天津大学

本科生毕业论文

题目：基于可信验证的区块链联邦隐私保护方案

学　　院　智能与计算学部

专　　业　软件工程

年　　级　2018

姓　　名　潘柯文

学　　号　3018216079

指导教师　许光全

独创性声明

本人声明：所呈交的毕业设计（论文），是本人在指导教师指导下，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，本毕业设计（论文）中不包含任何他人已经发表或撰写过的研究成果。对本毕业设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在论文中作了明确的说明。本毕业设计（论文）原创性声明的法律责任由本人承担。

论文作者签名：

年　　月　　日

本人声明：本毕业设计（论文）是本人指导学生完成的研究成果，已经审阅过论文的全部内容。

论文指导教师签名：

年　　月　　日

摘　要

分片区块链因其可扩展性得到广泛应用，但目前单节点维护用户信息在分片导致节点数变少的情况下极易造成隐私泄露。联邦思想为分片区块链的敏感信息存储提供了新思路。而可信验证也成了通信隐私保护的解决方案之一。

因此针对分片区块链体系架构，本文提出一种基于可信验证的敏感信息联邦分片机制，实现面向终端用户的联邦隐私保护方案，以解决数据在通信和存储时存在的信息泄露隐患。基于相关理论，本文首先设计了一种基于联邦思想的分片区块链模型，通过多节点分布式维护数据对用户的隐私信息进行差分存储，实现存储隐私保护。然后在区块链节点进行通信交易时设计可信验证机制对交易双方身份及平台信息进行验证，实现通信隐私保护。

之后本文以 C++函数编程对区块链交易存储和可信验证流程进行模拟仿真实验，实现原型系统的开发。实验发现联邦分片区块链确实可以在敏感信息存储时实现差分隐私保护，同时可信验证能有效保证数据在区块链中通信时不被披露。

关键词：可信验证，分片区块链，隐私保护，差分存储，网络安全

ABSTRACT

The sharding blockchain has been widely used because of its scalability. But at present, when a single node maintains user information, it is very easy to cause privacy disclosure when the number of nodes is reduced due to sharding. The idea of federation provides a new idea for the sensitive information storage of sharding blockchain. Trusted verification has also become one of the solutions for communication privacy protection.

Therefore, aiming at the blockchain architecture of sharding, this paper proposes a sensitive information federal sharding mechanism based on trusted verification to realize the federal privacy protection scheme for end users, so as to solve the hidden danger of information disclosure in data communication and storage. Based on relevant theories, this paper first designs a sharding blockchain model based on the idea of federation, which differentially stores users' privacy information through multi nodes distributed maintenance data to protect the storage privacy. Then, when the blockchain nodes carry out communication transactions, a trusted verification mechanism is designed to verify the identity and platform information of both parties to the transaction to protect the communication privacy.

Then, this paper uses C++ function programming to simulate the blockchain transaction storage and trusted verification process, and realizes the development of the prototype system. Experiments show that the federal sharding blockchain can indeed achieve differential privacy protection when storing sensitive information, and the trusted verification mechanism can ensure that the data will not be disclosed when communicating in the blockchain.

KEY WORDS: Trusted Verification, Sharding Blockchain, Privacy Protection, Differential Storage, Cyber Security

目 录

第一章　绪论

信息时代的到来使得区块链技术得以蓬勃发展。区块链去中心化、公开透明、不可篡改的特点有效保证了数据安全。但这样的优势在复杂网络情况下却极易导致敏感信息的暴露。因此如何在保证区块链数据安全性的前提下提高数据隐私性是目前广泛研究且亟需解决的问题。

1.1　研究背景及意义

区块链是一个公开透明且不可篡改的分布式系统。这样的机制保证了在区块链中传输的数据难以被造假，因此保证了数据的安全。但是，区块链这样公开透明的特性同时也是一把双刃剑，所有参与节点都记录了全局交易数据虽然保证了交易数据的安全公开，却也难免造成用户隐私的泄露。

随着区块链技术的不断发展，一种基于联邦分片机制的新型区块链系统成为解决隐私泄露问题的新思路。通过将节点以不同的算法，如计算 $n=2k$（n 表示第 n 个分片，k 表示节点地址的前 k 位）把整个区块链系统的节点划分进不同的分片的之中。通过基于状态的分片，节点可以存储系统状态视图的一部分，使得每个分片中的所有节点用来维护一个用户的数据。当用户之间传播交易时，会以分片中的某一节点接收交易，单独发送到和交易相关的节点用于共识，并接收和存储部分数据。同时又基于联邦思想可以多节点合作还原出原始的完整数据，在交易达成共识之后就能够让交易上链。

通过这种基于联邦思想的区块链分片机制，不仅通过多个节点共同维护一个用户的信息使得该用户不会轻易因某一节点受到恶意攻击而瘫痪；也使得用户分片的每个节点只存储完整消息的部分信息，当节点被恶意攻击时就不会导致完整信息的泄露，同时又可以进行分布式协作还原出原始的完整数据，因此实现了用户数据的差分隐私保护。另外，这种分片结构降低存储开销，提高交易吞吐量，实现区块链的扩展性。

但这样的机制仅能保证存储在节点上的数据不会轻易泄露。如果在节点间数据交易的过程中发送方是恶意节点，发送错误消息给接收方；或者接收方是故障节点，不能对收到的消息进行反馈、存储，甚至恶意节点，将收到的隐私消息在整个节点网络中披露，这将造成难以估计的损失。因此在节点间发送交易时，如何确定这些节点是可信的、故障的还是恶意的又成了新的问题。

所以在对敏感信息差分隐私保护的基础上，考虑引入可信验证机制，在发送交易之前通过可信验证平台对节点的身份进行验证来确保参与节点是否可信。身份验证具体包括平台身份验证和用户身份验证两部分。首先进行平台身份验证，通过验证度量结果的可信报告证明所使用的可信验证平台是可信的；然后利用可信验证平台对用户的私有秘密进行验证来证明用户身份是可信的。

综上，本文提出一种基于可信验证的区块链联邦隐私保护方案，通过采取联邦思想，实现分片区块链系统，对上链的敏感信息进行差分隐私保护。同时引入全流程的双向可信验证，证明交易双方的身份可信，实现一种基于可信验证的敏感信息联邦分片机制。

1.2 国内外研究现状

1.2.1 联邦分片区块链

ELASTICO 是第一个公开的分片区块链系统。其设定每个分片负责验证一组交易，并基于 PBFT 实现共识。最终的碎片验证所有从碎片接收到的交易，并将其发送到一个全局块，然后将其存储在系统的所有节点中。

ELASTICO 虽然为交易计算实现了分片，但并没有为系统中的存储和通信实现分片，因此被称为部分分片系统。由于每个节点接收并存储系统的完整信息，因此系统中不存在跨分片交易，但节点仍然存在较大的存储和带宽开销。

为了进一步减轻区块链中节点的开销，许多研究集中在完全分片上，即用于交易计算、存储和通信的分片。

Omniledger 是第一个实现全分片的基于分片的区块链系统。系统采用客户端驱动机制提交跨分片交易。然后，提出了另一个名为 Chainspace 的客户端驱动的分片系统，以支持通用智能合约的分片。

然而，客户端驱动的机制给典型的轻量级用户节点带来了额外的负担，并且容易受到恶意用户的拒绝服务（DoS）攻击。为了进一步提高性能，研究人员提出了几种分片驱动机制来处理跨分片交易。例如，RapidChain 在基于未使用事务输出（UTXO）的系统中提出了一种传输机制。对于每个跨分片交易，该机制首先通过子交易将所有涉及的 UTXO 传输到同一切分。然后，跨分片交易成为内部交易，可以在单个分片中处理。Monoxide 提出了基于账户/余额的系统中的重放机制。每个跨分片交易被划分为若干子交易，包括内部交易和中继交易。每个内部交易都对应于一个相关的分片。每两个连续的内部交易之间需要一个中继交易。

虽然上述完整的分片系统可以保证跨分片交易的原子性和一致性，但它们提交跨分片交易的工作量成倍增加。特别是每个跨分片交易都需要被分割成若干个子交易，整个分片系统需要验证和处理所有相关的子交易才能提交一个跨分片交易。这在吞吐量和确认延迟方面严重降低了分片性能。因此提出了一种分层分片区块链系统 Pyramid，其中一些分片可以存储多个分片的完整记录，从而跨分片的交易可以在这些分片内部进行处理和验证。

此外，对于能够将完整信息分别存储到各个分片当中的状态分片，传统的方式经常基于简单的数据映射，但这样的特定映射会导致占用大量内存。因此有研究提出了流量感知分片，它可以用内存有效映射来描述，减少内存使用。

综上所述，分片区块链系统的研究已经取得了一些进展，但现有的分片区块链系统更关注区块链的可扩展性，而对于在隐私保护方面的研究工作甚少。

联邦学习作为一种分布式训练的机器学习模型，其在保障了各方数据隐私的前提下能够解决“数据孤岛”问题，无疑给分片区块链系统的隐私保护提供了新的思路。虽然目前区块链与联邦学习结合的例子有很多，但大多是区块链赋能联邦学习以解决联邦学习集中式训练节点出现故障的问题，没有典型的联邦学习赋能区块链的例子。

因此考虑到联邦学习在隐私保护上的优势以及其与区块链异曲同工的分布式设计结构，本文考虑引入联邦思想，使其能够在分片中的节点收到消息时将消息进行切分，单独广播到分片的其他节点，验证以达成共识，并在每个节点上存储一些数据。同时又基于联邦思想可以多节点分布式协作还原出原始的完整数据，实现用户敏感数据的差分隐私保护。

1.2.2 可信验证机制

可信验证机制属于可信计算技术的一种具体应用场景。可信计算技术为计算机平台提供了一个相对安全的工作环境，具有硬件安全芯片，从而提高了各种计算设备的安全性。在国外，安全芯片依靠高度可信的平台模块（TPM），并得到 TCG 的推广。TPM 被为信任根，对引导过程中的硬件、操作系统和应用程序逐步进行测量的方法，从而建立对通用终端平台的信任，并广泛应用于各种计算机设备，如终端、PC、笔记本、服务器等。国内的可信计算标准由国家密码管理局制定，遵循该标准的芯片被称为可信密码模块（TCM）。

TCM 具有安全保护区域去存储某些隐私信息，如平台配置寄存器（PCR）、独立存储、易失性和非易失性存储。这些信息无法从 TCM 外部访问或更改。TCM 还具有 SMS4、SM3 和 SM2 三种类型的算法引擎，可以实现散列算法、对称密

码算法和非对称密码算法。TCM 还提供各种可信计算命令。这些命令允许使用受保护的机密信息访问 TCM 中各种受保护区域，以获得某些安全和可靠性特征。作为国内研究实施的可信密码模块，TCM 能够提供可信存储、可信报告、可信验证等可信密码服务。但与 TPM 不同的是 TCM 仅作为可信根的一部分被部署于可信平台控制模块 TPCM。而 TPCM 还能实现可信度量和可信控制等可信计算功能，能够实现主动可信，开启了可信计算 3.0 时代。

基于联邦的分片区块链虽然对数据进行了差分隐私保护，但这也就导致节点上仅存有用户部分数据，难以证明其可信度。可信计算作为一种保证信息系统可预见性的技术，常被用于区块链终端的安全通信研究。既能实现设备本身的完整性验证，又能实现设备之间的信任访问。为了将可信计算引入联邦分片区块链，需要在各个节点平台上构建 TCM 可信模块。有两种方式，一是基于原 TCM 硬件芯片。通过设备主板总线接口内置/外接 TCM 安全芯片，实现完整的 TCM 根信任功能；二是通过软件模拟 TCM 可信模块。通过软件编程模拟 TCM 提供的各种加密操作和可信功能。例如，ARM 处理器的 TrustZone 技术可以使用 TrustZone 构建功能齐全的 TCM 可信模块。

考虑到节点对可信验证要求的灵活性与复用性，本文采用程序模拟的 TCM 可信模块，从逻辑上自底向上进行层次可信验证。并设计可信报告、可信基准、可信验证三个模块完成可信身份验证。

对于区块链系统而言，所有的节点都是等重的，因此每个节点都应部署三个模块。但为了使实验结果更加直观，在本文中使用三个节点分别作为可信报告节点、可信基准节点和可信验证节点，以证明其可行性。可信基准节点生成可信基准值发送给可信验证节点，可信报告节点生成可信报告并发送给可信验证节点，可信验证节点利用可信基准值对可信报告进行验证，并输出验证结果。

1.3 本文主要工作

当前的研究中，并没有联邦分片机制与可信验证机制有效融合的方案，因此本文首先学习了分片区块链系统的结构框架、交易流程、优势劣势，并基于联邦思想探究其在差分隐私方面的应用；学习可信验证机制的概念、特点、执行流程以及应用范围等问题；探究可信验证在联邦分片区块链中的应用。

其次，本文基于对可信验证机制和联邦分片区块链的学习，针对分组/片新型区块链体系架构，提出一种基于可信验证的敏感信息联邦分组/片机制，实现面向终端用户的联邦隐私保护方案，并对方案进行分析和评价。

再者，本文设计相关实验，以探究引入可信验证机制对联邦分片区块链隐私安全性的影响，分析实验结果，得出结论，并与如预期结论进行对比。

最后，总结本次学习内容，对基于可信验证的区块链联邦隐私保护方案作出展望。

1.4 论文结构安排

论文第一章简述“联邦分片区块链”、“可信验证机制”的知识背景，探究结合了可信验证机制的联邦分片区块链的现实意义。并简述国内外对于联邦分片区块链以及可信验证机制的研究历史和现状。最后，简要介绍了本文的主要任务及其结构。

论文第二章是对相关工作的概述，具体体现为对于联邦分片区块链、可信验证机制的知识储备、前期研究。最后是对于结合了可信验证的联邦分片区块链的前期调研。

论文第三章基于对可信验证和联邦分片区块链的了解，设计出基于可信验证的区块链联邦隐私保护方案框架，并详细介绍两个核心功能的具体设计流程。

论文第四章设计基于可信验证的区块链联邦隐私保护方案相关实验，通过实验具象化联邦分片区块链进行交易的流程以及可信验证的整个流程，并对引入可信验证机制的联邦区块链的保密性和安全性的影响进行了总结和分析。

论文第五章是总结与展望部分，先分析总结了本次对于基于可信验证的区块链联邦隐私保护方案的学习与研究情况，以及模型设计、实验探究情况，然后对此研究方向今后的工作进行了展望。

论文最后是参考文献和致谢部分。

第二章　相关工作概述

为了将可信验证机制引入联邦分片区块链以探究其对隐私安全性的变化，我们首先需要进行前期的准备工作——对可信验证和联邦分片区块链的研究。

了解联邦分片区块链和可信验证的模型架构、运作机制有利于对其进行简化设计和模拟，从而将二者相结合，设计出目标模型，进行研究。

2.1　联邦分片区块链的研究

2.1.1　分片区块链概述

随着区块链技术的发展，其使用越来越广泛，目前的区块链系统已经无法满足大规模应用的需求。因此需要考虑在不增加单个节点工作量情况下对区块链扩容，增强区块链的可伸缩性。

目前，在不降低去中心化程度的情况下实现高性能的最有效的区块链扩容方案就是分片。分片的思想是“分而治之”，在区块链系统中，分片是指将事务和存储划分为不同的分组，每个分组并行处理交易，从而提升区块链的性能。分片的引入，如图 2-1 所示，将使得原本区块链中所有节点所做的相同的任务被分配到不同分片，每个分片处理不同的任务，独立运行自己的共识协议，分片之间并行处理，以此提升公链性能，也使得交易吞吐量及延迟得到有效的改善。

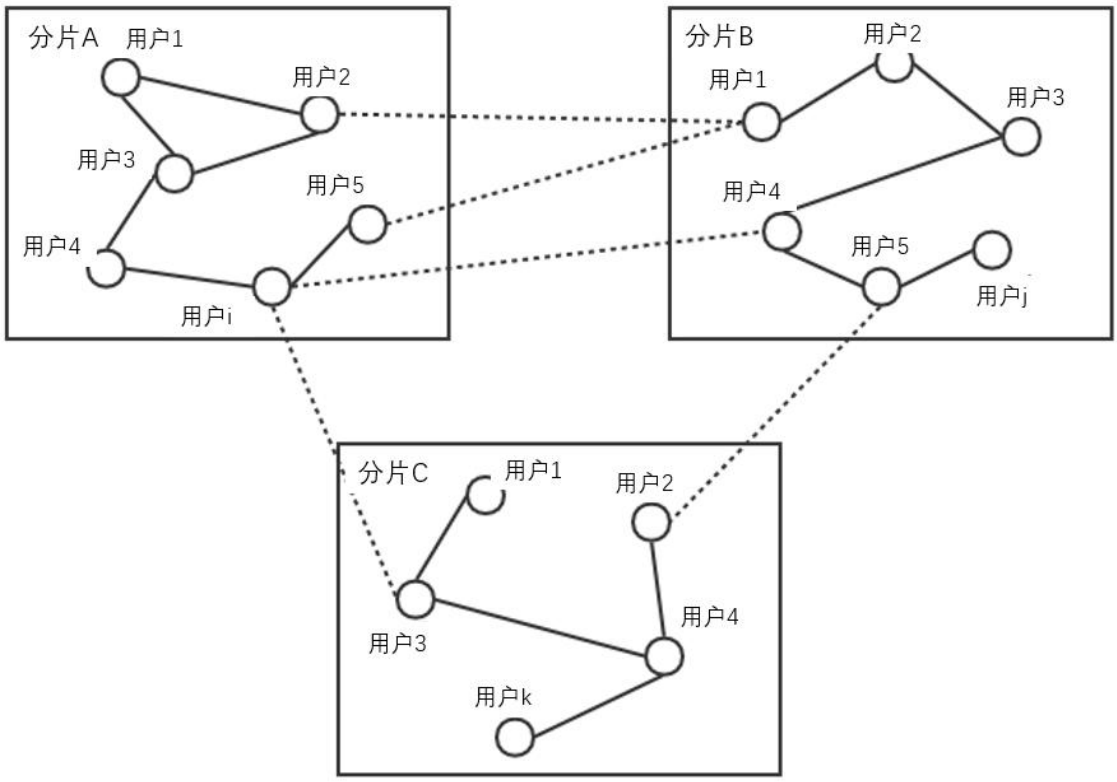

图 2-1　分片区块链的空间维度

分片区块链从功能上划分，可以分成网络分片、交易分片和状态分片。网络分片是将整个网络划分为多个部分，是对网络结构的划分。交易分片是将所有网络交易划分为几个部分进行验证和打包，网络的所有部分可以同时由不同的交易进行打包和控制，并行处理，从而提高了整个网络的效率。状态分片在每个分片中单独存储完整的账本信息，不再存储完整的区块链状态信息，并且是每个分片中维护的每个部分的账本信息。状态分片不仅根据网络分片和交易分片均衡事务处理，还为每个分片仅存储与该分片交易关联的状态信息，从而减少存储冗余。状态分片可以从本质上突破区块链的容量瓶颈。

分片区块链从程度上划分，可分为部分分片和完全分片和分层分片，具体节点的物理空间分布如图 2-2 所示。部分分片的典型例子就是 ELASTICO，其虽然实现了事务计算的分片，但没有实现系统中存储和通信的分片，因此被称为部分分片系统。由于每个节点接收和存储的系统信息都是完整的，因此系统中不存在跨分片事务。完全分片的典型例子就是 Monoxide，其在基于账户/平衡的系统中提出了一种中继机制。每个跨分片交易被分割成若干个子交易，包括内部交易和中继交易。每个内部交易都对应一个相关的分片，每两个连续的内部交易之间需要一个中继交易，保证了跨分片事务的原子性和一致性。完整分片的每个跨分片交易需要被分割成若干个子交易，需要验证和处理所有相关的子交易，才能提交

一个跨分片交易。分层分片的典型例子就是 Pyramid，其在完全分片的基础上设计了一种桥接分片。桥接分片用于连通多个独立分片，桥接分片中的节点可以共识并存储其相连的独立分片中的交易，并完成独立分片之间交易的发送。独立分片仅支持内部交易，桥接分片可以支持内部交易和交叉交易。

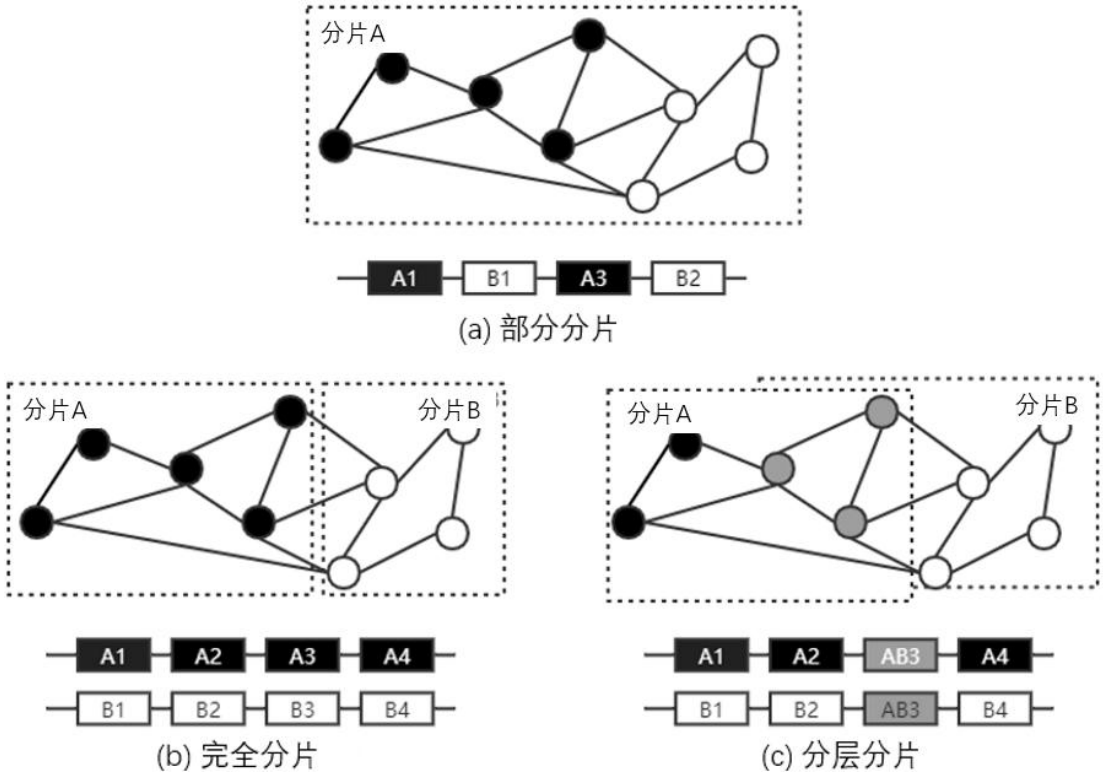

图 2-2　按程度分片分类图

2.1.2　分片区块链交易原理

以分层分片中的 Pyramid 作为例子，分片区块链的交易流程主要分为以下几步。

（1）分层分片组成

在节点加入网络时，需要解决一个 PoW 难题，并将解决方案提交，答案正确就可以进行身份注册。然后每个提交的节点会被随机分配一个 shardID。其中网络中的分片包括独立分片（i-shard）和桥接分片（b-shard）两种。

（2）跨片区块设计

对于跨片交易的来源有效性和结果有效性需要分别由对应的 i-shard 进行验证，而桥接两个 i-shard 的 b-shard 可以同时参与验证交易的来源有效性和结果有效性。而在交易通过验证以后，需要 b-shard 将其包装成区块才能进行共识。

（3）分层分片共识

Pyramid 中每个纪元包含多轮共识，每轮共识每个分片的 leader 都是被随机选举的。i-shard 的 leader 只能提交包含内部交易的区块，而 b-shard 的 leader 可以提交内部交易和跨片交易的区块。提交区块的流程包括三步，如图 2-3 所示。

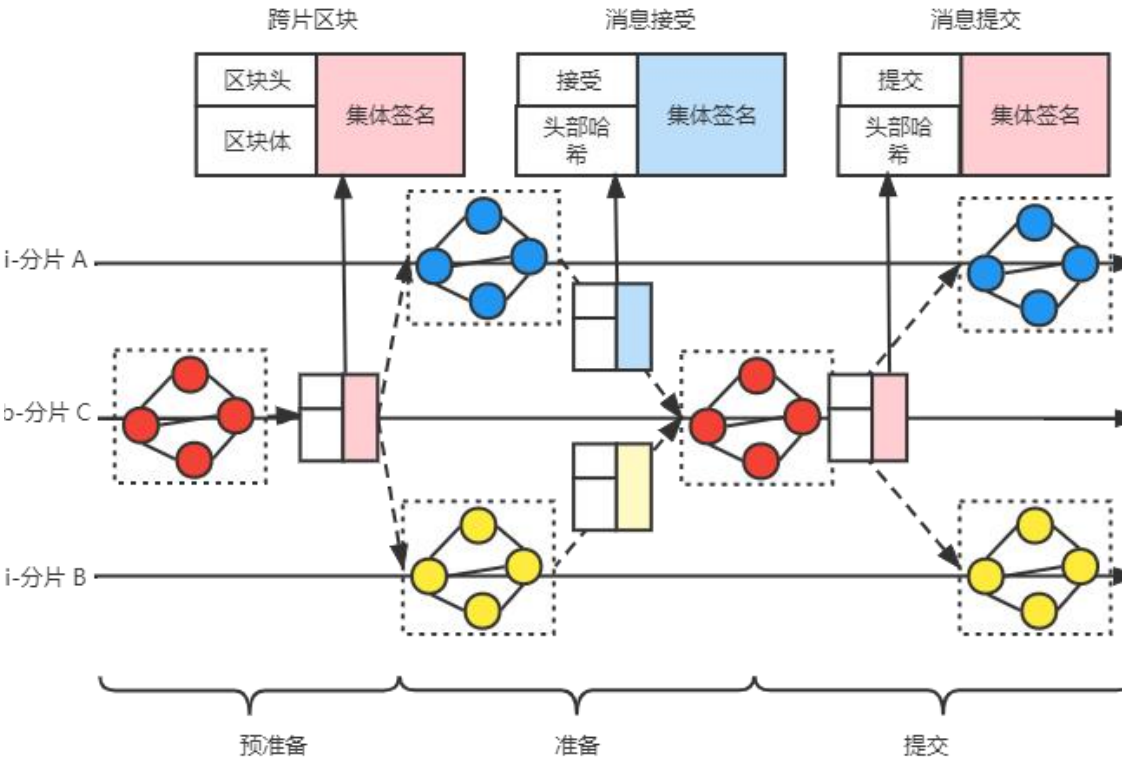

图 2-3　分层区块共识流程图

①区块预准备。b-shard 中的 leader 节点提出一个跨片区块（b-shard 中的节点都存储了关联的 i-shard 中的区块，因此可以验证交易）。然后 b-shard 中的节点基于 CoSi 协议[20]对该区块达成共识，产生集体签名。由于 i-shard 不知道该区块，该区块不能被提交，因此下一步需要把区块发给 b-shard 相关的 i-shard。

②区块准备。i-shard 接收到区块以后可用公钥验证签名。所有的 i-shard 对该区块达成共识，然后 i-shard 将 Accept 消息、Hash 头、集体签名发送到 b-shard。

③区块提交。b-shard 中的一个节点收到 Accept 就对 b-shard 其他节点广播，并把计数器-1。当计数器为 0 就可以确定区块能够在这轮被提交。然后 b-shard 产生新的集体签名、Hash 头、Commit 消息发送至 i-shard。

2.1.3 分片区块链技术挑战

尽管到目前为止，分片区块链技术已经有了很多成功的研究经验，但是其仍然存在很多技术难题，主要涉及分片内、分片间和系统层面三方面。

（1）对于分片内而言，首先存在 PoW 的 51%的攻击。对于使用 PoW 共识的区块链，将面临计算能力分散的问题。因此通常使用双层架构来确保分片的安全。其次存在 PBFT 共识的节点数量限制。PBFT 通信复杂度较高，所以节点过多会导致区块链性能明显下降，但节点过少又不能保证分片的安全。因此可以降低共识中 prepare 和 commit 阶段的通信次数来降低复杂度。最后存在女巫攻击。女巫攻击会导致一个节点伪造多个节点并谎称存有多份数据。因此主要通过 PoW 提高攻击门槛并引入身份验证机制解决问题。

（2）对于分片间而言，一方面是分片间的双花攻击。分片之间双花的攻击只能通过交叉通信来解决，系统必须在分片之间进行大量的通信以防止双花的问题。另一方面是跨分片的交易过载。分片机制虽然提升了性能，但也带来了跨片通信的额外开销。如果跨片交易过多，那么性能反而会降低。因此，在设计系统时要优化跨分片验证和交易的处理方式。

（3）对于系统层面而言，第一存在单点过热。当交易大多出现在某一分片，那么这个分片还是会发生延迟。因此在设计系统时需要考虑如何分散交易。第二需要动态调节。状态分片中的每个分片只存储网络中的部分子数据，因此在重新分配整个网络的状态时，必须确保整个系统的稳定性。

2.1.4 联邦分片机制

虽然分片区块链系统在区块链扩容方面起到了显著的成效，但当今大多数研究均集中在增加区块链中交易的吞吐量以及降低时延等性能方面的研究，对于隐私方面的研究也仅限于保证交易只在相关分片中共识。但如果节点遭到攻击，节点上存储的信息就会完全被泄露。因此本文考虑引入联邦思想，设计基于联邦思想的分片区块链进行差分隐私保护。

传统意义的分片是多个用户被分配到不同的分组当中，而联邦分片则是一个分片代表一个用户，分片中的众多节点共同去维护这个用户的信息，每个节点只存储完整信息的一部分。这样，当某个节点遭到攻击时，并不会泄露该用户完整的隐私数据。同时这样一个用户掌握多个节点的另一个优势就是当传统区块链的两个用户之间存在频繁交易时，很容易被攻击者检测到，但如果采用了联邦分片机制，用户就可以在每一个轮次随意获得消息发送方和接收方节点，一定程度上规避了被攻击的风险。综上，联邦分片机制的具体的拓扑结构如图 2-4 所示。

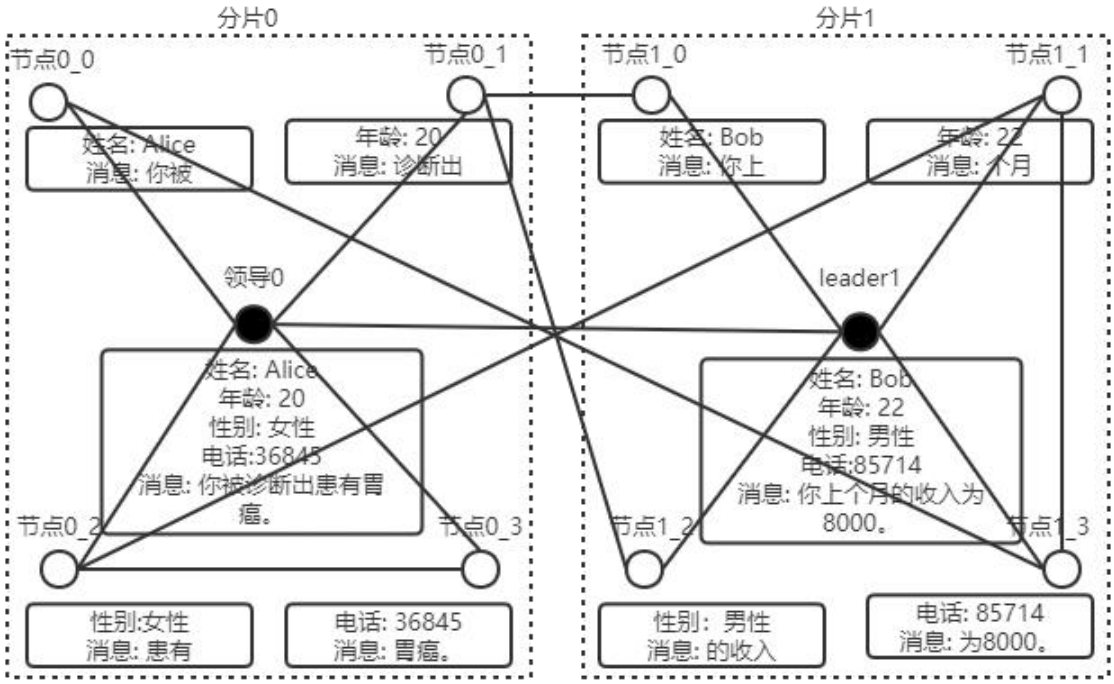

图 2-4　联邦分片机制拓扑结构

联邦分片中存在两种类型的节点：领导节点（leader node）和员工节点（staff node）。领导节点存储了该分片用户的完整信息，而员工节点则分布式地存储用户的部分信息。这主要是因为当分片中的节点接收到其余分片节点发来的消息并达成共识后，会将信息拆分并随机散布到分片的其他节点中。但是这样就存在一个弊端，当分片中的用户获取自身的完整信息时，从各个节点回收过来的消息可能无法按语序恢复成源信息。因此需要一个领导节点来存储用户的完整信息。虽然这样做如果领导节点被攻击就会导致用户隐私信息被泄露，但是大量的员工节点的加入使得领导节点被攻击的概率有效降低。此外，领导节点还主要负责在这一轮次中消息的发送和接收，便于消息的管理。因为每个轮次周期需要用户分片的节点进行洗牌，重新选举领导节点，所以即使两个用户之间存在频繁交易也不容易被察觉。根据这些理论，整个联邦分片的交易流程如图 2-5 所示。

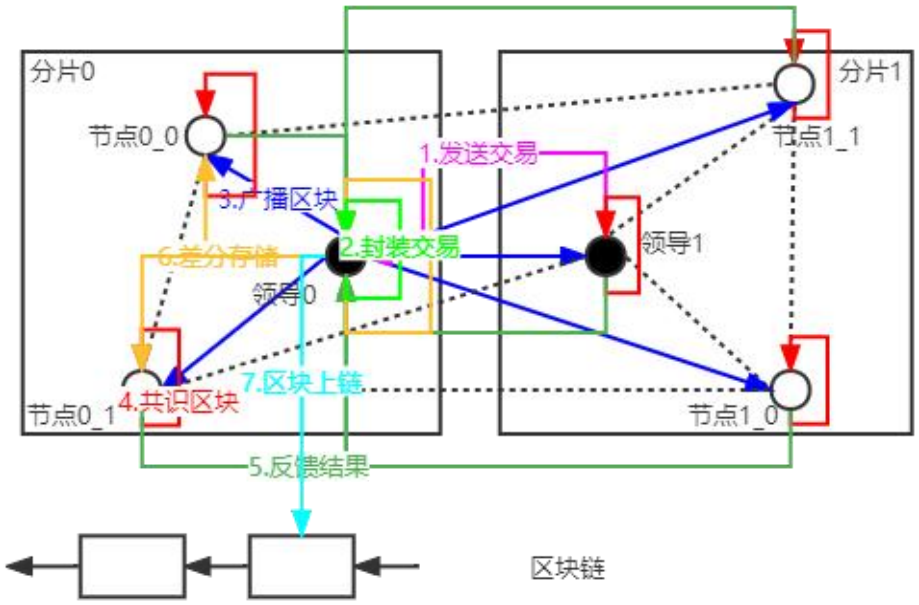

图 2-5　联邦分片交易流程图

整个联邦分片的交易流程可分为七步，发送交易、封装交易、广播区块，共识区块，反馈结果，差分存储和区块上链。

（1）发送交易

发送方的客户端从其分片的领导节点登录生成交易，在交易确认发送后需要验证发送有效性，在确认收到交易后需要验证接收有效性。

（2）封装交易

接收方节点收到交易在本地对交易进行一次运算，得到运算结果以后将交易封装成区块，便于消息后续的发送，同时也保护了数据隐私。

（3）广播区块

接收方节点将分装好交易的的区块广播给发送方节点和接收方节点所处的分片中的全部节点。

（4）共识区块

每个节点收到区块以后取出其中的交易分别进行一次运算，得到运算结果。

（5）反馈结果

每个节点将运算结果附上自身的签名返回给接收方节点，接收方节点对所有结果进行验证，如果超过 N-f（f 为恶意节点数量，拜占庭容错认为 3f+1≤N）个节点与接收方节点所计算的结果一致，表示该区块达成共识。

（6）差分存储

如果区块达成了共识，接收方节点就将其中的交易进行切分，分别存储到接收方节点所处分片的所有 staff 节点，同时把完整消息存储在自身。

（7）区块上链

在完成存储后，接收方节点就可以将区块上链，这笔交易就算完成。

2.2 可信验证机制的探究

2.2.1 可信验证概述

可信验证是可信计算的功能之一。在可信计算环境下，经常需要对远程用户的身份信息和平台信息的可信性进行验证。被验证方需要提供一个可信报告给验证方进行验证，验证方根据已有的基准值对可信报告进行验证的过程就称为可信验证。

可信身份验证基于信任根目录中的背书密钥 EK。EK 是一种非对称密钥，它唯一地存在于信任根中，不公开共享，并且可以被视为可信系统的报告根，代表了所位于的可信计算平台模块 TPM 的身份。但是如果在可信验证中频繁使用 EK 签名容易导致 EK 泄露。因此，为了提高报告受信任根的安全性，EK 通常用于生成用户和 CA 身份验证密钥的过程，而不参与其他受信任的报告或受信任的身份验证工作，并且 EK 公钥通常仅存储在 PrivateCA 证书中心。

因此可以由 PrivateCA 根据 EK 生成认证识别码 AIK（Attestation Identity Keys），让 AIK 代替 EK，作为 TPM 的身份象征。AIK 也是一个非对称密钥，只能用于签名，不可用于加解密，而且只能用于 TPM 内部信息的签名使用，这是为了防止攻击者进行 AIK 破解。

如果用户使用可信计算平台，则必须在响应该用户的平台上生成 AIK。用户可以使用 AIK 生成可信报告，并在外部确认当前可信根和可信根中的密钥的注册表状态信息。可以在可信报告中使用此信息，并将其与可信平台上的安全机制相结合，以实现用户身份和平台当前信任状态的远程身份验证。

而评价一个可信验证系统是否健全有效主要有以下几方面的要求。

在安全要求方面，基于可信根，系统引导加载程序、系统程序、通信设备的关键配置参数和通信应用程序可信，并且可以在关键应用程序执行链接上执行动态可信验证。检测到可靠性问题时，将生成警报，把验证结果记录以供检查，并发送到安全管理中心。在硬件启动和运行时，通过一致性验证或预安装的软件检测，可以检测和警告系统引导加载程序、系统程序、关键配置参数和操作关键应用程序的行为，方便后续操作。

在检查方法方面，首先，确定是否使用可信根来验证设备的引导系统、系统程序、关键配置参数和关键应用程序；其次，需要确保关键应用程序执行链接经

过了动态可信验证；此外，需要运行测试以确保设备不可靠时应报警；最后，验证结果将以审核记录的形式发送到安全管理中心。

在期望结果方面，第一，通信设备需要可靠的主芯片或硬件；第二，需要启动过程来根据可信系统引导加载程序、关键配置参数和可信根测量关键应用程序；第三，如果发现其可靠度受损，应及时报警，并将核查结果以审计记录的形式送交安全管理中心；第四，安全管理中心必须收到设备验证结果的记录。

2.2.2 可信验证流程

可信验证机制本质上可以抽象为四方参与的过程。如图 2-6 所示，参与方包括可信报告（TR）、可信基准（TS）、可信验证（TV）和密钥管理（CA）四个节点。其中，TR 节点是计算可信报告并提供给 TV 节点的验证。TS 节点可为 TV 节点生成可信基准值；TV 节点首先存储可信基准值，收到可信报告后，将根据可信基准值验证可信报告并计算验证结果；CA 节点主要负责可信报告相关密钥证书的发放。

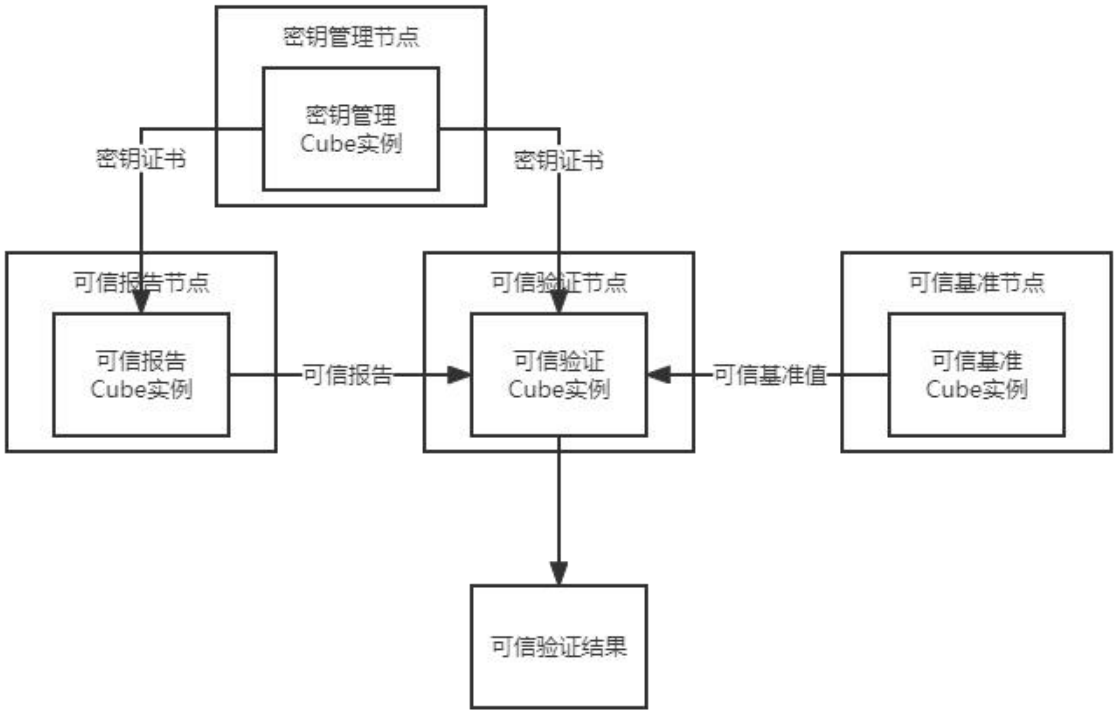

图 2-6　可信验证系统拓扑结构

可信验证的主要流程可分为签发密钥、可信基准、可信度量、可信报告、可信验证五部分。

（1）签发密钥

签发密钥阶段的主要流程就是 TR 实例和 TV 实例从 CA 实例中接收密钥的过程。这一过程主要在 TR 实例和 TV 实例部署完成后自动加载。其中，TR 实例从 CA 实例中获取被验证方（即 TR 节点自身）的 AIK 私钥（SK），TV 实例从 CA 实例中获取被验证方的 AIK 公钥（PK）。之后 TR 节点和 TV 节点就可以对所获得的密钥进行可信存储。在确定了验证方和被验证方之后，双方可以同时向 CA 节点请求密钥，没有先后顺序之分。具体实例间的路由信息如图 2-7 所示。

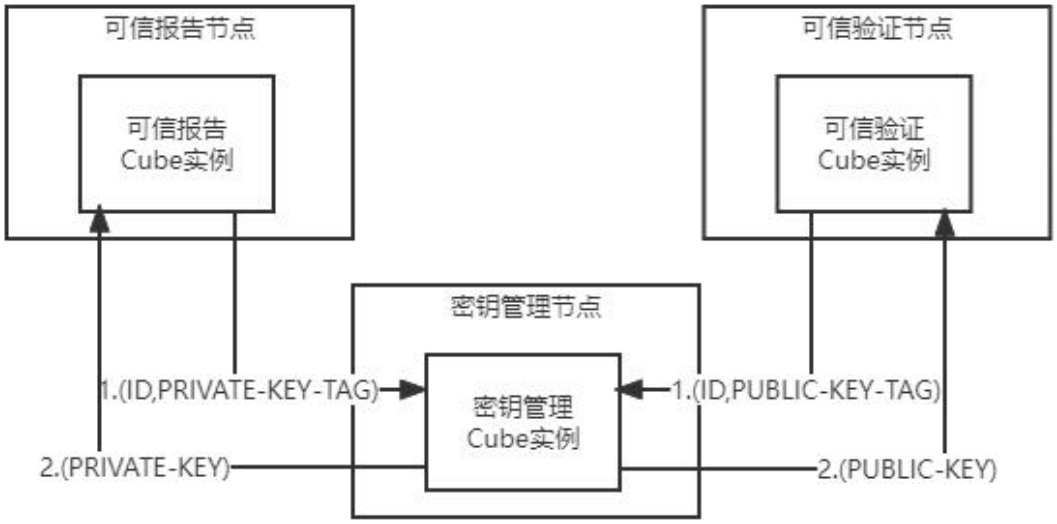

图 2-7　签发密钥接收路由

TR 实例向 CA 实例发送包含自身 ID 和所需密钥类型的消息（ID,PRIVATE_KEY_TAG），CA 实例收到消息以后就会将被验证方的 SK（PRIVATE_KEY）发送给 TR 实例。同理，TV 实例向 CA 实例发送包含被验证方 ID 和所需密钥类型的消息（ID,PUBLIC_KEY_TAG），CA 实例收到消息以后就会将被验证方的 PK（PUBLIC_KEY）发送给 TV 实例。

（2）可信基准

可信基准阶段主要流程就是当 TR 节点处于可信状态时，每次触发所有新的安全机制和部署策略时，TS 实例都必须在 TR 实例上测量安全机制和部署策略，将度量生成的 PCR 写入序列（PCR_sequence）作为可信基准值（standard_value）。之后 TS 实例就可以将 standard_value 发送给 TV 实例，用于后续的可信验证。PCR 是一个平台配置寄存器，用于存储平台的可信度量结果值。经过 SK 签名的 PCR 值（PCR_value）就是可信报告。而对平台每一步安全机制和载入策略的度量都会形成一个度量值，这些度量值按顺序相连就是一个写入序列。当前 PCR_value 是该序列的哈希值。基于哈希算法原理，攻击者无法编译另一个 PCR_sequence 来生成当前的 PCR_value，所以将 PCR_sequence 作为

standard_value 可以在可信验证阶段验证度量结果的可信性。具体实例间的路由信息如图 2-8 所示。

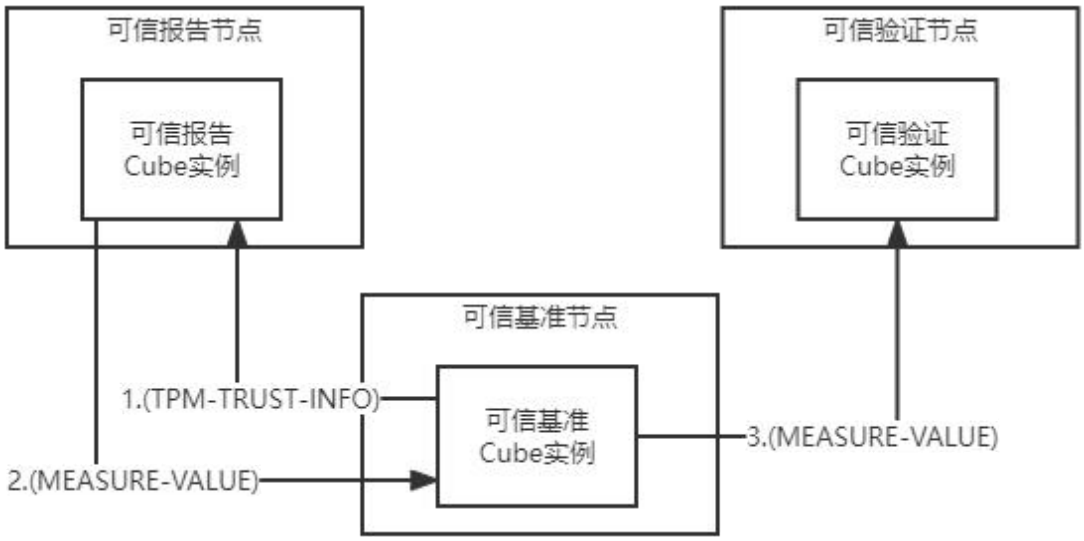

图 2-8　可信基准值生成流程路由

在确认 TR 节点属于可信运行的状态下，TS 实例对 TR 节点进行可信度量（TPM_TRUST_INFO）。之后就可以从 TR 节点处获得 PCR 写入序列作为可信基准值（MEASUER_VALUE）。在 TS 实例拿到可信基准值以后，就可以将其发送给 TV 实例进行可信存储，详细的可信基准值生成流程如图 2-9 所示。

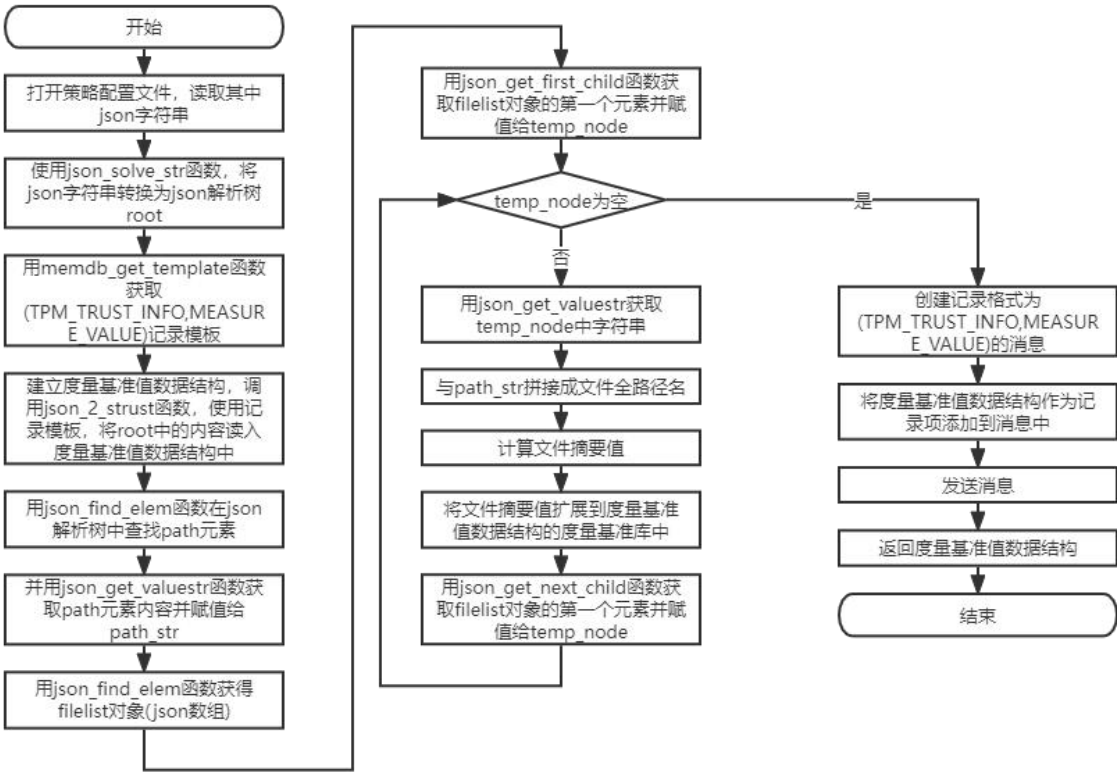

图 2-9 可信基准值生成流程图

（3）可信度量

可信度量阶段主要流程就是 TR 实例对自身可信平台的状态进行度量，并按照时间顺序生成 PCR 写入序列，然后将 PCR 写入序列的每次写入都按照式（2-1）的方法执行：

$$PCR_iNew = Hash(PCR_iOld\ value||value\ to\ add) \quad (2\text{-}1)$$

新保存的值与原始 PCR_value 合并，执行一次哈希操作，计算结果是新的 PCR_value。这样，如果度量结果集中的第 i 个值与预期值不匹配，则 PCR_sequence 中的第 i 个值也与预期值不匹配。而构建以 i+1 开头的新写入序列并在 PCR 中重建值与构造哈希算法冲突难度相同。

特定度量机制应根据平台的不同环境灵活设计。最简单的度量方法是测量平台上固定数据的完整性，可以分为平台实施策略和安全机制的度量。对于实施策略，主要度量是二进制可执行文件或实体内容配置文件的值，然后将度量结果和 standard_value 比较去判断是否遭到篡改。对于安全机制，在平台启动之前根据相同的过程计算摘要值，并将摘要值写入 PCR 中。

但如果度量平台中的关键数据是可变的，那上述度量方法就无法使用，因为可变部分的值无法确定。所以可以考虑用以下方法对可变数据进行度量：

①度量可变数据修改者身份。可信系统可以在配置者对数据进行修改的时候对配置者的身份信息进行度量，通过组合度量结果和可变数据计算的摘要值用作度量可变数据的结果。

②度量可变数据认证记录。修改者更改可变数据后，可信系统可以在审核完成后请求管理员查看可变数据并发送凭据。通过将凭据测量为可变数据的指标来生成摘要值作为可变数据度量结果。

③度量可变数据生成流程。可信系统可以设置安全策略，以便只有配置接口程序才能修改可变数据。度量安全策略、配置接口程序以及使用用户 ID 与可变数据合并以生成摘要值作为可变数据度量结果。

可信度量主要在 TR 节点内部执行，TR 实例在节点内部生成 PCR 写入序列和 PCR 值，并不涉及节点之间的信息路由。可信度量具体流程如图 2-10 所示。

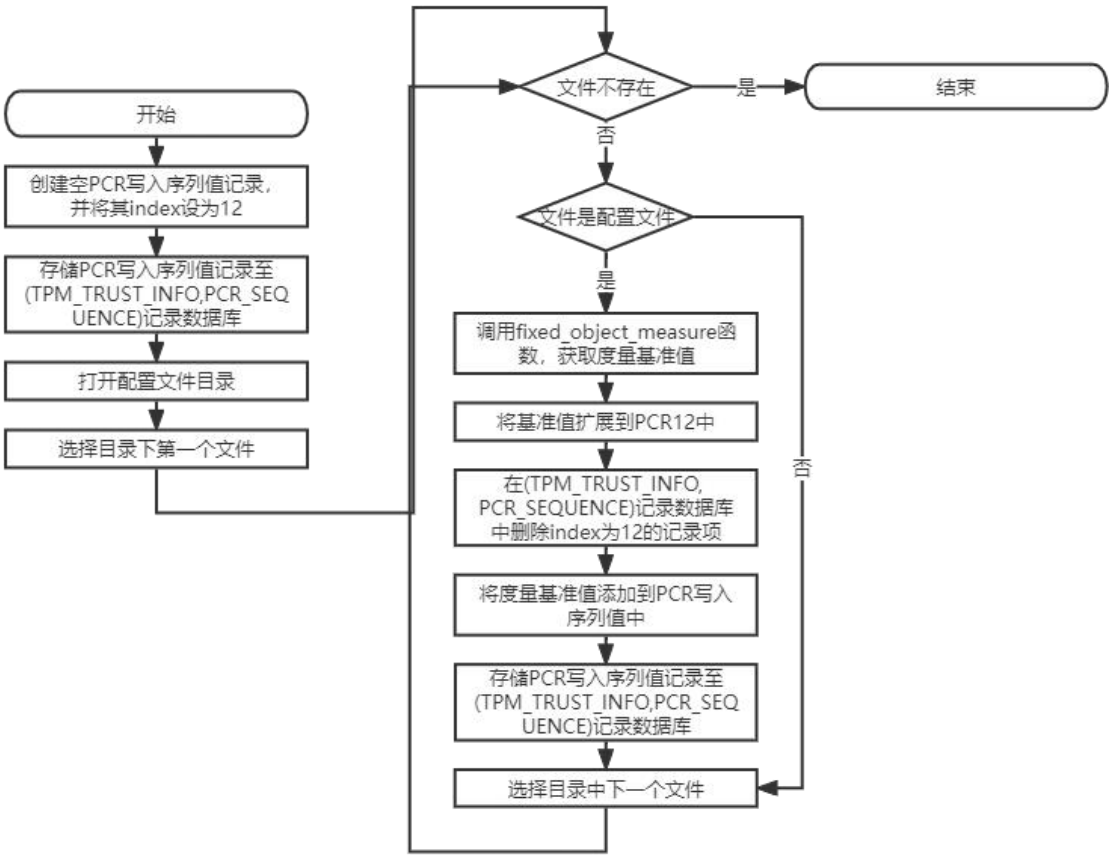

图 2-10 可信度量流程图

（4）可信报告

可信报告阶段主要流程就是 TR 实例使用从 CA 实例获得的 SK 对 PCR_value 进行加密生成可信报告（trusted_report），然后将 trusted_report 连同自身可信度量的 PCR_sequence 一起发给 TV 实例进行后续的可信验证工作。

节点根据 PCR_value 生成的 trusted_report 证明了 PCR_sequence 的真实性。对于固定数据对象，每个 PCR_sequence 对应于指标对象的度量情况，因此可以使用 PCR_value 来反映 PCR_sequence 是否可信。对于可变数据对象，PCR_sequence 是不确定的，PCR_value 无法确定 PCR_sequence 可信性。因此，除了将 trusted_report 和 PCR_sequence 发送到 TV 实例之外，还必须将可变对象度量的结果发送到验证程序，以便向验证程序发送足够的信息以确定 TR 节点的可靠性。具体实例间的路由信息如图 2-9 所示。

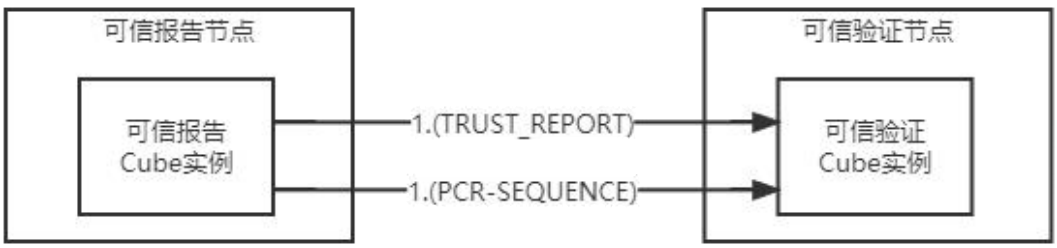

图 2-11　可信报告流程消息路由

在 TR 实例对自身所处节点完成可信度量后，使用 SK 对生成的 PCR 值加密生成可信报告（TRUST_REPORT），然后将可信度量阶段生成的 PCR 写入序列（PCR_SEQUENCE）与可信报告一起发给 TV 实例。可信报告生成的具体流程如图 2-12 所示。

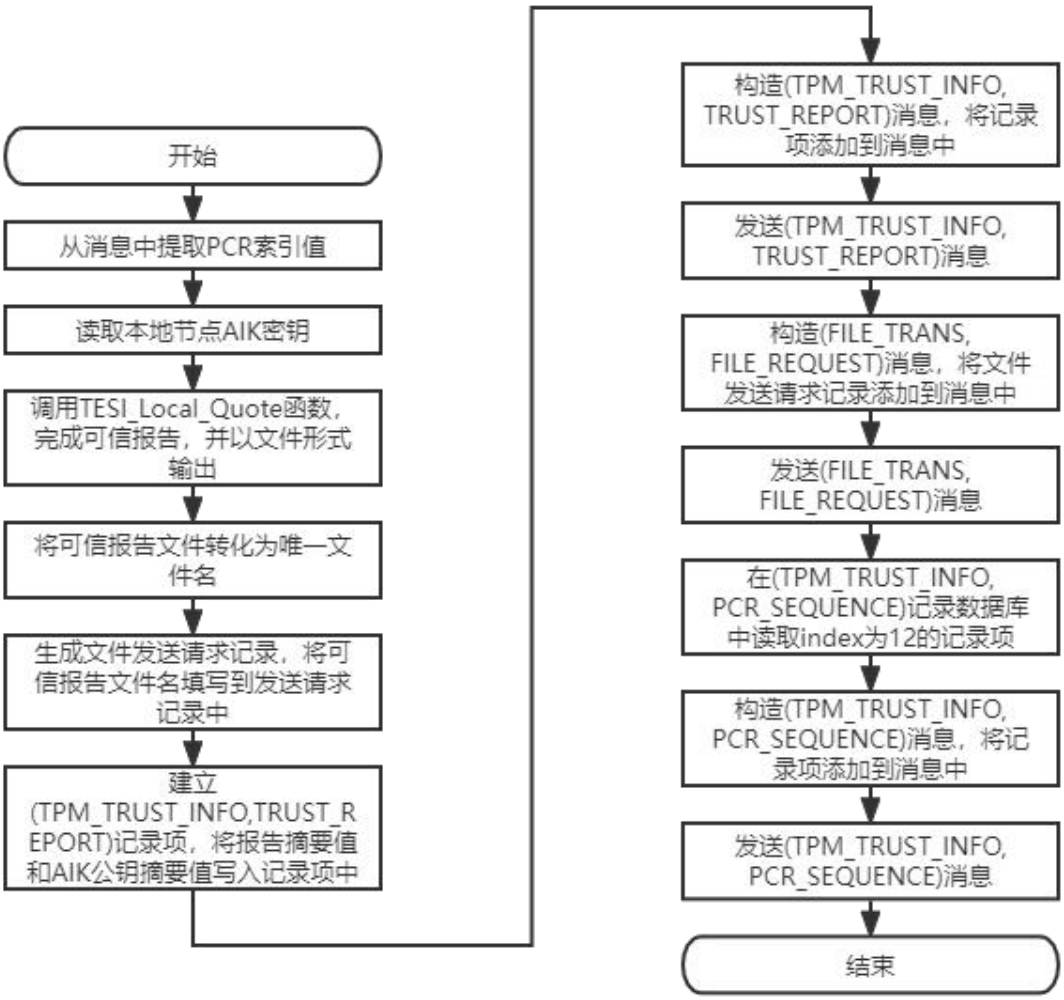

图 2-12 可信报告生成流程图

（5）可信验证

可信验证阶段的主要流程就是 TV 实例将从 TR 实例处获得的 trusted_report 以及 PCR_sequence，就可以开始对 TR 节点展开验证工作。具体工作可分为以下四步。

①验证 trusted_report 的可信性。TV 实例在收到 trusted_report 以后需要用从 CA 实例处拿到的被验证方的 PK 对 trusted_report 进行解密，如果能还原出原始信息，证明 trusted_report 可信，并取出 PCR_value。

②验证 PCR_sequence 真实性。TV 实例将从 TR 实例处收到的 PCR_sequence 按照生成规则生成一个预测值 PCR_value，并与 trusted_report 中的 PCR_value 比较，如果相同，证明 PCR_sequence 真实。

③验证度量结果真实性。对于固定数据对象，PCR_sequence 对应的度量值就是度量结果。如果 PCR_sequence 真实，就表示度量结果真实。但是对于可变数据对象，TV 实例在获得 TR 实例提交的对于可变数据对象的度量结果后，要计算其摘要值，将摘要值与 PCR_sequence 中可变数据对象对应的度量结果比对，如果相同，则表示度量结果真实。

④验证度量结果可信性。TV 实例需要将从 TR 实例处收到的 PCR_sequence 与其从 TS 实例处收到并已经存储在可信基准库中的 standard_value 进行比较，如果二者相同，则表示度量结果可信。

当以上四个步骤都通过验证后，就表示可信验证通过，确认被验证方平台以及身份信息真实可靠。简而言之，可信验证主要是 TV 节点内部的 TV 实例对从被验证方得到的 trusted_report 进行验证，判断 trusted_report 是否可信、PCR_sequence 是否真实、度量结果是否真实以及度量结果是否可信，并不涉及节点间的消息路由。可信报告生成的具体流程如图 2-13 所示。

图 2-13 可信验证程序流程图

2.2.3 可信验证技术挑战

可信验证作为一种可信密码服务，是可信计算的重要组成部分。因此可信验证面临的技术挑战就是可信计算面临的挑战。可信计算作为一种新型的技术实现了计算机体系结构的主动免疫，已主动监控的方式监视系统的行为，保证系统能够按照预期运行。但是目前已有的可信计算平台并不完善，还存在许多问题和挑战。

第一，可信计算是一种新的模型和保护方法，可改变某些用户的使用习惯。现阶段，可信计算的使用还不充分，需要政策支持，以加强自主创新体系建设。

第二，可信计算缺乏系统标准和行业标准，产品相对分散，不形成合力。

第三，在新的 IT 应用程序环境中（如大数据、云计算和物联网），如何确保可信计算安全并为业务应用程序提供安全支持也是亟需解决的问题。

第四，随着量子计算的发展，需要构建具有量子抗性的可信计算技术系统。因此如何在可信平台上设计高效的量子加密算法和协议，也是目前需要深入研究的问题。

2.3 基于可信验证的联邦分片区块链

虽然联邦分片区块链已经能有效地对数据进行差分隐私保护，但这种保护仅仅作用于数据存储上。在数据在网络中传递时，仍然存在隐私数据被披露的风险。主要的风险情况交易的创建、发送与接收阶段。在交易被一个可信节点接受以后就会被封装成区块，这使得交易被披露的风险减小。因此可以把风险攻击分为发送方为故障节点、发送方为恶意节点、接收方为故障节点、接收方为恶意节点四种情况。

（1）发送方为故障节点。如果发送方为故障节点，那当客户端从该节点接入节点网络时就可能无法发送消息。

（2）发送方为恶意节点。如果发送方为恶意节点，在进行交易发送时，发送方就有可能将消息广播到网络中的所有节点，导致消息披露。

（3）接收方为故障节点。如果接收方为故障节点，在收到交易后就可能不会把交易发送到其他节点进行共识，那交易就被停滞。

（4）接收方为恶意节点。如果接收方为恶意节点，在收到交易后就有可能把交易广播到网络中的所有节点中，导致消息披露。

因此为了有效解决交易在传播阶段的隐私泄露问题，考虑将可信验证机制引入联邦分片区块链中，保证在交易传输阶段发送方和接收方是安全可信的，以增强其隐私保护功能。

具体的操作步骤就是在交易发送之前，以发送方节点作为可信验证节点，接收方节点作为可信报告节点，并在不属于发送方与接收方所处分片的第三个分片任选一个节点作为可信基准节点对接收方的平台和身份进行可信验证。同理，再将接收方节点作为可信验证节点，发送方作为可信报告节点，再对发送方的平台和身份进行可信验证。当发送方和接收方都通过可信验证时，就证明两个节点可靠，可以进行交易传输。

2.4 本章小结

本章从原理、功能、运行流程以及技术挑战等方面介绍了分片区块链及可信验证机制的基本概念。并在分片区块链的基础上提出联邦分片区块链方案用于对信息进行差分隐私。同时也讨论了将可信验证机制引入到联邦分片区块链中的可行性和有效性，基于这些理论理论用于后续模型的设计和开发。

第三章　基于可信验证的区块链联邦隐私保护模型

为了更好地理解联邦分片区块链进行差分隐私保护的运行机制，以及更直观地探讨在引入了可信验证机制以后对联邦分片区块链安全性的提升效果，基于上一章对二者的学习和研究，设计出一个基于可信验证的区块链联邦隐私保护模型，并对模型进行结构上、功能上的研究和分析。

3.1　模型架构

整个基于可信验证的区块链联邦隐私保护模型构建于拜占庭容错系统之上，即假设系统总共存在 n 个节点，而存在 m 个恶意节点，其中 $n \geqslant 3m+1$。所以一般认为 $m \approx n/3$。

基于可信验证的区块链联邦隐私保护模型的主要功能如下：

（1）能够在数据存储时进行差分隐私保护。在通用状态分片区块链结构的基础上进行改良和优化，引入联邦思想进行差分隐私保护。需要具有初始化节点功能、发送交易功能、广播区块功能、共识区块功能、差分存储功能、区块上链功能以及节点作恶功能。

初始化节点的功能是在区块链网络中初始化一部分节点，将节点随机分配到不同的分片中并同时随机选举出 leader 节点。发送交易的功能是对接可信验证模块，将交易发送给接收方节点并验证发送方有效性。广播区块的功能是将交易封装成区块并广播到发送方和接收方所在分片的所有节点。共识区块的功能是每个节点分别计算所收到的区块并把结果连同自身签名返回给接收方，如果超过 2/3 的节点共识结果一致则表示达成共识。差分存储的功能是接收方把完整交易进行，并把交易分割成子交易随机分发给分片中其他节点。区块上链的功能是接收方节点把交易区块上传到链上，证明该笔交易达成。节点作恶的功能是节点可能是恶意或者故障的，使得发送或接收消息时可能出现不反馈或者全局广播消息的情况以及共识阶段可能出现不一样的计算结果。

（2）能够在数据通信时进行双方可信验证。依据可信计算中可信验证的基本思想针对节点的平台和身份信息进行可信验证，实现隐私保护。需要具有密钥签发功能、可信基准值生成功能、可信度量功能、可信报告生成功能、可信验证功能以及攻击反馈功能。

密钥签发的功能是 CA 节点将可信报告节点的 AIK 私钥发给可信报告节点，将 AIK 公钥发给可信验证节点。可信基准值生成的功能是在可信基准节点度量处于可信状态的可信报告节点，生成 PCR 写入序列作为可信基准值并发送给可信验证节点。可信度量的功能是可信报告节点度量其自身状态并生成 PCR 写入序列，同时根据哈希算法将 PCR 写入序列生成 PCR 值。可信报告生成的功能是可信报告节点将 PCR 值用 AIK 私钥加密形成可信报告，连同 PCR 写入序列一起发给可信验证节点。可信验证的功能是可信验证节点利用 AIK 公钥验证可信报告的可信性，利用 PCR 写入序列验证度量结果的真实性，利用可信基准值验证度量结果的可信性。攻击反馈的功能是设置多种攻击方式使得三种可信验证步骤失败时向客户端抛出异常。

综上，整个基于可信验证的区块链联邦隐私保护模型的基本功能如图 3-2 所示。

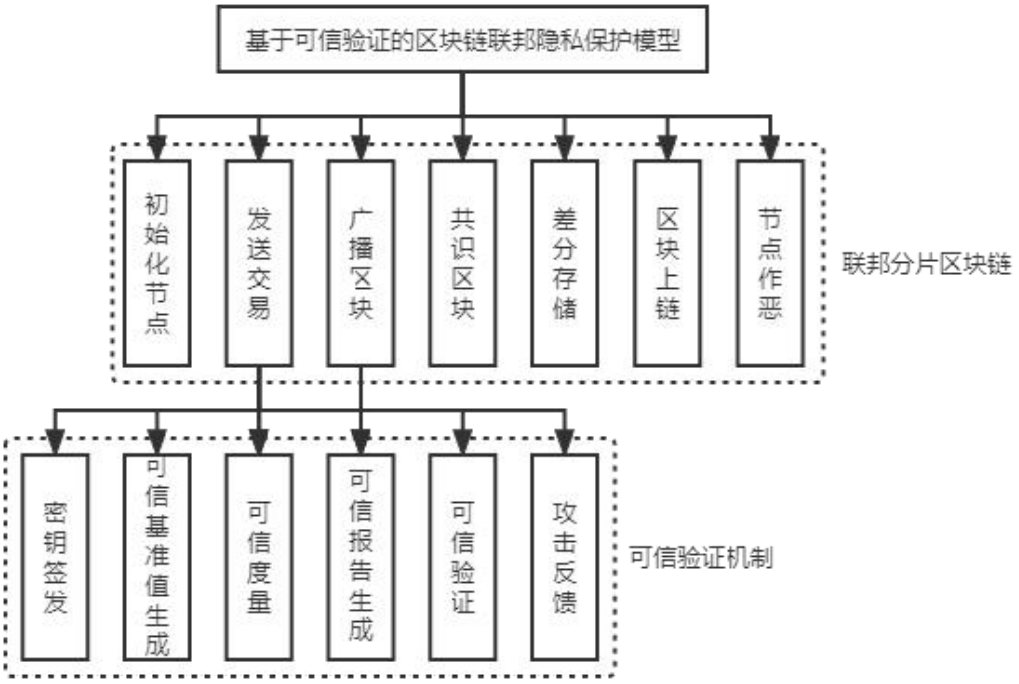

图 3-1　模型基本功能图

而具体的模型流程如图 3-1 所示。首先要部署好基于可信验证的联邦分片区块链，然后初始化节点。网络会把这一轮次的 leader 节点分配给客户端，当客户有发送交易的需求时通过客户端接入节点网络。在确定接收方节点地址后启动可信验证机制对发送方节点和接收方节点进行可信验证。未通过验证就抛出异常返回给客户端，否则就将交易发送给接收方节点。接收方节点将交易包装成区块后在发送方和接收方分片内对区块进行广播。然后相关节点收到区块后对其进行共

识并返回共识结果给接收方节点。如果超过 2/3 的节点共识结果一致，则表示达成共识，接收方节点将完整信息进行存储，然后将消息进行切分，随机发送给分片中的其他 staff 节点。完成差分存储以后，将区块上链，接受这一轮交易。

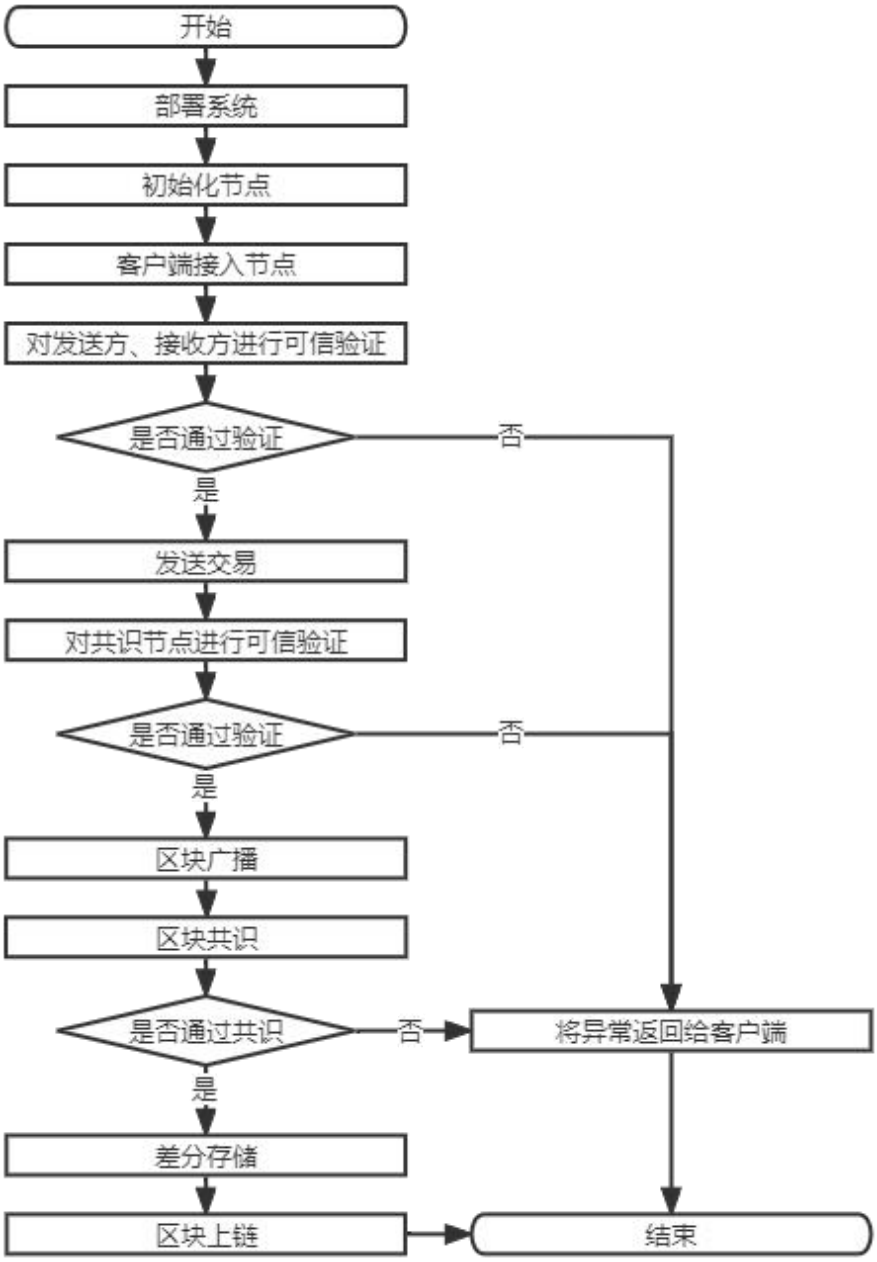

图 3-2　模型具体流程图

3.2　联邦分片区块链的存储隐私保护

联邦分片区块链首先会初始化一部分节点信息，然后让每个节点都计算一个 PoW 难题，计算结果被确认之后，就允许节点注册身份，加入到区块链网络中。

然后在每一轮中，每个节点都随机分配给一个分片，并在分片中随机选择一个作为 leader 节点，并将新的 leader 节点地址发送给每个用户的客户端。通过在每一轮随机分配节点进入分片和随机选取 leader 节点的方式，可以有效避免由于分片中节点数少而可能产生的 PoW 共识 51%攻击。其中有一个特殊节点为 CA 节点，其负责所有节点的密钥管理与分发，在初始化以后不会加入到任何分片中。

在某一轮中，当某一用户希望向其他用户发送消息时，他可以通过客户端进入区块链节点网络，系统在这一轮次的 leader 节点（sender）即用户的登入接口。同时根据用户希望发送消息的接收方，将接收方这一轮次的 leader 节点作为接收方节点（receiver），并将地址返回给 sender。然后分别调用可信验证机制对 sender 和 receiver 进行可信验证。通过可信验证后就将交易发送给 receiver，并对交易的发送有效性和接受有效性进行确认，即证明 sender 确实能发出了消息，receiver 确实收到消息。

receiver 收到交易以后就将交易打包，封装成区块，这样在一定程度上保证了信息的安全。然后将区块广播给 sender 和 receiver 所处的两个分片中的所有节点去共识。这样就保证了只有发送方和接收方的节点知道这笔交易，一定程度上保障了数据的隐私。

当每个节点收到区块时，会分析该区块并将交易数据提取到本地，并调用节点自身的算力去计算这笔交易信息，例如如果是转账交易就会计算转账后双方的余额会更新成多少。最后将计算的结果附带这个节点一个自身身份的签名反馈给 receiver。如果是可信节点，得出的结果会与 receiver 的计算结果相同；如果是故障节点，可能不会返回任何计算结果；如果是恶意节点，则可能返回一个错误值。当 receiver 收回所有共识结果后对和自身计算结果相同的的签名进行计数，如果超过 2/3 的结果一致，则表示达成共识，可以进行下一步。否则共识失败，这笔交易就被作废。

达成共识后 receiver 就会将这笔交易进行切分，如果分片中共有 N 个节点，就平分成 N-1 份。然后因为 receiver 就是该轮次的 leader 节点，因此将完整消息进行存储，然后将部分交易随机地分发给其余所有 staff 节点存储，这样就以联邦的形式完成了差分隐私。

之后 receiver 因为存储了完整信息，就可以将获取前一个区块的 Hash 值、时间戳和交易信息拼接，生成该区块自身的 Hash 值，将数据填充完整后就可以把区块上链了。当区块成功上链以后，这笔交易就算完成了。

但是为了确保模型的完备性。需要设置故障和恶意节点去保证模型在错误情况下能够及时抛出异常。在没有使用可信验证机制时，sender 和 receiver 会有 1/3 的概率成为故障节点或者恶意节点。当 sender 是故障节点时，不会发送客户端传输的交易给接收方；当 sender 是恶意节点时，会广播客户端传输的交易到整个网

络中导致消息泄露；当 receiver 是故障节点时，不会将交易封装成区块进行广播；当 receiver 为恶意节点时，会将交易封装成区块后广播到网络中所有节点，导致消息泄露。而由于基于拜占庭容错网络模型，因此无论是否使用可信验证机制，在区块共识阶段，总会难以避免因为故障节点不返回共识结果，恶意节点返回错误共识结果而导致共识失败，交易作废。

下面给出联邦分片区块链的存储隐私保护过程的伪代码：

算法 1：联邦分片区块链的存储隐私保护

输入： 交易信息 *s*

输出： 区块上链反馈信息 *r*

预处理： 联邦分片区块链系统部署完成且已完成 *leader* 选举

sender 发送交易 *s* 并验证有效性;
if (调用可信验证) {
  调用可信验证机制; }
if (*sender*==故障节点) {
  return; //不发送交易 }
if (*sender*==恶意节点) {
  披露交易给全网络;
  return; }
receiver 接收 *s* 并验证有效性;
if (*reveiver*==故障节点) {
  return; //不封装交易 }
receiver 封装 *s* 成区块；
if (*receiver*==恶意节点) {
  将区块广播至全网络;
  return; }
reveiver 将区块发至分片内其余节点;
节点共识交易;
返回共识结果给 *receiver*;
if (达成共识数<分片内总结点数*2/3) {
  return; //未达成共识 }
receiver 将交易切分;
reveiver 将切分后的交易差分存储到分片内其他节点，其

自身保留完整信息;

23 *receiver* 将区块上链;

24 **return** 上链信息 *r*;

基于以上逻辑，联邦分片区块链的具体流程如图 3-3 所示。

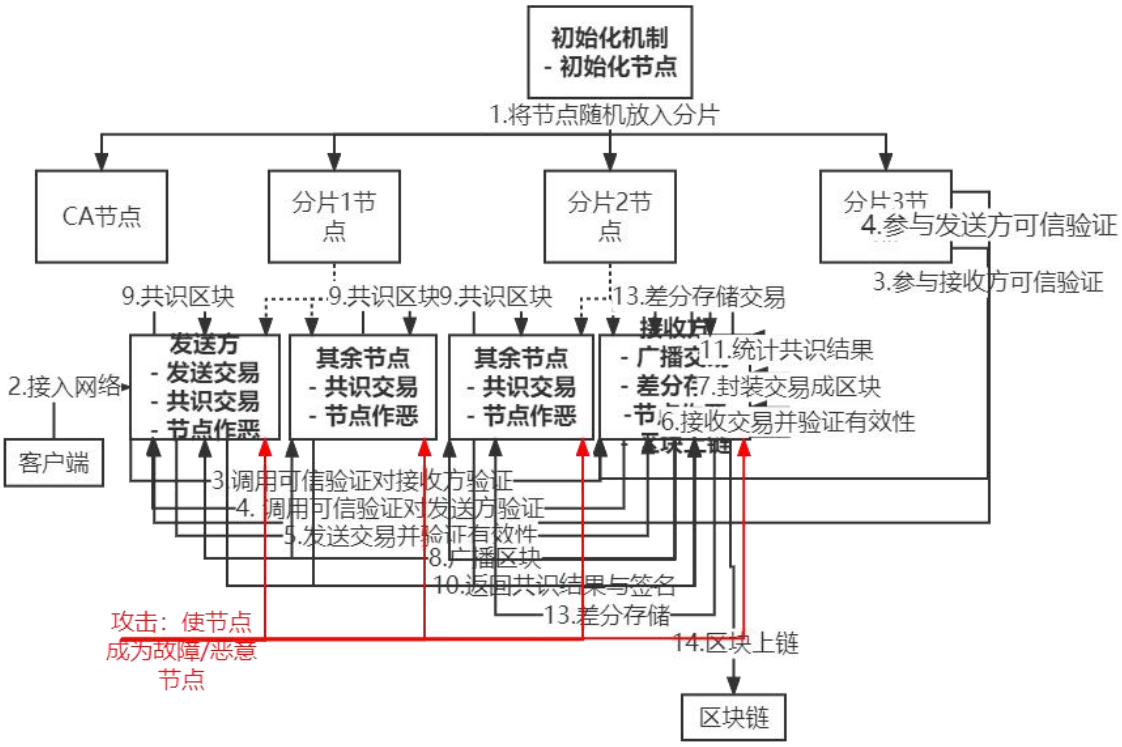

图 3-3　联邦分片区块链具体流程图

3.3　可信验证机制的通信隐私保护

因为区块链中的节点在加入网络时其内部的配置数据就不会发生变化，可以把节点信息看做固定数据对象。当联邦分片区块链调用了可信验证模块后，该模块就会启动。首先对 receiver 的可信状态进行验证，然后对 sender 的可信状态进行验证。以对 receiver 可信验证为例，将 receiver 作为可信报告节点（TR），sender 作为可信验证节点（TV），然后在不属于接收方和发送方的第三方分片中随机选取一个节点作为可信基准节点（TS），就可以展开可信验证。反过来同理。

开启可信验证以后，TR 节点就会将自身的 ID 和需要的密钥类型发送给密钥管理的 CA 节点去请求私钥，CA 节点收到指令后就向 TR 节点发送其自身的 AIK 私钥。TV 节点会把 TR 节点的 ID 和需要密钥类型发送给 CA 节点请求公钥，CA 节点会向 TV 节点发送 TR 节点的 AIK 公钥。

当 TR 节点处于可信状态下时，TS 节点就会对 TR 节点的平台和身份状态进行可信度量，按照时间顺序生成一个度量序列，即 PCR 写入序列（PCR_sequence）。由于此时的 TR 节点可信，就可以将其作为 TR 节点的可信基准值（standard_value）。然后 TS 节点就将 standard_value 发送给 TV 节点进行可信存储，将 standard_value 存入可信基准库（stantard_lib）中。

TR 节点可以对自身进行可信度量，主要度量安全机制和载入策略，然后将度量结果以 PCR_sequence 的形式呈现出来。之后就将 PCR_sequence 的每一个度量结果依次拼接当前 PCR 值（PCR_value）并进行哈希计算形成新的 PCR 值，直至将所有度量结果运行完就是当前最新的 PCR_value。

TR 节点将自身的 ID 和 PCR_value 进行拼接，并用已经获得的 AIK 私钥进行加密就生成了可信报告 trusted_report。然后把之前度量出来的 PCR_sequence 和 trusted_report 一起发给 TV 节点。

TV 节点收到 trusted_report 和 PCR_sequence 以后，就可以对 TR 节点进行可信验证。首先用 AIK 公钥对 trusted_report 解密，并将解密后的数据拆分，如果能够得到 TR 节点的 ID，则证明 trusted_report 可信，就可以取出 PCR_value。然后使用得到的 PCR_sequence 按照写入规则生成一个 PCR_value2，并与 trusted_report 中得到的 PCR_value 比较，如果相同，则证明可信度量真实。之后再拿 PCR_sequence 与 standard_lib 中的 standard_value 进行比较，如果一致，则证明度量结果真实。至此完成可信验证全部流程。

但是为了确保可信验证的完备性，需要分别对可信验证的不同步骤进行攻击，保证系统能识别错误行为并及时抛出异常。对 trusted_report 进行攻击，修改其中可信报告节点的 ID，则将导致可信报告不可信；对度量结果进行攻击，修改可信报告 PCR_value，则将导致度量结果不真实；对度量过程进行攻击，修改 PCR_sequence，则将导致度量结果不可信。

下面给出可信验证机制的通信隐私保护过程的伪代码：

算法 2：可信验证机制的通信隐私保护

输入： 当前可信状态 *i*

输出： 可信验证结果 *v*

预处理： 完成节点配置，密钥分发和可信基准值生成

1 *TR* 节点度量自身可信状态 *i* 生成 *PCR_sequence* 和 *PCR_value*;

2 *TR* 节点将 *PCR_value* 与自身 *ID* 拼接，并用 *AIK* 私钥加密生成可信报告 *trusted_report*;

3 *TR* 节点将 *trusted_report* 同 *PCR_sequence* 一起发给 *TV* 节点;

4 *TV* 节点接收 trusted_report 和 *PCR_sequence*;
5 *TV* 节点使用 *AIK* 公钥解密 *trusted_report*;
6 **if** (*ID*!=TR 节点真实 *ID*) {
7 **return** *v*=可信报告不可信; }
8 使用 *PCR_sequence* 按照写入规则生成新的 *PCR_value2*;
9 **if** (*PCR_value*!= *PCR_value2*) {
10 **return** *v*=度量结果不真实; }
11 **if** (*PCR_sequence*!=standard_lib 中存储的 *standard_value*) {
12 **return** *v*=度量结果不可信; }
13 **return** *v*=可信验证通过;

基于以上逻辑，可信验证机制的具体流程如图 3-4 所示。

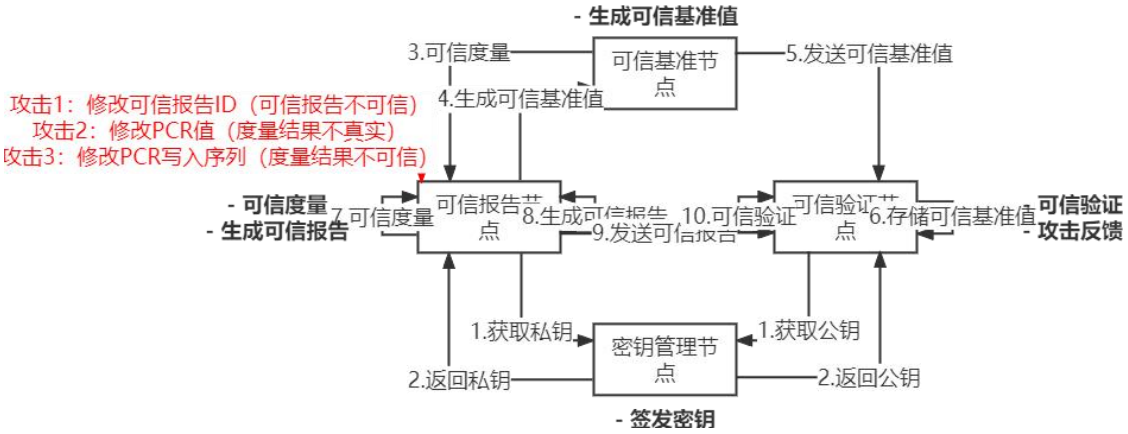

图 3-4　可信验证机制具体流程图

3.4　本章小结

本章基于对联邦分片区块链和可信验证机制的研究，对基于可信验证的区块链联邦隐私保护模型进行了整体框架模块化设计，明确了模型的具体功能。同时分别对联邦分片区块链的差分隐私存储和可信验证的通信隐私保护两个核心功能的具体结构进行了详细的分析与设计。

第四章　实验与结果分析

基于对联邦分片区块链和可信验证机制的原理理解和设计框架，通过实验证明联邦分片区块链在差分隐私保护上的可行性，以及确认引入可信验证机制后有助于提高联邦分片区块链的安全性。使用控制变量思想，设计对照实验对模型进行多维度攻击。并基于不同的攻击类型得出具有说服力的实验结果。

4.1　实验设计

为了证明基于联邦思想的分片区块链模型确实能够对隐私数据进行差分存储，以及在引入可信验证机制后系统具有更强的安全性。本文首先针对联邦分片区块链模型设计出原型系统，模拟出整个交易的传输、存储以及受到节点作恶时的及时反馈流程，证实提出的方案的可行性。同时设计对照实验，模拟在相同的网络环境下，在引入可信验证机制前后，系统中的交易在传输时的安全性。并对可信验证中可信报告节点中可信报告的 ID、PCR 值、PCR 写入序列分别进行攻击，探究可信验证机制在受到攻击时应该做出的反馈，以设计更优的系统。

本文以 15 各节点和 3 个分片为例展示从节点初始化进网络到一个轮次中一笔交易达成共识并成功上链的整个流程。整个实验分为两个模块：联邦分片区块链模块和可信验证模块。联邦分片区块链模块主要构建联邦分片区块链系统，在定义完节点、交易和区块结构体后，实现交易在系统中传输、共识、存储、上链的整个流程。同时也设计出节点故障或作恶的情况以及其对应的异常反馈机制，并能在交易发送前调用两次可信验证模块分别对发送方和接收方进行可信验证。可信验证模块主要在定义完可信报告节点、可信基准节点、可信验证节点和密钥分发节点结构体以后运行可信验证的整个流程。包括密钥分发、可信基准值生成、可信度量、可信报告生成、可信验证以及针对可信报告的三种攻击方式。具体的实验拓扑结构如图 4-1 所示。

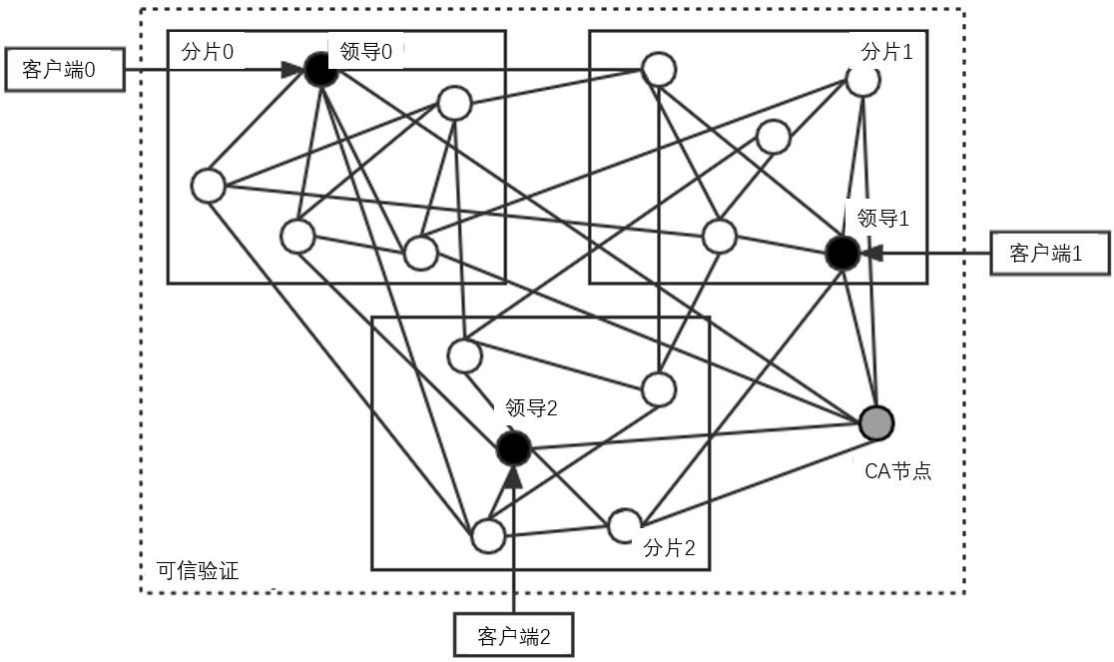

图 4-1　实验设计拓扑图

对于联邦分片区块链模块，首先分别针对节点、交易和区块三个对象构建结构体，并以向量的形式将其分别构成节点组、交易组和区块链，并设置标志位决定是否使用可信验证机制。然后构建初始化节点函数 init()，创建节点，在节点完成 PoW 难题后，随机分配节点进入分片，在每个分片中随机选取 leader 节点，并单独分配一个不进入分片的 CA 节点用于进行密钥管理。然后构建发送交易函数 sendInfo()，创建交易，并依据标志位来决定是否调用可信验证模块。如果没有调用，sender 有 1/6 的可能为故障节点，不发送消息并抛出异常。有 1/6 的可能为恶意节点，会将消息广播到全网节点并抛出异常。因为此时的交易已经被污染，表示不使用可信验证确实有风险，后面的操作就无意义了，系统就自动终止。但也有 2/3 的可能是可信节点，能够正常发送消息。如果调用可信验证模块，当检测到 sender 或 receiver 为故障/恶意节点时，程序就会自行终止。如果验证通过后，就可以发送消息，确认消息发出后就打印交易发送方有效。receiver 消息确认收到消息后就打印交易接收方有效。然后构建广播区块函数 broadcast()，receiver 将交易封装成区块，并依据标志位是否使用了可信验证来决定接收方此时的操作。如果没有使用可信验证，意味着 receiver 此时有 1/6 可能是故障节点，不会广播消息并抛出异常终止程序；有 1/6 可能是恶意节点，会将消息广播到全网节点并抛出异常终止程序；有 2/3 可能是可信节点，能够进行正常广播操作。之后就会把区块广播到 sender 和 receiver 所在分片的所有节点。之后构建共识区块函数 consensus()，首先基于拜占庭容错机制在这一阶段设置 1/3 的节点为不可信节点，其中也包括 sender。因为可信验证的有效期只有一个阶段，过了该阶段

以后节点就有可能变得不可信，但后续操作也无需 sender 可信了，所以此时也不需要对 sender 再次进行可信验证。然后相关分片内的每个节点就会对得到的区块里的信息进行计算共识。receiver 可以对共识结果进行计数，如果超过了参与共识总结点数的 2/3，就表示共识达成，否则打印出共识未达成，交易失败，终止程序。之后创建差分存储函数 difference()，receiver 将完整交易在自身节点存储，并根据分片内的节点数量将完整交易进行平均拆分成多份，并发送给分片内的其他节点分别存储部分交易数据。最后创建区块上链函数 upChain()，receiver 将完整交易数据、时间戳和前一区块 Hash 值相结合，创建当前 Hash 值，并填充当前 Hash 值、前一区块 Hash 值、时间戳以及交易数据，然后将区块上链完成链中的块以完成交易。

对于可信验证机制模块，首先分别构建可信报告节点、可信验证节点、可信基准节点和密钥管理节点的结构体，分别存储不同的功能，并创建函数 init()对节点初始化。然后基于 RSA 加密算法构造私钥和公钥，依次创建函数 BianaryTransform()进行二进制转换，将数据转换为二进制，逆向暂存 temp[]数组中，然后返回二进制的位数。函数 Modular_Exonentiation()进行反复平方求幂操作。函数 ProducePrimeNumber()生成 1000 以内的素数。函数 Exgcd()执行欧几里得扩展算法。函数 RSA_Initialize()用于密钥签发。保存产生的素数，然后随机取两个素数 p，q，将二者相乘得到 n，并用用欧几里德扩展算法求 e，d。将 e 与 n 拼接生成 AIK 公钥，将 d 与 n 拼接生成 AIK 私钥。然后将公私钥都存储到 CA 节点，完成密钥生成。并把私钥发送给 TR 节点，把公钥发送给 TV 节点，完成密钥分发。之后构建函数 trustedStandard()，用于生成 standard_value。用 TS 节点去度量 TR 节点的可信状态，生成 standard_value 序列，该数组用一个随机函数 rand()依次生成值，然后将数组发送给 TV 节点进行存储。之后构建可信度量函数 trustedMeasure()，用于生成 PCR_value 和 PCR_sequence。TR 节点用同样的度量方式产生随机数组作为对于自身的可信度量结果，即 PCR_sequence。然后简化 PCR 写入规则，用简单累加的方式依次累加 PCR_sequence 的值，代替 PCR_sequence 用哈希-拼接的方式生成 PCR_value。之后构建函数 RSA_Encrypt()生成 trusted_report。TR 节点将自身的 ID 和 PCR_value 拼接成数组，并使用私钥对数组进行加密，加密后的密文就是 trusted_report。然后构建函数 trustedReport()发送 trusted_report。将生成的 trusted_report 和可信报告节点自身度量出来的 PCR_sequence 一起发送给 TV 节点，完成 trusted_report 发送。最后构建函数 trustedVerify()对 trusted_report 进行可信验证。第一验证可信报告可信性，通过构建解密函数 RSA_Decrypt()用公钥对可信报告解密，还原出原文，对比其中存储的 TR 节点的 ID 和真实的 ID 是否一致判断可信报告是否可信。此时可以攻击可信报告中的 ID，使得 ID 不正确并抛出异常。第二验证度量结果真实性，将得到

的从 TR 节点发来的 PCR_sequence 按照写入规则累加生成 PCR_value，并与解密后的 trusted_report 中的 PCR_value 比较是否一致判断度量结果是否真实。此时可以攻击可信报告中的 PCR_value，使得 PCR_value 不正确并抛出异常。第三验证度量结果可信性，将从 TR 节点发来的 PCR_sequence 与 TV 节点已经存储的 standard_value 依次比较是否一致判断度量结果是否可信。此时可以攻击 PCR_sequence，使得 PCR_sequence 不正确并抛出异常。

根据以上实验设计，模块间的函数即调用架构如图 4-2 所示。

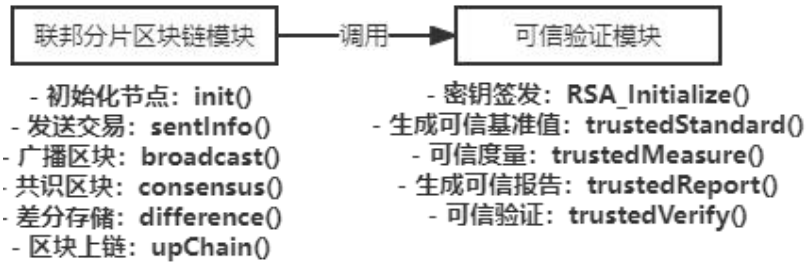

图 4-2　实验设计架构图

4.2　实验环境

由于目前没有现成的联邦分片区块链系统，针对现有开源的成熟平台的源码进行修改工程量较大。同时可信验证模块的部署繁琐，对计算机硬件要求高，且与区块链系统的对接复杂。因此本文仅做系统的原型开发，证实其可行性和有效性。故本实验的环境平台较为简单，具体实验环境的配置信息如表 4-1 所示。

表 4-1　个人笔记本电脑实验环境配置

实验软硬件条件	具体参数
处理器	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
机待 RAM	16.00 GB
操作系统	Windows11(64)
实验软件	Dev-C++
实验平台	Visual Studio Code

4.3 实验内容

4.3.1 联邦分片区块链实现

（1）基本配置

定义节点数 nodeNum、分片数 shardNum、交易数 txNum、当前交易编号 txindex、发送方节点编号 from、接收方节点编号 to、可信验证标志位 flag 的基本信息。同时创建节点、交易、区块的结构体，并将其使用数组进行初始化，构成节点数组 nodelist、交易数组 txlist、当前区块 block 和区块链 bolckchain。节点结构体 Nodes 包括节点编号 nodeid、节点所在分片 shardid、区分主从节点 flag、存储交易信息 info、临时存放当前交易信息 temp 五个属性。交易结构体 Txs 包括交易编号 txid、交易内容 txinfo、交易签名 signature 三个属性。区块结构体 Blocks 包括区块哈希值 hash、前一个区块哈希值 prevhash、上链时间戳 timestamp、交易数据 data 和集体签名 count 五个属性。具体代码如图 4-3 所示。

```
const int nodeNum=15; // 节点数
const int shardNum=3; // 分片数
const int txNum=10000; // 交易数
int txindex = 0; // 当前交易编号
int from; // 发送方节点编号
int to; // 接收方节点编号
int tag=1; // 表示是否使用可信验证，0为不使用，1为使用

struct Nodes // 节点
{
    int nodeid; // 节点编号
    int shardid; // 节点所在分片
    int flag;  // 区分主从节点
    vector<string> info; // 存储交易信息
    string temp; // 临时存放当前交易信息
};

struct Txs // 交易
{
    int txid; // 交易编号
    string txinfo; // 交易内容
    string signature; // 交易签名
};

struct Block // 区块
{
    string hash; // 区块哈希值
    string prevhash; // 前一个区块哈希值
    string timestamp; // 上链时间戳
    Txs data; //交易数据
    int count=0; // 集体签名
};

Nodes nodelist[nodeNum]; // 节点数组
Txs txlist[txNum]; // 交易数组
Block block; // 当前区块
vector<Block> bolckchain; // 区块链
```

图 4-3　基本配置代码

（2）初始化节点

构建函数 init()进行节点网络初始化。将配置的所有节点确定好 nodeid 后依次完成 PoW 参与进网络，并随机分配到不同分片中，记录 shardid。然后设置最先加入分片的节点 flag 为 0，表示为 leader 节点，其余为 1。最后单独设置 CA 节点，不加入任何分片，用来进行密钥信息管理。具体代码如图 4-4 所示。

```
/*
    1.初始化节点
    完成PoW后注册身份信息（节点ID），并通过节点分配算法将节点分配到不同分片中
*/
void init(){
    cout << "step1: 初始化节点" << endl;
    for(int i=0; i<nodeNum; i++){
        nodelist[i].nodeid = i;
        cout << "节点 " << i << " 完成PoW，成功参与" << endl;
        nodelist[i].shardid = i%shardNum;
        cout << "节点 " << i << " 加入分片 " << nodelist[i].shardid << " 中" << endl;
        if(i<shardNum){
            nodelist[i].flag=0;
        } else {
            nodelist[i].flag=1;
        }

    }
    cout << "节点 " << nodeNum << " 完成PoW，成功参与" << endl;
    cout << "节点 " << nodeNum << " 为CA节点，不加入任何分片" << endl;
    cout << "----------------------------------------" << endl;
    sentInfo();
}
```

图 4-4　初始化节点代码

（3）发送交易

构建函数 sentInfo()创建、验证并发送交易。随机选择 from 和 to，并并针对 txindex 创建交易，完成 txid、signature 和 txinfo 的配置。然后判断 flag，若为 0 表示未使用可信验证，则 from 有 1/6 概率为故障节点，不发送交易，并抛出异常终止程序；有 1/6 概率为恶意节点，会将交易广播到全网络节点，并抛出异常终止程序；有 2/3 概率为可信节点，正常发送交易。若 flag 为 1，表示使用可信验证机制，则将对 from 和 to 的分别验证交给可信验证模块，通过验证就打印验证通过的消息。之后对交易进行发送，如果 from 确认交易已发送就打印相关信息证明发送方有效性，如果 to 确认收到交易就打印相关信息证明接收方有效性。具体代码如图 4-5 所示。

```
/*
    2.发送交易
    发送方节点生成交易，验证该交易的有效性（发送方及接收方的可信验证），并发送到接收
*/
void sentInfo(){
    cout << "step2: 发送交易" << endl;
    from = rand() % 3;
    to = (from+1) % 3;
    string info = "hello world! it is a test info.";
    txlist[txindex].txid=txindex;
    txlist[txindex].signature="node"+to_string(from);
    txlist[txindex].txinfo=info;

    if(tag == 0){ // ！！！没有可信验证的突发情况：1.发送方是故障或恶意节点
        int f1 = rand() % 6; // 节点网络有1/3的坏节点，包括故障和恶意
        if(f1 == 1){ // 1.1 如果是故障节点
            cout << "节点故障，不发送任何消息" << endl;
            cout << "----------------------------------------" << endl;
            return;
        }else if(f1 == 2){ // 1.2 如果是恶意节点
            cout << "分片 " << from <<" 中的节点 " << from << " 发送消息" << endl;
            cout << "发送方满足交易条件" << endl;
            // 将消息广播给全网络节点
            for(int i = 0; i < nodeNum; i++){
                if(i != from)
                    cout << "分片 " << nodelist[i].shardid <<" 中的节点 " << nodelis
            }
            cout << "节点恶意，交易被披露" << endl;
            cout << "----------------------------------------" << endl;
            return;
        }
    }else if(tag == 1){ // ！！！使用了可信验证机制
        cout << "发送方节点通过可信验证" << endl;
        cout << "接收方节点通过可信验证" << endl;
    }

    cout << "分片 " << from <<" 中的节点 " << from << " 发送消息" << endl;
    cout << "发送方满足交易条件" << endl;
    cout << "分片 " << to <<" 中的节点 " << to << " 接收消息" << endl;
    cout << "接收方满足交易条件" << endl;
    cout << "发送消息为: " << info << endl;
    nodelist[to].temp = info;
    cout << "----------------------------------------" << endl;
    broadcast();
}
```

图 4-5　发送交易代码

（4）广播区块

构建函数 broadcast()对交易进行封装和广播。to 收到交易后会将交易打包成区块 block。如果 flag 为 0，则 to 有 1/6 之一的概率是故障节点，不会广播区块，抛出异常并终止程序；有 1/6 的概率是恶意节点，会将区块广播到全网节点，抛出异常并终止程序；有 2/3 的概率是可信节点，能够正常向 from 和 to 所处的区

块的所有节点广播区块。如果 flag 为 1，能进行到此步说明 to 已经通过可信验证，则将消息广播到除了 to 本身以外，from 和 to 所处的区块的所有节点。节点收到区块就打印收到区块的日志。具体代码如图 4-6 所示。

```
/*
    3.广播区块
    接收方主节点收到消息后将交易封装成区块，并在发送方和接收方的所有节点广播该区块。
*/
void broadcast(){
    cout << "step3: 广播区块" << endl;
    block.data = txlist[txindex];
    cout << "交易 " << txlist[txindex].txid << " 封装成区块" << endl;

    if(tag == 0){ // ！！！没有可信验证的突发情况：2.接收方是故障或恶意节点
        int f2 = rand() % 6; // 节点网络有1/3的坏节点，包括故障和恶意
        if(f2 == 1){ // 2.1 如果是故障节点
            cout << "节点故障，不广播任何消息" << endl;
            cout << "----------------------------------------" << endl;
            return; // 不广播任何消息
        } else if(f2 == 2){ // 2.2 如果是恶意节点
            // 将消息广播给全网络节点
            for(int i = 0; i < nodeNum; i++){
                if(i != to){
                    nodelist[i].temp = nodelist[to].temp;
                    cout << "分片 " << nodelist[i].shardid <<" 中的节点 " << nodeli
                }
            }
            cout << "节点恶意，交易被披露" << endl;
            cout << "----------------------------------------" << endl;
            return;
        }
    }

    // ！！！使用了可信验证机制
    // 只广播给发送方和接收方相关分片的节点
    for(int i = 0; i < nodeNum; i++){
        if((nodelist[i].shardid == from || nodelist[i].shardid == to) && i != to){
            nodelist[i].temp = nodelist[to].temp;
            cout << "分片 " << nodelist[i].shardid <<" 中的节点 " << nodelist[i].no
        }
    }
    cout << "----------------------------------------" << endl;
    consensus();
}
```

图 4-6　广播区块代码

（5）共识区块

构建函数 consensus()对区块中的数据进行共识。设置计数器 join 对参与共识节点进行统计。因为处于拜占庭容错网络中，所以会有 1/3 的节点不可信，所以在进行共识之前通过 rand()随机设置不可信节点，修改其共识结果。然后将每个

节点的共识结果与 to 中的进行比较，如果一致则达成共识，并将 block 中的 count 加一。如果最后的 count 数超过 join 数的 2/3，表明该区块达成共识，并打印相关结果。具体代码如图 4-7 所示。

```
/*
    4.共识区块
    区块发送到的所有节点对区块进行计算并打印结果发送给接收方节点，超过2/3
*/
void consensus(){
    cout << "step4: 共识区块" << endl;
    int join = 0; // 统计参与共识的节点数
    for(int i = 0; i < nodeNum; i++){
        if(nodelist[i].shardid == from || nodelist[i].shardid == to){
            int tik = rand()%3; //拜占庭容错机制
            if(tik == 0){
                nodelist[i].temp = "";
            }
            if(nodelist[to].temp.compare(nodelist[i].temp) == 0){
                cout <<  "分片 " << nodelist[i].shardid <<" 中的节点 " <<
                block.count++;
            }else{
                cout <<  "分片 " << nodelist[i].shardid <<" 中的节点 " <<
            }
            join++;
        }
    }
    cout << "参与共识节点数为：" << join << endl;
    cout << "达成共识节点数为：" << block.count << endl;
    int threshold = join - (join/3);
    if(block.count>=threshold){
        cout << "*****交易区块达成共识*****" << endl;
        cout << "--------------------------------------" << endl;
        difference();
        upChain();
    }else{
        cout << "*****交易区块未达成共识*****" << endl;
        cout << "--------------------------------------" << endl;
    }
    txindex++;
}
```

图 4-7　共识区块代码

（6）差分存储

构建函数 difference()对交易进行差分存储。to 在达成共识后将完整交易存进自身节点 info 中，然后统计所处分片中 staff 节点个数将交易字符串进行平均切分成 staff 节点数量的份数。然后将切分好的部分交易分别发送给 staff 节点进行

存储。最后每个节点打印出自己存储的交易数据。具体代码如图 4-8 所示。

```
/*
    5.差分存储
    接收方主节点将消息切分，随机分给接收方所在分片的其他子节点存储
*/
void difference(){
    cout << "step5: 差分存储" << endl;
    nodelist[to].info.push_back(nodelist[to].temp);
    cout << "分片 " << nodelist[to].shardid << " 中节点 " << nodelist[to].nodei
    int shardnode=0; // 统计分片中节点个数
    for(int i=shardNum;i<nodeNum;i++){
        if(nodelist[i].shardid==nodelist[to].shardid && nodelist[i].flag==1){
            shardnode++;
        }
    }
    int sp=nodelist[to].temp.size() / shardnode + 1; // 切分后每段消息的长度
    string sub=nodelist[to].temp; // 临时存储消息内容
    for(int i=shardNum;i<nodeNum;i++){
        if(nodelist[i].shardid==nodelist[to].shardid && nodelist[i].flag==1){
            nodelist[i].info.push_back(sub.substr(0,sp));
            cout << "分片 " << nodelist[i].shardid << " 中节点 " << nodelist[i]
            shardnode--;
            if(shardnode!=0){
                sub = sub.substr(sp);
            }
        }
    }
    cout << "---------------------------------------" << endl;
}
```

图 4-8　差分存储代码

（7）区块上链

构建函数 upChain()将区块上链。完成差分存储后 to 就获取前一个区块的哈希值 prevhash、当前时间戳 timestamp，并用”hash”+”id”的方式模拟区块哈希构建方式以生成该区块的哈希值 hash，将这三个数据填充到区块中。最后将区块中的所有消息打印出来，并将区块放进 blockchain 中，完成上链。具体代码如图 4-9 所示。

```
/*
    6.区块上链
    交易存储完毕后，主节点就将交易上链
*/
void upChain(){
    cout << "step6: 区块上链" << endl;
    block.hash = "hash"+to_string(txindex);
    if(!bolckchain.empty()){
        block.prevhash = bolckchain.back().hash;
    }else{
        block.prevhash = "hash-1";
    }
    time_t nowtime;
    struct tm* p; // 当前时间
    time(&nowtime);
    p = localtime(&nowtime);
    block.timestamp=to_string(p->tm_hour) + ":" + to_string(p->tm_min)
    bolckchain.push_back(block);
    cout << "当前区块哈希值: " << block.hash << endl;
    cout << "上一区块哈希值: " << block.prevhash << endl;
    cout << "当前区块时间戳: " << block.timestamp << endl;
    cout << "区块中交易编号: " << block.data.txid << endl;
    cout << "区块中交易数据: " << block.data.txinfo << endl;
    cout << "区块中交易签名: " << block.data.signature << endl;
    cout << "区块成功上链！" << endl;
    cout << "--------------------------------------" << endl;
}
```

图 4-9　区块上链代码

4.3.2　可信验证实现

（1）基本配置

分别创建可信报告、可信验证、可信基准和密钥管理结构体，并通过 init() 函数为每个结构体初始化一个节点，将可信报告节点设为 node0，可信验证节点设为 node1，可信基准节点设为 node2，密钥管理节点设为 node3。可信报告结构体 Report 包括节点编号 id、PCR 写入序列 PCR_sequence、PCR 值 PCR_value 和私钥 private_key 四个属性。可信验证结构体 Verify 包括节点编号 id、可信基准库 recordlib、公钥 public_key、可信报告 trusted_report 和 PCR 写入序列 PCR_sequence 五个属性。可信基准结构体 Standard 包括节点编号 id 和可信基准值 stantard_value 两个属性。密钥管理结构体包括节点编号 id、私钥 private_key 和公钥 public_key 三个属性。具体代码如图 4-10 所示。

```
struct Report // 可信报告
{
    int id;
    vector<int>  PCR_sequence; // PCR写入序列
    int PCR_value; // PCR值
    int private_key[2]; // 私钥
};

struct Verify // 可信验证
{
    int id;
    vector<int> recordlib; // 从可信基准节点获得的可信基准值
    int public_key[2]; // 公钥
    long long trusted_report[2]; // 可信报告
    vector<int>  PCR_sequence; // PCR写入序列
};

struct Standard // 可信基准
{
    int id;
    vector<int> standard_value; // 从可信报告节点测量的可信基准值
};

struct CA // 密钥管理
{
    int id;
    int private_key[2]; // 私钥
    int public_key[2]; // 公钥
};

// 初始化节点
Report node0;
Verify node1;
Standard node2;
CA node3;

/*
    0.初始化
*/
void init(){
    node0.id=0;
    node1.id=1;
    node2.id=2;
    node3.id=3;
    cout << "可信验证流程开始:" << endl;
    cout << "--------------------------------------" << endl;
}
```

图 4-10　基本配置代码

（2）签发密钥

首先定义明文 Plaintext、密文 Ciphertext 和密钥所需的 n、e、d。然后定义

二进制转换函数 BianaryTransform()将输入的数据不断转换模 2 除 2 转换成二进制形式，并统计二进制位数。定义反复平方求幂函数 Modular_Exonentiation()调用 BianaryTransform()计算 e 的倒数对(p-1)(q-1)的取模结果求得密钥中的 d。定义大素数生成函数 ProducePrimeNumber()逐个数比较并相乘翻倍生成 1000 以内所有素数。定义欧几里得扩展算法 Exgcd()求 m、n 最大公约数同时找到整数 x、y 使得它们满足贝祖等式。定义 RSA 初始化函数 RSA_Initialize()，调用 ProducePrimeNumber()取出 1000 内素数保存在 prime[]数组中，并从中随机宣传两个素数作为 p、q，将 p、q 相乘得到 n，最后调用 Exgcd()求得 e 和 d，就生成了密钥。将(e，n)封装成 public_key，将(d，n)封装成 private_key 存入 node3 中，并打印结果。然后将 private_key 发送给 node0，将 public_key 发送给 node1，分别存储并打印。具体代码如图 4-11 所示。

```
//RSA初始化
void RSA_Initialize()
{
    //取出1000内素数保存在prime[]数组中
    int prime[5000];
    int count_Prime = ProducePrimeNumber(prime);

    //随机取两个素数p,q
    srand((unsigned)time(NULL));
    int ranNum1 = rand()%count_Prime;
    int ranNum2 = rand()%count_Prime;
    int p = prime[ranNum1], q = prime[ranNum2];

    n = p*q;

    int On = (p-1)*(q-1);


    //用欧几里德扩展算法求e,d
    for(int j = 3; j < On; j+=1331)
    {
        int gcd = Exgcd(j, On, d);
        if( gcd == 1 && d > 0)
        {
            e = j;
            break;
        }

    }
    cout << "step1: 签发密钥" << endl;
    cout << "生成公钥 (e, n) : e = " << e << " n = " << n << endl;
    cout << "生成私钥 (d, n) : d = " << d << " n = " << n << endl;
    node3.private_key[0] = e;
    node3.private_key[1] = n;
    node3.public_key[0] = d;
    node3.public_key[1] = n;
    node0.private_key[0] = node3.private_key[0];
    node0.private_key[1] = node3.private_key[1];
    node1.public_key[0] = node3.public_key[0];
    node1.public_key[1] = node3.public_key[1];
    cout << "向node0分发私钥: [" << node0.private_key[0] << " , " <<
    cout << "向node2分发公钥: [" << node1.public_key[0] << " , " <<
    cout << "---------------------------------------" << endl;
}
```

图 4-11　签发密钥代码

（3）生成可信基准值

构建函数 trustedStantard()用于生成可信基准值。使用 rand()函数模拟生成一个 PCR 写入序列的数组，并分别保存到 node2 的 stantard_value 和 node0 的 PCR_sequence，然后打印出可信基准值 stantard_value，并将 stantard_value 发送

给 node1 的 recordlib 进行存储并打印结果。具体代码如图 4-12 所示。

```
/*
    2.生成可信基准值，node2计算node0可信基准值，将可信基准值（写入序列）
*/
void trustedStandard(){
    for(int i=0;i<5;i++){
        int temp=rand() % 100;
        node0.PCR_sequence.push_back(temp);
        node2.standard_value.push_back(temp);
    }
    cout << "step2: 生成可信基准值" << endl;
    cout << "可信基准节点node2获得可信基准值: ";
    for (vector<int>::iterator iter = node2.standard_value.begin(); iter
    {
        cout << *iter << " ";
    }
    cout << endl;
    node1.recordlib=node2.standard_value;
    cout << "可信验证节点node1存储可信基准值: ";
    for (vector<int>::iterator iter = node1.recordlib.begin(); iter != n
    {
        cout << *iter << " ";
    }
    cout << endl;
    cout << "----------------------------------------" << endl;
}
```

图 4-12　生成可信基准值代码

（4）可信度量

创建函数 trustedMeasure()进行对 node0 的可信度量。因为 rand()函数的随机性，在生成可信基准值的同时也对 node0 自身进行了可信度量，以保证随机数组的一致性。然后将度量后的 PCR_sequence 打印出来，并按照将每个度量结果累加的方法模拟 PCR 写入规则去生成一个 PCR_value，并将 PCR_value 打印出来。具体代码如图 4-13 所示。

```
/*
    3.可信度量，node0度量可信报告节点的安全机制和载入策略，对PCR写入序列进行处理，
*/
void trustedMeasure(){
    node0.PCR_value=0;
    for (vector<int>::iterator iter = node0.PCR_sequence.begin(); iter != nod
    {
        node0.PCR_value+=*iter;
    }
    cout << "step3: 可信度量" << endl;
    cout << "可信报告节点node0获得PCR写入序列: ";
    for (vector<int>::iterator iter = node0.PCR_sequence.begin(); iter != nod
    {
        cout << *iter << " ";
    }
    cout << endl;
    cout << "可信报告节点node0计算PCR值: " << node0.PCR_value << endl;
    cout << "--------------------------------------" << endl;
}
```

图 4-13　可信度量代码

（5）生成可信报告

创建函数 RSA_Encrypt()进行可信报告加密。将 node0 的 id 和 PCR_value 拼接成明文数组 Plaintext，并调用 Modular_Exonentiation()使用 private_key 对 Plaintext 加密生成可信报告 Ciphertext 并打印出来。构建函数 trustedReport()调用函数 RSA_Encrypt()，并将 Ciphertext 和 PCR_sequence 发送给 node1 进行存储，并打印出相关数据。具体代码如图 4-14 所示。

```
/*
    4.生成可信报告：node0将PCR值用AIK私钥加密，即为可信报告。node0将可信报告、公钥证书(
*/

//RSA加密
void RSA_Encrypt()
{
    cout << "step4: 生成可信报告" << endl;
    Plaintext[0] = node0.id;
    Plaintext[1] = node0.PCR_value;
    for(int i = 0; i < 2; i++)
        Ciphertext[i] = Modular_Exonentiation(Plaintext[i], d, n);
    cout << "使用私钥 (d, n) 加密，可信报告 (密文) 为: [" << Ciphertext[0] << ", " <
}

void trustedReport(){
    RSA_Encrypt();
    for(int i = 0; i < 2; i++)
        node1.trusted_report[i]=Ciphertext[i];
    cout << "可信验证节点node1存储可信报告: [" << node1.trusted_report[0] << ", " <<
    node1.PCR_sequence=node0.PCR_sequence;
    cout << "可信验证节点node1存储PCR序列: ";
    for (vector<int>::iterator iter = node1.PCR_sequence.begin(); iter != node1.PCI
    {
        cout << *iter << " ";
    }
    cout << endl;
    cout << "----------------------------------------" << endl;
}
```

图 4-14　生成可信报告代码

（6）可信验证

构建函数 RSA_Decrypt ()进行可信报告解密。将收到的 trusted_report 使用 public_key 解密还原出 Plaintext 数组。构建函数 trustedVerify()进行可信验证。首先调用 RSA_Decrypt ()并取出数组中的 id 与 node0 的真实 id 进行比较，如果一致则可信报告可信。此时可以设置攻击修改 Plaintext 中的 id 值，使可信报告不可信并抛出异常。其次将 PCR_sequence 按照累加的写入规则生成新的 PCR 值，并与 trusted_report 中的 PCR_value 比较，如果一致则度量结果真实。此时可以设置攻击修改 PCR_value，使度量结果不真实并抛出异常。最后将 PCR_sequence 和 recordlib 中的每个度量结果一次比较，如果一致则度量结果可信。此时可以设置修改 PCR_sequence，使度量结果不可信并抛出异常。具体代码如图 4-15 所示。

```
/*
    5.可信验证:
    (1)通过公钥证书还原出被验证方ID证明公钥可信，若可信，则使用公钥将可信报告解
    (2)将PCR写入序列按照写入规则生成一个值，与解密后的PCR值比较，若相同，print
    (3)比较基准库的值和PCR写入序列，若相同，print（度量结果可信）
*/

//RSA解密
void RSA_Decrypt(){
    for(int i = 0; i < 100; i++)
        Ciphertext[i] = Modular_Exonentiation(Ciphertext[i], e, n);
}

void trustedVerify(){
    cout << "step5: 可信验证" << endl;

    cout << "(1) 验证可信报告是否可信" << endl;
    RSA_Decrypt();
    // 攻击1
    // Ciphertext[0] = rand() % 100 + 1;
    cout << "使用公钥 (e, n) 解密，PCR值 (明文) 为: [" << Ciphertext[0] <<
    cout << "node0真实ID: " << node0.id << endl;
    cout << "公钥还原node0的ID: " << Ciphertext[0] << endl;
    if(node0.id == Ciphertext[0]){
        cout << "=====可信报告可信=====" << endl << endl;
    }else{
        cout << "=====可信报告不可信=====" << endl;
        return;
    }

    cout << "(2) 验证度量结果是否真实" << endl;
    cout << "可信报告所存储PCR值: " << Ciphertext[1] << endl;
    cout << "利用所得PCR写入序列生成PCR值: ";
    int tempPCR=0;
    for (vector<int>::iterator iter = node1.PCR_sequence.begin(); iter
    {
        tempPCR+=*iter;
    }
    // 攻击2
    // tempPCR*=2;
    cout << tempPCR << endl;
    if(Ciphertext[1] == tempPCR){
        cout << "=====度量结果真实=====" << endl << endl;
    }else{
        cout << "=====度量结果不真实=====" << endl;
        return;
    }
```

```
    // 攻击2
    // tempPCR*=2;
    cout << tempPCR << endl;
    if(Ciphertext[1] == tempPCR){
        cout << "=====度量结果真实=====" << endl << endl;
    }else{
        cout << "=====度量结果不真实=====" << endl;
        return;
    }

    cout << "(3) 验证度量结果是否可信" << endl;
    cout << "可信基准库值: ";
    for (vector<int>::iterator iter = node1.recordlib.begin(); iter != 
    {
        cout << *iter << " ";
    }
    cout << endl;
    cout << "所得PCR写入序列值: ";
    // 攻击3
    // int attack = rand() % 5;
    // node1.PCR_sequence[attack]*=2;
    for (vector<int>::iterator iter = node1.PCR_sequence.begin(); iter
    {
        cout << *iter << " ";
    }
    cout << endl;
    int count=0;
    for(int i=0;i<5;i++){
        if(node1.recordlib[i]==node1.PCR_sequence[i]){
            count++;
        }
    }
    if(count==5){
        cout << "=====度量结果可信=====" << endl;
    }else{
        cout << "=====度量结果不可信=====" << endl;
        return;
    }
}
```

图 4-15　可信验证代码

4.4　结果分析

4.4.1　未使用可信验证

联邦分片区块链系统主要目的是实现数据的差分隐私保护，保证其能基于联邦思想分布式地存储到分片中的大量节点中。但在拜占庭容错网络中，对于没有使用可信验证的联邦分片区块链系统，发送方节点和接收方节点均会有 1/3 的概

率为不可信节点。因此系统中的交易在传输过程中仅有 4/9（2/3 * 2/3）的概率能正常传输，其余情况均会由于节点不可信而产生隐私数据暴露的问题。因此具体运行结果分为以下几类：

（1）正常运行

如图 4-16 所示，在正常情况下，程序依次会执行初始化节点、发送交易、广播区块、共识区块、差分存储、区块上链的六步流程。

```
step1: 初始化节点
节点 0 完成PoW, 成功参与
节点 0 加入分片 0 中
节点 1 完成PoW, 成功参与
节点 1 加入分片 1 中
节点 2 完成PoW, 成功参与
节点 2 加入分片 2 中
节点 3 完成PoW, 成功参与
节点 3 加入分片 0 中
节点 4 完成PoW, 成功参与
节点 4 加入分片 1 中
节点 5 完成PoW, 成功参与
节点 5 加入分片 2 中
节点 6 完成PoW, 成功参与
节点 6 加入分片 0 中
节点 7 完成PoW, 成功参与
节点 7 加入分片 1 中
节点 8 完成PoW, 成功参与
节点 8 加入分片 2 中
节点 9 完成PoW, 成功参与
节点 9 加入分片 0 中
节点 10 完成PoW, 成功参与
节点 10 加入分片 1 中
节点 11 完成PoW, 成功参与
节点 11 加入分片 2 中
节点 12 完成PoW, 成功参与
节点 12 加入分片 0 中
节点 13 完成PoW, 成功参与
节点 13 加入分片 1 中
节点 14 完成PoW, 成功参与
节点 14 加入分片 2 中
节点 15 完成PoW, 成功参与
节点 15 为CA节点, 不加入任何分片
----------------------------------------
step2: 发送交易
发送方节点通过可信验证
接收方节点通过可信验证
分片 2 中的节点 2 发送消息
发送方满足交易条件
分片 0 中的节点 0 接收消息
接收方满足交易条件
发送消息为: hello world! it is a test info.
----------------------------------------
```

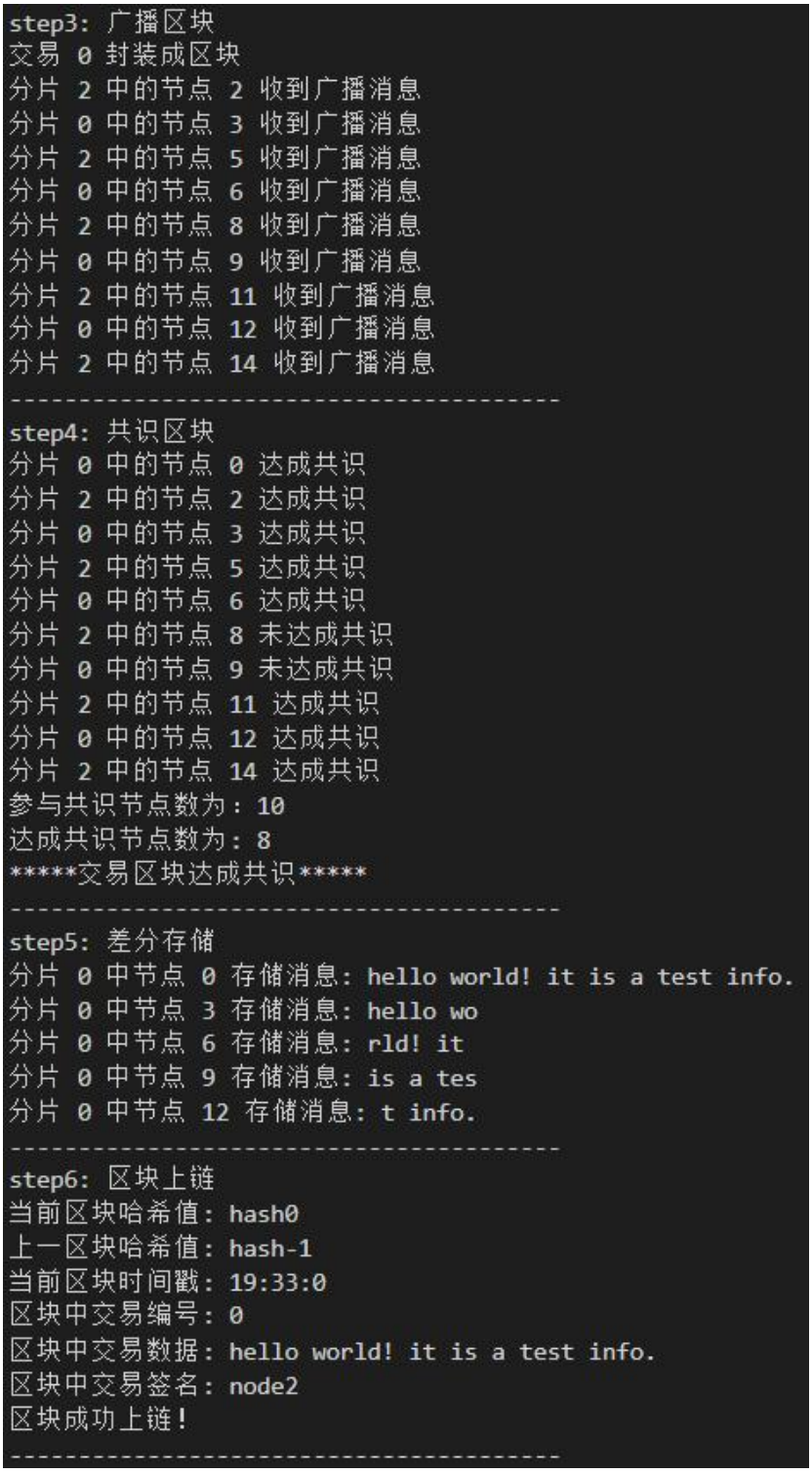

图 4-16　正常情况运行结果图

（2）发送方故障

如图 4-17 所示，当发送方节点故障时，在发送交易阶段就不会发送交易给接收方，而程序也会抛出异常，就此关闭。

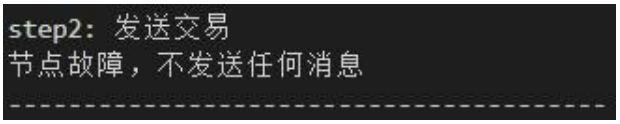

图 4-17　发送方故障结果图

（3）发送方恶意

如图 4-18 所示，在发送方节点恶意时，在发送交易阶段会将交易广播给网络中所有节点，而不仅仅发给接收方，使得隐私数据暴露，而程序也会抛出异常，就此关闭。

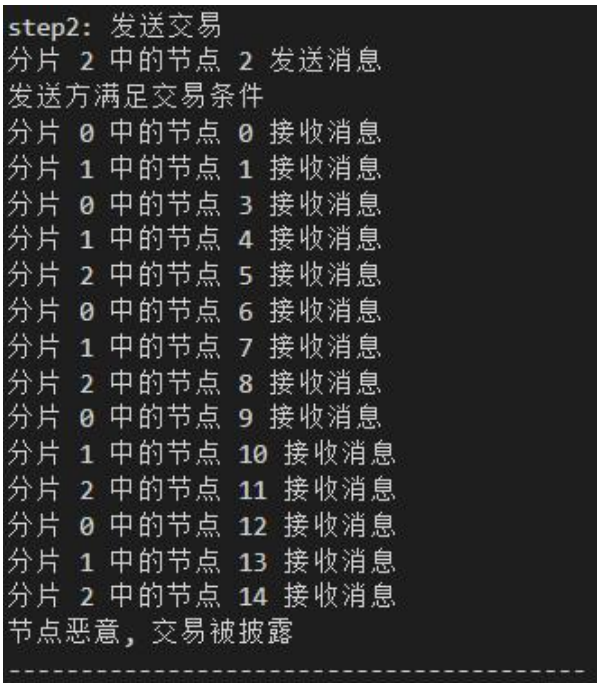

图 4-18　发送方恶意结果图

（4）接收方故障

如图 4-19 所示，当接收方节点故障时，在广播区块阶段就不会广播区块给相关节点，而程序也会抛出异常，就此关闭。

图 4-19　接收方故障结果图

（5）接收方恶意

如图 4-20 所示，当接收方节点恶意时，在广播区块阶段会将区块广播给无关分片中的节点，使得隐私数据被披露，而程序也会抛出异常，就此关闭。

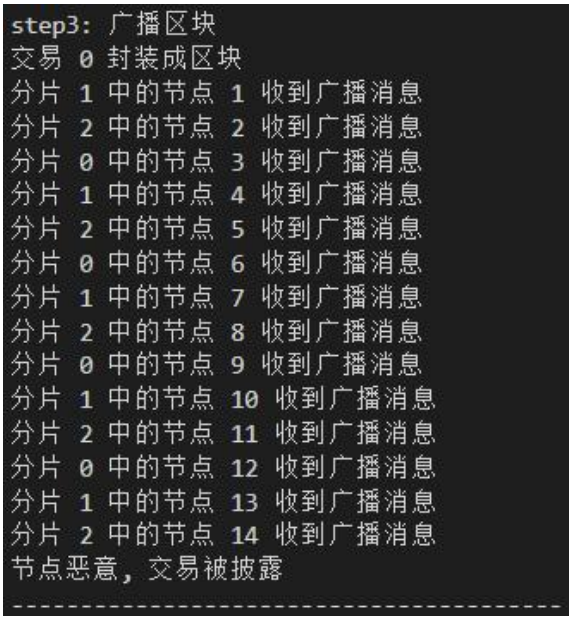

图 4-20　接收方恶意结果图

4.4.2　使用可信验证

在使用了可信验证机制以后，所有针对发送方和接收方的故障或恶意情况均可在可信验证流程中被验证，如果节点不可信，在验证中就会被抛出异常，终止程序。因此规避了在不知道节点可信状况下让节点发送或接收消息的情况，使得隐私数据在节点间传输时得到了有效的保护。在可信验证中，除了节点可信以外，还会因为针对可信报告 id、可信报告 PCR 值、PCR 写入序列的攻击行为。而一个可信节点变成故障或恶意节点往往也能通过这些度量值的变化体现出来。因此具体运行结果分为以下几类：

（1）节点可信

如图 4-21 所示，在正常情况下，程序依次会执行签发密钥、生成可信基准值、可信度量、生成可信报告、可信验证的五步流程。

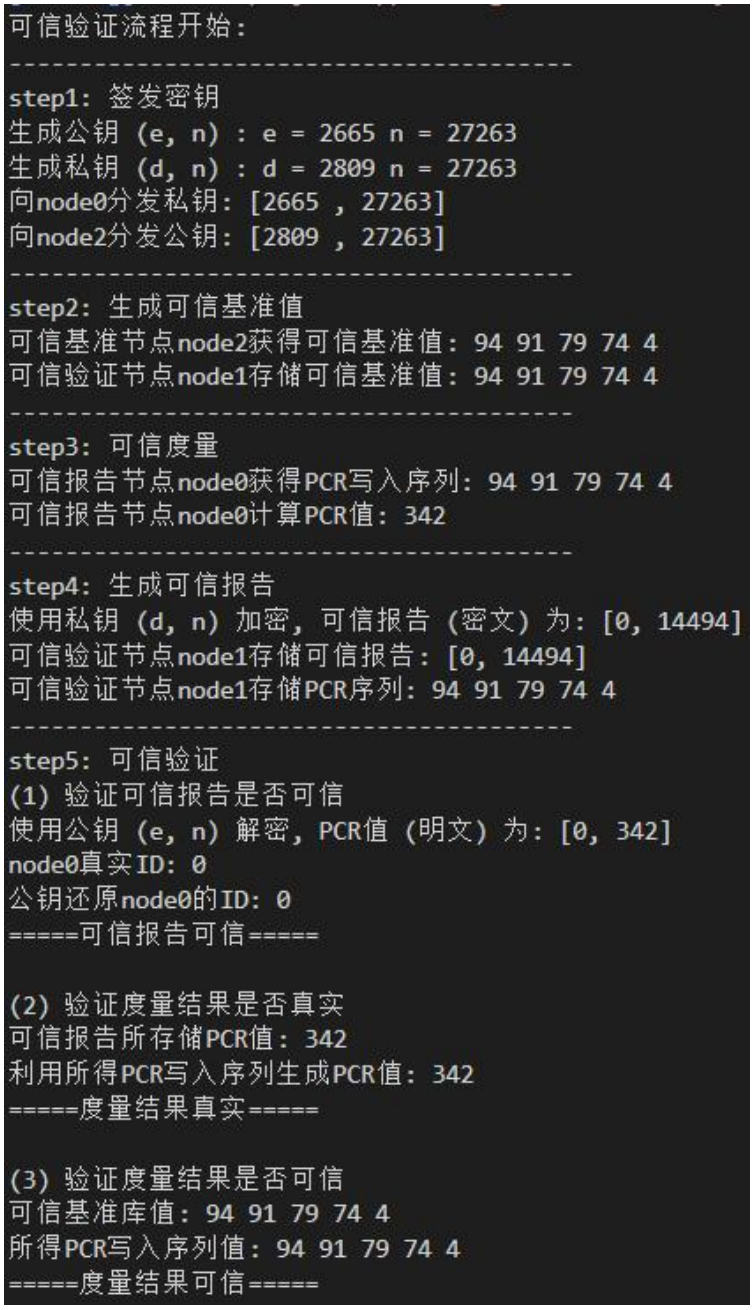

图 4-21　节点可信结果图

（2）可信报告不可信

如图 4-22 所示，当可信验证节点拿公钥解密了可信报告后发现其中存储的可信报告节点 id 与可信报告实际 id 不符，则表示可信报告不可信，而程序也会抛出异常，就此关闭。

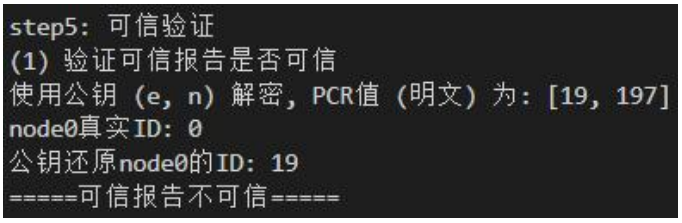

图 4-22　可信报告不可信结果图

（3）度量结果不真实

如图 4-23 所示，当可信验证节点将 PCR 写入序列按照写入规则生成 PCR 值后发现与可信报告中的 PCR 值不符，则表示度量结果不真实，而程序也会抛出异常，就此关闭。

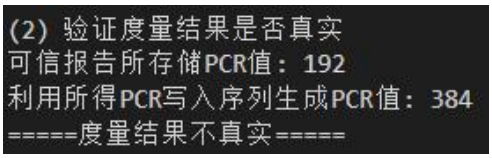

图 4-23　度量结果不真实结果图

（4）度量结果不可信

如图 4-24 所示，当可信验证节点将 PCR 写入序列与可信基准库已经存储的可信基准值依次比较，不符时则表示度量结果不可信，而程序也会抛出异常，就此关闭。

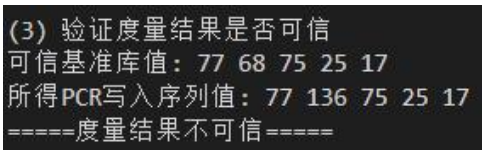

图 4-24　度量结果不可信结果图

4.4.3　综合分析

根据上述实验结果，可以发现联邦分片区块链系统确实能够有效对交易的隐私数据进行差分存储，保证当某一节点被攻破时仅仅会泄露一部分用户数据，完整数据不会泄露。同时每轮重新分配节点进分片和重新选举 leader 的机制也有效防御了 PoW 的 51%攻击，频繁交易的用户也不容易攻击者被检测到。

但在拜占庭容错的网络中，节点的故障或恶意是难以避免的。因此一般的联邦分片区块链系统在发送阶段发送方节点难以避免会有一定概率导致交易不发

送或者全网发送导致隐私泄露；也可能在广播区块阶段时接收方节点存在一定概率导致区块不广播或者全网广播导致隐私泄露。

但在使用了可信验证以后节点的故障或恶意问题会在验证阶段由于可信报告 id 值、可信报告 PCR 值或 PCR 写入序列与可信状态下的不一致而被判定为节点已经不可信，即节点变成故障或恶意节点。此时可信验证机制就会直接将异常抛出，终止交易继续发送，防止交易中的隐私数据被泄露。

4.5 本章小结

本章使用 C++程序语言对基于可信验证的联邦分片区块链进行了简化模拟实验，完成了其原型系统的开发。实现了基于可信验证的联邦分片区块链的安全性、稳定性与完备性实验，并对实验结果进行了具体分析。为了得出相关结论，本章首先对联邦分片区块链进行开发，证明了其对于数据差分隐私保护的有效性。之后使用了控制变量法设置了是否使用可信验证的对比实验，并在每组实验中可能会出现的错误情况进行了详尽分析与处理，使得系统更加健壮完备。同时对每一种情况的结果分别分析，得出了可信验证机制确实能够保证联邦分片区块链在交易传输时隐私数据安全不被泄露的结论，证明了方案的可行性。

第五章　总结与展望

本文通过对联邦分片区块链以及可信验证机制的研究，设计了基于可信验证的联邦分片区块链模型，并设计了相关的实验对系统进行原型开发，证明了可信验证机制的引入使得联邦分片区块链的安全性得到了有效的提升，确认了基于可信验证的区块链联邦隐私保护方案的可行性。最后通过总结和展望，对本文的工作进行了回顾和安排，并提出了改进模型的建议。

5.1　工作总结

本文通过对当前复杂网络环境的了解以及对隐私安全重要性的认知，分析出了人们在不可信的网络下进行数据传输和存储时对其隐私保护的需求。为了满足这一需求，本文通过对分片区块链、差分隐私和可信验证的深入学习与研究，设计出了基于可信验证的联邦分片区块链模型，实现对数据的差分隐私保护。并设计了相关的实验进行验证，确认联邦分片区块链确实能够实现差分隐私存储，同时也证明可信验证机制确实能有效保护数据在传输时不被泄露。从而得出结论。

本文第一章从研究背景方面论述了基于可信验证机制的联邦分片区块链的现实意义和必要性。并分别对联邦分片区块链和可信验证机制的国内外研究现状进行了总结，使我对二者有了更加深刻的认识与理解，为后期模型的设计与开发提供了方向指引。本文第二章分别对联邦分片区块链和可信验证进行了深入的理论知识学习研究，为模型的设计与实现奠定了坚实的理论基础。联邦分片区块链方面，首先对分片区块链的基本概念进行了研究，包括模型结构、应用场景以及功能和范围上的分类等。然后详细介绍了分片区块链的具体运行逻辑以及目前面临的技术挑战。最后基于联邦思想提出了联邦分片区块链，并简要介绍了其运行流程。可信验证方面，首先对可信验证的基本概念和具体要求进行了探究。然后详细介绍了可信验证的总体流程。最后介绍了具有可信验证功能的可信计算目前面临的技术挑战。本文第三章选择在拜占庭容错网络中设计基于可信验证的联邦分片区块链模型。首先构建出模型的总体流程和总体架构模块。并分别对联邦分片区块链和可信验证机制的细节上进行了详细的设计分析，最终构建完整的模型。最后从结构和功能上对模型设计进行了分析与评价。本文第四章针对模型的设计进行了相关实验开发了原型系统加以测试，并通过分析实验结果得出结论。首先基于 C++编程语言的功能特性和运行效能对系统设计进行了简化，便于实现。然后以可行性和完备性为原则明确了联邦分片区块链的具体实现方式，以及针对不

可信节点使得隐私暴露时的异常处理。之后针对可信验证机制设计了其具体实现方法以及异常处理机制，并根据控制控制变量方法设计了是否使用可信验证的不同结果。最后使用 C++对上述的实验设计进行复现，分析结果并将其与预测结果进行比较以得出结论。本文第五章对全文进行了梳理，并提出系统今后进一步的优化方案。

通过本次论文的撰写，我自身也收获颇多。首先，我的相关知识储备不断丰富。为了设计并实现基于可信验证的区块链联邦隐私保护方案，我深入学习了网络安全方面的相关知识，对可信验证、分片区块链和差分隐私保护等领域都进行了一定的了解。这不仅用于开发系统，也为我未来的相关研究奠定了良好的理论基础。其次，我也对科研有了新的认识，并掌握了科研能力。作为论文类毕设，主要以科研为向导，探究的问题都没有现成的参考资料与实现方法。必须自己提前研究发展现状和理论知识，整合分析大量研究资料，按照自己的理解去分析和设计模型，构建原型系统，最后对实验结果进行分析和总结，得出结论。在这个过程中为了使设计的模型更加完备，需要不断考虑模型会存在的弊端并加以解决。这也让我对科研的方法有了一定的领悟和总结。在遇到新的问题时，首先得对相关背景知识的学习，然后对问题作出初步的分析和预测，接下来进行动手实践，在实践中得出结果，并将结果与当初的预测进行对比，分析区别，找到错因，自我反思，从而得到提升。

5.2 工作展望

基于可信验证的区块链联邦隐私保护方案目前只是从理论上进行了分析，并设计开发了简单的原型系统证明了其可行性及有效性。但是当该方案正式部署到相关平台落地成实际项目时，仍需要在细节上进行更为详尽的处理，包括联邦分片区块链和可信验证机制的总体流程以及两种机制的对接等，都是后续需要不断完善的地方。后续，可以考虑在 Hyperledger Fabric 上完成联邦分片区块链的开发，在 Cube-tcm 上完成可信验证机制的开发。使得该方案真正能够实现对用户隐私数据的差分保护。

通过本次毕业论文的撰写，我对可信验证、分片区块链、分布式差分隐私以及基于可信验证的联邦分片区块链有了一定的了解和掌握。分片区块链是目前区块链扩容的有效方案之一。将联邦思想引入分片区块链又使得这种系统能够为隐私保护所用，保证完整数据能够被拆分并分布式地存放到各个相关节点，这样当某一节点被攻破，不会使得用户完整数据的泄露。而可信验证的加入，在通信过程中交易信息被公开。又使得整个系统的安全性和有效性得到了更好的提升。而

这种对数据隐私的保护，也是未来区块链发展的必然趋势。

在今后的学习工作中，我也将对分片区块链和可信验证机制长期关注，并进行相关知识的积累，以便对基于可信验证的联邦分片区块链有更加深入的研究。

参考文献

[1] 可信计算 3.0 工程初步[M].人民邮电出版社:学术中国·院士系列, 201705.239.Borko H, Bernier C L. Indexing Concepts and Methods[M]. New York: Academic Pr, 1978:6.

[2] 田娜.采用多轮验证的ETH 2.0状态分片抗合谋攻击方案设计[D].大连海事大学, 2020.

[3] 付金华.高效能区块链关键技术及应用研究[D].战略支援部队信息工程大学,2020.DOI:10.27188/d.cnki.gzjxu.2020.000011.

[4] 冯登国,刘敬彬,秦宇,等.创新发展中的可信计算理论与技术[J].中国科学:信息科学,2020,50(08):1127-1147.Koya S, Yutaka W. Jointing Nitride Ceramics: JP, 07011305[P].1995-10-16.

[5] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, et al. Towards Scaling Blockchain Systems via Sharding[P]. Management of Data,2019.

[6] Jiaping Wang, Hao Wang. Monoxide: Scale Out Blockchain with Asynchronous Consensus Zones.[J]. IACR Cryptology ePrint Archive,2019,2019:

[7] Mizrahi A, Rottenstreich O. Blockchain State Sharding with Space-Aware Representations[J]. IEEE Transactions on Network and Service Management, 2020, 18(2): 1571-1583.

[8] Hou D, Zhang J, Man K L, et al. A systematic literature review of blockchain-based federated learning: Architectures, applications and issues[C]. 2021 2nd Information Communication Technologies Conference (ICTC). IEEE, 2021: 302-307.

[9] Kokoris-Kogias E, Jovanovic P, Gasser L, et al. Omniledger: A secure, scale-out, decentralized ledger via sharding[C]. 2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018: 583-598.

[10] Luu L, Narayanan V, Zheng C, et al. A secure sharding protocol for open blockchains[C]. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 17-30.

[11] Al-Bassam M, Sonnino A, Bano S, et al. Chainspace: A sharded smart contracts platform[J]. arXiv preprint arXiv:1708.03778, 2017.

［12］Zamani M, Movahedi M, Raykova M. Rapidchain: Scaling blockchain via full sharding[C]. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 931-948.

［13］Hong Z, Guo S, Li P, et al. Pyramid: A layered sharding blockchain system[C]. IEEE INFOCOM 2021-IEEE Conference on Computer Communications. IEEE, 2021: 1-10.

［14］Li K, Zhou H, Tu Z, et al. Distributed network intrusion detection system in satellite-terrestrial integrated networks using federated learning[J]. IEEE Access, 2020, 8: 214852-214865.

［15］Shen C, Zhu L, Hua G, et al. A Blockchain Based Federal Learning Method for Urban Rail Passenger Flow Prediction[C]. 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2020: 1-5.

［16］Trusted Computing Group. TCG Specification Architecture Overview, Specification Revision 1.4[J]. 2007.

［17］Liang J, Zhang M, Leung V C M. A reliable trust computing mechanism based on multisource feedback and fog computing in social sensor cloud[J]. IEEE Internet of Things Journal, 2020, 7(6): 5481-5490.

［18］Wang G, Tian D, Gu F, et al. Design of terminal security access scheme based on trusted computing in ubiquitous electric internet of things[C]. 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC). IEEE, 2020, 9: 188-192.

［19］Sun X, Wang Y. Network Security Access Model Based on TCM[J]. Communications Technology, 2019, 52(05): 1219-1223.

［20］Luu L, Narayanan V, Zheng C, et al. A secure sharding protocol for open blockchains[C]. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 17-30.

［21］Wang G, Shi Z J, Nixon M, et al. SoK: Sharding on Blockchain[C]. the 1st ACM Conference. ACM, 2019.

［22］Syta E, Tamas I, Visher D, et al. Keeping authorities" honest or bust" with decentralized witness cosigning[C]. 2016 IEEE Symposium on Security and Privacy (SP). Ieee, 2016: 526-545.

[23] Alfandi O, Otoum S, Jararweh Y. Blockchain solution for iot-based critical infrastructures: Byzantine fault tolerance[C]. NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2020: 1-4.

致　谢

谨以简短的致谢感谢一直以来帮助过我的老师、同学、前辈、亲友们。