北京大学

硕士研究生学位论文

题目：一种基于联盟链的跨链系统的设计与实现

姓　　名：	叶德鹏
学　　号：	1801210702
院　　系：	软件与微电子学院
专　　业：	计算机技术
研究方向：	金融大数据
导师姓名：	郁莲　教授

二〇二一 年 四 月

版权声明

摘要

伴随着区块链技术的发展，联盟链被广泛地应用于司法仲裁、身份识别、医疗卫生等领域。对于这些领域的部分场景，产生了链与链之间跨链交互的需求，例如链间司法取证、链间个人信息授权和链间医疗健康数据共享。目前联盟链跨链所面临的难题可以概括为三个方面：首先，联盟链底层的实现架构各不相同，底层数据结构存在差异，实现联盟链之间的数据互认存在一定困难；其次，不同联盟链的数据信任机制不同，导致联盟链间的跨链数据存在信任风险；最后，由于各联盟链的通信协议、接口协议各不相同，导致联盟链间的互访受到阻碍。

针对以上问题，本文设计了跨链协议，包括跨链注册，数据互认，数据互信，隐私保护和异常处理等。接着，基于跨链协议，设计了包括数据层，交互层，事务层和治理层的跨链系统。本文的主要工作如下：（1）根据联盟链的特点和联盟链跨链场景，设计了联盟链跨链协议，该协议对跨链涉及的核心数据结构进行了抽象定义，并对跨链流程进行了规范（2）基于跨链协议，在实验室已有项目 ssbc 联盟链的基础上进行了升级开发，设计实现了跨链系统。跨链系统的实现主要包括：基于可验证随机函数以及门限签名实现预言机，并基于预言机实现了区块链间的区块头同步；通过生成及同步抽象区块头以及对抽象交易、抽象回执的打包实现了联盟链间的数据互认；基于默克尔证明以及门限签名实现跨链信息的合法性验证，从而实现了联盟链间的数据互信；基于跨链资产锁定以及异常回滚保证了跨链事务的原子性；基于日志记录、权限管理、追溯机制、监管机制实现了跨链的有效治理。

最后，本文对所设计的跨链系统进行了全面的功能测试以及鲁棒性测试。测试结果表明，跨链系统区块头同步、抽象交易打包、默克尔验证等功能均能正常运行，并且系统具备较强的稳定性。

关键词：联盟链，跨链，数据互认，数据互信，默克尔验证

Design and Implementation of a Cross-chain System Based on Alliance Chain

Depeng Ye(Computer Technology)
Directed by Lian Yu

ABSTRACT

With the development of Blockchain technology, the Consortium Blockchain is widely used in judicial arbitration, identification, medical health and other fields. For some scenarios in these fields, there is a need for cross-chain interaction between chains, such as interchain judicial forensics, inter-chain personal information authorization, and inter-chain medical data sharing. At present, the problems faced by Consortium Blockchain across chains can be summarized in three aspects: Firstly, the implementation architecture of the bottom of the consortium chain is different, the underlying data structure is different, so it is difficult to achieve mutual recognition of data between the consortium chains; Secondly, The data trust mechanism of the consortium chain is different, which leads to the trust risk of the cross-chain data between the consortium chains; Finally, due to the different communication protocols and interface protocols of the consortium chains, the mutual access between the consortium chains is hindered.

In response to the above problems, this article designed a cross-chain protocol, including cross-chain registration, data mutual recognition, data mutual trust, privacy protection and exception handling. Then, based on the cross-chain protocol, a cross-chain system including data layer, interaction layer, transaction layer and governance layer is designed. The main work of this paper is as follows: (1) According to the characteristics of the consortium chain and the cross-chain scenario of the consortium chain, the consortium chain cross-chain protocol is designed. The agreement abstracts the core data structure involved in the cross-chain and carries out the cross-chain process. (2) The specification based on the cross-chain protocol, upgraded and developed on the basis of the existing project SSBC consortium chain in the laboratory, and the cross-chain system is designed and implemented. The realization of the cross-chain system mainly includes: The oracle based on verifiable random functions and threshold signatures, and the synchronization of block headers between blockchains based on the oracle; By generating and synchronizing abstract block headers, abstract transactions and abstract receipts The packaging realizes the mutual recognition of data between alliance chains;

the legality verification of cross-chain information is realized based on Merkel proof and threshold signature, thereby realizing the mutual trust of data between the alliance chains; Based on the cross-chain asset lock and abnormal rollback, it guarantees the cross-chain information. The atomicity of chain transactions; Based on log records, authority management, traceability mechanisms, and supervision mechanisms, effective cross-chain governance is realized.

Finally, this paper conducts a comprehensive functional test and robustness test on the designed cross-chain system. The test results show that the cross-chain system block header synchronization, abstract transaction packaging, Merkel verification and other functions can all operate normally, and the system has strong stability.

KEY WORDS: Consortium Blockchain, Cross-chain, Mutual Acceptance of Data, Data Mutual Trust, Merkel Verification

目录

第一章 绪论 1
1.1 研究背景及意义 1
1.2 研究现状 2
1.3 研究内容 4
1.4 论文结构 4
第二章 相关技术介绍 6
2.1 默克尔证明 6
2.1.1 默克尔树 6
2.1.2 默克尔树在区块链中的应用 7
2.2 预言机技术 8
2.3 可验证随机函数 9
2.4 门限签名技术 12
2.5 本章小结 13
第三章 跨链系统需求分析 14
3.1 跨链系统的基本功能需求 14
3.2 跨链系统的非功能性需求 16
3.3 本章小结 17
第四章 跨链协议与架构设计 18
4.1 跨链协议 18
4.1.1 跨链注册机制 18
4.1.2 数据互认机制 18
4.1.3 数据互信机制 21
4.1.4 跨链数据隐私保护机制 22
4.1.5 异常处理机制 23
4.2 系统架构设计 23
4.2.1 数据层 24
4.2.2 交互层 25
4.2.3 事务层 25
4.2.4 治理层 26
4.3 本章小结 26
第五章 系统详细设计与实现 27

5.1 数据层 27
5.1.1 抽象区块头存储模块 27
5.1.2 抽象交易打包模块 27
5.1.3 抽象回执打包模块 30
5.2 交互层 31
5.2.1 区块头同步模块 31
5.2.2 跨链寻址模块 33
5.2.3 交易验证模块 35
5.3 事务层 37
5.3.1 跨链资产锁定机制 38
5.3.2 回滚机制 38
5.4 治理层 39
5.4.1 日志模块 39
5.4.2 追责机制 39
5.4.3 权限管理 40
5.4.4 监管机制 41
5.5 本章小结 42
第六章 系统测试与评估 43
6.1 测试环境 43
6.2 系统测试 43
6.2.1 功能模块测试 43
6.2.2 异常场景测试 46
6.3 系统评估 48
6.4 本章小结 48
第七章 总结与展望 50
7.1 回顾和总结 50
7.2 下一步工作展望 50
参考文献 51
附录 A 术语表 54
致谢 55

第一章　绪论

1.1　研究背景及意义

自比特币面世以来，区块链技术[1]越来越受到业界的关注与重视。区块链技术因其分布式存储、基于 P2P 网络的底层通信、数据上链需要共识、深度融合密码学等特性，逐渐成为构建价值网络的重要基础设施[2]。自 2008 年 11 月 1 日比特币白皮书[3]发布至今，区块链技术已从最初的可编程数字货币发展到可编程金融，目前已进入 2.0 阶段[4]，该阶段的一个主要特征是智能合约[5]的普遍应用。当前，世界主要经济体都在推进区块链技术的布局，换言之，随着研究的深入，区块链技术激发并引领了各行业的技术革新和社会的产业变革[6]。

根据许可程度的不同，区块链可以分为公链、联盟链以及私链三种类型[7,8]。公链的开放程度最高，达到了完全的去中心化，任意节点都可以随意加入。因此，用户加入公链无任何门槛，不需要身份认证，链上的数据也完全公开透明。这使得公链拥有近乎完全自由的特性，这是它能快速发展的一个主要原因，但同时也带来了许多问题，例如：监管困难，数据无任何隐私保护，作恶节点易加入等等。因此，公链并不太适合企业使用。私有链开放程度最低，只能由单一机构运作，但因其未能体现区块链分布式账本的理念，所以无法得到广泛应用。联盟链的开放程度介于公链和私链之间，加入联盟链需要获得链内组织的许可，联盟链上的数据只向链内成员公开，具备一定的隐私保护性。联盟链一般链上无原生数字代币，侧重基于智能合约实现复杂的链上应用以服务于链上组织成员之间的协作。所以联盟链的链上交易一般无需手续费，参与节点打包区块无需挖矿，节点进行共识无需争取记账权以获得代币激励。正是基于联盟链极低的交易成本、完善的数据隐私保护机制、相对公链更快的交易速度、作恶门槛高、作恶行为可及时控制等特性，越来越多的企业和组织开始使用联盟链进行成员间协作。

智能合约的图灵完备性[9]，使联盟链得以在社会的许多领域中落地和应用。基于联盟链，百度公司开发了“图腾”平台，该平台可用于保护原创图片的版权。除此之外，在常见的需要溯源的场景如商品溯源、药品溯源中，联盟链通过上链物品从厂家生产、质检部门抽检、物流公司转运的全流程记录使得用户知悉物品的真实性，一旦出现质量问题可查询每一个环节的记录信息实现溯源[10,11]。

尽管近几年联盟链发展迅速，链上应用层出不穷，但总体而言，这种发展主要是以小规模组织、局部使用为主，少有形成联盟链与联盟链之间交互协作的多链互联生态，主要阻碍是联盟链之间的跨链交互[12]的实现。当前联盟链跨链面临着三个层面的难题：

第一，联盟链底层的实现架构各有不同，首当其冲的就是底层数据结构的差异性，这使得联盟链之间难以实现数据互认。二是不同联盟链间数据信任机制的差异，导致了联盟链之间难以实现数据互信。第三，不同联盟链的通信协议、接口协议不尽相同，导致联盟链之间难以实现互联互通。总的来说，联盟链上的数据由于其组织内部公开透明，组织外部不可见的隐私保护机制，使单独的联盟链成为信息孤岛[13]，信息只能在链内使用，无法发挥其最大的价值[14]。此外，在联盟链中存在着成员身份认证机制，提高了链外成员的加入门槛，造成了联盟链中信任存在边界。如何打破这一边界，将信任延伸至链外是一个难题。这就使得如同一个个信息孤岛的联盟链又像是一个个信任孤岛，进而进一步制约了联盟链的发展。本文关于跨链交互的研究目标正是推倒阻碍链间信息流动的高墙，把众多的信息孤岛、信任孤岛连接、贯通起来，让各个联盟链上的业务能够实现合作协同，真正发挥联盟链的力量[12,15-17]。

1.2 研究现状

随着跨链需求的日益增多，越来越多的跨链项目被提出，跨链技术也得到了蓬勃发展。以太坊[18]创始人 Buterin 从总体上将跨链技术划分为三种机制：公证人机制、侧链/中继机制以及哈希时间锁机制[19]。

公证人机制是一种选举中间公证人作为跨链中介的机制，实质上就是公证人承担着两条链之间的信息传递和交易确认的责任。Corda[20]是公证人机制的代表项目之一，该项目于 2016 年推出，其主要实现方式是由跨链各方共同选举出验证者作为交叉验证人，交叉验证人对每次跨链交易的数据进行真实性验证。公证人机制具有实现简单、兼容性强等特点，对参与跨链的区块链本身无特殊要求，所以对于异构链之间的跨链有很好的支持。但是公证人机制对于中间公证人的信任度要求较高，跨链数据的可信性取决于公证人的信任程度，因此，这种实现机制存在一定的信任风险，且其与区块链的去中心化理念有所违背。

侧链/中继机制是通过同步目标链的区块头以在本链上生成目标链的轻客户端[21]，然后基于轻客户端对目标链上发生交易的合法性进行默克尔验证以实现跨链数据可信的机制。其本质上是通过默克尔证明来解决跨链交易可信性的问题。侧链机制的代表项目之一为 BTC Relay[22]，该项目开展于 2016 年，该项目通过在以太坊网络上对比特币网络中发生的交易进行合法性验证，实现了以太坊对比特币的跨链。BTC Relay 项目的实现是在以太坊上部署一个存储比特币网络区块头的智能合约，通过以太坊上 Relayer 的提交实时更新比特币区块头，然后基于区块头和交易的 Merkle Path 对比特币交易进行默克尔验证。中继机制的代表项目有 Cosmos[23]和 Polkadot[24]等。Cosmos 于

2016 年被提出，该项目的实现是基于枢纽以及分区构建了 Cosmos 网络，并设计了一种旨于在区块链间进行通信的 IBC 协议。Polkadot 项目于同年被提出，该项目基于中继链来转发各个参与跨链的区块链网络的交易，与此同时，中继链也会同步各个参与跨链的区块链的区块头信息。在 Polkadot 中定义了四个参与跨链的角色，分别是：运行全节点的 Collator，发现非法行为并进行举报的 Fisherman，拥有权益并进行资产委托的 Nominator，以及封装区块的 Validator。侧链/中继机制的实现较为复杂，但是与公证人机制相比，该机制实现了去中心化的跨链。但因为这一机制需要在本链上生成所跨目标链的轻客户端，所以对于参与跨链的区块链网络本身也有一些要求，例如上面的 BTC Relay 因为无法在比特币网络中实现以太坊的轻客户端，只能单向跨链，不能实现比特币对以太坊的跨链。

哈希时间锁机制是一种跨链双方在约定的时间内给出解锁资产所需密钥的跨链机制，其本质是跨链一方在使用密钥解锁对方锁定资产的同时密钥暴露，对方基于该密钥可解锁己方锁定资产。哈希时间锁机制的代表项目之一为 2015 年提出的 Lightning Network（闪电网络）[25]，该项目的实现方式是将大部分的交易置于比特币链下执行以提高比特币的交易处理速度。哈希时间锁机制实现简单，并且与侧链/中继机制一样，该机制的实现也是去中心化的。但是这种机制的缺点也比较明显，首先是兼容性差，基于哈希算法实现的特性要求参与跨链的双方使用相同的哈希算法[26]，并且由于需要对资产进行锁定和解锁，因此跨链的双方需要具有一定的可编程能力如支持智能合约。其次，基于该机制的跨链本质上只能是跨链的资产交换，跨链双方都应在对方链上具有账户，跨链结果是双方的资产交换到在对方链上的账户中。如表 1.1 所示，本文对各个跨链机制的特点进行了总结与比较[17]。

表 1.1　跨链机制特点比较

跨链机制	具体实现	优点	缺点
公证人机制	选举一个或多个公证人作为跨链的中介，对跨链双方的交易数据进行真实性验证	实现简单；兼容性强，对参与跨链的区块链网络无要求	跨链数据的可信性依赖于公证人的可信性，存在信任风险
侧链/中继机制	通过同步目标链的区块头在本链实现目标链的轻客户端，基于跨链交易的 merkle path 进行默克尔验证	去中心化；应用场景丰富：支持跨链资产转移、跨链读写等	实现困难；要求参与跨链的区块链网络具备实现外链轻客户端的能力

续表 1.1　跨链机制特点比较

哈希时间锁机制	跨链双方基于相同随机数的哈希值对自身资产进行锁定，在约定时间内，跨链一方使用随机数解锁对方资产时，随机数暴露，对方基于该随机数解锁己方资产	去中心化，实现简单	兼容性差，存在一定局限性；只支持跨链资产互换

1.3　研究内容

由于联盟链上一般没有原生代币[27]，且主要基于智能合约实现业务协同，所以联盟链之间的跨链更多是链上数据的跨链转移[28]。但是，当前业界大部分的跨链方案更注重公链上的资产的跨链转移[29,30]，而对数据跨链的关注不够。随着联盟链的快速发展，链间数据互信共享、业务高效协同需要跨链系统的支持。从应用场景来看，哈希时间锁机制仅支持跨链资产互换，因此不适用于联盟链跨链系统；从去中心化角度考虑，公证人机制存在信任风险，且在多联盟链跨链场景下，如何选择共同的公证人是一个难题，因此公证人机制也不太适用于联盟链跨链系统。联盟链由于其基本支持智能合约故具有在本链上实现外链轻客户端的能力，即天然支持中继机制。并且联盟链间的跨链场景丰富，即跨链系统需要支持跨链资产转移、跨链读、跨链写等多个场景，因此，本文在深入分析探究当前业界主流跨链技术的基础上基于中继机制以及实验室项目 ssbc 联盟链，设计并实现了一种基于联盟链的跨链系统。

本文的主要研究工作如下：

（1）对现有的跨链项目进行深入探究与总结，并对其优点和不足进行归纳总结。

（2）设计联盟链跨链协议，定义抽象区块头、抽象交易、抽象回执等数据结构，规范跨链流程。

（3）基于可验证随机函数以及门限签名[31-33]实现了预言机机制，并基于预言机机制实现了区块链间的区块头同步。

（4）基于抽象区块头、抽象交易、抽象回执实现数据互认机制。

（5）基于默克尔证明机制、门限签名机制实现数据互信机制。

（6）设计数据层、交互层、事务层、治理层，实现跨链系统的所有功能模块。

（7）对跨链系统进行全面的测试与评估。

1.4　论文结构

第一章主要介绍了联盟链跨链的背景和目前跨链技术发展的现状，并对研究现状进行总结，简要描述了本文开展的工作。

第二章主要介绍了本文所设计的跨链系统的相关技术，包括可验证随机函数、预

言机技术、门限签名技术和默克尔证明。

第三章结合应用场景对联盟链跨链系统的需求进行分析与说明。

第四章详细介绍了本文所设计的联盟链跨链协议以及跨链系统的架构设计。

第五章主要从数据层、交互层、事务层、治理层等方面对跨链系统各功能模块的设计与实现进行了详细介绍。

第六章针对跨链系统进行了全面的测试，并根据测试结果对跨链系统作出整体评估。

第七章主要是对本文研究工作的总结和展望。

第二章　相关技术介绍

本章是对本文所设计的跨链系统相关的技术做一个简要的介绍说明，主要包括默克尔证明、预言机技术、可验证随机函数以及门限签名四个方面的内容。

2.1　默克尔证明

2.1.1　默克尔树

默克尔证明是一种以默克尔树[34-36]为基础的证明机制，默克尔树是一种树，也被称为哈希树。默克尔树大多为二叉树，在区块链中主要用于证明特定区块中是否存在特定交易。

一般来说，默克尔树具有以下特征：

（1）默克尔树的所有叶子节点均为对应交易的 Hash 值。

（2）默克尔树从下往上第二层非叶子节点开始，每一个非叶子节点的值都由其孩子节点的组合值 Hash 获得。

（3）只要任意一个叶子节点对应的数据块中有任何的数据变化，这种变化都将导致生成一棵完全不同的默克尔树，默克尔树的树根也会完全不同。

一棵简单的默克尔树结构图如图 2.1 所示。

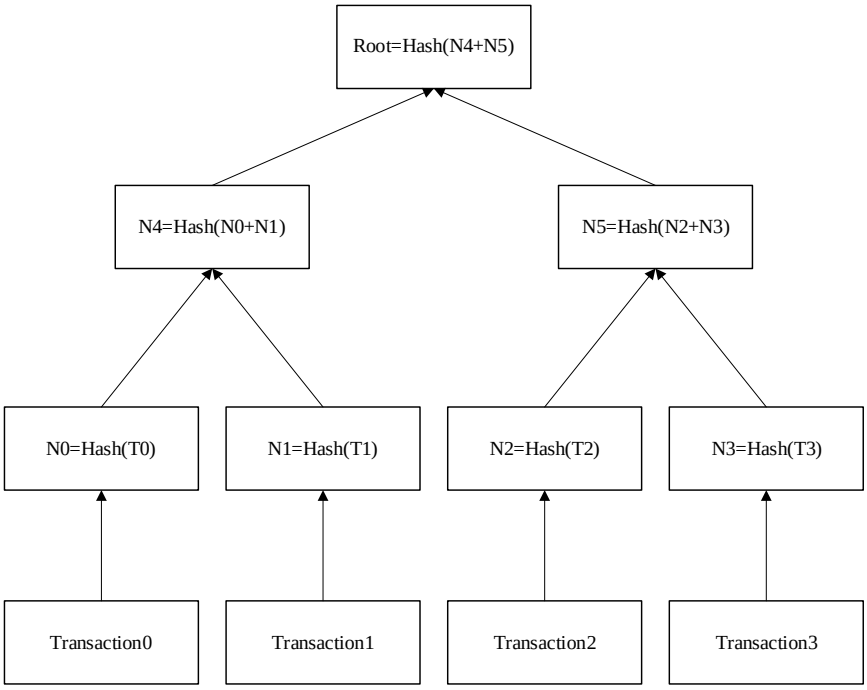

图 2.1　默克尔树结构图

2.1.2　默克尔树在区块链中的应用

在区块链中默克尔树的生成方式如下：首先，将交易列表中的所有交易按顺序排列，为每笔交易依次生成一个 Hash 值作为叶子节点，然后从左到右将每两个叶子节点的值组合 Hash 生成父节点的 Hash 值。依此类推，从下向上最终生成一个默克尔树树根即 Merkle Root。若叶子节点的个数为奇数个，那么一个简单的解决方案是：复制最后一个叶子节点凑成偶数个，然后开始生成默克尔树。

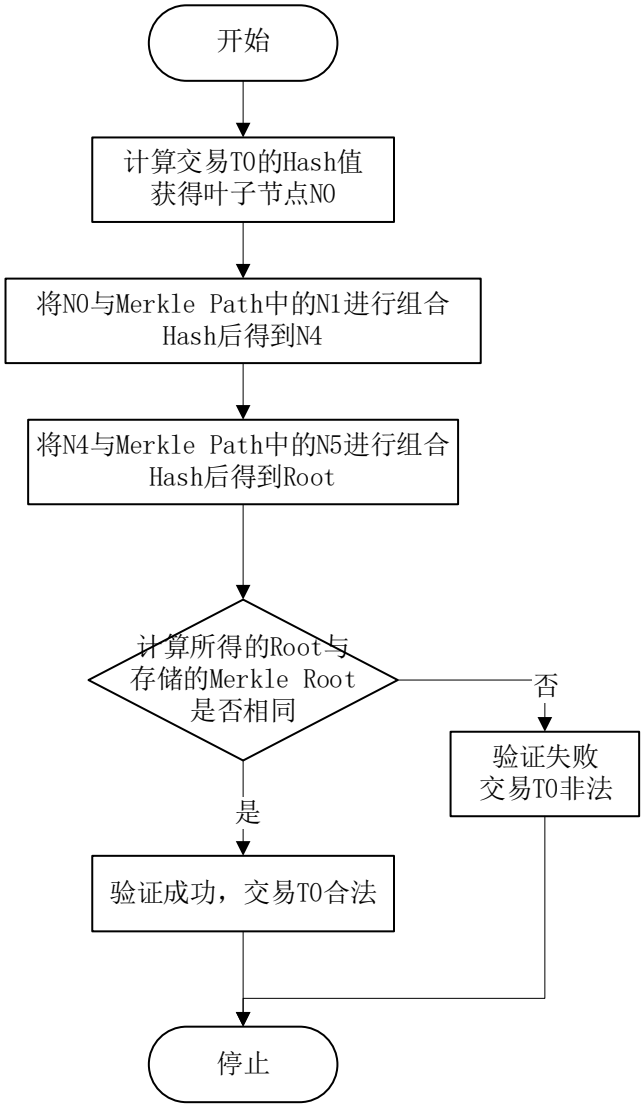

图 2.2　交易 T0 合法性验证流程图

一般而言，区块链中只有全节点才会保存全部的区块数据，而普通的轻节点并不存储所有的区块数据，只是将所有的区块头保存到本地。当轻节点需要验证一笔交易的合法性时，轻节点基于本地存储的指定区块头中的 Merkle Root 和接收到的交易的

Merkle Path 进行默克尔验证。对比计算得出的树根 Hash 与取出的 Merkle Root 是否相同，两者相同则证明交易合法。

以图 2.1 为例，为了证明交易 T0 的合法性，只需要由 N1 和 N5 这两个值所构成的 Merkle Path 即可。交易 T0 合法性证明的流程图如图 2.2 所示。

具体的步骤说明如下：

Step1：计算交易 T0 的 Hash 值，得到 N0，其中 N0=Hash(T0)。

Step2：将 N0 与 Merkle Path 中的 N1 进行组合，计算父节点 N4 的值：N4=Hash(N0+N1)。

Step3：将 N4 与 Merkle Path 中的 N5 进行组合，计算父节点即 Merkle Root 的值：Merkle Root=Hash（N4+N5）。

Step4：将计算得到的 Merkle Root 与从本地获取到的 Merkle Root 进行对比，如果相同则证明该区块中确实存在交易 T0，否则，证明交易 T0 不存在。

2.2 预言机技术

区块链外信息写入区块链内的机制，一般被称为预言机[37]。区块链本质上是一种分布式账本[38]，所有的信息上链都需要达成共识，对于链上的智能合约，给定相同的输入，所有节点执行应该得到相同的结果。从另一个角度讲，区块链是一个确定性的系统，确定性体现在每一个节点都能验证区块内的交易，验证方式是在本地执行与交易相同的流程，无论何时何地，执行完流程得到的结果应该与交易结果完全相同。这种确定性就导致了区块链的封闭性，也就是说区块链是无法直接调用外部信息接口的，因为一旦调用了外部接口，则结果就是不确定的，也就无法达成共识。因此，预言机机制的产生与发展正是为了解决区块链系统对于外部信息的依赖问题[39,40]。

一般而言，智能合约在区块链上的执行是需要触发条件的，一旦这种触发条件依赖于链外信息，如某农作物保险合约依赖于当地天气的实时输入，此时就必须依靠预言机来提供链外天气信息，通过预言机将数据传递给链上智能合约。预言机的原理图如图 2.3 所示。

根据实现体系结构的不同，预言机可以分为中心化预言机与去中心化预言机。中心化预言机只有一个预言者，一般为单一可信数据中心处理来自链上的链外数据请求，请求方接收单一结果，这种实现方式存在一定的中心化风险，数据的可信度依赖于数据提供方的可信度。去中心化预言机具有多个预言者，针对同一数据请求，请求方能获取到来自不同预言者的多个结果，因此数据的可信性可以通过多个预言者提供的结果相互验证。

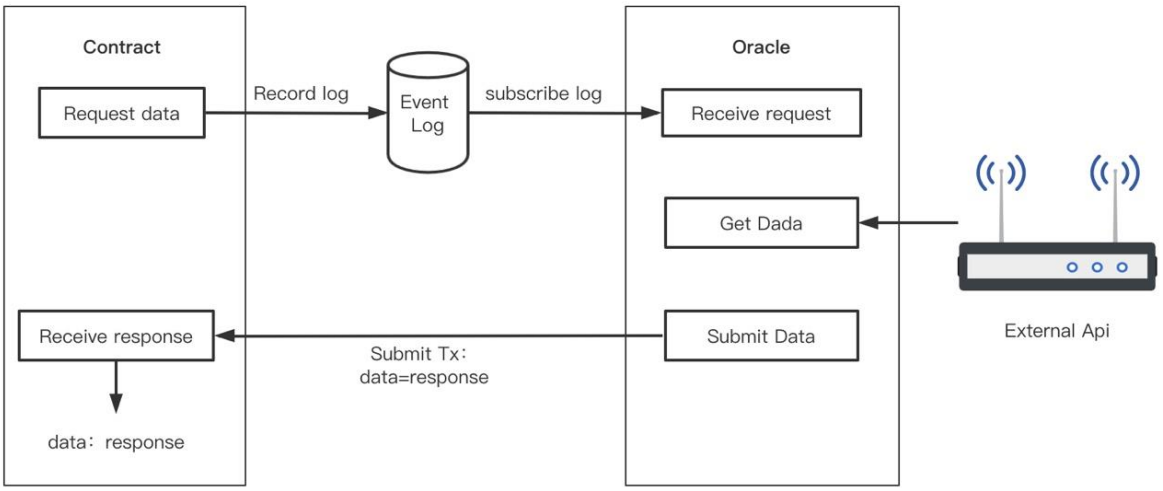

图 2.3　预言机原理图

2.3　可验证随机函数

可验证随机函数是一种在分布式网络中，由本地生成随机数且该随机数可以被全网其他节点进行验证的随机函数[41,42]。要理解可验证随机函数，需要从理解随机函数入手，一般情况下，一个标准的随机函数应该具有如下的特性：

（1）对于一个任意的输入值，它的输出值应该是随机的，没有任何规律可循。

（2）随机函数的值域应该尽可能对称概率分布，即均匀分布。

随机函数的特性与哈希函数的特性相似，一个简单的哈希函数可以根据 info 的输入计算得到结果 result，即 result=hash(info)。将该哈希函数进行改造，与加密技术结合，每次计算不但需要输入 info 还需要输入密钥 secret，即 result=hash(info,secret)。那么，每次验证计算结果的合法性时需要出示密钥 secret，这就引出了一个问题：如何在不暴露密钥 secret 的前提下仍然能够对计算结果 result 进行合法性验证？融合了哈希函数以及非对称加密技术的可验证随机函数的出现，为此类问题的解决提供了思路。

通常情况下，要实现可验证随机函数需要先实现其子函数，子函数如表 2.1 所示。

表 2.1　可验证随机函数的子函数

子函数名称	功能说明
VRF_Gen（）	该函数用来生成一对公私钥对，公钥 PK，私钥 SK
VRF_Hash()	该函数用来计算出 VRF 结果，即生成了一个随机数
VRF_Proof()	该函数用来生成 VRF 证明
VRF_P2H（）	该函数用来验证随机数结果与随机数证明的对应关系

续表 2.1　可验证随机函数的子函数

VRF_Verify()	该函数基于公钥 PK 以及证明 Proof 验证随机数结果的合法性

可验证随机函数一次生成结果并进行验证的流程如图 2.4 所示，具体的操作步骤如下：

Step1：节点 A 在本地调用 VRF_Gen()来生成公私密钥对。

Step2：节点 A 基于私钥 SK 以及输入 Info 调用 VRF_Hash()生成随机数结果。

Step3：节点 A 基于私钥 SK 以及输入 Info 调用 VRF_ProofI()生成随机数结果对应的证明。

Step4：节点 A 向节点 B 发送随机数结果及证明。

Step5：节点 B 调用 VRF_P2H()验证结果与证明的对应关系。

Step6：验证成功则节点 A 继续将公钥以及输入 Info 发送至节点 B 并继续执行 Step7，验证失败则此次验证未通过，流程结束。

Step7：节点 B 基于节点 A 的公钥，证明信息，以及输入信息验证结果的合法性。

Step8：验证成功则此次验证通过，否则将无法通过此次验证。

综上，可验证随机函数的特性总结如下：

（1）由于生成的结果包含了生成者的签名，其他验证者可以基于生成者的公钥进行验签，故 VRF 具备唯一性，即无法伪造或者篡改他人生成的 VRF 结果。

（2）整个随机结果生成的过程均是在本地进行的，无需与其他节点进行通信，因此 VRF 具备一定的高效性。

（3）除具备一般随机函数的特性外，可验证随机函数还具备的一个特性是对于相同的输入，不管执行多少次，其输出的结果都不变。这是因为在分布式网络中，可验证随机函数不仅需要保证由某一个节点生成的随机数全网可验证，而且还需要保证该节点无法通过多次运行随机函数而得到所需结果。

（4）任何节点都能够基于接收到的 VRF 证明对该证明对应的 VRF 结果进行验证，即 VRF 具有全网可验证性，且该验证过程不需要暴露 VRF 结果生成者的私钥。

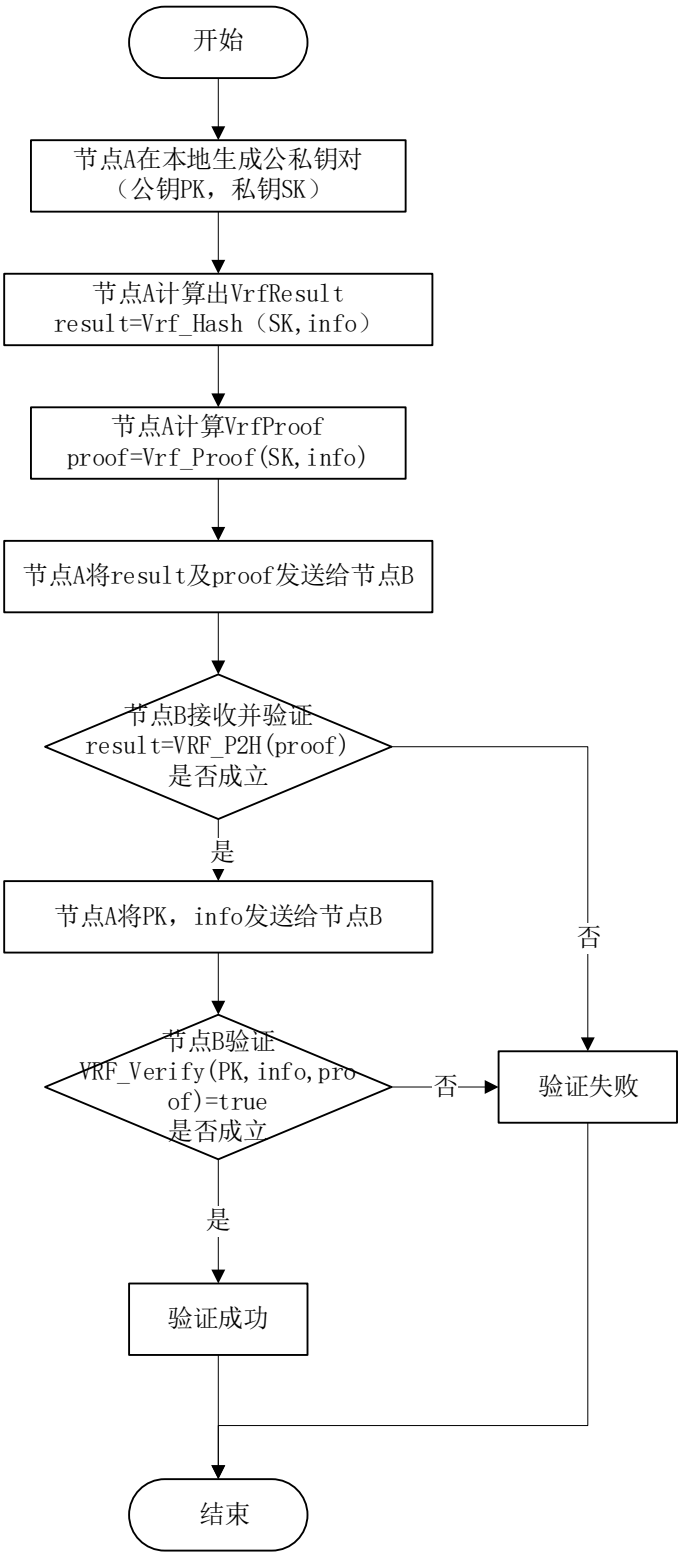

图 2.4　可验证随机函数执行流程图

2.4 门限签名技术

数字签名技术是一种用于验证消息传输者真实身份和消息完整性的技术[31]。一般情况下，针对一个特定的消息，一个节点签名后就可以了。但这种常规的单点签名技术存在一定的局限性，在特定的分布式网络场景中，特定的信息或决策需要多个参与者的签名才能生效。针对这种场景需求，提出了一种多人签名方案即多重签名。多重签名的实现方案是每一个参与者都对提案信息进行签名，只有所有参与者都签名完毕，才能验证该信息是合法的，但是多重签名的本质还是若干个单点签名的集合。这种签名机制的结构难以动态调整，即参与者无法中途退出或加入，并且验签复杂，验签时需要对签名集合中的每一个单点签名进行验签，存在一定的应用局限性。门限签名技术的出现打破了多重签名的局限，同时又保留了多重签名的优点，它是一种在分布式网络中由多个节点参与签名的签名方案，它主要包括密钥的生成和分配、节点签名、收集和验证等环节。

门限签名的技术原理如图 2.5 所示。首先，基于门限密钥生成函数生成一个公共公钥以及分属于不同参与者的私钥分片，每一个参与者获取自己的私钥分片。接着，在分布式网络中，针对同一消息，每一个参与者基于自身私钥分片对消息进行签名生成签名分片。最后，收集足够多的签名分片，当签名分片数大于等于门限阈值时，生成最终的签名，并且该签名可以通过公共公钥进行合法性验证。

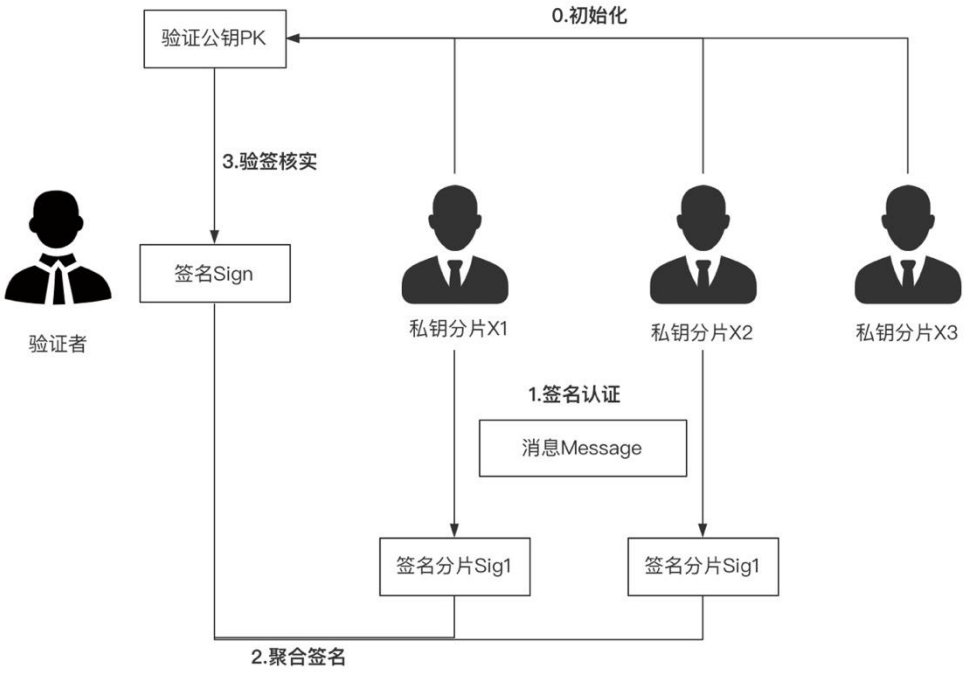

图 2.5　门限签名的技术原理图

具体的步骤说明如下：

Step1：初始化，生成验证公钥 PK 以及私钥分片集合，每个参与者都将获得自己的私有密钥分片。

Step2：签名认证，参与者基于自身私钥分片对消息进行签名。

Step3：聚合签名，收集所有参与者的签名分片，当收集数大于等于门限阈值时生成最终签名。

Step4：验签核实，验证者基于验证公钥 PK 对最终签名的合法性进行验证。

综上所述，现对门限签名的特征归纳如下：

（1）只有收集到的参与者生成的有效签名数大于等于阈值时，才能生成最终签名。

（2）签名结构可以动态调整，参与者可以随时退出或加入。

（3）容错能力高，只要生成的有效签名数大于阈值，即使部分节点作恶或部分节点的私钥丢失，仍然可以生成最终签名。

2.5　本章小结

本章主要介绍了与本文所设计的跨链系统相关的一些技术，包括默克尔证明、预言机、可验证随机函数以及门限签名。基于默克尔证明技术可以对其他链上的交易进行验证以确定该交易是否存在于指定区块中；基于预言机技术可实现链与链之间的区块头同步；基于可验证随机函数可从同步区块头的服务节点中选出提案节点；基于门限签名技术其他节点可验证提案节点提案的合法性，并发送签名分片，提案节点收集满足阈值要求的签名分片，从而生成最终签名。

第三章　跨链系统需求分析

3.1　跨链系统的基本功能需求

本文基于联盟链的跨链系统设计，最终目标是满足不同场景下联盟链的跨链需求，主要包括资产价值的跨链转移以及数据价值的跨链转移。资产价值的跨链转移是指用户可以基于链上的匿名身份，在加密世界自由地进行资产交换，实现资产的自由流通。数据价值的转移，本质上是数据的传输、信息的交互，是指信息无损、安全地在链间传输的过程。针对资产与数据转移的需求分析进行梳理和总结，现归纳跨链系统应满足的基本功能需求如下：

（1）跨链协议

不同联盟链之间进行跨链交互需要遵循一定的跨链交互规范，即需要制定用于规范跨链交互的跨链协议。一个合格的跨链协议需要从跨链机制、跨链数据结构的定义以及跨链流程等各个方面做出详细明确的规定与说明。

（2）跨链注册机制

参与跨链交互的双方要求能够通过获取对方的通信地址感知到交互对象的存在，并基于该通信地址将跨链消息传递至对方，因此跨链系统需要实现跨链注册机制。跨链双方在开始跨链交互前需要在对方链上进行注册，登记本链的基础信息以及本链的通信地址等相关信息。

（3）信息传输

跨链交互的本质是链与链之间的信息传输，稳定的信息传输通道是实现交互双方信息交流的基本保障。

（4）信息安全

信息安全是首当其冲需要考虑的问题，为了保证跨链消息在链间传输过程中的安全性及完整性，避免消息被第三方恶意截获读取或者恶意篡改，跨链系统需要实现消息的加解密机制以及签名机制，即消息发送方在发送消息之前需要基于接收方公钥对消息进行加密并基于自身私钥对消息摘要进行签名，消息接收方将基于自身私钥对消息进行解密并基于发送方公钥对消息摘要进行验签。

（5）数据互认[43,44]

由于区块、区块头、交易等结构体的定义规范不同，不同区块链的底层实现各有不同，这导致了联盟链间的差异性。因此，跨链系统需要基于已有的差异性实现跨链数据的互认。当参与跨链的双方为异构链时，需要满足将源链交易中继至目标链后，目标链能够正确解析来自源链的交易。基于同样的原因，跨链双方在同步对方的区块头后，需

要满足正确解析对方区块头数据的需求，需要能够基于解析出的 Merkle Root 进行默克尔验证。

（6）数据互信

联盟链的执行环境是一个可信环境，但是这种信任存在边界。跨链双方均未能参与对方链上数据共识上链的过程，所获取到的跨链数据均是由各自的中继节点接收与传递。因此，跨链系统需要通过对跨链数据的合法性验证，实现跨链数据的互信性。

（7）交易事务的原子性

跨链交易事务的原子性是指一笔跨链交易可能会分解为在源链和目标链上的多个执行步骤，跨链系统需要保证一次跨链交易中属于同一事务的所有执行步骤要么全部成功执行，要么全部执行失败并正常回滚，不允许出现部分步骤执行成功而其余步骤执行失败的情况。

（8）资产跨链转移

链上资产的跨链转移是联盟链跨链的一个核心目标，故跨链系统必须满足源链成功将本链资产转移至目标链的需求。

（9）跨链读写

与公链不同，联盟链更注重于智能合约的开发以实现组织成员间的业务协同。因此，联盟链之间跨链交易的较大比例仍然是数据价值的转移。从本质上说，数据价值转移是源链向目标链发起一个跨链读请求，以获取目标链上指定数据信息，或源链向目标链发起一个跨链写请求，以更新目标链上指定数据的状态的过程。跨联盟链读写是数据价值转移的基础，也是跨链系统功能的基本要求。

跨链系统功能需求的用例图如图 3.1 所示。

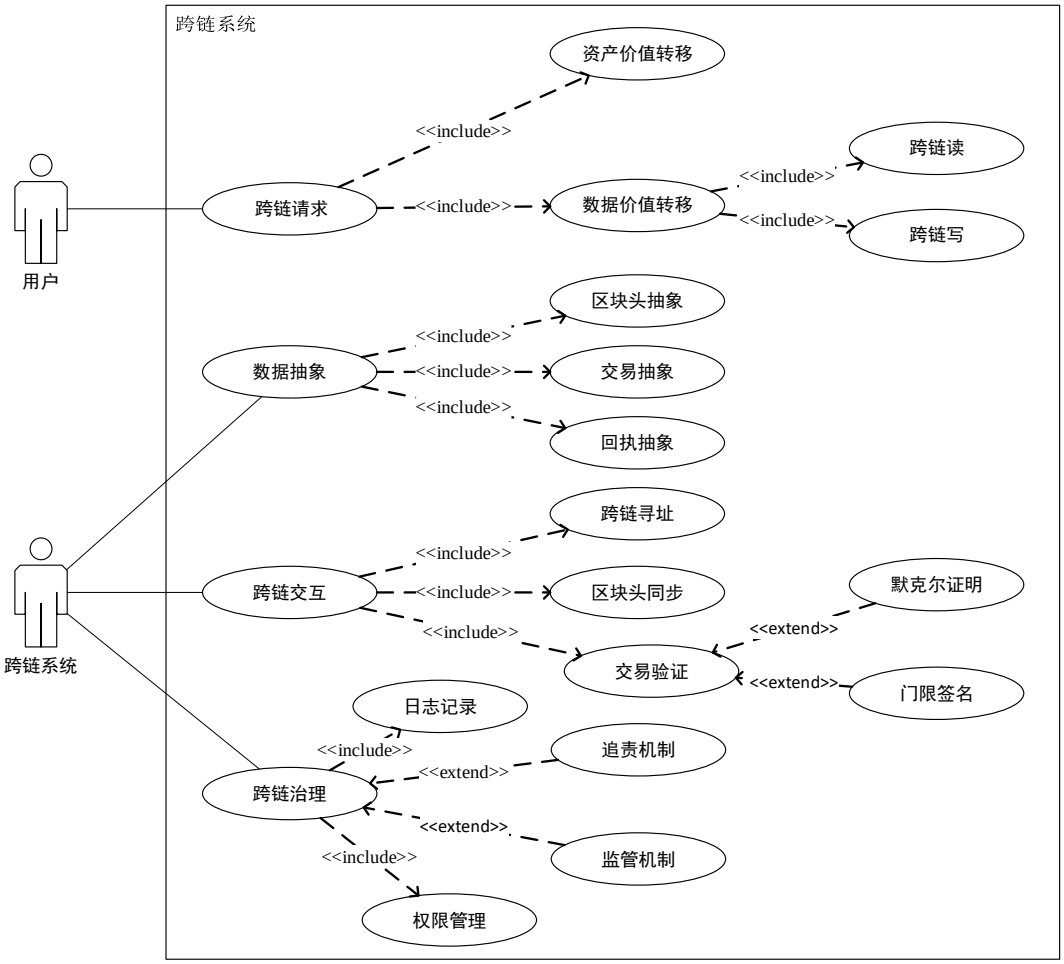

图 3.1　跨链系统用例图

3.2　跨链系统的非功能性需求

除了基本的功能性需求外，从跨链系统的完备性出发，跨链系统还应满足一些非功能性需求。本文对这些需求进行了梳理，归纳如下：

（1）权限管理

本论文所讨论的跨链系统是基于联盟链设计与实现的，联盟链本身针对不同账户对链上不同资源的操作有一定的权限设置。所以，针对链与链之间的跨链资源操作，跨链系统需要具备细粒度的权限管理，只有被授予权限的用户才能对指定的资源进行跨链操作。

（2）隐私保护

尽管区块链上的数据是公开透明的，但对于联盟链来说，开放是有界限的，只有联盟链内的成员才有权限读取链上数据。换而言之，当链外一个具有可读权限的用户发起跨链读请求时，需要保证跨链读结果在传输至源链后不会被源链其他节点解析读取，

只有具有权限的读交易发起者才能将完整的读结果解析读出。因此，这就要求跨链系统必须设计实现跨链数据隐私保护机制，以保护联盟链数据的链外隐私。

（3）容错性

首先，跨链系统需要对恶意节点的恶意行为具备容错性，在跨链过程中，恶意节点可能会有发起假抽象区块头存储提案、篡改跨链读结果后发起门限签名等作恶行为。跨链系统需要确保在小范围内作恶不会影响系统的正常运行。

其次，跨链系统还需要对跨链过程中可能出现的异常情况具有容错能力。在跨链过程中，可能会发生服务节点或者中继节点连接超时甚至宕机等突发异常，跨链系统需要确保正在发生的跨链交易能够正常回滚以及严重程度不高的异常能够迅速恢复。如果异常情况严重到影响了跨链交易的正常进行，跨链系统需要保证这种异常不会对联盟链系统本身造成影响。

（4）扩展性

随着联盟链自身的迭代升级，跨链系统应该实时地进行升级与拓展，这就要求跨链系统应该具备较强的扩展性。跨链系统应尽量采用分层模块化设计，模块与模块之间需要做到高内聚、低耦合；每个模块都应可以独立进行优化扩展，从而满足跨链系统的扩展性需求。

3.3　本章小结

在这一章中，本文从资产跨链转移和数据跨链转移两个角度，对基于联盟链设计的跨链系统进行了详细的需求分析。从跨链数据的互认性、互信性，跨链事务的原子性等方面分析了跨链系统的基本功能需求；从权限管理、隐私保护、容错性等方面分析了跨链系统的非功能性需求。

第四章　跨链协议与架构设计

4.1　跨链协议

本文基于联盟链特性以及联盟链跨链场景设计了联盟链跨链协议。该协议是对跨链交互所涉及的跨链数据结构、跨链流程等各方面的规范。本文所设计的跨链协议主要包括了跨链注册机制、联盟链间数据互认机制、跨链数据可信机制、跨链数据隐私保护机制和异常处理机制五个方面。

4.1.1　跨链注册机制

参与跨链交易的双方在跨链交易开始前，需要分别在对方的链上进行跨链注册。跨链注册的主要内容分为两个模块，首先是注册链的基本信息，其次是注册链用于提供区块头同步服务的节点列表和用于提供消息中继服务的节点列表。跨链注册完成后，跨链双方的服务节点和中继节点将根据对方的 IP 地址与端口号进行握手连接。跨链注册信息的结构体定义如表 4.1 所示。

表 4.1　跨链注册信息结构体定义

```
1   //golang
2   type RegisterInformation struct {
3       Id string   //链名
4       Meta string   //json 序列化后的基本信息
5       Relayers []Node   //中继节点列表
6       Servers []Node   //服务节点列表
7   }   //注册信息
9   //节点信息
10  type Node struct {
11      Id string   //节点 Id
12      PublicKey []byte   //节点公钥
13      IP string   //节点 IP 地址
14      Port string   //节点端口号
15  }
```

4.1.2　数据互认机制

跨链交易的本质是跨链消息在链间的传递以及基于跨链消息链与链之间的协作，

因此，跨链的核心就在于链间数据的流动。然而，基于不同底层体系结构实现的联盟链之间存在着差异，这种差异主要表现为底层数据结构的不同。不同链的区块、区块头、交易等结构体均有所差异，例如相同含义的字段命名不同：在 A 链上默克尔树根可能被命名为 Merkle Hash，而在 B 链上可能被命名为 Merkle Root；同样，不同链上相同含义的字段类型也可能不同。这种差异给跨链带来的问题是，链 A 的一笔交易，或者一个区块头，在链 B 上无法解析更无法使用，也就是说，难以实现链与链之间的数据互认。因此，本文提出的跨链协议对跨链交互所涉及的核心结构体进行抽象，主要包括区块头、交易以及回执。所有参与跨链的联盟链均需要在发送跨链数据前按照本协议进行数据抽象，作为数据互认的基础。

跨链双方在进行区块头同步之前，需要将本链区块头进行抽象生成抽象区块头。所以在实际的跨链交易过程中，跨链双方实际上同步的是对方的抽象区块头。抽象区块头除了包含一般区块头的常见字段如：区块高度、区块哈希、前一区块哈希等，还具有一些特殊字段，例如为区分区块头所属链的区块链 ID 字段以及为验证最终提交区块头合法性的门限签名字段等。具体的抽象区块头数据结构如表 4.2 所示。

表 4.2　抽象区块头结构体定义

```
1   //golang
2   type AbstractBlockHeader struct {
3       ChainId string   //区块链 ID
4       Height int   //区块头高度
5       Hash []byte   //区块 hash
6       PreHash []byte   //前一区块 hash
7       MerkleRoot []byte   //区块 merkle 树根 hash
8       Sign ThresholdSignature   //签名
9   }   //抽象区块头
10  type ThresholdSignature struct{
11      Threshold int   //阈值
12      PublicKey []byte   //公钥
13      Signature []byte   //最终签名
14      Submitter []byte   //提交者签名
15      Shares [][]byte   //达到门限阈值时参与签名者集合
16  }   //门限签名
```

在发送跨链交易之前，源链需要将跨链交易打包成一个可以被对方链解析的抽象交易，而抽象交易除了包含一般交易的常见字段，例如交易 ID、时间戳、源账户、目标账户等，还有一些特殊字段，例如标识跨链源链和目标链的源链 ID 和目标链 ID，为确保交易超时及时回滚而设置的超时字段，为进行默克尔验证而设置的交易默克尔证明字段等。具体的抽象交易数据结构如表 4.3 所示。

表 4.3　抽象交易结构体定义

```
//golang
//抽象交易
type AbstractTran struct {
    SourceChainId string   //源链链 ID
    DestChainId string   //目标链链 ID
    TimeStamp string   //时间戳
    TimeOut string   //超时设置
    UserCertificate []byte   //交易发起用户证书
    TransId []byte   //交易 Id
    Type string   //交易类型
    Status string   //交易状态
    From string   //源账户
    To string   //目标账户
    Value int   //交易额
    Param CrossTranParam   //交易参数
    Proof MerkleProof   //交易默克尔证明
}
//交易参数
type CrossTranParam struct {
    ContractName string   //智能合约名
    ContractFunc string   //智能合约函数名
    ContractArgs []string   //调用参数
}
```

目的链向源链发送跨链请求执行结果之前，需要将执行结果打包成源链可解析的抽象回执，抽象回执大致类似于抽象交易的域设置，主要是添加了回执域，回执域中除了本次交易的响应结果之外，还包含对结果的签名，回执域的数据结构见表 4.4。

表 4.4　回执域的数据结构

```
1  //golang
2  type CrossTranResp struct {
3      Data []byte   //响应结果
4      Signature []byte   //签名
5  }
```

4.1.3　数据互信机制

联盟链的跨链应用场景一般可以归纳为三类，分别是跨链转账、跨链读以及跨链写。其中跨链转账和跨链写会在目标链上发起新交易，最终改变目标链的链上状态。而目标链处理跨链读请求则会直接调用链上 query 原语，不会改变链上状态。综上，本协议依据跨链操作是否会改变目标链的链上状态将跨链交易分为两组：第一组包含跨链转账交易以及跨链写交易，该组交易会改变目标链的链上状态；第二组为跨链读交易，该组交易不会改变目标链的链上状态。本文在跨链系统的设计中，针对这两组交易分别设计了跨链数据可信机制。

（1）目标链的链上状态会改变

跨链转账交易以及跨链写交易，它们本质上都是在源链或目标链上发起新的子交易，并且改变了链上的状态，跨链交易是否能够正常执行取决于子交易是否已成功执行。因此，如何使对方链信任子交易确实在本链上产生了，并且得到共识上链了，是解决数据可信问题的核心。本跨链协议所设计的第一种跨链数据可信机制就是解决这一类跨链场景下的数据信任问题，该机制基于默克尔证明机制实现。

默克尔证明机制可以使跨链一方在不需要获取另一方全量链上数据的情况下，仍然能够快速证明另一方链上指定交易的真实存在性。该跨链场景下使用默克尔证明机制有一个前提和两个出发点：前提是参与跨链的联盟链一般都需要支持智能合约，满足生成外链轻客户端的技术条件；出发点之一是任何跨链转账交易都得确保交易双方在各自的链上拥有所宣称的资产；出发点之二是跨链交易的结束需要确保跨链事务的各个步骤已在各自链上得到正确的执行和上链。

为实现默克尔证明机制，本协议在设计抽象交易与抽象回执结构体时增加了默克尔证明字段，并且定义了相关的接口。默克尔证明字段及接口定义如表 4.5 与 4.6 所示。

表 4.5　默克尔证明字段结构体定义

```
1      type MerkleProof struct {
```

续表 4.5　默克尔证明字段结构体定义

```
2        MerklePath [][]byte   //默克尔验证路径
3        TransHash []byte   //交易 hash
4        Height int   //交易所在区块高度
5        MerkleIndex []int64   //默克尔验证 index
6    }
```

表 4.6　默克尔证明接口定义

```
1    type MerkleProve interface{
2        NewMerkleTree()   //生成 merkle 树
3        GetMerklePath()   //生成交易 merkle path
4        VerifyMerkleProof()   //默克尔验证
5    }
```

（2）目标链的链上状态未改变

跨链读交易因为只涉及到对对方链上指定数据的读取，所以对方链上不需要发起新的交易来完成此次跨链读交易。该场景下的数据可信机制基于门限签名机制实现。首先，目标链处理跨链读请求的服务节点在本链直接调用 query()原语，获取到读结果，然后该服务节点将查询请求以及查询结果进行广播，最后，其他服务节点对查询结果进行门限签名。其他服务节点将自身调用 query()原语获取到查询结果，并对比本地查询结果与接收到的广播结果，两者一致则进行签名并将签名发送给广播节点。广播节点收集到满足门限阈值的签名后生成最终的签名，基于该签名、节点自身签名以及收集到的签名列表进行回执打包。

综上，基于门限签名的数据可信机制有如下特点：

a.避免了在目标链上新增新的查询交易，只需要调用 query()原语，具有高效性。

b.采用去中心化的方案保证查询结果在链外的可信性验证，且通过打包回执时附带节点自身签名以及参与门限签名的参与者签名列表，可以起到预防节点组团作恶对读结果进行造假的作用，并为事后追责提供支持。

4.1.4　跨链数据隐私保护机制

联盟链上的数据只对链内组织成员公开，联盟链外数据是非公开的，应当受到隐私保护，因此本文中跨链协议设计了跨链数据隐私保护机制，该机制核心内容如下：

（1）在定义跨链交易结构体时增加发起交易用户的证书字段 UserCertificate，用户证书中包含了用户的公钥。

（2）源链在发送跨链交易前将跨链交易数据打包成抽象交易，其中应包括发起跨链交易用户的证书。

（3）目标链将执行结果打包成抽象回执后发送，若本次跨链交易为跨链读交易，则目标链需要基于发起交易用户证书中的公钥对跨链读结果进行加密，并将密文打包进抽象回执。

（4）源链发起跨链读交易的用户在接收到经过加密的跨链读结果后，基于自身私钥对加密结果进行解密，最终得到跨链读结果。

4.1.5　异常处理机制

跨链的安全性和稳定性受多方面因素的影响，例如中继连接的稳定性、作恶节点的攻击等。本跨链协议要求跨链系统具备标准的异常处理机制，面对可能发生的常见异常情况可以及时处理与解决。异常处理机制的设计如下：

首先，对每一笔跨链交易都进行超时设置，交易超时后全部参与方进行回滚。其次，在链上部署新的智能合约时，对写函数提出规范要求：任何可能会被跨链调用的写函数都应配套编写对应的回滚函数。在跨链调用目标链写函数时，目标链在接收到跨链写请求后需要进行回滚函数存在性检测，若目标写函数无对应的回滚函数则应直接返回跨链失败。最后，针对资产的跨链转移，源链上的跨链转账合约需要实现 Lock 与 UnLock 接口。用户转移资产将先被锁定在锁定账户中，若本次跨链转账完成则焚毁锁定账户资产，若本次跨链转账发生异常，则需要解锁锁定账户中的资金，将资金退回至源账户。

4.2　系统架构设计

本文所设计跨链系统的架构主要分为数据层、交互层、事务层及治理层四层，如图 4.1 所示。

图 4.1　跨链系统架构图

4.2.1　数据层

跨链交互的本质是数据在链间的传输，所以数据层的数据抽象尤为重要。为实现跨链协议中所设计的数据互认机制，本文在数据层设计了三个模块，依次是抽象区块头存储模块、抽象交易打包模块、抽象回执打包模块。基于该层，跨链系统设计类图如图 4.2 所示。

抽象区块头模块主要功能是将本链的区块头按照跨链协议中抽象区块头的数据结构定义进行抽象后进行存储，本链节点可以通过该模块获取到本链指定区块高度区间内的抽象区块头，为交互层的区块头同步模块提供支持。抽象交易打包模块实现的主要功能是将本链发起的跨链交易按照跨链协议中抽象交易的数据结构定义进行打包。抽象回执打包模块的主要功能为将本链执行完跨链请求后的结果按照跨链协议中抽象回执的数据结构定义并进行打包。

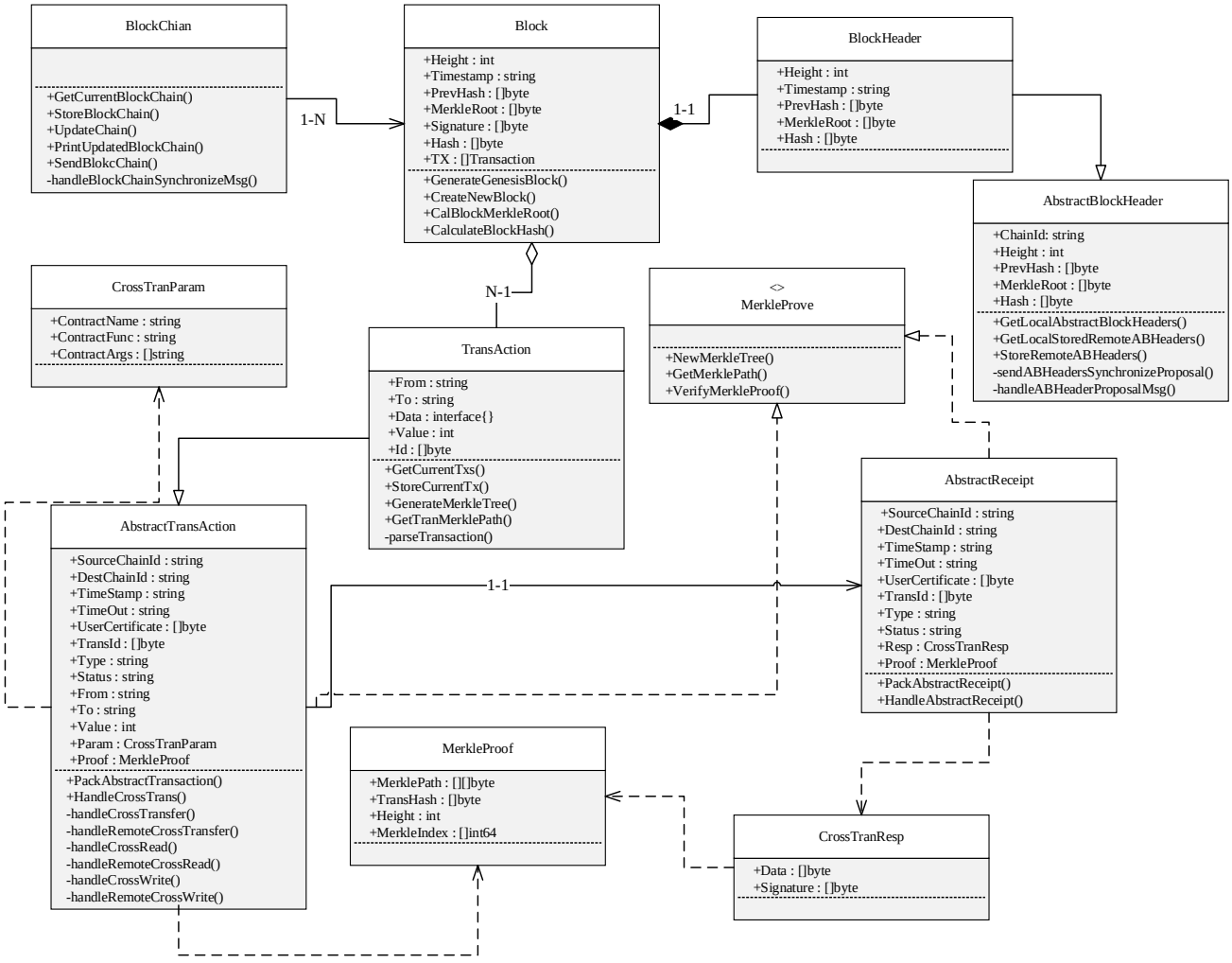

图 4.2　跨链系统类图

4.2.2　交互层

交互层作为本研究中跨链系统的核心，由区块头同步模块、跨链寻址模块、交易验证模块三个模块组成，目的在于解决跨链数据交互及跨链数据可信的问题。

区块头同步模块的主要功能是实现跨链双方同步对方链的区块头以在本链构建对方链的轻客户端的需求，为交易验证模块提供支持。跨链寻址模块的主要功能为：解析用户发起的跨链请求并定位到目标链的通信地址，实现跨链请求的转发操作。交易验证模块的主要功能是为本链的跨链交易生成交易验证证明信息，并基于对方链传输的证明信息对跨链交易进行合法性验证。

4.2.3　事务层

事务层的设计是为了保证跨链事务的原子性，主要包括跨链资产锁定机制及回滚机制两个模块。

在跨链资产转移的应用场景下，跨链转账合约在跨链交易成功结束前将用户转出资产锁定于指定账户中，交易成功执行后焚毁锁定资产，若交易失败跨链转账合约将锁定资产解锁并退回交易发起用户账户中。回滚机制的设计是在跨链操作异常中断时，保证已发生的链上状态的更新全部回滚，恢复原状态。

4.2.4 治理层

治理层的设计是实现跨链系统的有效治理的基本保障，主要分成日志模块、追责机制、权限管理以及监管机制四个模块。

日志模块的主要功能是记录跨链事务的每一步操作，并且在事务完成时生成供监管节点审计的加密监管日志，为监管机制提供支持。在发现跨链系统存在恶意行为时，追责机制的设计保证了迅速定位作恶节点并及时作出应对措施。权限管理主要是对用户跨链操作的不同粒度的授权与权限审核。监管机制的实现基础是日志模块的实现，监管节点通过同步监管日志达到对链间跨链交易审计的目的。

4.3 本章小结

本章对所设计的跨链协议与系统架构做了详细的阐述说明。针对跨链协议，以跨链注册机制的介绍为起点，从背景及具体设计两个角度对数据的互认、互信及隐私保护机制进行了剖析，最后以异常处理机制的介绍为终点，全面立体地展现了本文中的跨链协议；通过对各层组成模块的模块功能分析，完成了对系统架构的说明。

第五章　系统详细设计与实现

5.1　数据层

数据层的实现主要包括抽象区块头存储模块、抽象交易打包模块及抽象回执打包模块，接下来将分节对此三个模块进行详细说明。

5.1.1　抽象区块头存储模块

在跨链系统的设计中为了实现联盟链间数据互认，通过定义抽象区块头的数据结构，对链与链之间同步的区块头做了一层抽象。在跨链协议中规定了，参与区块头同步的双方，必须先将本链区块头按照协议定义的抽象区块头的结构进行抽象处理后，才能发送给对方链。为了将本链区块头抽象成满足跨链协议规定的抽象区块头的格式要求，本文设计实现了抽象区块头存储模块，该模块的实现主要通过在链上部署抽象区块头存储合约，合约的主要函数如表 5.1 所示。

表 5.1　抽象区块头存储合约的主要函数列表

方法	描述
GetLocalAbstractHeaders()	获取到本链指定高度区间内的抽象区块头列表
GenerateLocalAbstractHeaders()	生成本链指定高度区间内的抽象区块头列表
GenerateLocalNewAbstractHeader()	生成本链最新区块对应的抽象区块头

在区块链启动后的初始化阶段，调用 GenerateLocalAbstractHeaders()函数生成本链创世区块头所对应的创世抽象区块头。该合约监听链上新区块生成事件，一旦有新区块的生成将触发调用 GenerateLocalNewAbstractHeader()函数，生成该新区块对应的抽象区块头并存储。抽象区块头的存储方式与本链区块头的存储方式类似，都是基于前一区块头的 hash 值维护一个不断增长的区块头链，新增抽象区块头的 PreBlockHash 指向前一抽象区块头。该模块为交互层的区块头同步模块提供支持，进行区块同步的服务节点调用 GetLocalAbstractHeaders()函数，获取到本链指定高度区间内的抽象区块头列表，将抽象区块头列表打包成区块头同步消息发送给对方链的服务节点。

5.1.2　抽象交易打包模块

为实现链间的数据互认，本文的跨链协议还对链与链之间传输的交易进行了一层抽象，并给出了抽象交易的数据结构定义。跨链协议规定，源链在将本链发起的跨链交

易发送至目标链之前，需要按照抽象交易的数据结构要求将跨链交易进行封装打包。因此，本文基于 PackCrossAbstractTransaction()函数的实现设计实现了抽象交易打包模块，函数源代码如图 5.1 所示。

```
func PackCrossAbstractTransaction(t meta.CrossTran, height int, sequence int) meta.CrossTran {
    var pf meta.CrossTranProof
    var txs []meta.Transaction
    txs=make([]meta.Transaction,0)
    if t.Type==commoncon.CrossTranTransferType{
        //首先根据区块高度获取到指定的区块
        cBcs := GetCurrentBlockChain()
        bc := cBcs[height]
        //获取到区块中所有的交易
        txs = bc.TX
        //生成该交易的merkle proof
        tranHash, merklePath, merkleIndex := merkle.GetTranMerklePath(txs, sequence)
        pf = meta.CrossTranProof{
            MerklePath:  merklePath,
            TransHash:   tranHash,
            Height:      height,
            MerkleIndex: merkleIndex,
        }
    }
    var ct meta.CrossTran
    uc, _ := json.Marshal(*util.LocalPublicKey)
    ct = meta.CrossTran{
        SourceChainId:   t.SourceChainId,
        DestChainId:     t.DestChainId,
        TimeStamp:       time.Now().String(),
        TimeOut:         time.Now().Add(time.Hour).String(),
        UserCertificate: uc,
        TransId:         txs[sequence].Id,
        Type:            t.Type,
        Status:          commoncon.StatusDeal,
        Param:           meta.CrossTranParam{},
        Proof:           pf,
    }
    return ct
}
```

图 5.1　PackCrossAbstractTransaction()源代码

基于 PackCrossAbstractTransaction()函数的一次抽象交易打包流程如图 5.2 所示。具体的抽象交易打包操作流程说明如下：

Step1：判别此次打包的交易是否为跨链转账交易。

Step2：若不是跨链转账交易则执行 Step3，否则执行步骤 Step4。

Step3：基于以入参传进函数的交易信息，将抽象交易结构体的部分属性如源账户 From 进行填充并对剩余属性，如超时设置 TimeOut 进行赋值，生成抽象交易后，执行 Step9。

Step4：调用 GetCurrentBlockChain()函数获取当前的区块链数据。

Step5：根据传进参数 Height 获取跨链资产锁定交易所在区块。

Step6：调用 GetBlockTransactions()函数获取指定区块中的交易列表。

Step7：调用 Merkle 验证模块中的 GetMerklePath()函数，传入交易列表以及交易序号 sequence 生成 Merkle Proof。

Step8：基于以入参传进函数的交易信息以及生成的 Merkle Proof 生成抽象交易。

Step9：返回打包完成的抽象交易。

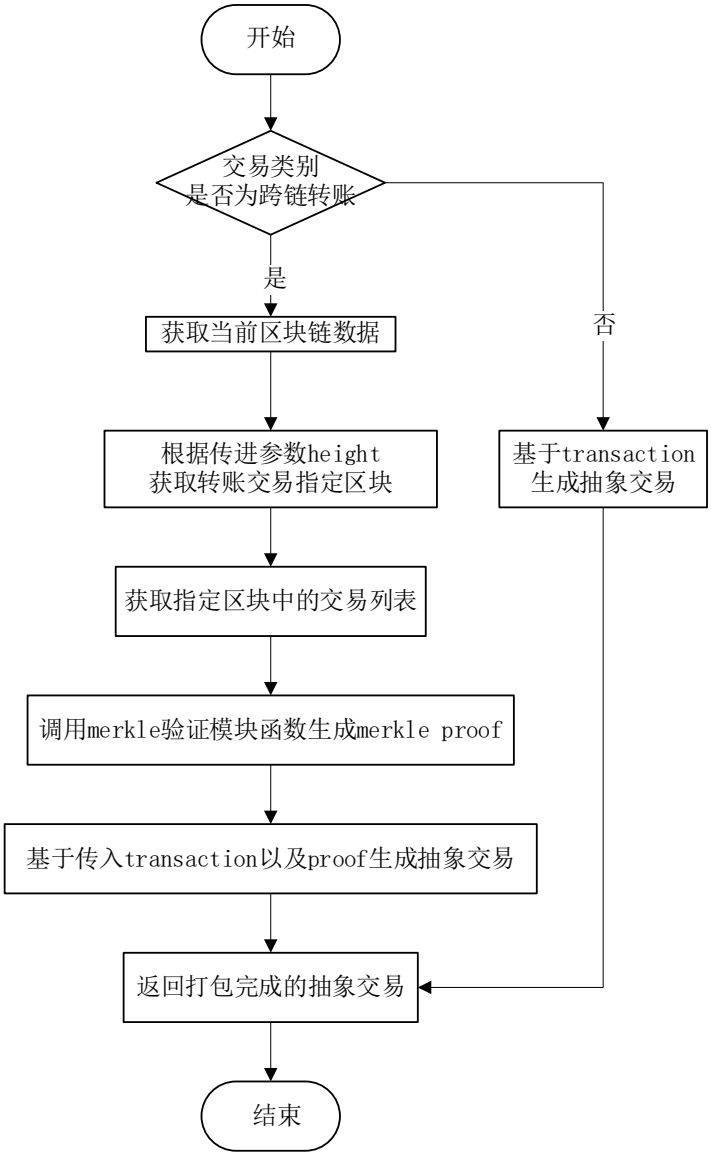

图 5.2　抽象交易打包流程图

5.1.3 抽象回执打包模块

为实现数据互认，除了对链与链之间传输的交易以及区块头的结构体进行规范，跨链协议还通过定义抽象回执的数据结构对执行结果的传输作出了规范。跨链协议规定：目标链在执行完跨链请求后需要按照抽象回执的格式，将执行结果进行打包。该功能是通过抽象回执打包模块实现的，且由于跨链协议中对抽象回执的结构体定义与对抽象交易的结构体定义大致相同，抽象回执打包功能的实现与抽象交易打包功能的实现基本类似，核心步骤都是生成本链新交易的默克尔证明。由于跨链读交易并未在目标链上新增新的交易，故目标链打包回执时无需生成默克尔证明，但是需要基于源链上发起跨链读交易的用户的公钥对读结果进行加密，并基于密文、跨链读结果的门限签名打包生成跨链回执。若跨链交易并非跨链读交易，则打包流程与跨链转账交易的打包流程基本一致，需要生成 Merkle Proof 后再进行打包，一次抽象回执的打包流程如图 5.3 所示。

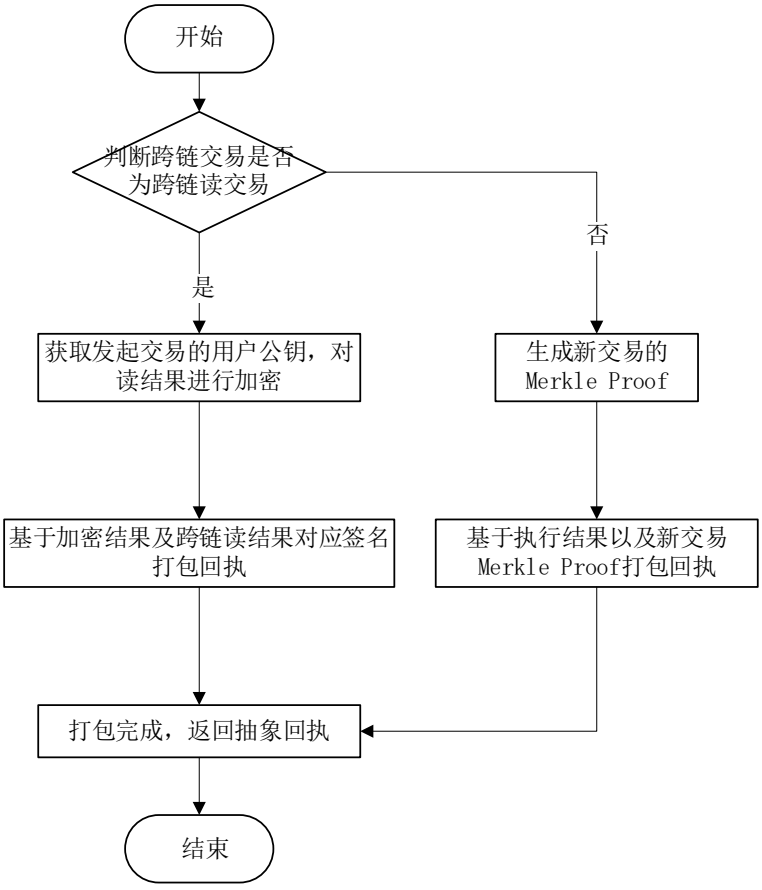

图 5.3 抽象回执打包流程图

具体的抽象回执打包操作流程说明如下：

Step1：判断此次跨链交易是否为跨链读交易。

Step2：若不是跨链读交易，则生成本链新交易的 Merkle Proof，基于执行结果以及新交易的 Merkle Proof 打包回执后，执行 Step5；若是跨链读交易，则执行 Step3。

Step3：获取发起交易用户的公钥，对跨链读结果进行加密。

Step4：基于加密结果及跨链读结果的签名打包回执。

Step5：打包完成，返回抽象回执。

5.2　交互层

交互层主要是实现该层中区块头同步模块、跨链寻址模块、交易验证模块三个模块的功能。

5.2.1　区块头同步模块

本跨链系统实现跨链数据可信的基础是默克尔证明机制，而默克尔证明机制的实现依赖于所跨目标链的区块头信息，所以如何在本链上获取到目标链的区块头信息，是实现默克尔证明机制的关键，在本文中将该问题简称为区块头同步问题。目前，关于区块头同步问题的一个解决方案是 BTC Relay，该方案实现了在以太坊上获取比特币网络的区块头信息，具体的实现方式如下：以太坊节点中部分节点充当比特币区块头的提交者（relayer），并在以太坊上部署了一个保存 relayer 提交信息的智能合约。当有 relayer 提交新的区块头信息时，智能合约基于简单的验证函数即可验证所提交信息的合法性，若信息合法则提交成功。总结 BTC Relay 的区块头同步方案发现，该方案无需任何第三方的参与，并且能够保证已提交区块头的合法性。

但是区块链网络中并非每个节点都愿意主动充当 relayer，也无法保证每一个 relayer 都能诚实提交正确的区块头信息，因此提交节点的积极参与以及提交信息的合法性验证是区块头同步问题的两大核心问题。BTC Relay 的简单区块头同步机制之所以能够保证区块头信息的积极提交以及提交信息的合法性验证，是通过给予提交者一定的手续费激励，以及 BTC 所基于的 PoW 工作量证明共识机制和最长链准则[45]。当有新区块头信息提交时，验证函数首先会取出该区块的父区块，从而量定所提交区块头对应的累积工作量。若累积工作量大于目前智能合约中保存的已有区块头的最大累积工作量，那么该区块头保留，但是，不满足最大累积工作量时，该区块也不会立即丢弃。因为 BTC 共识机制决定了暂时的分叉现象很常见，该区块所在分支未来仍可能会发展成为最长的主链。当新区块信息经过验证并在最长链上时，每次调用该区块头信息进行默克尔验证都会付给提交者一笔手续费。若提交者作恶，虽然虚假区块头不会第一时

间被检验非法丢弃，但是随着新的区块头信息的不断提交，虚假区块头因为不在最长链上被淘汰丢弃。这种验证机制的本质在于 BTC 是基于 PoW 共识的，每一个新区块的生成都需要耗费大量的算力，而个人的算力无法与整个区块链网络的算力匹敌，故最长链准则会逐渐淘汰掉虚假区块。

对于联盟链来说，链上一般没有原生币，而且链上交易一般无手续费，所以直接采用给予手续费的方式，去激励链上节点主动提交区块头信息的方法行不通。再者，即使有节点主动提交外链区块头信息，一般联盟链的共识机制都不是工作量证明机制，比如 ssbc 采用的是高效的 PBFT 共识，那么仅靠合约本身的函数功能很难验证一个外链的区块头是否合法。所以，针对以上两个难题，结合联盟链的特性，本系统设计实现了基于可验证随机函数以及门限签名的预言机，并基于预言机机制实现了区块头同步方案，该方案的详细实现如下：

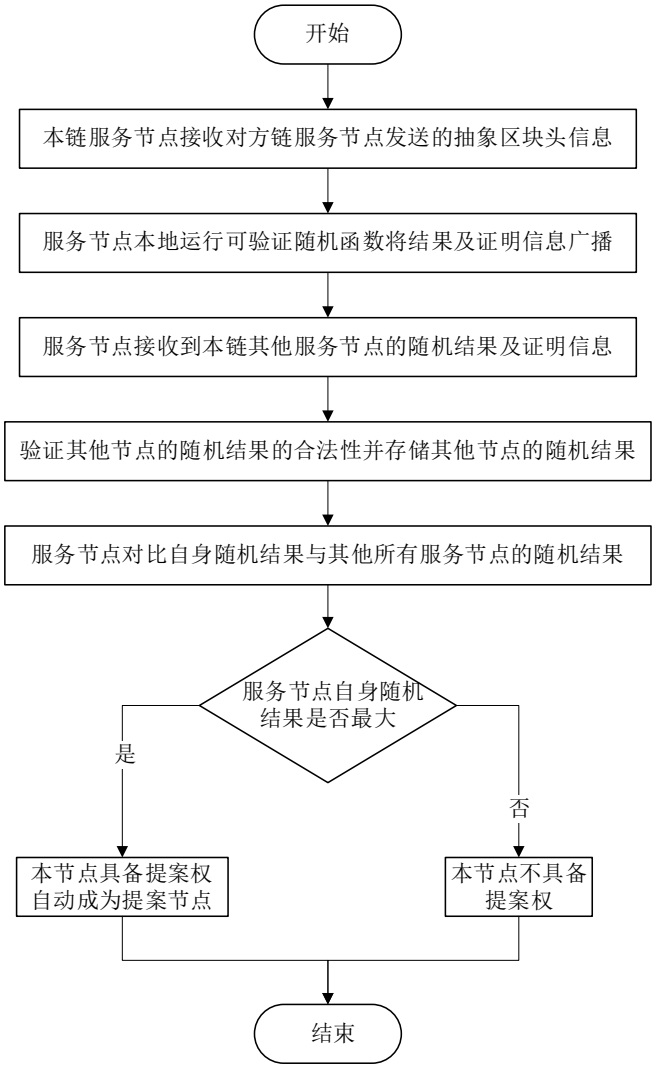

图 5.4　选取提案节点流程图

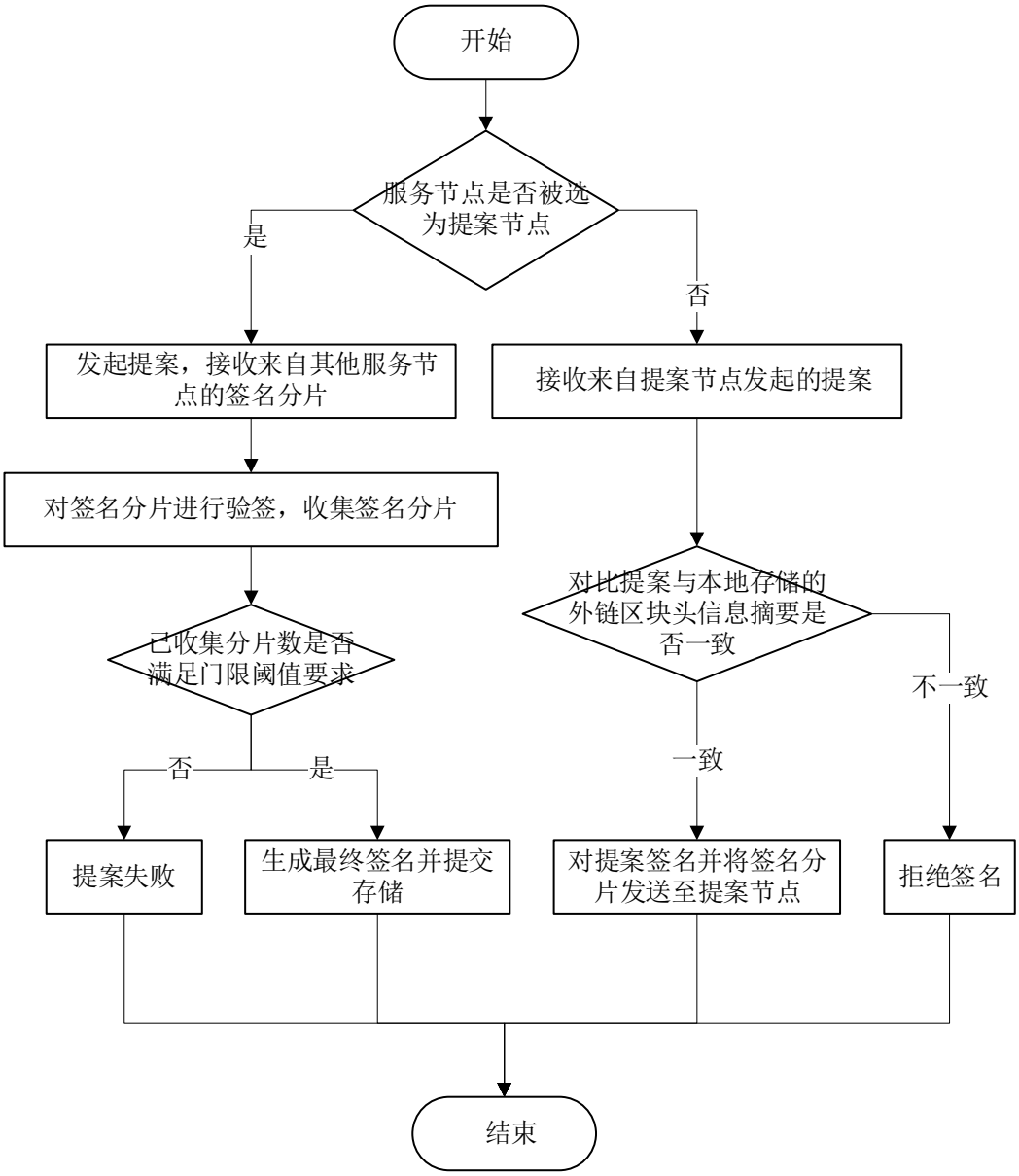

图 5.5　对提案进行门限签名流程图

5.2.2　跨链寻址模块

不同的联盟链调用本链上资源的方式各不相同，本链处理用户跨链请求的第一步就是解析出跨链请求的类型，并定位到请求所对应的目标链以及目标资源。本系统设计的跨链寻址模块主要由两个部分组成：第一个部分是跨链寻址协议，跨链寻址协议规范了用户发起的跨链请求的格式，所有参与跨链的用户将按照统一的资源请求的格式发起跨链请求，所有参与跨链的联盟链将按照该统一格式对用户的跨链请求进行解析处理。第二部分是链名解析合约，链名解析合约是部署在本链上的一个可以针对基于跨链寻址协议生成的跨链请求进行解析的智能合约。通过这两个部分的实现，用户

将发起标准格式的跨链资源请求，本链将请求解析处理后打包成跨链交易发送至目标链。

跨链寻址协议如下：

[chain_meta:][//chain_name?][type][contract_name][func_name][req_args]，各字段定义如表 5.2 所示。

表 5.2　字段定义说明表

方法	描述
chain_meta	所跨联盟链底层实现标识符，该字段用于区分所跨链类型，用于判别是否为同构链
chain_name	所跨联盟链的链名
type	本次跨链请求类型
contract_name	本次跨链请求合约名
func_name	本次跨链请求合约中函数名
req_args	本次跨链请求函数参数

例如底层基于 ssbc 实现的联盟链 ssbcA 上部署了一个管理学生成绩的智能合约 SocreManagement，该智能合约对外暴露的可以支持跨链调用的函数为获取学生成绩 GetStudentScore()，一个在同样是底层基于 ssbc 实现的联盟链上的具备跨链调用 ssbcA 所有智能合约读函数权限的用户，发起了一笔查询学生 A 成绩的跨链读请求，该请求应如下：

ssbc://ssbcA?type=query&contractName=SocreManagement&func_name=GetStudentScore&req_args={"name":"StudentA"}。

链名解析合约的主要功能为：在本链注册新的参与跨链的联盟链、对本链用户发起的跨链请求进行解析，定位到目标链的通信地址以及对已经注册过的跨链信息进行更新等，链名解析合约的主要函数如表 5.3 所示。

表 5.3　链名解析合约主要函数列表

方法	描述
RegisterNewChain()	注册一个新的跨链网络
UpdateChainMeta()	对已注册的跨链信息进行更新
ResolveChain()	根据链名解析出目标链的通信地址集
AddNewRelayer()	目标链跨链中继节点集合新增
DeleteRelayer()	目标链跨链中继节点集合删减

续表 5.3　链名解析合约主要函数列表

AddNewServer()	目标链跨链服务节点集合新增
DeleteServer()	目标链跨链服务节点集合删减

5.2.3　交易验证模块

本跨链系统基于默克尔证明以及门限签名机制实现了跨链数据的可信性，目标链接收到来自源链的跨链交易以及源链接收到来自目标链的交易回执时，都需要对交易以及回执进行合法性验证，本文将该过程称之为交易验证，为实现该过程，本系统在交互层设计实现了交易验证模块。一个跨链事务的执行通常分解为多个步骤，比如跨链转账交易，只有前置操作即源链跨链资产锁定完成，后置的操作即目标链新铸资产转给目标账户才会执行。所以交易验证模块在一定程度上支持了跨链事务的原子性。另外，跨链消息在传输过程中可能会被截取篡改，交易验证模块不但保证了数据的可信性还保证了消息的真实性。

本模块是基于默克尔证明机制以及门限签名机制实现，主要函数列表如表 5.4 所示。

表 5.4　交易验证主要函数列表

方法	描述
parseCrossTrans()	解析跨链交易
GenerateNewMerkleTree()	生成新的 merkle 树
GetTranMerklePath()	生成一笔交易的 merkle path
VerifyMerkleProof()	基于交易中的 merkle path 与 merkle root 进行交易验证
VerifyThsProof()	对交易中的门限签名进行验证

源链打包跨链交易发送至目标链，目标链进行交易验证，流程图如图 5.6 所示。

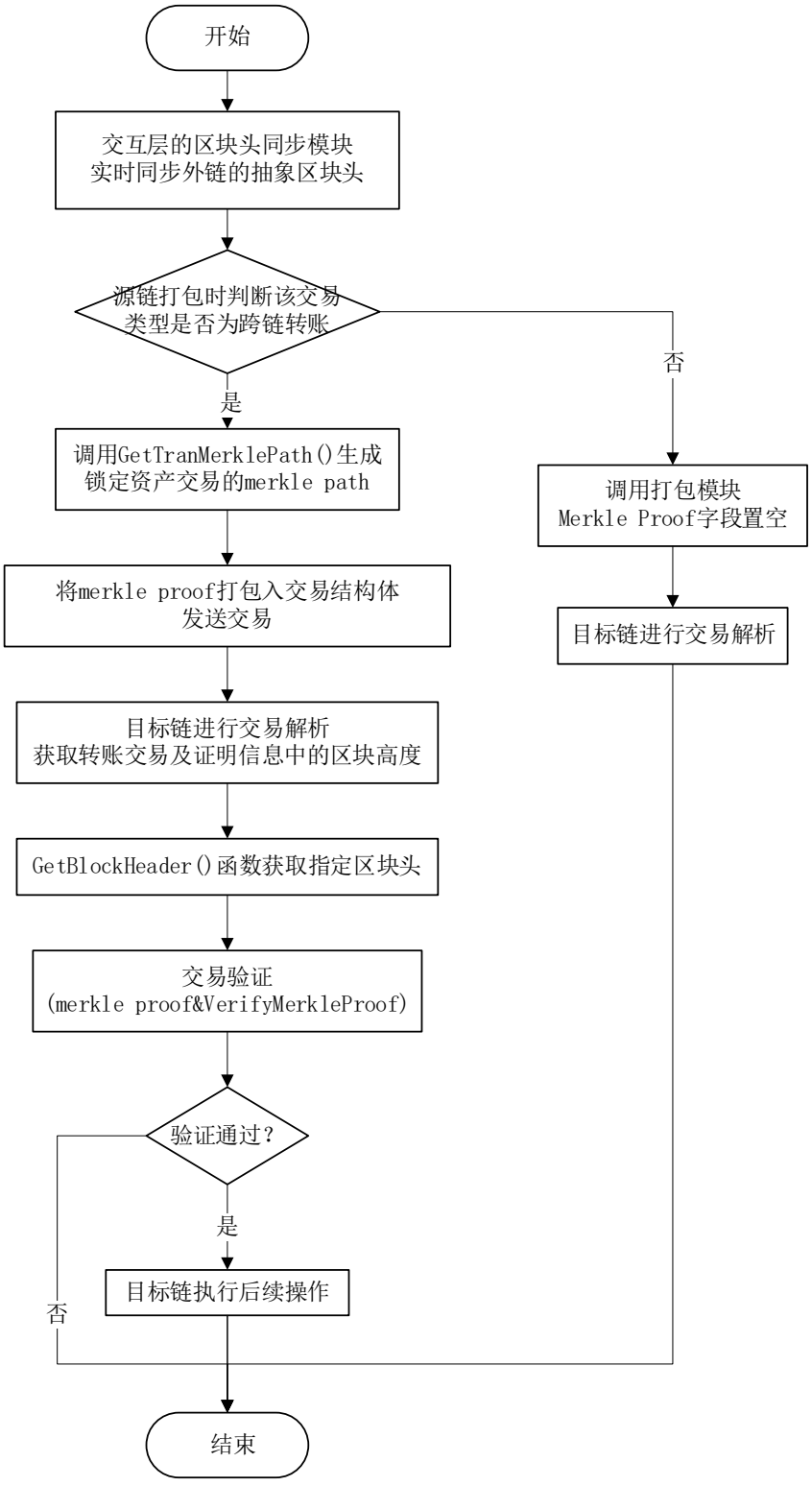

图 5.6　交易验证流程图

目标链打包回执发送至源链，源链进行回执验证的流程操作如图 5.7 所示。

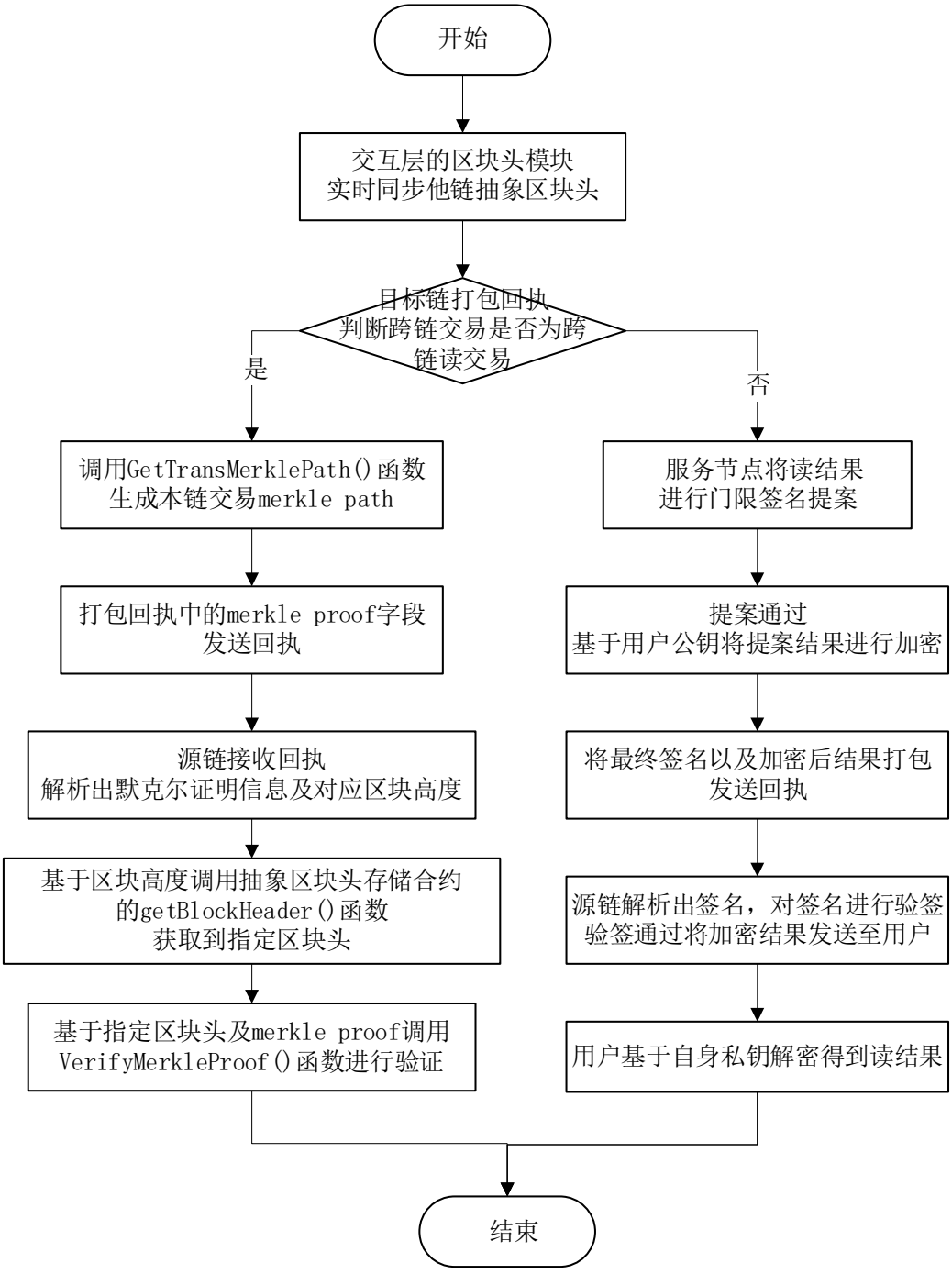

图 5.7　回执验证流程图

5.3　事务层

本文的跨链系统针对不同的跨链场景，设计了针对性的事务机制来保证跨链事务的原子性。其中，针对跨链转账场景，设计实现了跨链资产锁定机制来保证跨链转账的

原子性；针对跨链写场景，设计实现了回滚机制来保证跨链写交易的原子性。

5.3.1 跨链资产锁定机制

跨链资产锁定机制是通过在链上部署跨链转账合约实现的，该合约使用的主要函数如表 5.5 所示。

表 5.5 跨链资产锁定机制主要函数列表

方法	描述
LockAssets()	将跨链转移资产锁定至锁定账户
UnlockAssets()	解锁锁定资产，将资产退回到源账户
Burn()	焚毁锁定资产
Cast()	新铸资产
Transfer()	转移资产

按照跨链协议的规定，源链用户发起一笔跨链转账交易时，首先调用 LockAssets()函数将转移资产锁定至锁定账户，待该锁定资产交易共识上链，源链将跨链交易进行打包发送至目标链。目标链进行默克尔验证，验证通过后调用 Cast()函数新铸资产，并调用 Transfer()函数将新铸资产转账至目标账户，待转账交易共识上链后，目标链打包回执发送至源链，源链对回执进行默克尔验证，验证通过则调用 Burn()函数焚毁锁定资产。在跨链转账过程中若发生异常如等待目标链回执超时，则源链调用 UnlockAssets()函数将锁定资产解锁并退回到源账户。通过资产锁定机制的实现，保证了跨链转账事务的原子性。

5.3.2 回滚机制

跨链协议规定，参与跨链的联盟链上所有智能合约的写函数，若需要被链外调用，则必须有配套编写的回滚函数，无配套回滚函数的写函数必须限制不允许被外链调用。这是为了一旦跨链写交易发生异常，可以通过所有跨链操作回滚的方式，确保跨链写事务的原子性。以学生成绩管理合约为例：合约中包括了一个读函数 GetStudentScore(studentID)，函数功能为读取指定学生的成绩，一个写函数 UpdateStudentScore(studentID,newScore)，该函数功能为更新学生成绩，则该写函数需要编写配套的回滚函数才能被外链调用。

首先，编写获取最新改动前的成绩的函数 GetLastUpdatedScore(studentID)，该函数功能为每次调用 UpdateStudentScore(studentID,newScore)会触发 GetStudentScore(studentID)函数的执行，获取到最近更新前的学生成绩并将该成绩保存。其次，编写回滚函数

UpdateRollBack(studentID)，执行回滚函数首先会调用 GetLastUpdatedScore(studentID)函数获取到最近更新前的成绩，然后调用 UpdateStudentScore(studentID,newScore)函数，将 newScore 参数赋值为最近更新前的成绩，则回滚函数执行完成。只有 UpdateStudentScore(studentID,newScore)函数配套编写了 UpdateRollBack(studentID)回滚函数，且回滚函数的触发机制设定为异常回滚触发，那么 UpdateStudentScore(studentID,newScore)函数才能被链外有写权限的用户调用。通过该回滚机制的实现，保证了跨链写事务的原子性。

5.4　治理层

治理层主要是实现本层的日志模块、追责机制、权限管理以及监管机制四个模块的功能。

5.4.1　日志模块

本层实现的日志模块不但能为跨链事务的原子性提供支持，而且能为本系统设计的追责机制提供支持。

日志模块的实现主要是在链上部署了跨链事务日志记录合约。合约方法主要有两个：LogStore(transactionID,stepID,Operation,status)函数以及 LogQuery(transactionID)函数。跨链交易处理的关键步骤都会调用日志记录合约的 LogStore()函数进行日志存储，基于事务的唯一 ID 记录下事务操作每一步的状态。

一个典型的事务记录如表 5.6 所示。

表 5.6　事务记录样例

TransactionID	StepID	Operation	Status	Extra
3deds1w4s	1	Chain1 节点 A 发起对 Chain2 合约 C 的跨链读请求	已完成	
3deds1w4s	2	Chain1 打包跨链交易完成，向 Chain2 发送	已完成	
3deds1w4s	3	Chain1 接收到 Chain2 回执，验证回执	已完成	

5.4.2　追责机制

本系统通过实现去中心化的预言机以及门限签名机制解决了跨链流程参与节点的单节点信任风险问题。但是，若联盟链中作恶节点的个数超出阈值，则仍然会使得作恶行为成功。

本系统有两个场景可能会出现作恶行为成功，一个场景为：区块头同步场景，若参与同步的服务节点中作恶节点的个数大于门限签名的阈值，则服务节点可能提交篡改后的假区块成功。另一个可能出现作恶的场景为：跨链读结果提交。若参与跨链读结果签名的节点中，作恶节点的个数大于门限签名的阈值，则服务节点可能提交篡改后的查询结果成功。总结发现这两种作恶场景的本质相同，均为作恶节点数与门限签名阈值出现失衡，在这种情况下门限签名机制失效，即无法保证跨链数据的可信性。

为实现这种场景下的追责机制，跨链协议在抽象区块头以及跨链回执的数据结构设计上增加了门限签名属性，该属性包括了最终签名、提交者签名以及参与者签名列表。一旦发现提交区块为假区块或者跨链读结果被篡改，可以根据提交者签名以及参与者签名列表追溯到作恶节点，及时对作恶节点做出问责惩罚措施。

5.4.3 权限管理

本系统通过在链上部署权限管理合约，实现对跨链请求的权限管理。权限管理合约从用户及链上资源两个维度维护了用户与资源间的权限关系。

从用户角度出发，权限管理合约记录了本链上的节点有权限访问哪些外链的具体哪些资源，用户维度的权限记录如表 5.7 所示。

表 5.7　用户维度的权限记录

用户 ID	资源	是否具备权限	说明
0x123	ssbc.score.*	Y	本链用户 0x123 对于 ssbc 链上的 score 合约的所有函数均具备调用权限
0x234	ssbc.score.getScore()	Y	本链用户 0x234 对于 ssbc 链上的 score 合约的读函数 getScore()具备读权限
0x345	ssbc.score.setScore()	N	本链用户 0x345 对于 ssbc 链上的 score 合约的写函数 setScore()不具备写权限

从链上资源角度出发，权限管理合约记录了本链上的资源针对不同链外节点开放不同的权限，链上资源维度的权限记录如表 5.8 所示。

表 5.8　资源维度的权限记录

资源	用户 ID	说明
score.*	ssbc.NodeA	本链 socre 合约的所有函数都能被 ssbc 链的 A 节点调用

续表 5.8　资源维度的权限记录

score.getScroe()	ssbc.NodeB	本链 score 合约的读函数 getScore()能被 ssbc 链的 B 节点调用
score.setScore()	ssbc.NodeC	本链 score 合约的写函数 setScore()能被 ssbc 链的 C 节点调用

权限管理合约的主要函数如表 5.9 所示。

表 5.9　权限管理的主要函数列表

方法	描述
ApplicatePermission()	权限申请
RegisterPermission()	权限登记
RemovePermission()	权限移除
QueryPermission()	权限查询

权限信息结构体定义如表 5.10 所示。

表 5.10　权限信息结构体

字段	类型	描述
ChainId	string	链 ID
NodeId	string	节点 ID
Source	string	链上资源

5.4.4　监管机制

为了实现对跨链交易的有效监管，本系统在治理层设计实现了监管机制。监管机制的实现主要依赖日志模块的支持，在日志记录合约中新增函数 GenerateSupervisionLog()，该函数在每次交易事务完成后触发，该函数生成特定格式的监管日志，监管日志结构如下表 5.11 所示。

表 5.11　监管日志结构

字段	描述
crossTransId	跨链交易唯一 ID
sourceChain	源链
destChain	目标链
transData.request	交易请求

续表 5.11　监管日志结构

transData.type	交易类别
transData.params	交易参数
transData.resp	交易结果
transData.signatures	签名列表
timestamp	时间戳

生成的监管日志将进行加密处理，监管节点可以通过跨链请求同步所有的监管日志，也可以申请成为联盟链的成员直接在链上读取监管日志。

5.5　本章小结

本章首先介绍了数据层的设计与实现，通过实现抽象区块头存储模块、抽象交易打包模块、抽象回执打包模块这三个模块，本系统实现了链间的数据互认。接着，本章从区块头同步、跨链寻址、交易验证三个方面详细介绍了交互层的设计与实现，在此基础上，本系统实现了链间数据互信。接下来，本章介绍了旨在保证跨链事务原子性的事务层实现方法。本章最后阐述了治理层如何实现跨链治理。

第六章　系统测试与评估

6.1　测试环境

为对本文设计的基于联盟链的跨链系统进行测试，本文搭建了测试环境。

本跨链系统是基于 ssbc 联盟链设计与实现的，ssbc 联盟链采用的共识机制为 PBFT，为满足测试要求需要至少运行四个节点。本跨链系统为了测试跨链功能需要至少运行两条链，故本文在服务器上运行了两个分别包含四个节点的 ssbc 同构链。

服务器配置参数如表 6.1 所示。

表 6.1　服务器配置参数

名称	参数
处理器	2.6 GHz Intel Core i7
内存	16 GB 2400 MHz DDR4
硬盘	250G
操作系统	macOS Mojave 版本 10.14.6
Docker 版本	19.03.8
Go 版本	1.14.4 darwin/amd64

6.2　系统测试

6.2.1　功能模块测试

测试环境准备就绪，本文对跨链系统的所有功能模块进行了测试，主要包括如下内容：

（1）在开始跨链之前，有跨链需求的两条链分别在对方链上进行跨链信息注册，登记本链的基本信息、服务节点的地址以及中继节点地址等信息，跨链信息注册如图 6.1 所示。

图 6.1　跨链信息注册

（2）在完成跨链注册之后，跨链双方的服务节点以及中继节点进行握手建立 TCP 长连接，握手连接如图 6.2 所示。

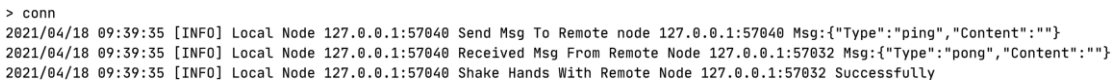

图 6.2　TCP 握手

（3）双方的服务节点在建立起 TCP 长连接之后，基于该连接进行抽象区块头同步，即本链服务节点向对方服务节点发送本链的抽象区块头，同时接收对方服务节点发送的对方链的抽象区块头。本链服务节点在接收到对方服务节点传送的抽象区块头之后第一时间存储至本地，等待后续的提交流程，抽象区块头同步如图 6.3 所示。

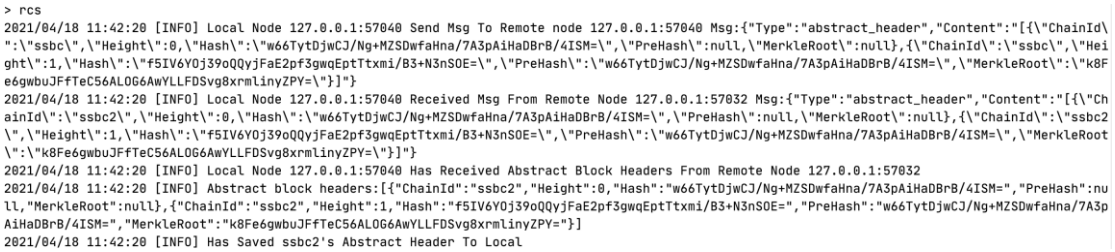

图 6.3　抽象区块头同步

（4）抽象区块头同步后进行抽象区块头提交功能测试。首先本链所有的服务节点在本地运行可验证随机函数，生成随机数结果以及该随机数结果对应的证明信息。本链服务节点将运行产生的随机数结果保存至本地，然后将该结果以及对应的证明信息打包成 VRF 消息广播至本链其他所有的服务节点，同时接收来自其他服务节点的 VRF 消息广播。接收到广播消息的服务节点首先验证对方产生随机数结果的合法性，验证完成后将对方的随机数结果保存至本地，VRF 执行如图 6.4 所示。

图 6.4　执行可验证随机函数

（5）当服务节点接收到其他所有的服务节点的广播后，取出本地保存的随机数集合，对比本地生成的随机数结果以及随机数集合中所有的随机数，判断本地生成的随机数结果是否最大。本地生成随机数结果最大的服务节点将自动变为提案节点，具备提案权限，提案节点选取如图 6.5 所示。

```
2021/04/18 11:46:06 [INFO] Vrf Result Verification Successful
2021/04/18 11:46:06 [INFO] Current Received VRF Results:index=0,result=6.994739841547616e+18
2021/04/18 11:46:06 [INFO] Current Received VRF Results:index=1,result=4.40647596978898 23e+18
2021/04/18 11:46:06 [INFO] Local VRF Result:6.994739841547616e+18
2021/04/18 11:46:06 [INFO] Local Node Has Right To Propose
2021/04/18 11:46:06 [INFO] Local Node Initiates An Abstract Block Header Storage Proposal
```

图 6.5　提案节点选取

（6）提案节点将之前同步获得的外链的抽象区块头打包成提案消息进行广播，其他不具备提案权的服务节点将对该提案信息进行门限签名：其他服务节点解析所收到的提案消息中的抽象区块头与自身本地存储的抽象区块头进行哈希摘要后对比，摘要值一致则签名发送给提案节点，否则拒绝签名，提案验证如图 6.6 所示。

```
2021/04/18 11:46:06 [INFO] Received Msg:{"Type":"AbstractHeaderProposal","Content":"[{\"ChainId\":\"ssbc\",\"Height\":0,\"Hash\":\"o3WqkfuN1W/lffC3
pcM9ztmCyfq0qDRRh3aiy+tXv0o=\",\"PreHash\":null,\"MerkleRoot\":null}]"}
2021/04/18 11:46:06 [INFO] Verifying Proposal……
2021/04/18 11:46:06 [INFO] Proposal Verification Successful, Local Node Make Signs
```

图 6.6　提案验证

（7）提案节点收集其他所有服务节点发来的针对提案的签名信息，提案节点第一时间对签名信息进行验签，当收集到的有效签名数大于等于阈值后，生成最终的门限签名。提案节点对提案结果进行签名，打包最终的门限签名、自身签名、收集到的签名列表以及提案结果后，进行提交存储，提案提交如图 6.7 所示。

```
2021/04/18 11:46:06 [INFO] Received Msg:{"Type":"HeaderProposalSignMsg","Content":"{\"Hash\":\"k5Wk3SS409wb9RCsa9cbZgyt9R93BasHcemX3Spb/K8=\",\"Pub
Key\":{\"N\":163723431889356020057929579802553748296929782926972289699442688814030402868113128640164543937983125759548317466970943460122320116924
4142512750312953281265133668798578179795338131745277032462340684871618485268098863396494459901480468507097124256586993216259820206751552983897393284249323952762722409590594441,\"E\":65537},\"Sign\":\"B3B4mUz4w/ZGi7PYCJBVGXMTr027sUbhQ2P2pyCVGyeOh/IWBtUlnwg+UWIXZTvikWhkaV1lT8I98gDXW/YADAlvpVQPr0
1z2U9aw2ar4Fbv7CRukdEA4yna8nyXWDu7DhTcqklmOv8qSRbzF2B6mcoL8ngNUtZaOJlLLWYW4RI=\",\"Threshold\":1}"}
2021/04/18 11:46:06 [INFO] Signature Verifying:{"Hash":"k5Wk3SS409wb9RCsa9cbZgyt9R93BasHcemX3Spb/K8=","PubKey":{"N":163723431889356020057929579802553748296929782926972289699442688814030402868113128640164543937983125759548317466970943460122320116924541425127503129532812651336687985781797953381317452770324623406848716184852680988633964944599014804685070971242565869932162598202067515529838973932842493239527627224095905944 41,"E":65537},"Sign":"B3B4mUz4w/ZGi7PYCJBVGXMTr027sUbhQ2P2pyCVGyeOh/IWBtUlnwg+UWIXZTvikWhkaV1lT8I98gDXW/YADAlvpVQPr01z2U9aw2ar4Fbv7CRukdEA4yna8nyXWDu7DhTcqklmOv8qSRb
zF2B6mcoL8ngNUtZaOJlLLWYW4RI=","Threshold":1}
2021/04/18 11:46:06 [INFO] Signature Verification Succeeded
2021/04/18 11:46:06 [INFO] Proposal Signature Verification is Successful
2021/04/18 11:46:06 [INFO] The Proposal Signature Reaches The Threshold Requirement, Local Node Submits The Abstract Block Header for Storage! Abst
ract Block Header:[147 149 164 221 36 184 211 220 27 245 16 172 107 215 27 102 12 173 245 31 119 5 171 7 113 233 151 221 42 91 252 175]
2021/04/18 11:46:06 [INFO] The Abstract Block Header Is Stored Successfully
```

图 6.7　提案提交

（8）源链 ssbc1 上的账户 A 在本地发起一笔跨链交易，交易内容为向目标链 ssbc2 上的账户 B 转账 10。源链首先将跨链转移资产锁定至锁定账户中，锁定资产交易上链后，源链将最新生成的区块头进行抽象后同步到目标链，交易发起如图 6.8 所示。

```
> cto-{"SourceChainId":"ssbc","DestChainId":"ssbc2","Type":"CTT","From":"QmQeDk5u1Pw7jYJMTsKFTT9i53uw1cY3YpA1NSj93DoWyQ","To":"QmZxarfNST2jYnrqXGci
2VQq4LK12ARaE4r6emZSdUCAkX","Value":10}
2021/04/18 11:53:12 [INFO] New Block Generated
```

图 6.8　发起跨链交易

（9）源链生成锁定资产交易的默克尔证明信息后，打包跨链交易，发送至目标链，交易打包如图 6.9 所示。

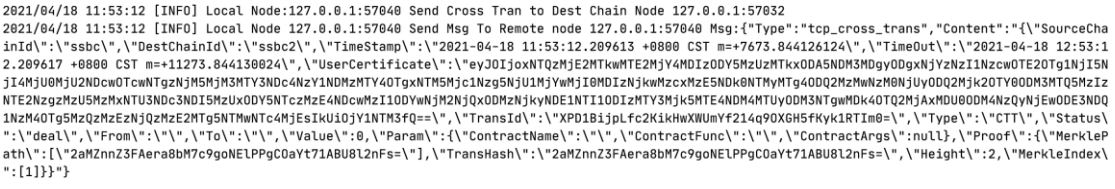
2021/04/18 11:53:12 [INFO] Local Node:127.0.0.1:57040 Send Cross Tran to Dest Chain Node 127.0.0.1:57032
2021/04/18 11:53:12 [INFO] Local Node 127.0.0.1:57040 Send Msg To Remote node 127.0.0.1:57040 Msg:{"Type":"tcp_cross_trans","Content":"{\"SourceCha
inId\":\"ssbc\",\"DestChainId\":\"ssbc2\",\"TimeStamp\":\"2021-04-18 11:53:12.209613 +0800 CST m=+7673.844126124\",\"TimeOut\":\"2021-04-18 12:53:1
2.209617 +0800 CST m=+11273.844130024\",\"UserCertificate\":\"eyJOIjoxNTQzMjE2MTkwMTE2MjY4MDIzODY5MzUzMTkxODA5NDM3MDgyODgxNjYzNzI1NzcwOTE2OTg1NjI5N
jI4MjU0MjU2NDcwOTcwNTgzNjM5MjM3MTY3NDc4NzY1NDMzMTY4OTgxNTM5Mjc1Nzg5NjU1MjYwMjI0MDIzNjkwMzcxMzE5NDk0NTMyMTg4ODQ2MzMwNzM0NjUyODQ2Mjk2OTY0ODM3MTQ5MzIz
NTE2NzgzMzU5MzMxNTU3NDc3NDI5MzUxODY5NTczMzE4NDcwMzI1ODYwNjM2NjQxODMzNjkyNDE1NTI1ODIzMTY3Mjk5MTE4NDM4MTUyODM3NDQ
1NzM4OTg5MzQzMzEzNjQzMzE2MTg5NTMwNTc4MjEsIkUiOjY1NTM3fQ==\",\"TransId\":\"XPD1BijpLfc2KikHWXWUmYf214q9OXGH5fKyk1RTIm0=\",\"Type\":\"CTT\",\"Status\
":\"deal\",\"From\":\"\",\"To\":\"\",\"Value\":0,\"Param\":{\"ContractName\":\"\",\"ContractFunc\":\"\",\"ContractArgs\":null},\"Proof\":{\"MerkleP
ath\":[\"2aMZnnZ3FAera8bM7c9goNElPPgCOaYt71ABU8l2nFs=\"],\"TransHash\":\"2aMZnnZ3FAera8bM7c9goNElPPgCOaYt71ABU8l2nFs=\",\"Height\":2,\"MerkleIndex\
":[1]}}"}

图 6.9　跨链交易打包发送

（10）目标链接收到跨链交易后，解析出跨链交易中的默克尔证明信息。目标链基于证明信息中的锁定交易所在区块高度以及实时同步的抽象区块头，取出指定的抽象区块头。目标链基于指定区块头的 Merkle Root 以及交易携带的默克尔证明信息进行默克尔验证，交易验证如图 6.10 所示。


2021/04/18 11:53:12 [INFO] Local Node 127.0.0.1:57032Received Cross Transaction From Remote Node 127.0.0.1:57040
2021/04/18 11:53:12 [INFO] Cross Transaction:{"SourceChainId":"ssbc","DestChainId":"ssbc2","TimeStamp":"2021-04-18 11:53:12.209613 +0800 CST m=+767
3.844126124","TimeOut":"2021-04-18 12:53:12.209617 +0800 CST m=+11273.844130024","UserCertificate":"eyJOIjoxNTQzMjE2MTkwMTE2MjY4MDIzODY5MzUzMTkxODA
5NDM3MDgyODgxNjYzNzI1NzcwOTE2OTg1NjI5NjI4MjU0MjU2NDcwOTcwNTgzNjM5MjM3MTY3NDc4NzY1NDMzMTY4OTgxNTM5Mjc1Nzg5NjU1MjYwMjI0MDIzNjkwMzcxMzE5NDk0NTMyMTg4OD
Q2MzMwNzM0NjUyODQ2Mjk2OTY0ODM3MTQ5MzIzNTE2NzgzMzU5MzMxNTU3NDc3NDI5MzUxODY5NTczMzE4NDcwMzI1ODYwNjM2NjQxODMzNjkyNDE1NTI1ODIzMTY3Mjk5MTE4NDM4MTUyODM3N
TgwMDk4OTQ2MjAxMDU0ODM4NzQyNjEwODE3NDQ1NzM4OTg5MzQzMzEzNjQzMzE2MTg5NTMwNTc4MjEsIkUiOjY1NTM3fQ==","TransId":"XPD1BijpLfc2KikHWXWUmYf214q9OXGH5fKyk1R
TIm0=","Type":"CTT","Status":"deal","From":"","To":"","Value":0,"Param":{"ContractName":"","ContractFunc":"","ContractArgs":null},"Proof":{"MerkleP
ath":["2aMZnnZ3FAera8bM7c9goNElPPgCOaYt71ABU8l2nFs="],"TransHash":"2aMZnnZ3FAera8bM7c9goNElPPgCOaYt71ABU8l2nFs=","Height":2,"MerkleIndex":[1]}}
2021/04/18 11:53:12 [INFO] Merkle Proof Verifying……
2021/04/18 11:53:12 [INFO] Merkle Verification Success

图 6.10　交易验证

（11）默克尔验证完成后目标链新铸资产转账至账户 B，待该转账交易上链后，目标链向源链同步最新抽象区块头，同时生成该转账交易的默克尔证明信息，打包回执发送至源链，回执打包如图 6.11 所示。


2021/04/18 11:56:44 [INFO] Local Node 127.0.0.1:57040 Has Received Remote Node 127.0.0.1:57032 Sent Transaction Receipt
2021/04/18 11:56:44 [INFO] Transaction Receipt:{"SourceChainId":"ssbc2","DestChainId":"ssbc","TimeStamp":"2021-04-18 11:53:12.213399 +0800 CST m=+7
661.081877727","TimeOut":"2021-04-18 12:53:12.213402 +0800 CST m=+11261.081881534","UserCertificate":"eyJOIjoxNjM3MjM0MzE4ODkzNTYwMjAwNTc5Mjk1Nzk4M
DI1NTM3NDgyOTY5Mjk3ODI5MjY5NzIyODk2OTk0NDI2ODg4MTQwMzA0MDI4NjgxMTMxMjg2NDAxNjQ1NDM5Mzc5ODMxMjU3NTk1NDgzMTc0NjY5NzA5NDM0NjAxMjIzMjAxMTY5MjQ0NTQxNDI1
MTI3NTAzMTI5NTMyODEyNjUxMzM2Njg3OTg1NzgxNzk3OTUzMzgxMzE3NDUyNzcwMzI0NjIzNDA2ODQ4NzE2MTg0ODUyNjgwOTg4NjMzOTY0OTQ0NTk5MDE0ODA0Njg1MDcwOTcxMjQyNTY1ODY
5OTMyMTYyNTk4MjAyMDY3NTE1NTI5ODM4OTczOTMyODQyNDkzMjM5NTI3NjI0MDk1OTA1OTQ0NDEsIkUiOjY1NTM3fQ==","TransId":"cwisppkk0NRhwNApmpeJ2P0l+6l6E5RdY8K+0
9/krms=","Type":"CTT","Status":"success","Resp":{"Data":null,"Signature":"aR5jA8gFK510qXgC88lrLxsP8Z4dTUelP9KgcIJfDrz4/pl78jCW5eT7hyJlpTjRcz7/UGIxm
k3hkJ1LHZrQAwLlfOhC/JUZ6i1fb0zVUyvl4gYDy5/lOI4ffpOi2wb1xmzf3RRryLjtt2fVhL4u7pTI1UojlZCVsn9gpfppbII=","Proof":{"MerklePath":["3Fi/60cnbuR7rKKRuijs+
4nSg12ofkAHDX9mWun8qDk="],"TransHash":"3Fi/60cnbuR7rKKRuijs+4nSg12ofkAHDX9mWun8qDk=","Height":3,"MerkleIndex":[1]}}

图 6.11　交易回执打包发送

（12）源链接收到目标链的回执后将进行解析，对回执进行默克尔验证，验证成功后源链将锁定资产焚毁，此次跨链交易结束，回执验证如图 6.12 所示。

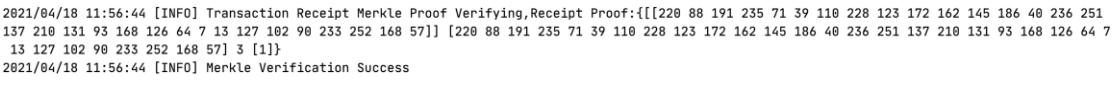
2021/04/18 11:56:44 [INFO] Transaction Receipt Merkle Proof Verifying,Receipt Proof:{[[220 88 191 235 71 39 110 228 123 172 162 145 186 40 236 251
137 210 131 93 168 126 64 7 13 127 102 90 233 252 168 57]] [220 88 191 235 71 39 110 228 123 172 162 145 186 40 236 251 137 210 131 93 168 126 64 7
 13 127 102 90 233 252 168 57] 3 [1]}
2021/04/18 11:56:44 [INFO] Merkle Verification Success

图 6.12　回执验证

6.2.2　异常场景测试

除对系统的功能进行全面测试外，本文还对跨链系统的鲁棒性进行了测试。

本文基于不同的异常等级设计了不同的异常测试场景，测试系统在不同异常情况下的稳定性。具体测试详情如表 6.2 所示。

表 6.2　异常场景测试

序号	异常测试场景	异常等级	测试结果	是否符合预期
1	在跨链过程中直接关停一个正在传输跨链信息的中继节点使之宕机	高	跨链系统因超时触发回滚，同时，与宕机中继节点通信的对方链中继节点，在中继节点列表中重新选择了节点进行握手通信，跨链系统恢复正常	是
2	在跨链过程中关停所有中继节点	极高	跨链系统因超时触发回滚，跨链系统停止工作，但是对联盟链本身无影响	是
3	在抽象区块头同步中，一个服务节点恶意同步假区块头，且对方链接收假区块头的服务节点为提案节点	中	提案节点发起提案，本链其他节点对提案进行验证，验证不通过，拒绝签名，提案节点收集到 0 个签名，提案失败，触发下一轮提案	是
4	在抽象区块头同步中，一个服务节点恶意同步假区块头，且对方链接收假区块头的服务节点为非提案节点	低	接收假区块头的服务节点因验证提案节点的提案不通过，拒绝签名，但提案节点接收到了其他服务节点的签名，收集签名数大于阈值，提案成功	是
5	在区块头同步过程中关停一个服务节点使之宕机	高	与宕机服务节点通信的对方链服务节点，在本链登记信息中找到到对方链所有服务节点，在服务节点中重新选择一个节点进行握手通信，继续抽象区块头的同步	是
6	在区块头同步过程中关停所有服务节点	极高	区块头同步模块停止工作，默克尔证明模块因无法获取到对方链指定高度的抽象区块头，验证失败，触发回滚，跨链系统停止工作，但对联盟链本身无影响	是
7	在抽象区块头存储提案中，提案节点将抽象区块头篡改后发起提案	低	其他服务节点验证提案不通过，拒绝签名，提案节点收集到 0 个签名，提案失败，触发下一轮提案	是
8	进行抽象区块头同步的所有节点中，有 n 个节点作恶，且作恶节点为提案节点，t 为门限签名阈值且 t>n-1	极高	所有的诚实节点验证提案不通过，拒绝签名，其他作恶节点进行签名，提案节点收到了 n-1 个签名，因小于门限签名阈值提案失败，触发下一轮提案	是

续表 6.2　异常场景测试

9	进行抽象区块头同步的所有节点中，有 n 个节点作恶，且作恶节点为提案节点，t 为门限签名阈值且 t<=n-1	极高	所有的诚实节点验证提案不通过，拒绝签名，其他作恶节点进行签名，提案节点收到了 n-1 个签名，因大于等于门限签名阈值提案成功，作恶成功，跨链系统异常	是
10	进行抽象区块头同步的所有节点数为 m，有 n 个节点为作恶节点，且提案节点为诚实节点，t 为门限签名阈值，且 m-n>=t+1	极高	所有诚实节点验证提案通过，进行签名，所有作恶节点放弃签名，提案节点收到大于等于门限签名阈值的签名数，提案成功	是
11	进行抽象区块头同步的所有节点数为 m，有 n 个节点为作恶节点，且提案节点为诚实节点，t 为门限签名阈值，且 m-n<t+1	极高	所有诚实节点验证提案通过，进行签名，所有作恶节点放弃签名，提案节点收到的签名数小于门限签名阈值，提案失败，作恶成功，跨链系统异常	是

6.3　系统评估

本文对基于联盟链实现的跨链系统进行了全面的功能测试，以及设计可能发生异常的测试场景对跨链系统的鲁棒性进行测试，测试结果分析如下。

本跨链系统基于联盟链 ssbc，联盟链 ssbc 使用的是 PBFT 共识机制，即总节点数为 n，作恶节点数为 f，当 n>=3f+1 时，联盟链系统能保证稳定性。

本跨链系统参与跨链的服务节点数为 n，作恶节点数为 f，门限签名的阈值为 t，当作恶节点为提案节点时，为满足跨链系统稳定性，f-1<t，即门限签名的阈值要满足大于作恶节点数减一。当提案节点为诚实节点时，为满足跨链系统稳定性，需要满足 n-f-1>=t，即诚实节点数大于等于阈值加一，当 n=3f+1 时，t<=2f，即门限签名的阈值小于等于作恶节点数的两倍。本系统将门限签名的阈值 t 设置为服务节点数的二分之一，即 t=n/2，当 n=3f+1 时，即 n>=4 时满足 t>f-1 且 t<=2f。故联盟链执行环境满足拜占庭容错即总结点数 n 大于等于三倍的作恶节点数加一时，跨链系统设置门限签名阈值为参与跨链的服务节点数的二分之一，可保证跨链系统的稳定性。

6.4　本章小结

本章对本文设计的基于联盟链的跨链系统进行了全面的功能测试，测试结果表明本系统的核心功能模块：基于预言机机制实现的抽象区块头同步模块、基于默克尔证明机制以及门限签名机制实现的交易验证模块等均能正常运行，最后对系统的稳定性

作出了评估。

第七章　总结与展望

7.1　回顾和总结

跨链技术是连接链与链之间的桥梁，通过跨链技术的实现，区块链之间能够做到信息交互，价值转移，发挥出区块链技术更大的潜力。

本文基于联盟链，设计与实现了一种跨链系统。首先介绍了跨链技术的背景，随着区块链技术的发展，出现了越来越多的信息孤岛，需要实现跨链去连接众多的信息孤岛。其次，通过对主流跨链机制的总结以及对主流跨链项目的深入探究分析，介绍了跨链技术的发展现状。然后，对所设计跨链系统涉及的相关技术进行了简要介绍。

在核心设计章节，本文首先从基本功能需求以及非功能性需求两个角度对跨链系统进行了需求分析，并基于需求分析设计了跨链协议以及对系统的整体架构进行设计。然后，基于所设计的四层架构逐层介绍了每层若干模块详细的设计与实现，包括旨于实现数据互认的数据层；由基于预言机机制实现的区块头同步模块、基于默克尔证明和门限签名实现的交易验证模块以及跨链寻址模块组成的交互层；为保证跨链事务原子性的事务层以及旨于实现跨链系统有效治理的治理层。最后，本文对所设计的跨链系统进行了全面的功能性以及鲁棒性测试，测试结果表明本文所设计系统能满足联盟链跨链需求且具备较强的稳定性。

7.2　下一步工作展望

本文基于联盟链特性，设计了联盟链跨链协议，对核心数据结构进行了定义，对跨链流程进行了规范，基于 ssbc 联盟链实现了一种跨链系统。从性能上看该跨链系统仍有很大的提升空间，故下一步的优化是完善项目代码提升系统性能。从实现机制上看，本系统从去中心化预言机的实现、门限签名机制、默克尔证明机制等方面保证了跨链系统的去中心化，另外还设计实现了追责机制。但是，本系统仍然存在节点组团作恶成功的可能性，未来研究的重要方向是设计实现一定的机制，使得节点作恶行为能够更早地被发现，提高作恶节点需要面临的作恶成本。

参考文献

[1] Strawn G.BLOCKCHAIN[J].IT professional,2019, 21 (1): 91-92.

[2] 蔡维德,郁莲,王荣,刘娜,邓恩艳.基于区块链的应用系统开发方法研究[J].软件学报,2017,28(06):1474-1487.

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

[4] 袁勇,王飞跃.区块链技术发展现状与展望[J].自动化学报,2016,42(04):481-494.

[5] Szabo N. Smart contracts: Building blocks for digital markets[EB/OL].http://www.truevaluemetrics.org/DBpdfs/BlockChain/Nick-Szabo-Smart-Contracts-Building-Blocks-for-Digital-Markets-1996-14591.pdf.

[6] 何蒲,于戈,张岩峰,鲍玉斌.区块链技术与应用前瞻综述[J].计算机科学,2017,44(04):1-7+15.

[7] DINH T, WANG J, CHEN G, et al. BLOCKBENCH: A Framework for Analyzing Private Blockchains [J]. 2017.

[8] LI Z, KANG J, YU R, et al. Consortium Blockchain for Secure Energy Trading in Industrial Internet of Things [J]. IEEE Transactions on Industrial Informatics, 2017.

[9] Karbasi A H, Shahpasand S.A post-quantum end-to-end encryption over smart contract-based blockchain for defeating man-in-the-middle and interception attacks[J].Peer-to-peer networking and applications,2020, 13 (5): 1423-1441.

[10] 刘家稷,杨挺,汪文勇.使用双区块链的防伪溯源系统[J].信息安全学报,2018, 3 (3): 17-29.

[11] Sunny J, Undralla N, Madhusudanan Pillai V.Supply chain transparency through blockchain-based traceability: An overview with demonstration[J].Computers & industrial engineering,2020, 150.

[12] 郭朝,郭帅印,张胜利,宋令阳,王晖.区块链跨链技术分析[J].物联网学报,2020,4(02):35-48.

[13] 李希明,土丽艳,金科.从信息孤岛的形成谈数字资源整合的作用[J].图书馆论坛,2003(06):121-122+61.

[14] 王洁,魏生,戴科冕.基于区块链的科技金融大数据开放共享体系研究[J].现代计算机(专业版),2018(22):52-58+78.

[15] 魏昂.一种改进的区块链跨链技术[J].网络空间安全,2019,10(06):40-45.

[16] 路爱同,赵阔,杨晶莹,王峰.区块链跨链技术研究[J].信息网络安全,2019(08):83-90.

[17] 李芳,李卓然,赵赫.区块链跨链技术进展研究[J].软件学报,2019,30(06):1649-1660.

[18] Buterin V. A next-generation smart contract and decentralized application platform[EB/OL].[04-12].https://github.com/ethereum/wiki/wiki/White-Paper.

[19] Buterin V. Chain interoperability[EB/OL].https://www.r3.com/download/chain-interoperability.

[20] Brown R, Carlyle J, Grigg I, et al. Corda: An introduction[EB/OL].https://docs.corda.net/_static/corda-introductory-whitepaper.pdf.

[21] Reilly E, Maloney M, Siegel M, et al. An IoT Integrity-First Communication Protocol via an Ethereum Blockchain Light Client[C]: 53-56.

[22] Consensys. BTC Relay's documentation[EB/OL].http://btc-relay.readthedocs.io/en/latest/.

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

[24] Wood G. Polkadot: Vision for a heterogeneous multi-chain framework[EB/OL].https://github.com/polkadot-io/polkadotpaper/raw/master/PolkaDotPaper.pdf.

[25] Poon J, Dryja T. The Bitcoin lightning network: Scalable off-chain instant payments[EB/OL].https://lightning.network/lightning-network-paper.pdf.

[26] Kuznetsov A, Shekhanin K, Kolhatin A, et al. Performance of Hash Algorithms on GPUs for Use in Blockchain[C]: 166-170.

[27] Kranz J, Nagel E, Yoo Y.Blockchain Token Sale: Economic and Technological Foundations[J].Business & information systems engineering,2019, 61 (6): 745-753.

[28] Guziur J, Pawlak M, Poniszewska-Marańda A, et al.: Light Blockchain Communication Protocol for Secure Data Transfer Integrity, Cham: Springer International Publishing,2018: 194-208.

[29] Sigwart M, Frauenthaler P, Spanring C, et al.Decentralized Cross-Blockchain Asset Transfers[J],2020.

[30] Liu W, Wu H, Meng T, et al.AucSwap: A Vickrey auction modeled decentralized cross-blockchain asset transfer protocol[J].Journal of systems architecture,2021, 117: 102102.

[31] 石贤芝,林昌露,张胜元,唐飞.标准模型下高效的门限签名方案[J].计算机应用,2013,33(01):15-18.

[32] 王斌,李建华.无可信中心的(t,n)门限签名方案[J].计算机学报,2003(11):1581-1584.

[33] 伍忠东,谢维信,喻建平.一种安全增强的基于椭圆曲线可验证门限签名方案[J].计算机研究与发展,2005(04):705-710.

[34] 张晓燕,范冰冰.基于 Merkle 树的移动平台文件完整性校验[J].计算机系统应用,2010,19(09):173-176+162.

[35] 孟浩华,曹波,袁慧,董亮.一种基于 Merkle-Tree 的云存储数据持有性检查方案[J].计算机与数字工程,2017,45(07):1387-1390.

[36] Bruschi F, Rana V, Pagani A, et al.Tunnelling Trust into the Blockchain: a Merkle Based Proof System for Structured Documents[J].IEEE access U6 - ctx_ver=Z39.88-2004&rft.genre=article&rft.jtitle=IEEE+access&rft.date=2020-10-01&rft.pub=IEEE&rft.eissn=2169-3536&rft.spage=1&rft.epage=1&rft.externalDocID=6287639&paramdict=en-US U7 - Journal Article,2020: 1-1.

[37] 徐忠,邹传伟.区块链能做什么、不能做什么?[J].金融研究,2018(11):1-16.

[38] Gradstein H L, Krause S K.Distributed Ledger Technology (DLT) and blockchain[J],2017.

[39] Lo S K, Xu X, Staples M, et al.Reliability analysis for blockchain oracles[J].Computers & electrical engineering,2020, 83: 106582.

[40] Nelaturu K, Adler J, Merlini M, et al.On Public Crowdsource-Based Mechanisms for a Decentralized Blockchain Oracle[J].IEEE transactions on engineering management,2020, 67 (4): 1444-1458.

[41] Fuchsbauer G: Constrained Verifiable Random Functions, Cham: Springer International Publishing,2014: 95-114.

[42] Dodis Y.Efficient construction of (distributed) verifiable random functions[J].Lecture notes in computer science,2003, 2567: 7-17.

[43] 刘阳,李栋,张天石,曾鹏.第五讲:工业互联网应用协议及数据互认标准进展与分析[J].仪器仪表标准化与计量,2020, No.215 (05): 35-38+43.

[44] Council C O.Decision of the Organization for Economic Co-operation and Development Council concerning the Mutual Acceptance of Data in the Assessment of Chemicals, adopted by the Council at its 535th meeting on 12 May 1981[J].Annali Dellistituto Superiore Di Sanità,2002, 38 (1): 87-93.

[45] Gervais A, Karame G O, Wüst K, et al. On the security and performance of Proof of Work blockchains[C]: 3-16.

附录 A　术语表

缩写	全称	中文含义
PBFT	Practical Byzantine Fault Tolerance	实用拜占庭容错协议
PoW	Proof of Work	工作量证明
VRF	Verifiable Random Function	可验证随机函数
BTC	Bitcoin	比特币
TCP	Transmission Control Protocol	传输控制协议
P2P	peer-to-peer	对等计算机网络
IBC	Inter Blockchain Communication Protocol	区块链间通信协议

致谢

岁月辗转成歌，时光流逝如花，即将告别多彩自在的校园生活走向社会开始新的征程，在这交接之际，复盘我的研究生生涯，在三年的学业与实践过程中，我成长了许多，回看生活中的点点滴滴，这得益于师长、同学、朋友及家人的陪伴和鼓励，感恩每一个走进我生命里的人，协助我顺利地走到了现在。

首先，要特别感谢我的导师郁莲老师。非常感谢郁老师在研究生入学之初选择了我，使我有幸成为导师大家庭中的一员。郁老师是一个很认真的人，时刻保持着对科研的严谨与不懈追求，督促着我们的坚持与进步。在每一次的组会上，认真地倾听我们的见解，并给出精准的指导与建议，使我在这三年的学习中，每天都保持着知识的更新与迭代。在研究方向的选择上，特别感谢郁老师给了我极大的选择权，作为区块链研究方向的独苗苗，老师以推荐文献阅读、联系学长等方式给我提供了莫大的帮助，使我得以在感兴趣的方向上坚持下去。此次关于毕业论文的撰写，郁老师从选题到最终的定稿，在持续一年里，倾听我的每一个想法，引导我做更深入的研究，在迷茫时给我指引方向，并不断地鼓励我，使我有信心、有动力在反复的修改中坚持下来。经师易遇，人师难遇，感恩郁莲老师的谆谆教导，我将铭记于心，终生难忘。

其次，感谢在软微学习过程中所遇到的所有老师与同学，是老师们用专业的知识、无私的精神对我进行了知识的灌溉，丰富了知识的储备，是同学们的深切关怀、互帮互助使我开心顺利地度过了我的集体生涯。

最后，非常感谢我的家人以及我的女朋友，因为学业的追求忽略了对他们的陪伴，感恩他们的理解与支持，作为我永远的坚强后盾，感谢父母无私的养育之恩以及女友多年来不离不弃的陪伴。

至此，我的学生时代正式结束，但吾生有涯，而知也无涯，在未来的路途中，我会依旧保留我的赤子之心，朝着人生理想而奋斗！

北京大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名：叶德鹏　日期：2021年04月19日

学位论文使用授权说明

（必须装订在提交学校图书馆的印刷本）

本人完全了解北京大学关于收集、保存、使用学位论文的规定，即：

- 按照学校要求提交学位论文的印刷本和电子版本；
- 学校有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务，在校园网上提供服务；
- 学校可以采用影印、缩印、数字化或其它复制手段保存论文；
- 因某种特殊原因需要延迟发布学位论文电子版，授权学校□一年/□两年/□三年以后，在校园网上全文发布。

（保密论文在解密后遵守此规定）

论文作者签名：叶德鹏　导师签名：

日期：2021年04月19日