Cross-chain deals and adversarial commerce. Proc. of the VLDB Endowment, 2019, 13(2): 100–113. [doi: 10.14778/3364324.3364326]

[15] Thyagarajan SAK, Malavolta G, Moreno-Sanchez P. Universal atomic swaps: Secure exchange of coins across all blockchains. In: Proc. of the 2022 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2022. 1299–1316. [doi: 10.1109/SP46214.2022.9833731]

[16] WeBankBlockchain. WeCross: The blockchain interoperability platform white paper. 2020 (in Chinese). https://pdf.dfcfw.com/pdf/H3_AP202003101376070183_1.pdf

[17] Kava. Leading the world to Web3. 2025. https://www.kava.io/

[18] Komodo. Join the trustless (d)experience. 2025. https://komodoplatform.com/en/

[19] Decred—Money evolved. 2025. https://decred.org/

[20] Core lightning—Lightning network implementation focusing on spec compliance and performance. 2024. https://github.com/Elements-Project/lightning

[21] Lightning network daemon. 2022. https://github.com/lightningnetwork/lnd

[22] Han RC, Lin HY, Yu JS. On the optionality and fairness of atomic swaps. In: Proc. of the 1st ACM Conf. on Advances in Financial Technologies. Zurich: ACM, 2019. 62–75. [doi: 10.1145/3318041.3355460]

[23] Xue YJ, Herlihy M. Hedging against sore loser attacks in cross-chain transactions. In: Proc. of the 2021 ACM Symp. on Principles of Distributed Computing. ACM, 2021. 155–164. [doi: 10.1145/3465084.3467904]

[24] Heilman E, Lipmann S, Goldberg S. The arwen trading protocols. In: Proc. of the 24th Int'l Conf. on Financial Cryptography and Data Security. Kota Kinabalu: Springer, 2020. 156–173. [doi: 10.1007/978-3-030-51280-4_10]

[25] Mazumdar S. Towards faster settlement in HTLC-based cross-chain atomic swaps. In: Proc. of the 4th IEEE Int'l Conf. on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA). Atlanta: IEEE, 2022. 295–304. [doi: 10.1109/TPS-ISA56441.2022.00043]

[26] Nadahalli T, Khabbazian M, Wattenhofer R. Grief-free atomic swaps. In: Proc. of the 2022 IEEE Int'l Conf. on Blockchain and Cryptocurrency. Shanghai: IEEE, 2022. 1–9. [doi: 10.1109/ICBC54727.2022.9805490]

[27] Xue YJ, Jin D, Herlihy M. Invited paper: Fault-tolerant and expressive cross-chain swaps. In: Proc. of the 24th Int'l Conf. on Distributed Computing and Networking. Kharagpur: ACM, 2023. 28–37. [doi: 10.1145/3571306.3571388]

[28] Lamport L. My TLA+ home page. 2025. https://lamport.azurewebsites.net/tla/tla.html

[29] Binance. Binance: Your UK gBinance: The world's most trusted cryptocurrency exchange to buy, trade & invest in cryptoateway into Web3. 2025. https://www.binance.com

[30] Hoenisch P, Mazumdar S, Moreno-Sanchez P, Ruj S. LightSwap: An atomic swap does not require timeouts at both blockchains. In: Proc. of the 2023 Data Privacy Management, Cryptocurrencies and Blockchain Technology. Copenhagen: Springer, 2023. 219–235. [doi: 10.1007/978-3-031-25734-6_14]

[31] Zie JY, Deneuville JC, Briffaut J, Nguyen B. Extending atomic cross-chain swaps. In: Proc. of the 2019 Data Privacy Management, Cryptocurrencies and Blockchain Technology. Luxembourg: Springer, 2019. 219–229. [doi: 10.1007/978-3-030-31500-9_14]

[32] Manevich Y, Akavia A. Cross chain atomic swaps in the absence of time via attribute verifiable timed commitments. In: Proc. of the 7th IEEE European Symp. on Security and Privacy. Genoa: IEEE, 2022. 606–625. [doi: 10.1109/EUROSP53844.2022.00044]

[33] Shlomovits O, Leiba O. JugglingSwap: Scriptless atomic cross-chain swaps. arXiv:2007.14423, 2020.

[34] Shadab N, Houshmand F, Lesani M. Cross-chain transactions. In: Proc. of the 2020 IEEE Int'l Conf. on Blockchain and Cryptocurrency. Toronto: IEEE, 2020. 1–9. [doi: 10.1109/ICBC48266.2020.9169477]

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

[36] Clark E, Georgiou C, Poon K, Chrobak M. On HTLC-based protocols for multi-party cross-chain swaps. In: Proc. of the 35th Int'l Symp.

on Algorithms and Computation. Dagstuhl: Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2024. 22: 1–22: 15. [doi: 10.4230/LIPICS.ISAAC.2024.22]

[37] Chan E, Chrobak M, Lesani M. Cross-chain swaps with preferences. In: Proc. of the 36th IEEE Computer Security Foundations Symp. Dubrovnik: IEEE, 2023. 261–275. [doi: 10.1109/CSF57540.2023.00031]

[38] Rueegger J, Machado GS. Rational exchange: Incentives in atomic cross chain swaps. In: Proc. of the 2020 IEEE Int'l Conf. on Blockchain and Cryptocurrency. Toronto: IEEE, 2020. 1–3. [doi: 10.1109/ICBC48266.2020.9169408]

[39] Belotti M, Moretti S, Potop-Butucaru M, Secci S. Game theoretical analysis of cross-chain swaps. In: Proc. of the 40th IEEE Int'l Conf. on Distributed Computing Systems. Singapore: IEEE, 2020. 485–495. [doi: 10.1109/ICDCS47774.2020.00060]

[40] Ladóczki B, Bíró J, Tapolcai J. Stochastic analysis of the success rate in atomic swaps between blockchains. In: Proc. of the 4th Int'l Conf. on Blockchain Computing and Applications. San Antonio: IEEE, 2022. 41–46. [doi: 10.1109/BCCA55292.2022.9922365]

[41] ZmnSCPxj. [Lightning-dev] an argument for single-asset lightning network. 2018. https://diyhpl.us/~bryan/irc/bitcoin/bitcoin-dev/linuxfoundation-pipermail/lightning-dev/2018-December/001752.txt

[42] Zhuo F, Song ZX, Jia LP, Zhang HW, Li ZC, Sun Y. Enabling fast settlement in atomic cross-chain swaps. In: Proc. of the 19th EAI Int'l Conf. on Security and Privacy in Communication Networks. Hong Kong: Springer, 2025. 395–412. [doi: 10.1007/978-3-031-64948-6_20]

[43] Winzer F, Herd B, Faust S. Temporary censorship attacks in the presence of rational miners. In: Proc. of the 2019 IEEE European Symp. on Security and Privacy Workshops. Stockholm: IEEE, 2019. 357–366. [doi: 10.1109/EUROSPW.2019.00046]

[44] Nadahalli T, Khabbazian M, Wattenhofer R. Timelocked bribing. In: Proc. of the 25th Int'l Conf. on Financial Cryptography and Data Security. Springer, 2021. 53–72. [doi: 10.1007/978-3-662-64322-8_3]

[45] Tsabary I, Yechieli M, Manuskin A, Eyal I. MAD-HTLC: Because HTLC is crazy-cheap to attack. In: Proc. of the 2021 IEEE Symp. on Security and Privacy. San Francisco: IEEE, 2021. 1230–1248. [doi: 10.1109/SP40001.2021.00080]

[46] Wadhwa S, Stoeter J, Zhang F, Nayak K. He-HTLC: Revisiting incentives in HTLC. In: Proc. of the 2023 Network and Distributed System Security Symp. San Diego: The Internet Society, 2023. [doi: 10.14722/ndss.2023.24775]

[47] Deshpande A, Herlihy M. Privacy-preserving cross-chain atomic swaps. In: Proc. of the 2020 Financial Cryptography and Data Security Int'l Workshops. Kota Kinabalu: Springer, 2020. 540–549. [doi: 10.1007/978-3-030-54455-3_38]

[48] Cao L, Wan ZY. Anonymous scheme for blockchain atomic swap based on zero-knowledge proof. In: Proc. of the 2020 IEEE Int'l Conf. on Artificial Intelligence and Computer Applications (ICAICA). Dalian: IEEE, 2020. 371–374. [doi: 10.1109/ICAICA50127.2020.9181875]

[49] Nie P, Wang M, Wang X, Zhao W, Duan SS, Jia KT, Zhang GY. Towards a relay chain-based cross-chain solution for heterogeneous blockchains. Chinese Journal of Computers, 2026, 49(3): 638–660 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2026.00638]

[50] Zhang PY, Tao S, Chen J, Chen ZH. A two-stage cross-chain access control mechanism in relay chain environment. Journal of Computer Research and Development, 2026, 63(1): 255–272 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.202440972]

[51] Wei QY, Zhao XF, Zhu XY, Zhang WH, Lu YH. Formal analysis of cross-chain protocol IBC. Ruan Jian Xue Bao/Journal of Software, 2025, 36(11): 4953–4974 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7356.htm [doi: 10.13328/j.cnki.jos.007356]

[52] Lyu YY, Feng RT, Wang ZM, Liu JC, Wu HW, Li XH. Formal verification and improvement of XCMP protocol: Improving security of cross-chain interaction. Journal of Software, 2026 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7479.htm [doi: 10.13328/j.cnki.jos.007479]

[53] Xu JH, Ackerer D, Dubovitskaya A. A game-theoretic analysis of cross-chain atomic swaps with HTLCs. In: Proc. of the 41st IEEE Int'l Conf. on Distributed Computing Systems. Washington: IEEE, 2021. 584–594. [doi: 10.1109/ICDCS51616.2021.00062]

[54] Strong Nash equilibrium. 2025. https://en.wikipedia.org/wiki/Strong_Nash_equilibrium

[55] Etherscan. The ethereum blockchain explorer. 2025. https://etherscan.io/

附中文参考文献

[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]

[16] 微众银行区块链团队. WeCross 技术白皮书: 区块链跨链协作平台. 2020. https://pdf.dfcfw.com/pdf/H3_AP202003101376070183_1.pdf

[49] 聂鹏, 王玫, 王馨, 赵伟, 段斯斯, 贾珂婷, 张国艳. 面向异构区块链系统的中继链跨链方案. 计算机学报, 2026, 49(3): 638–660. [doi:

10.11897/SP.J.1016.2026.00638]
[50] 张佩云, 陶帅, 陈健, 陈子寒. 中继链环境下一种 2 阶段跨链访问控制机制. 计算机研究与发展, 2026, 63(1): 255–272. [doi: 10.7544/issn1000-1239.202440972]
[51] 魏秋阳, 赵旭峰, 朱雪阳, 张文辉, 卢奕函. 区块链跨链协议 IBC 形式化分析. 软件学报, 2025, 36(11): 4953–4974. http://www.jos.org.cn/1000-9825/7356.htm [doi: 10.13328/j.cnki.jos.007356]
[52] 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. http://www.jos.org.cn/1000-9825/7479.htm [doi: 10.13328/j.cnki.jos.007479]

作者简介

卓凤, 博士生, 主要研究领域为区块链, 跨链.
段田田, 博士, 特别研究助理, CCF 学生会员, 主要研究领域为区块链, 分布式系统.
贾林鹏, 博士, 副研究员, CCF 专业会员, 主要研究领域为区块链性能优化, 跨链互操作.
李忠诚, 博士, 研究员, 博士生导师, CCF 高级会员, 主要研究领域为计算机网络, 区块链.
孙毅, 博士, 研究员, 博士生导师, CCF 杰出会员, 主要研究领域为区块链, 社交视频.