南京大学

研 究 生 毕 业 论 文

（申请工程硕士学位）

论 文 题 目	基于 PROV 模型的 Diem 安全溯源系统
作 者 姓 名	常家鑫
学科、专业名称	工程硕士（软件工程领域）
研 究 方 向	软件工程
指 导 教 师	陈振宇 教授

2022 年 05 月 20 日

学　　号：**MF20320011**

论文答辩日期：**2022 年 05 月 20 日**

指 导 教 师：　　**（签字）**

Diem Secure Provenance System Based on PROV Model

By

Jiaxin Chang

Supervised by

Professor Zhenyu Chen

A Thesis
Submitted to the Software Institute
and the Graduate School
of Nanjing University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Engineering

Software Institute

May 2022

学 位 论 文 原 创 性 声 明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

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，本论文不句含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，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：_____

日期：　　年　　月　　日

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于 PROV 模型的 Diem 安全溯源系统

工程硕士（软件工程领域）专业 2020 级硕士生姓名：常家鑫

指导教师（姓名、职称）：陈振宇 教授

摘 要

数据溯源通过记录数据转换过程追踪数据的来龙去脉。数据溯源过程中产生的溯源信息对于数据溯源至关重要。通过区块链技术存储溯源信息能够一定程度上提高数据溯源的可信性。目前，区块链数据溯源相关研究大多采用将溯源信息定义为智能合约变量的方式进行上链存储。由于缺少对智能合约中定义变量的安全性检查，开发人员可引入诸如资产复制、资产重用、资产丢失等智能合约缺陷。一旦合约缺陷被利用将难以保障溯源数据的安全性。此外，当前数据溯源多针对某一具体场景实现，难以支持其他场景下的数据溯源，溯源系统通用性较低。

为进一步提升溯源数据安全性，避免由于合约变量安全问题导致溯源数据遭到破坏，本文提出了基于 Diem 区块链平台的溯源数据存储方案。Diem 区块链引入了资源的概念。Diem 原生数字资产 Diem Coin 和智能合约中定义的变量均可看作资源类型的一种。资源归 Diem 中的账户所有，且资源不能被复制、或隐式销毁，只能在不同程序地址上移动。Diem 区块链的这一特点从程序语言的角度解决了数字资产的保护问题，保障了智能合约中数字资产的安全性。本文基于 PROV 数据溯源模型并结合抽象化模板的思想解决数据溯源通用性较低的问题。PROV 模型适用于较多场景下的数据溯源。本文通过溯源模板将 PROV 模型具体化表示，使用溯源模板代表不同的数据溯源场景，一定程度提升了数据溯源的通用性。

在技术实现方面，存储层采用 Diem 区块链平台存储溯源数据，溯源服务层采用 Springboot 框架、星际文件系统 IPFS 管理溯源逻辑，前端采用 Vue 框架进行模块化开发。采用 Docker 部署系统服务，提升系统可用性和可扩展性。最终对系统开展功能测试和性能测试。功能上，实现了具有一定通用性的可信数据溯源。性能上，系统各接口在吞吐量、并发量等指标均能达到可用性要求。综上所述，系统能够为多场景下数据溯源提供稳定可靠服务，提高溯源数据可信性。为数据溯源的发展做出了贡献。

关键词：数据溯源，Diem，PROV 模型，安全

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Diem Secure Provenance System Based on PROV Model

SPECIALIZATION: Software Engineering

POSTGRADUATE: Jiaxin Chang

MENTOR: Professor Zhenyu Chen

Abstract

Data provenance traces the origin of data by documenting the data transformation process. The provenance information generated in the data provenance process is crucial to data provenance. Storing provenance information through blockchain technology can improve the credibility of data provenance to a certain extent. At present, most of the research related to blockchain data provenance adopts the way of defining provenance information as smart contract variables for storage on the chain. Due to the lack of security checks on the variables defined in smart contracts, developers can introduce smart contract defects such as asset replication, asset reuse and asset loss. Once the contract defects are exploited, it will be difficult to guarantee the security of provenance data. In addition, the current data provenance is mostly implemented for a specific scenario, and it is difficult to support data provenance in other scenarios, and the generality of the provenance system is low.

In order to further improve the security of provenance data and avoid the destruction of provenance data due to the security of contract variables, this paper proposes a provenance data storage scheme based on Diem blockchain platform. Diem blockchain introduces the concept of resources. Diem native digital asset Diem coin and the variables defined in smart contract can be regarded as a kind of resource type. The resource belongs to the account in diem, and the resource cannot be copied or implicitly destroyed, but can only be moved on different program addresses. This feature of Diem blockchain solves the protection of digital assets from the perspective of program language and ensures the security of digital assets in smart contracts. Based on the PROV data provenance model and the idea of abstract template, this paper solves the problem of low universality of data provenance. PROV model is suitable for data provenance in many scenarios. In this paper, the PROV model is expressed concretely through the

provenance template, and the provenance template is used to represent different data provenance scenarios, which improves the universality of data provenance to a certain extent.

In terms of technical implementation, the Diem blockchain platform is used for storing provenance data in the storage layer, the Springboot framework and IPFS, an interstellar file system, are used for managing provenance logic in the provenance service layer, and the Vue framework is used for modular development in the front-end. Docker is used to deploy system services to improve system availability and scalability. Functional and performance tests were finally carried out on the system. Functionally, trusted data provenance with a certain degree of versatility was achieved. Performance-wise, the system interfaces can meet the availability requirements in terms of throughput, concurrency and other indicators. In summary, the system can provide stable and reliable services for data provenance in multiple scenarios and improve the trustworthiness of provenance data. It contributes to the development of data provenance.

Keywords: Data Provenance, Diem, PROV Model, Secure

目录

表 目 录

图 目 录

第一章　引言

1.1　选题背景和意义

互联网技术的蓬勃发展使得网络世界中的数据爆发式增长。借助互联网技术，海量的数据在不同的平台、系统、组织之间流转和融合。数据本身所携带的信息往往具有重要价值。在不同领域，数据逐渐成为新的生产资料。生产中经常见到的数据大多由原始数据加工融合而来。除此之外，不同数据之间往往存在先后和依存关系。数据来源的多样性以及数据间关系的复杂性往往使得数据的可信性和安全性不能得到保障 [1]。如何保证数据的可信性和安全性成为一个亟待解决的问题。数据溯源为解决这一问题提供了思路。

作为一种溯本追源的技术，数据溯源通过追踪路径完成数据历史状态和演变过程的重现，最终实现数据历史档案的追溯 [2]。通过数据溯源技术重现数据产生过程，对数据来源进行进行记录追踪，一定程度上解决了数据来源模糊，数据可信性不高的问题。传统的数据溯源系统多采用中心化方式存储数据，这种存储方式存在单点故障问题且易导致数据库遭受内部、外部攻击。中心化数据库中的数据易被恶意篡改，数据的安全性难以得到保障 [3]。安全存储溯源数据成为进一步提高溯源数据可信性的关键。

区块链 [4] 是一种基于互联网的去中心化信任管理机制，其难以篡改、可溯源等特性为可信的数据溯源提供了新的解决途径。目前，已有在以太坊 [5]、超级账本 [6] 区块链平台上数据溯源的相关研究 [7–9]。智能合约是实现数据溯源的关键，开发者通过编写智能合约实现具体的数据溯源逻辑。然而，在以太坊、超级账本区块链平台中，由于缺少对智能合约中定义变量的安全性检查，开发人员可引入诸如资产复制、资产重用、资产丢失等智能合约缺陷 [10]。这些缺陷可导致严重的安全问题。例如，由于智能合约缺陷已引发诸如 DAO[11] 攻击和 Parity 多重签名钱包合约漏洞 [12] 等重大安全事件，损失的价值总额高达数百万美元。

Diem[1]是由 Facebook 主导开发，以区块链技术为基础技术之一，由 Diem 协会负责管理和运营的加密数字货币及其配套的金融基础设施。相较于其他区块链平台，Diem 安全性高，可保障资金和金融数据的安全。在智能合约方面，Diem 采用 Move[10] 语言。Move 引入了资源的概念，Diem 原生数字资产 Diem Coin

[1]Diem. https://www.diem.com/en-us/

和智能合约中定义的变量均可看作资源类型的一种。Move 语言的关键特性是其具备实现自定义资源类型的能力。Move 规定资源类型不能被复制，不能被隐式地销毁，只能在不同程序地址上移动。此外，资源类型只能由声明该类型的模块创建或销毁。在这种机制下，开发者可在智能合约中基于资源类型定义各种数字资产，这些数字资产均能得到和 Diem Coin 同等程度的保护。Move 语言的这一特点从程序语言的角度解决了数字资产的保护问题，保障了智能合约中数字资产的安全性。与 Diem 智能合约定义数字资产的方式不同，其他区块链平台直接将智能合约中的普通变量视为数字资产。在其他区块链平台中，针对数据资产的安全性检查，只能由开发者自己维护。因此，采用 Diem 区块链将溯源数据定义为数字资产进行存储和管理可有效保障溯源数据的安全性。

数据溯源模型描述了数据溯源的基本思路和大体步骤。PROV 模型最初由万维网联盟提出 [13]。与其他数据溯源模型 [14–16] 相比，PROV 模型不仅适用于较多场景下的数据溯源，而且拥有丰富的说明文档。PROV 模型提出了数据溯源中实体、代理、活动三种重要类型并详尽描述了三种类型之间的关系。本文借助溯源模板将 PROV 模型中描述的类型和关系进行具体化。并将溯源模板、原始数据等溯源信息定义为 Diem 中的资源类型进行存储。最终，本文实现了基于 PROV 模型的的 Diem 安全溯源系统。系统通过 Diem 资源存储溯源信息，使得溯源信息不能被随意复制、重用或销毁，保证溯源信息的安全存储。数据上链过程由智能合约完成，并通过多方共识验证，防止篡改。

1.2 国内外研究现状及分析

本文重点研究利用 Diem 区块链实现安全的数据溯源，下面将分别从数据溯源技术和基于区块链的数据溯源系统两个方面描述国内外研究现状。

1.2.1 数据溯源技术

数据溯源（Data Provenance）的概念最早于 20 世纪 90 年代提出 [2]。在不同领域，数据溯源有着不同的含义。Simmhan 等人认为从源数据派生到数据产品的流程信息是数据溯源 [17]。在数据库领域，Buneman 等人将数据溯源定义为“数据的起源以及数据在数据库之间的移动”[18]。在数据仓库系统中，戴超凡等人将其定义为“数据从产生到死亡全生命周期内的演变信息”[19]。在地理信息系统中，Lanter 等人认为描述数据演变过程的信息是数据溯源 [20]。在本文中，溯源表示不同场景下数据之间关系的追溯和数据真实性的验证。

数据溯源模型是实现数据溯源的关键，目前，关于数据溯源模型也有较多研究。Bowers 等人提出了用于医疗领域的时间-值中心模型，根据时间以及医疗事

件流 ID 追溯原始数据 [14]。在第一届国际溯源与注释研讨会 (International Provenance and Annotation Workshop,IPAW) 中, 参会成员提出了开放起源模型 (Open Provenance Model,OPM)，后逐渐成为数据溯源模型的标准 [15]。在第二届 IPAW 会议上，Sahoo 等人提出了 Provenir 模型，该模型采用物化视图的方法进行数据存储 [16]。Provenir 模型解决了 OPM 模型在实际应用中存在的概念术语模糊等问题，被广泛应用于海洋，传感器等领域。在 OPM 基础上，万维网联盟修改了 OPM 的缺陷并发布了新的 PROV 模型及相关文档 [13]。目前，PROV 模型已在数据库、工作流等领域取得广泛应用 [21]。

在数据溯源方法上，Chiticariu 等人提出了标注法，标注法通过在原始数据中添加重要的标记信息实现数据溯源 [22]。标记法需要额外的存储空间存储标记信息。Poulov 等人提出了反向查询法，反向查询法通过构造逆置函数追溯原始数据 [23], 反向查询法减少了标注法中因标记而产生的存储代价。明华等人针对标记法消耗存储空间的缺点，设计了列存储的方法压缩存储空间，一定程度上减轻了标记法对了存储空间的消耗 [2]。此外，徐飞等人通过引入追踪图和追踪路径的概念提出了一种新的数据追踪方法 [24]。Wang 等人针对特定数据格式类型提出了双向指针追踪法 [25]。Gadang 等人提出了位向量存储定位法，该方法记录数据处理路径实现数据溯源 [26]。

数据溯源已在多个领域取得应用。在数据库领域, Chiticariu 等人采用注释方式提出了 DBNotes 数据溯源系统 [22]。Widom 等人提出了经典的 Trio 系统, Trio 系统在传统关系型数据库中加入了数据溯源功能 [27]。在工作流领域，MyGrid 团队提出了 Taverna 系统[2]，利用 if、for 等程序语言中的控制语句梳理工作流中各步骤关系，进而实现数据溯源。UC Davis 等人提出了 Kepler 开源系统，通过追踪数据的历史记录，将溯源结果提供给用户 [28]。在区块链领域，Pingcheng 等人实现了 LineageChain 系统，在超级账本区块链中引入 Merkle DAG ，并增加跳表索引，实现支持细粒度的、安全高效的区块链数据溯源功能 [29]。Miguel 等人基于以太坊和超级账本提出了用于农产品供应链追溯的 AgriBlockIoT 系统 [30]。

1.2.2 基于区块链的数据溯源系统

区块链技术可以理解为一种由多个节点共同维护的数据库系统，节点之间通过共识机制达成一致，多个节点存储同一份数据。区块链概念最早由 2008 年中本聪所发表的论文 [31] 发展而来。自概念提出以来，先后出现了比特币

[2]Taverna. http://taverna.sourceforge.net/

(Bitcoin)[3]、以太坊 (Ethereum)[4]、超级账本 (Hyperledger Fabric)[5]、Diem[6]等区块链平台。Diem 区块链由 Facebook 提出，旨在构建一个“全新的”加密货币系统。Diem 设计 Move 语言实现智能合约编程，与其他区块链平台的智能合约相比具有更高的安全性，可避免出现由智能合约编写逻辑产生的问题。区块链蓬勃发展的同时，也催生出各种基于区块链的应用。目前，区块链已在数据溯源 [32]、金融货币 [33]、数字资产 [34] 等领域得到广泛应用。

区块链去中心化、不可篡改、可溯源的特性对于解决传统数据溯源中溯源信息难以安全存储的问题具有重要意义。目前，已有相关基于区块链的数据溯源研究。Liang 等人提出了一种在云技术环境下区块链数据溯源体系框架 ProvChain，该框架可监控记录用户数据操作，利用区块链对数据记录进行验证，实现云技术环境下的数据溯源 [35]。刘耀宗等人在 RFID 大数据领域引入区块链技术，实现 RFID 大数据全链路追踪，利用区块链去中心化的特性保障溯源数据的安全管理 [36]。张国英等人采用公钥密钥技术验证数据溯源参与方真实性验证，并通过建立数据溯源模型表示溯源数据，最终设计智能合约实现溯源信息的链上存储，保证了链上溯源信息的真实可靠 [37]。Nugent 等人将区块链技术应用于临床实验中，通过区块链技术管理临床实验中产生的相关数据，提高临床实验数据的透明度 [38]。Montecchi 等人建立一套商品生产信息溯源体系，全流程跟踪存储商品生产数据，最终将商品溯源信息提供给用户 [39]。任浩方等人设计物联网数据溯源模型，提出区块链平台中物联网溯源数据的存储和查询方法，基于以太坊平台下实现溯源数据的安全存储和高效查询 [40]。以上工作多利用智能合约中的普通变量存储溯源数据，由于缺少对于智能合约变量的安全性检查，易导致开发人员引入相关缺陷。本文考虑将溯源数据抽象为 Move 资源进行存储，基于 Diem 区块链平台实现溯源系统。

1.3 本文的工作

本文设计了 PROV 模型表示下的 Diem 安全数据溯源方法，并以此实现了基于 PROV 模型的 Diem 安全溯源系统，主要工作如下：

在方法上，针对当前区块链数据溯源中存在的溯源数据安全问题，本文设计了一种 PROV 模型表示下的 Diem 安全数据溯源方法。该方法在 PROV 模型的基础上，结合溯源模板建立数据溯源模型。在此基础上，通过将溯源数据抽象

[3]Bitcoin. https://bitcoin.org/en/

[4]Ethereum. https://ethereum.org/zh/

[5]Hyperledger Fabric. https://cn.hyperledger.org/

[6]Diem. https://www.diem.com/en-us/

为 Diem 资源进行存储的方法保障溯源数据的安全性，通过遍历有向无环图的方法完成溯源数据查询。本文通过建立溯源模板提升数据溯源场景的通用性，以溯源模板为基础采集不同类型的源数据，并将采集到的数据以资源的形式存储至区块链中。当需要进行数据溯源时，先从溯源模板中解析数据之间的关系，之后从区块链中读取目标溯源字段的原始数据。

在系统上，本文根据上述方法实现了基于 PROV 模型的 Diem 安全溯源系统。系统分为以下四个重要模块：(1) 系统提供溯源模板管理功能。溯源模板分为对象模板和任务模板两种类型。对象模板类似于编程语言中的 Class 类，用于描述复杂的数据对象。任务模板可以设置不同的溯源阶段、每个溯源阶段可选择已创建的对象模板作为当前阶段所包含的属性。(2) 系统提供数据采集功能。不同场景下的数据通常有不同的表现形式，系统将采集到的数字、文件、时间、字符串等各种异构数据统一转化成 IPFS 哈希值进行存储。(3) 系统提供数据上链功能。数据上链过程通过智能合约完成。链上数据经多节点验证后共同存储，保护链上信息不被篡改。系统将溯源模板、原始数据哈希等溯源信息上链，保障数据的安全性。(4) 系统提供数据溯源功能。数据实体及溯源关系被抽象为 DAG，针对 DAG 中的一个节点溯源只需搜索 DAG 中所有与之相关的前驱节点，搜索过程中出现的节点和边都将作为溯源结果返回。

1.4 本文的组织结构

本文对基于 PROV 模型的 Diem 安全溯源系统进行了详细的介绍，本文的组织结构如下：

第一章：引言。介绍了本文的背景与意义，分别从数据溯源技术、基于区块链的数据溯源系统两个研究领域给出了相关研究现状，并对本文的主要工作进行总结。

第二章：技术综述。阐述本系统相关的数据溯源技术和 Diem 区块链。首先，从数据溯源概述、数据溯源方法以及数据溯源模型三个方面介绍数据溯源技术。之后，介绍了 Diem 区块链的基本概念、Move 智能合约编程语言、Diem 与其他区块链的对比。

第三章：技术研究。首先，系统给出系统数据溯源的整体框架和流程。之后，在 PROV 模型的基础上，建立本文的数据溯源模型。随后，给出了系统溯源数据的可信存储方案。最后，提出基于有向无环图的数据溯源方法。

第四章：需求分析与概要设计。首先对项系统进行简要概述，然后进行需求分析，总结出功能性和非功能性需求，同时给出系统所涉及用例。围绕 4+1 视图对系统总体设计进行阐述并给出系统的持久化模型设计。

第五章：详细设计与实现。将系统划分为溯源模板管理、溯源数据管理、溯源任务管理、数据溯源四大模块。具体地，详细介绍了四大模块中溯源模板、数据采集、数据上链、溯源任务配置以及数据溯源的实现。

第六章：系统测试与案例分析。本节通过设计测试样例对系统进行功能测试和性能测试，并对测试结果进行分析。之后，以众包测试场景为例，详细描述了众包测试溯源的全部流程。

第七章：总结与展望。对本文工作进行全面总结，并对当前工作中存在不足的地方进行分析，对后续发展做出展望。

第二章　技术综述

本章主要对系统使用的数据溯源、Diem 区块链技术进行介绍。在数据溯源方面，本文主要描述了数据溯源的概念、数据溯源常用的方法以及数据溯源模型。在区块链方面，本文首先对 Diem 进行概述，之后详细介绍了 Move 智能合约编程语言以及不同区块链平台之间的差异。

2.1　数据溯源技术

2.1.1　数据溯源概述

每个数据都会经历从最初产生到最终删除或存档的生命周期 [19]。在一份数据的生命周期中数据会经历查询、存储及各种加工处理的过程。数据溯源用于追溯查询数据在生命周期内被改变和处理的信息。一次完整的数据溯源活动一般包括溯源数据采集、溯源数据存储和溯源数据应用等三个部分。下文将分别介绍这三个部分：

1) 溯源数据采集：溯源数据是完成数据溯源的关键，溯源数据决定了如何获取数据历史档案。溯源数据所包含的内容是数据溯源研究工作者一直关注的问题。文献 [41] 认为溯源数据的内容应该包括 who、when、where、how、which、what、why 七种信息。显然，这种溯源数据记录方式非常详尽，但也存在着存储开销较大的问题，且每个数据可能缺失以上七种信息中的一种或多种。

2) 溯源数据存储：溯源数据采集完成之后即需要考虑溯源数据存储的问题。在溯源数据存储中，需要从数据存储形式和数据存储介质两方面考虑如何存储溯源信息。在数据存储形式上，一种方法是通过扩充原始数据属性记录溯源信息，另一种是以树形结构方式组织溯源数据。在数据存储介质上，传统数据溯源一般存储于中心化的关系型数据库中，现阶段也有较多采用各种区块链平台作为数据存储介质的方案。

3) 溯源数据应用：完成数据存储后，即可进行溯源数据的查询。系统通过数据溯源方法查找已存储的溯源数据，并将数据溯源结果提供给数据溯源需求方。一次完整的数据溯源经过以上三个步骤后最终完成。

2.1.2　数据溯源模型

数据溯源模型的构建对于数据溯源至关重要。数据溯源模型规定了溯源数据从采集、存储到应用的统一标准，根据数据溯源模型可初步确定数据溯源的

基本思路。由万维网联盟发布的 PROV 模型得到了专家和相关技术人员的肯定，与其他数据溯源模型相比，PROV 模型拥有丰富的语义词汇和数据模型，具有广阔的发展前景 [37]。本节将详细介绍 PROV 模型的具体内容。

相较于 OPM 模型和 Provenir 模型，PROV 模型更加简洁，且对于概念的定义更为准确。在 PROV 模型中，用户可使用 RDF、XML 或 JSON 格式描述溯源数据。此外，借助于 PROV 模型，异构系统之间的溯源信息可方便地进行转换、推理。PROV 模型共由 12 个说明文档组合而成。表2.1所示为 PROV 推荐的 4 个标准，其余八个为工作草案。PROV-O 提供一个 OWL2 本体，主要用于数据关联和语义网领域。PROV-DM(PROV 数据模型)，则由三大类型和七种关系组成，是 PROV 标准的核心。PROV-N 主要用于提供用户友好的数据溯源符号。PROV-CONSTRAINTS 中规定了溯源模型相关的约束条件，用于解决溯源信息的验证问题。

表 2.1: PROV 数据溯源推荐标准

推荐标准	描述
PROV-O	提供一个 OWL2 本体，主要用于数据关联和语义网领域
PROV-DM	PROV 数据模型，由三大类型和七种关系组成，是 PROV 标准的核心
PROV-N	提供用户友好的数据溯源符号
PROV-CONSTRAINTS	描述溯源模型相关的约束条件，用于解决溯源信息的验证问题

PROV-DM 是一种用于数据溯源的通用型模型 [42]。PROV-DM 模型包含两个重要的组件，分别是类型和关系。类型包含实体（Entity），活动（Activity）和代理（Agent）三个要素。实体是指在数据溯源中需要被记录信息的事物，如一个网页，一本书籍等物理的或者抽象的事物。活动的概念比较广泛，活动描述了一个实体如何变为现在的状态。针对实体的动作如修改、转换以及处理等都可以称为活动。代理强调责任的概念，对已经存在的实体、正在发生的活动或者另一个代理承担某种形式责任的事物都被称为代理。

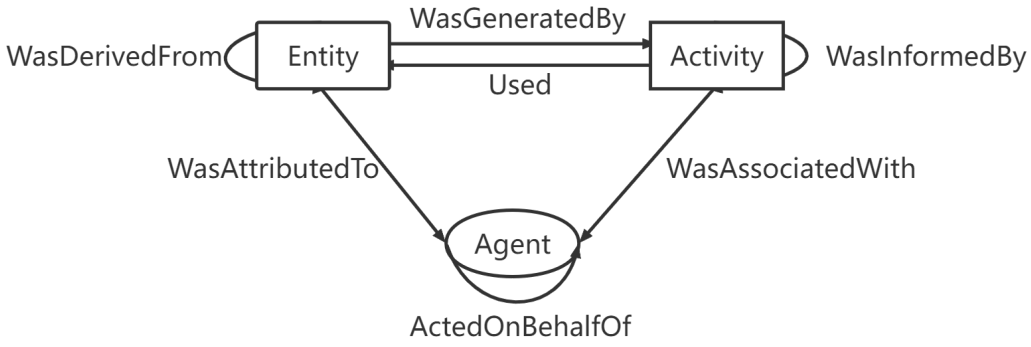

图 2.1: PROV 核心结构

各个类型之间存在不同的关系。以电子报表 PROV-DM 模型为例，各个类型的关系如图 2.1所示。电子报表为一个实体，一次对电子报表修改的活动可产生新版本的电子报表实体，用 wasGenerated 表示。在这次修改的活动中，活动也利用了前一个版本的电子报表实体，用 used 表示。本次活动产生的新版本实体与旧版本实体之间的衍生关系用 wasDerivedFrom 表示。一个活动同时也可以通知调用其他活动，这种关系用 wasInformedBy 表示。在 PROV 数据模型中，活动与代理之间的关系用 wasAssociated 表示，表示代理对活动负有责任。实体与代理之间的关系表示为 wasAttributedTo。代理与代理之间还存在着 ActedOnBehalfOf 的关系。

PROV-DM 模型不仅适用于较多场景下的数据溯源，而且拥有丰富的说明文档。以上两点是本文选用 PROV-DM 模型构建 Diem 区块链平台下的数据溯源模型的原因。此外，Diem 区块链安全的智能合约以及链上数据不可篡改的特点保证了数据溯源的可信性。

2.1.3 数据溯源方法

现阶段，数据溯源方法主要是标注法和反向查询法。两种数据溯源方法分别适用于不同特点下的数据溯源且都取得了较为广泛的应用。本节将详细介绍这两种数据溯源方法。

标注法通过在原始数据中添加重要属性达到记录数据溯源信息的目的。例如在测试报告中，通过添加测试报告的时间、测试方、测试需求信息即可在获取该测试报告的同时获取到该数据报告的出处信息。这种事先进行标记并携带溯源信息的方法虽然实现方式简单，但在实际的数据溯源应用中十分有效。标记法的实现方式决定了其不适用于大型系统的数据溯源，原因是在大型系统中很

难获取用于标记的溯源信息。此外，大量的标记信息需要额外的存储开销。极端情况下，标记信息所需的存储空间将远高于原始数据所需的存储空间。

反向查询法是指通过构造逆向函数对溯源查询求逆实现数据溯源的方法。反向查询法不需要对原始数据进行额外标注，只需在数据溯源需要时通过逆向函数得到原始数据的溯源信息。这种方法的核心在于逆向函数的构造。逆向函数的优劣直接决定了数据溯源所需的时间开销和存储开销。相比于标注法，反向查询法虽然实现较为复杂，但减少了标记所需的存储空间，只需存储少量的数据即可实现数据溯源。此外，并非所有的函数都有逆向函数，这就限制了反向查询法的使用场景。

2.2 Diem 区块链

2.2.1 Diem 概述

Diem（原名为 Libra，后更名为 Diem）最早于 2019 年 6 月 18 日被 FaceBook 公司推出。Diem 的愿景是为全球数亿人提供一个稳定的数字货币交易媒介。从设计角度来看，Diem 区块链可看做是一个去中心化、可编程的数据库，数据库中保存的数据均经过密码学验证，可保证数据的真实可信。Diem 在该数据库中维护了一个以键值对形式存储的账本状态，该键值对存储着账户地址和账户数据之间的映射。在账本状态中，存在账户、账户地址、资源和模块四个重要概念。这四个概念的详细描述如下：

账户：与以太坊类似，Diem 使用账户/余额模型编码账本状态 [5]。在 Diem 区块链中，账户是资源和模块的容器。每个账户可包含多个资源和模块。账户使用账户地址唯一标识。

账户地址：账户地址是一个 256-bit 的值。创建新账户的第一步是生成新的公开密钥和私有密钥。Diem 使用公开密钥的加密哈希值作为账户地址。私有密钥由用户或客户端保存。用户必须使用私有密钥才可发起交易。

模块：模块可看作是 Diem 区块链中的智能合约。它定义了更新 Diem 区块链全局状态的规则。模块使用 Diem 提出的全新编程语言 Move[10] 编写。模块只包含代码，不包含数据。在用户提交的交易中包含交易脚本，交易脚本可调用一个或多个模块完成 Diem 账本状态的更新。

资源：资源存储了各种简单变量类型或复杂变量类型的值。资源的类型以及资源的创建、修改、删除、发布规则均在模块中被指定。Move 语言为资源提供了特殊的安全保证，一个资源永远无法复制，只能移动。此外，资源只能由声明该资源的模块创建或销毁。

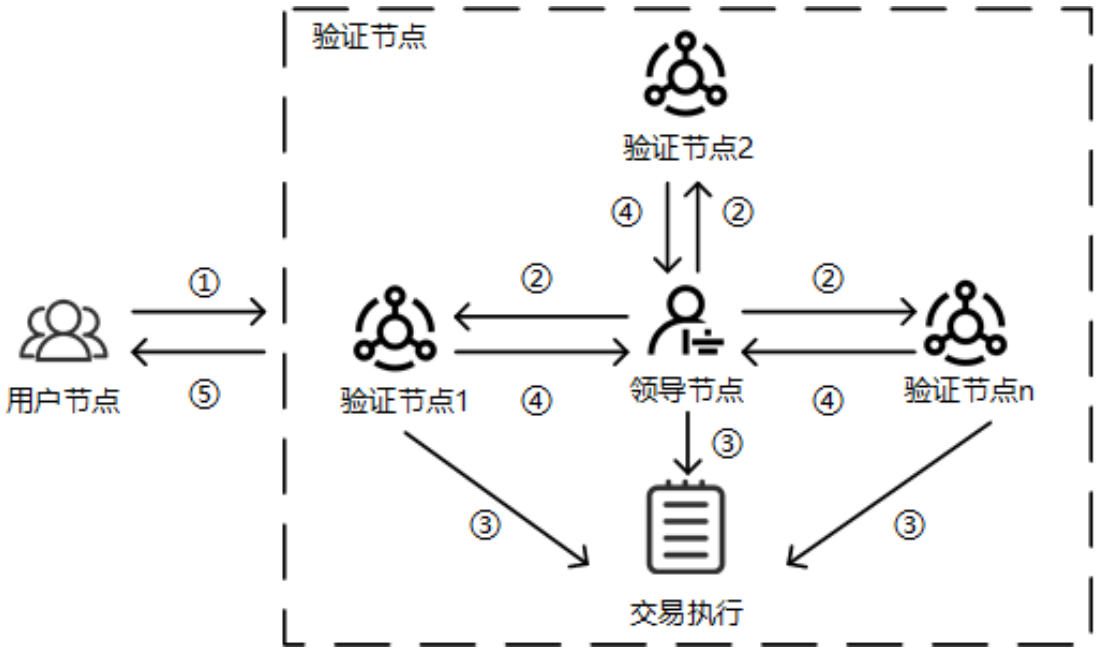

图 2.2: Diem 节点结构

在 Diem 区块链中，包含用户节点和验证节点两种类型的实体。用户节点可提交交易或者查询交易。验证节点用于处理交易并维护整个 Diem 区块链数据库。如图 2.2所示，用户节点和验证节点之间的交互过程可描述为如下步骤：(1) 客户端提交交易至验证节点，请求各验证节点达成共识执行交易并将交易结果保存至区块链中。(2) 验证节点可轮流成为领导者节点，领导者节点将用户节点直接提交的交易或者其他验证节点间接提交的交易打包成区块后广播给其他验证节点。(3) 在各节点收到领导者节点的广播的区块后，验证节点会执行区块中的所有交易。由于验证节点存在作恶的可能性，不能把交易结果直接存储。(4) 针对步骤三中产生的交易结果，验证节点对其进行投票以达成一致，达成一致后才能存储交易结果。(5) 交易提交成功后，验证节点将交易和交易结果一并存储。验证节点还将生成代表当前数据库完整状态的签名，并把该签名作为回应的一部分返回给用户节点。

2.2.2 Move 智能合约编程语言

Diem 的使命是为全球数十亿人提供一个简单的全球货币和金融基础设施。为实现这一目标，Diem 提出了一种安全灵活的编程语言 Move 语言 [10]。用户使用 Move 语言可实现自定义的交易逻辑和智能合约编程。Move 语言具有以下特征。

一等资源 (Frist-Clase Resources)：在以太坊中，针对原生数字资产以太币 (ETH) 的操作具有一系列验证保护过程。这些验证保护过程如：针对一笔以太币转账，以太坊虚拟机 (EVM) 需要验证交易发起者是否为该以太币的实际拥有者、该账户是否有足够的余额进行转账等。然而，针对基于合约变量定义的数字资产，以太坊中缺乏对其进行各种操作的保护机制，合约变量的安全性检查操作只能由开发者自己维护。开发者需要避免引入因数字资产复制、重用、销毁而导致的严重错误。为解决这一问题，Move 引入了资源的概念，Diem 原生数字资产 Diem Coin 和智能合约中定义的变量均可看作资源类型的一种。Move 语言的关键特性是其具备实现自定义资源类型的能力。Move 规定资源类型不能被复制，不能被隐式地销毁，只能在不同程序地址上移动。此外，资源类型只能由声明该类型的模块创建或销毁。在这种机制下，开发者可在智能合约中基于资源类型定义各种数字资产，这些数字资产均能得到和 Diem Coin 同等程度的保护。

安全性：使用 Move 语言编写的代码需要进行内存、类型、资源等类型的安全性检查。一般来说，程序安全性检查动作可发生在编译时或运行时。Move 语言的安全检查则介于两者之间。Move 将可执行代码设计为一种被称作类型字节码 (typed bytecode) 的形式。类型字节码被发布上链前，字节码验证器 (Bytecode Verifier) 将检查其是否满足内存安全、类型安全和资源安全的要求。若字节码满足安全性要求，则可将其发布并交由字节码解释器 (Bytecode Interpreter) 运行。

灵活性：Move 通过交易脚本提升灵活性。交易脚本是一段可执行的 Move 代码，每个交易中都包含一个交易脚本。交易脚本可以直接执行，也可以在一笔交易中调用多个已发布的模块 (Move 中的智能合约)。交易脚本机制使得 Diem 可以在一笔交易中灵活实现对多人转账。

```
1 public main(payee: address, amount: u64) {
2     let coin: 0x0.Currency.Coin =
          0x0.Currency.withdraw_from_sender(copy(amount));
3     0x0.Currency.deposit(copy(payee), move(coin));
4 }
```

图 2.3: 点对点转账交易代码示例图

接下来将结合具体示例，详细描述 Move 语言的三个主要特征。如图 2.3所示为一个点对点支付的交易脚本示例。该脚本有转账金额和交易接收方地址两个输入。脚本的目的是实现将制定金额 amount 转移至账户 payee 中。转账步骤如下：首先，从 0x0.Currency 模块中调用 withdraw_from_sender 方法，该方法的

返回值 Coin 是一个资源值，其类型为 0x0.Currency.Coin。其中，0x0 是存储模块的账户地址、Currency 是模块的名称、Coin 则是在 Currency 中声明的资源，针对 Coin 资源的操作均需要在 Currency 模块中定义。之后，发送方调用 deposit 方法，将指定数量的 Coin 资源转移至收款方。

在上述转账程序中，Move 语言保证资源不能被复制、重用或丢失。Move 通过这种机制，提高了 Diem 区块链中的数字资产的安全性。若上述脚本存在以下三种操作，将不能通过 Move 的安全性检查。

将 copy(coin) 替代 move(coin) 以复制资源：每次读取某一 Move 变量 x，都必须指定其用法是移出变量 (使 x 不可用) 还是复制值 (使 x 继续可用)。u64 类型和 address 类型等不受限制的值可以复制或者移动。但是，资源值只能移动，Move 不支持资源类型的复制操作，尝试复制操作将在字节码验证时不能通过安全性检查。

两次使用 move(coin) 以重用资源：若在上述代码末尾添加一行 0x0.Currency.deposit(copy(payee1), move(coin)) 将导致同一 coin 资源被花费两次。Move 机制将拒绝这样的程序，资源 coin 在第一次 move(coin) 操作后将变得不可用，第二次使用 move(coin) 将触发字节码验证错误。

忽视 move(coin) 以丢失资源：上述点对点转账脚本中，若将包含 move(coin) 的代码行删除将同样导致字节码验证出错。Move 语言的这种机制帮助开发人员避免了丢失资源的风险。

2.2.3 Diem 与其他区块链的对比

表 2.2: 区块链平台部分架构对比

平台名称	数据层	共识层	合约层
比特币	基于交易	PoW	脚本
以太坊	基于账户	PoW/PoS	Solidity
超级账本	基于账户	PBFT	Go/Java
Diem	基于账户	DiemBFT	Move

表 2.2所示为区块链平台部分架构对比。本节重点比较比特币、以太坊、超级账本和 Diem 区块链平台分别在数据层、共识层和合约层的区别。

在数据层，比特币采用基于交易的模型，其他区块链平台采用基于账户的模型。基于交易的模型中，多笔交易通过交易哈希指针连接为链条状，交易可通过链条追踪至源头。基于交易的模型中，交易的链式存储便于实现交易的验证，

但账户余额查询效率较低。以太坊、超级账本和 Diem 区块链采用的基于账户的模型则可根据账户地址快速查询账户状态。此外，基于账户的模型更加方便智能合约的构建。

在共识层，主要有工作量证明机制 (Proof of Work，PoW)[43]、权益证明机制 (Proof of Stake，PoS)[44]、实用拜占庭协议 (Practical Byzantine Fault Tolerant，PBFT)[45] 和 Diem 拜占庭协议 (Diem Byzantine Fault Tolerance，DiemBFT)[46]。其中，PoW 是比特币的共识算法，该机制存在的主要问题是数据难以计算，造成大量资源浪费。相较于 PoW 机制，PoS 机制解决了 PoW 中资源过度消耗的问题，但其安全性低于 PoW 机制。PBFT 具有较高的共识效率且容忍作恶节点的存在，更适用于联盟链和私有链场景。Diem 采用 DiemBFT 共识机制，DiemBFT 是一个为 Diem 设计的具有较高鲁棒性的共识机制，是一种基于 BFT 的新型共识算法，在扩展性和一致性上达到了较高的水平。

在合约层，比特币仅支持脚本，可扩展性性较低。以太坊提出智能合约的概念并采用 Solidity 语言开发智能合约，实现自定义合约功能。超级账本智能合约的编写支持 Java、Go 等高级语言。Diem 提出了新的编程语言 Move，以提高智能合约中资源类型的安全性。

本文最终选择 Diem 区块链平台存储溯源数据。将待上链溯源数据定义为 Diem 中的资源类型，保证溯源数据的安全存储。

2.3 本章小结

本章从数据溯源技术和 Diem 区块链两方面对本文所需的基础知识进行描述。在数据溯源方面，着重描述了数据溯源的概念，并在此基础上详细介绍了 W3C 推荐的 PROV 模型内容和常见的数据溯源方法。在 Diem 区块链方面，首先对 Diem 中的关键概念进行介绍。之后详细介绍了 Move 智能合约编程语言。最后，详细对比了 Diem 区块链与其他区块链平台的差异，并给出了本文选择 Diem 区块链平台作为溯源数据存储的原因——Diem 区块链中定义的资源具有较高的安全性。

第三章　数据溯源方案技术研究

本章主要介绍基于 PROV 模型的 Diem 安全数据溯源的整体方法。首先介绍本文数据溯源的整体流程。之后，针对流程中较为重要的基于 PROV 的溯源模型表示、基于 Diem 的溯源数据存储方案以及基于有向无环图的数据溯源方法进行详细描述。

3.1　数据溯源整体流程

图 3.1所示为系统数据溯源流程图。其主要包括了以下五个步骤，分别是溯源模板定义、PROV 模型表示、异构数据采集、数据上链、数据溯源。下面将详细介绍各步骤的具体作用以及步骤之间的关系。

步骤一：溯源模板定义包括对象模板定义和任务模板定义。任务模板可使用已定义好的对象模板扩充任务模板中支持的数据类型。一个对象模板由多个基础数据类型 (数字、字符串、时间、文件) 组成。在对象模板中可定义多个字段，在每个字段中需要规定字段的名称、类型和描述信息。一个任务模板由多个阶段组成。每个阶段可以由多个对象模板或基础数据类型组成。在任务模板中定义的每一个字段，除需要指定其字段名称、类型和描述信息外，还需要指定字段之间的依赖关系。后一阶段的字段可依赖于前一阶段的字段。

步骤二：PROV 模型表示基于溯源模板的定义而生成。一个任务模板可理解为一个具体场景下的 PROV 模型。关于 PROV 模型的具体表示过程将在下文作进一步阐述。

步骤三：异构数据采集是数据溯源的源头。数据采集依赖于溯源模板进行。在这一步骤中，数据提供方将原始数据录入溯源系统。

步骤四：数据采集完成将发起数据上链请求。多个区块链节点达成共识后通过智能合约将数据存储于区块链中。数据上链方案将在下文详细阐述。

步骤五：本文基于有向无环图的方法实现数据溯源。数据溯源的结果以一张代表数据之间关系的树状图表示。数据溯源结果还包括从区块链中取出的原始数据和数据上链时产生的交易信息。

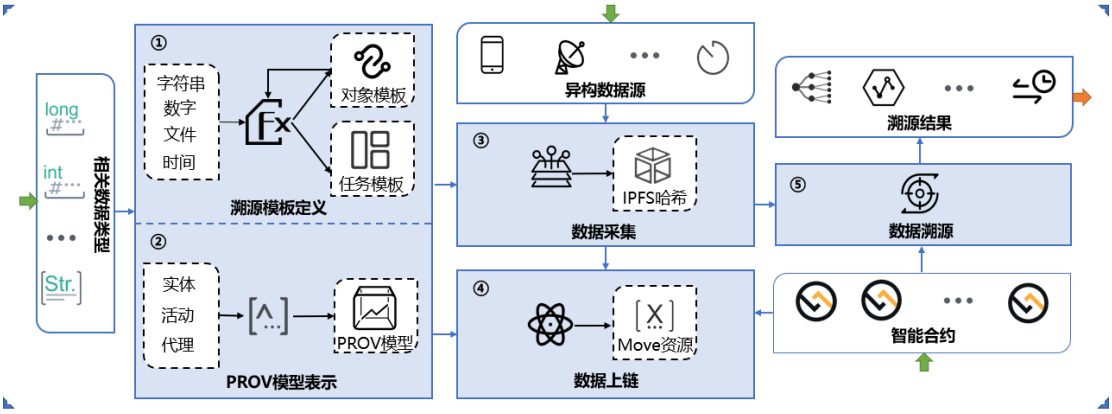

图 3.1: 溯源流程图

3.2 基于 PROV 模型的溯源模型表示

本文在第二章数据溯源模型部分介绍了 PROV 模型核心结构以及结构中实体、活动、代理的重要概念。本节将 PROV 模型中关于实体、活动、模型的概念具体化。具体地，结合溯源模板的定义和结构，下文将详细介绍 PROV 模型中三个概念的实际含义。

实体：在溯源模板中，每个字段是需要被实际记录的数据。本文的实体是指溯源模板中的每个字段。每个实体具有的属性如表 3.1所示。

表 3.1: 实体组成

标识	属性	描述
name	字段名称	当前字段的名称
type	字段类型	当前字段取值支持的数据类型
desc	字段描述	当前字段的补充解释
value	字段取值	当前字段的真实值

活动：溯源模板除描述了字段的基本信息，还定义了不同阶段之间字段的关系。后一阶段内的一个字段可以由前面阶段的若干字段转换而来。这种转换可以是修改、增加、删除等具体活动。本文使用字段之间的依赖关系来统一描述这种转换。因此，在溯源模板中，字段之间的依赖关系代表了 PROV 模型中的活动。每个活动具有的属性如表 3.2所示。

表 3.2: 活动组成

标识	属性	描述
cur_entity	产生实体	本次活动所产生的新的实体
pre_entity	依赖实体	本次活动依赖的实体名称

代理：代理通常是指活动的具体操作者。上述数据采集流程中的数据提供方可理解为 PROV 模型中的代理。本文中代理所具有的属性如表 3.3所示。

表 3.3: 代理组成

标识	属性	描述
agent_name	代理人	代理人的名字也即本文的数据提供方
time	时间	代理操作活动的时间

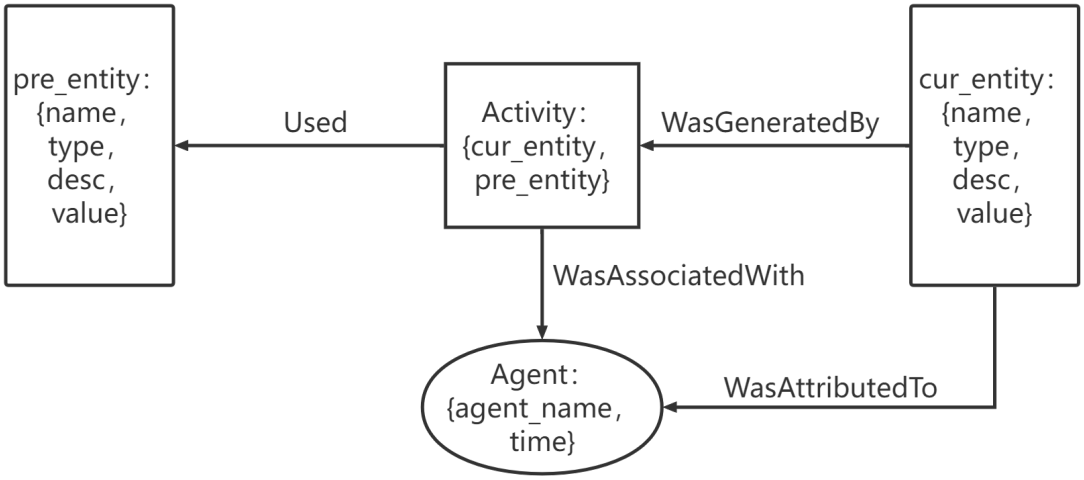

图 3.2: 关系表达实例

图 3.2描述了本文中实体、活动和代理之间的关系。该图表示存在一个代理即数据提供方 Agent 通过活动 Activity 使用前一个实体 pre_entity 生成新的实体 cur_entity。

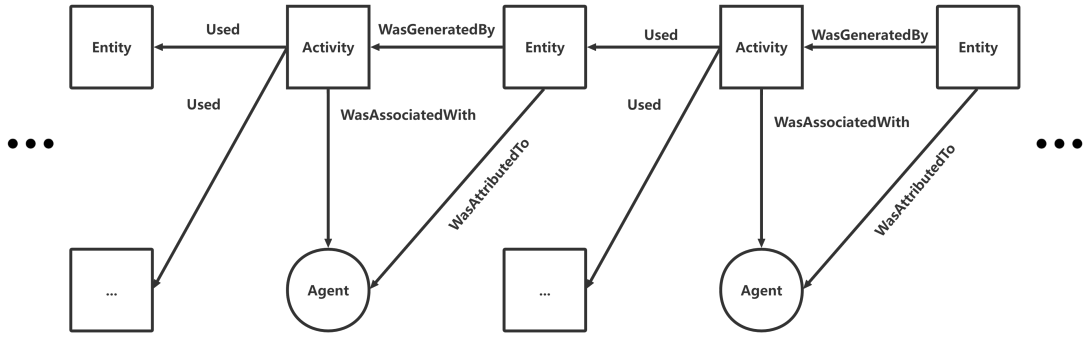

图 3.3: 溯源链

基于上述关系表示，多个活动、实体、代理组成的溯源链如图 3.3所示。一个活动可以使用多个老版本的实体用于生成新版本的实体。新版本的实体在后续流程中又可以作为老版本再次被使用。通过这种方式最终形成一条长链存储于区块链中。

3.3 基于 Diem 的溯源数据存储方案

采集到的异构数据作为溯源原始数据需要存储于区块链中。溯源模板和任务模板定义了实体的属性和实体之间的关系，这种信息同样需要使用区块链存储提升溯源可信性。此外，溯源任务相关信息也将存储于区块链中。以上数据存储于链上后，可进行数据溯源活动。区块链网络中数据的读取和写入都将通过智能合约进行。

由上述可知，区块链网络中需要存储不同类型的数据信息。这些数据信息若未经处理直接进行上链存储，将会有以下两方面弊端。首先，链上存储信息过多，易造成区块链中数据读取和写入效率变慢。其次，复杂类型的数据格式难以进行统一管理上链，编码复杂程度变高。为解决以上问题，本文提出如图 3.4所示的区块链数据存储方案。具体流程描述如下:

步骤一：将待上链数据统一转化为文件并上传至 IPFS 中，IPFS 将生成一个唯一的哈希值。该哈希值代表了该文件，且哈希值全局唯一，一旦文件被改动，哈希值也将改变。

步骤二：获取步骤一得到的 IPFS 哈希值，向区块链网络发起交易提案，各节点共识通过后，调用智能合约完成哈希值的数据上链并返回一个区块链交易发起方地址。

步骤三：当需要获取原始数据时，首先需要根据步骤二中的交易发起方地

址查询到区块链中存储的 IPFS 哈希值。之后，通过 IPFS 哈希值即可得到原始数据。

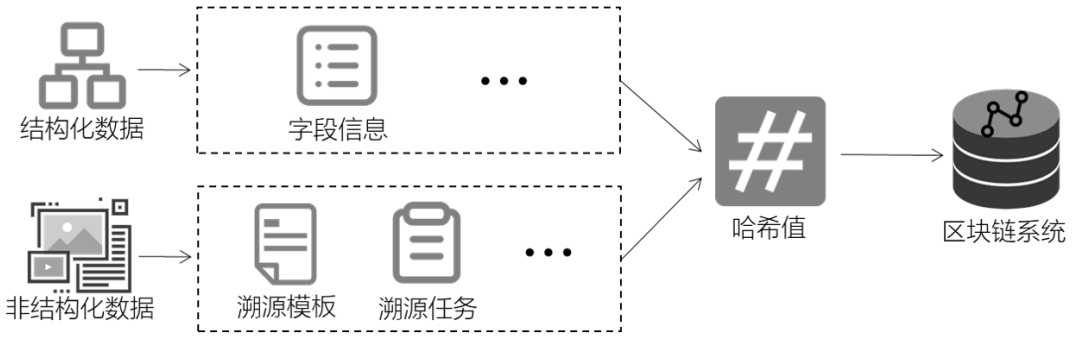

图 3.4: 区块链数据存储方案

如图 3.5所示为 Diem 智能合约代码。代码中定义了资源类型结构体 Counter1。Counter1 中存在一个 u64 的整型数据类型 i。由于 i 的数据类型为 u64 整型，其存储范围为闭区间 $[0,2^{64}\text{-}1]$。IPFS 哈希值长度通常为 66 位十六进制数。本文将 66 位十六进制数以 16 位为单位分割为五组。在每一组中，将 16 位十六进制数转化为十进制，其范围大小同样为 $[0,2^{64}\text{-}1]$。可完全存储于 u64 整型中。因此，本文在同一 Diem 账户地址下，发起五笔交易，调用对应的智能合约将 IPFS 哈希值分五次完成数据上链。

```
address 0x064efe5a7b4340c8a8419f1f40dc71b8{
module Counter1 {
    use 0x1::Signer;
    resource struct Counter1 { i: u64 }
    public fun publish(account: &signer, i: u64) {
      move_to(account, Counter1 { i })
    }
    public fun get_count(addr: address): u64 acquires Counter1 {
        borrow_global<Counter1>(addr).i
    }
    ...}
}
```

图 3.5: Diem 智能合约

3.4 基于有向无环图的数据溯源方法

数据流转过程中，不同数据实体经过多个转换活动可变为新的状态或产生新的数据实体。这种转换过程可以抽象为有向无环图表示。图 3.6左半边描述了一个有向无环图的示例。图中的每一个节点代表了一个数据实体，节点之间的边表示两个数据实体之间存在依赖关系。其中，箭头末尾的节点依赖于箭头开端的节点。图 3.6右半边表示对数据实体 E9 进行数据溯源所得到的结果。数据溯源将原始有向无环图的箭头逆置，并找出与目标溯源数据相关的数据实体节点。当前溯源结果可描述为 E9 依赖于 E7 和 E8，E7 依赖于 E4，E4 由 E1 和 E2 共同加工而来。

基于有向无环图的数据溯源方法可描述为如下过程。

步骤一：从区块链中读取溯源模板信息，以溯源模板为依据获取模板中阶段和字段信息，并建立原始有向无环图。

步骤二：将原始有向无环图箭头方向逆置，以目标溯源数据为起点，采用宽度优先搜索算法找出所有与目标溯源数据相关的节点和边，并将结果记录下来。

步骤三：根据步骤二中的搜索结果，建立新的有向无环图作为溯源结果提供给溯源方。

步骤四：遍历新的有向无环图，并从区块链中读取图中每个节点的原始数据 IPFS 哈希值。

步骤五：根据 IPFS 哈希值逐一获取图中所有节点的原始数据。

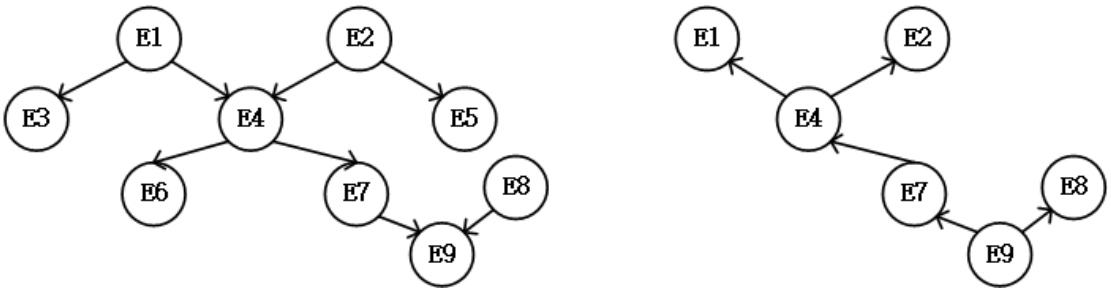

图 3.6: 有向无环图示例

3.5 本章小结

本章首先从宏观上介绍了数据溯源的整体流程。之后基于 PROV 模型描述本文数据溯源中的三个要素实体、代理和活动。紧接着，基于 Diem 区块链，本文描述了原始数据存储至区块链中的过程。最后，介绍了基于有向无环图的数据溯源方法。

第四章　溯源系统需求分析与总体设计

在数据溯源模型与数据溯源方法确定之后，本章对系统进行需求分析与概要设计。首先，对系统整体进行概述。之后，分别从用例分析、功能性需求分析、非功能性需求分析三个角度整体阐述系统需求。其次，完成系统架构设计，划分系统功能模块，阐述系统 4+1 视图。最后，设计系统的持久化模型。

4.1　系统概述

为解决传统数据溯源数据存储不安全，数据溯源场景单一固定的问题。本文设计了数据溯源模型和方法，并根据系统需求分析实现基于 PROV 模型的的 Diem 安全溯源系统。本系统的涉众主要有平台方、数据提供方、溯源需求方三种角色。本文通过设计智能合约实现溯源数据的链上存储和查询，保障溯源数据的可信性。并通过溯源模板机制，一定程度上解决了数据溯源场景单一的缺陷。系统采用 B/S 架构，自上而下分为前端展示层、溯源服务层和区块链服务层。系统将实现以下目标：实现溯源模板的新建、删除、修改、上链功能；实现溯源数据采集、溯源数据分字段上链的溯源数据管理功能。实现溯源任务创建、溯源任务删除、溯源任务查看的溯源任务管理功能；之后在溯源任务查看模块中进行追踪溯源、生成溯源关系图并展示溯源信息。

4.2　需求分析

4.2.1　系统涉众分析

本系统涉众主要有溯源平台方、数据提供方、溯源需求方。详细描述如下：

溯源平台方：平台方拥有较高权限，是系统溯源模板的提供者和管理者。溯源模板来自于不同场景下的数据溯源需求。平台提供或创建对应场景下的数据溯源模板。此外，平台方还可参与溯源数据的管理以及溯源任务的管理。

数据提供方：数据提供方是系统原始数据的提供者，数据提供方主要负责数据采集和数据上链。在数据提供方完成数据采集之后，系统会提示用户进行数据上链操作。

溯源需求方：溯源需求方是系统的主要服务对象。溯源需求方为实现数据溯源需求需要创建溯源任务，完成溯源任务创建中的各个步骤。最后，系统将提供给用户对于目标数据进行溯源的溯源结果。

4.2.2 用例分析

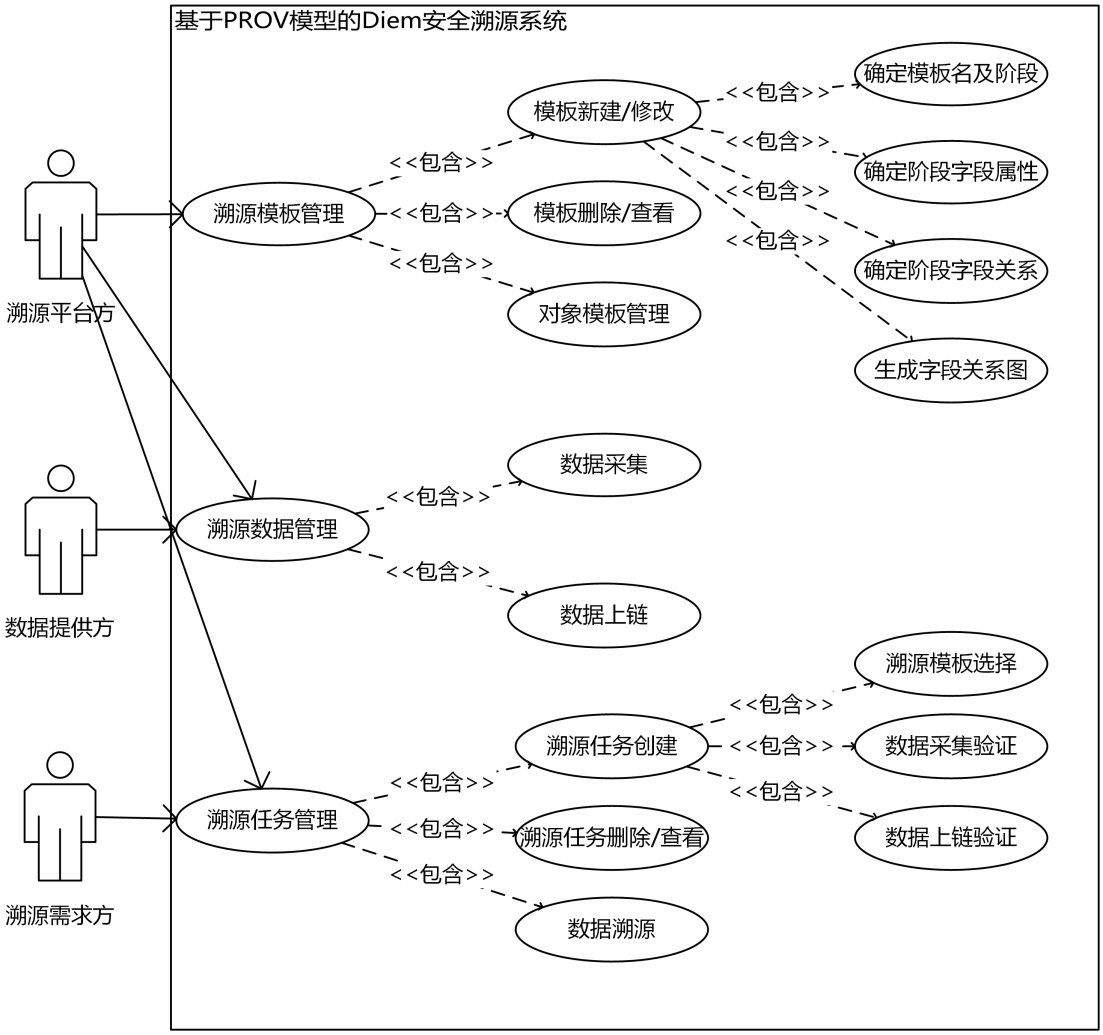

图 4.1: 系统用例图

通过对系统进行用例分析，得到如图 4.1所示系统用例图。系统主要包括溯源模板管理、溯源数据管理以及溯源任务管理三个用例。其中，溯源模板管理包括溯源模板新建/修改，溯源模板删除、溯源模板查看以及对象模板管理功能；溯源模板新建/修改共经历确定模板名及阶段、确定阶段字段属性、确定阶段字段关系以及生成字段关系图四个步骤。溯源数据管理模块包括数据采集和数据上链两个用例。溯源任务管理包括溯源任务创建、溯源任务查看以及溯源任务删除功能。在溯源任务创建模块中包含溯源模板选择、数据采集验证、数据上链验证三个阶段，在溯源任务查看中可进行数据溯源。

在溯源模板管理模块中，主要分为任务模板新建/修改、任务模板删除以及任务模板搜索/查看三个板块。每一个任务模板可包含多个阶段，同时每个阶段也可包含多个字段，在字段间存在前驱依赖关系。所谓前驱依赖关系是指下一阶段的某一个字段可以选择上级阶段的一个或多个字段作为其前驱依赖。当用户点击至某个板块中时，就能看到系统为该板块提供的功能任务。在任务模板新建/修改模块，用户点击该模块后，共需经历确定模板名及阶段、确定阶段字段属性、确定阶段字段关系以及生成字段关系图四个步骤。确定模板名及阶段需要用户输入任务模板的名称、阶段个数以及各阶段的名称；确定阶段字段属性需要用户在每个阶段均指定当前阶段所包含的字段及字段属性，字段属性包含字段名称、字段类型以及字段描述，其中字段类型支持数字、字符串、文件以及自定义的对象模板。确定阶段字段关系需要用户指定各个阶段的不同字段的前驱依赖。生成字段关系图是指在用户完成前三个步骤后，系统将根据用户的输入，自动生成一个用于描述模板中所有字段关系的树状图。各板块之间相互关联，当用户新建任务模板后，任务模板的搜索、查看、修改、删除和下载等功能才能正常使用。溯源模板管理模块用例描述如表 4.1所示。

表 4.1: 溯源模板管理用例描述

用例 ID	UC1
用例名称	溯源模板管理
参与者	普通用户
前置条件	用户注册并成功登录系统
后置条件	保存至系统服务器等待用户后续操作
正常流程	1. 用户注册并登录溯源系统； 2. 用户选择选择溯源模板管理功能； 3. 用户进行任务模板的新建/修改； 4. 用户搜索/查看/下载/删除已创建模板。
可选流程	3.a 确定模板名称及阶段。 3.b 确定每个阶段字段属性。 3.c 确定阶段字段关系及关系。 5.a 根据任务模板名称搜索任务模板。 5.b 根据任务模板创建日期搜索任务模板。 5.c 根据任务模板使用的对象模板搜索任务模板。
优先级	高

表 4.2所示为溯源任务创建用例描述。溯源任务创建主要分为溯源任务模板选择、溯源数据采集、溯源数据上链三个模块。在溯源任务模板选择中，用户需要确认本次溯源任务名称以及选择创建完成的任务模板。在溯源数据采集模块中，依赖于所选择的任务模板，在任务模板的所有阶段均需要进行溯源数据的采集，完成溯源数据采集功能后方可进入下一阶段。在溯源数据上链模块，溯源数据采集模块中采集到的每一个真实数据均需要依次进行上链，保证数据的真实性，便于后续的数据溯源。

表 4.2: 溯源任务创建用例描述

用例 ID	UC2
用例名称	溯源任务创建
参与者	普通用户
前置条件	用户注册并成功登录系统，且用户完成任务模板的创建/修改
后置条件	保存至系统服务器等待用户后续操作
正常流程	1. 注册并登录溯源系统； 2. 完成创建/修改任务模板； 3. 确定溯源任务名并选择任务模板； 4. 验证完成所有字段的数据采集； 5. 验证采集完成后所有数据是否上链。
可选流程	3.a 用户输入非重复的溯源任务名称。 3.b 用户选择已创建的任务模板。
优先级	高

数据采集用例描述如表 4.3所示。数据采集的目的是采集用户溯源数据，数据格式支持数字、字符串、时间、文件形式。溯源数据采集以列表的形式展示阶段内字段内容，包含字段名称、字段类型、字段取值、交易发起方地址以及字段详情链接；点击字段详情可查看字段名称、字段类型、字段描述以及字段依赖信息；填写过程会有进度条提示填写进度。

表 4.3: 数据采集用例描述

用例 ID	UC3
用例名称	数据采集
参与者	普通用户
前置条件	用户注册并成功登录系统，且完成溯源任务模板选择
后置条件	等待数据上链
正常流程	1. 根据任务模板选择阶段； 2. 阶段内逐个字段进行数据采集。
可选流程	2.a 用户点击字段详情查看字段详细内容。 2.b 用户查看采集进度。
优先级	高

表 4.4所示为数据上链用例描述。数据上链的前提是用户已经完成溯源数据的采集。当溯源数据采集完成后，用户点击数据上链按钮，系统将在后台完成数据上链的过程并显示上链等待动画。上链完成之后，用户会收到上链是否成功的提示信息。若上链成功，用户则可在界面中查看看当前字段上链的交易发起方地址；若上链失败，则提示用户进行再次数据上链操作。

表 4.4: 数据上链用例描述

用例 ID	UC4
用例名称	数据上链
参与者	普通用户
前置条件	用户注册并成功登录系统，且完成溯源数据采集
后置条件	将数据保存至区块链中，等待用户后续操作
正常流程	1. 阶段内逐个字段进行数据采集并点击数据上链； 2. 等待上链完成提示并查看交易发起方地址。
可选流程	2.a 查看上链完成提示信息。 2.b 上链提示成功时，可在界面查看交易发起方地址。
优先级	高

表 4.5描述了数据溯源用例的相关信息。在用户注册并成功登录系统并完成溯源任务的配置后，即可进行数据溯源的操作。用户选择溯源任务管理模块查找需要溯源的溯源任务，查找过程支持根据溯源任务名，任务模板名称以及日

期进行搜索。点击所查找的溯源任务后可以看到该溯源任务所具有的全部阶段以及全部的字段。选择对应的字段，点击数据溯源即可对该字段数据进行溯源。此外，用户也可查看该字段的字段详情信息以及该字段数据在上链时的交易信息。数据溯源的结果以树状图的形式表示。在树状图中，两个节点之间存在连线即代表两个数据存在关联。父节点是子节点的数据来源，根据树状图中的箭头即可从子节点到父节点查找到该数据的最初来源。树状图中的节点代表着某一字段的链上真实数据。点击树状图中的节点，可以看到该字段数据在上链时的交易发起方地址。此外，用户也可点击“真实数据”按钮，查看该字段在区块链上的真实数据，防止数据遭到破坏。

表 4.5: 数据溯源用例描述

用例 ID	UC5
用例名称	数据溯源
参与者	普通用户
前置条件	用户注册并成功登录系统，且完成溯源任务创建
后置条件	等待用户后续操作
正常流程	1. 用户点击溯源任务管理查找需要溯源的溯源任务； 2. 用户选择溯源任务并点击“任务详情”按钮； 3. 用户选择溯源任务的某一阶段中的某一字段； 4. 用户点击“数据溯源”按钮进行数据溯源； 5. 查看树状图形式的溯源结果，查看数据的来龙去脉； 6. 点击“真实数据”按钮，查看链上真实数据。
可选流程	1.a 可根据任务名、模板名及日期搜索溯源任务。 3.a 用户可点击“字段详情”按钮查看字段的详细信息。 3.b 用户可查看该字段数据上链的交易信息。 5.a 用户点击树状图节点可查看交易发起方地址。 6.a 对于文件类型数据，用户可点击进行下载。
优先级	高

4.2.3 功能性需求分析

Diem 安全溯源系统主要包括对象模板管理、溯源模板管理、溯源任务管理、数据溯源、Diem 服务五个功能性需求。下面将详细介绍每种功能性需求。对象模板管理。面对复杂多变的真实使用场景，为提升溯源系统的通用性，采用对

象模板的方式对系统进行扩展。对象模板类似于编程语言中类的概念，一个对象模板可以由多个属性组成，属性包含属性名称、属性类型、属性描述，其中，属性类型可以是数字、日期、字符串、文件中的一种。通过这种设计，系统所能支持的待溯源字段类型更加丰富。对象模板管理支持对象模板的创建、修改、删除、搜索和详情查看。

溯源模板管理。任务模板是对溯源任务的抽象，它描述和定义了同类型溯源任务中可能涉及到的多个阶段、各种字段以及字段之间的关联。溯源模板管理支持任务模板的创建、修改、删除、搜索和详情查看。任务模板中的字段由字段名称、字段类型、字段描述、前驱关系组成。其中，字段类型可以是已经定义的对象模板、时间、数字、字符串；前驱关系表示该字段的前驱字段由哪几个字段组成。创建完成的任务模板即可在后续的溯源任务中被使用。通过对象模板和任务模板的机制，系统的通用性和可扩展性得到进一步的提高。

溯源任务管理。在完成对象模板和任务模板的创建后才可创建溯源任务。在溯源任务中主要完成数据采集和数据上链的功能。以任务模板为参照，用户分阶段逐字段完成数据采集和上链，数据采集支持数字、字符串、时间、文件、对象模板类型。数据采集和数据上链是系统进行数据溯源的前提。

数据溯源。系统中的一次溯源任务在完成数据采集和上链后即可进行数据溯源。数据溯源是对原始数据和链上交易信息的追溯，原始数据在采集完成之后将数据摘要存储于 Diem 区块链中保证真实数据不会遭受恶意破坏。在区块链中，Diem 维护了多个 <K,V> 对，其中 K 为账户地址，V 则是当前账户保存在区块链中的数据。通过这样的机制，在数据溯源时，利用 K 值即可取出存储于区块链中的数据，完成链上数据的获取。一次数据溯源操作，系统会将当前字段的原始数据和树状关系图一并提供给用户。此外，针对当前字段，系统也将返回用户该字段的上链时产生的交易信息。

Diem 服务。区块链的使用保证了数据的真实性和安全性，系统业务逻辑和区块链之间的通信与交互则是系统实现的关键。通过 Diem 服务将区块链与系统业务逻辑之间的交互进行抽象，提供如账户创建、交易查询，交易提交、链上数据获取、数据上链等常用接口。Diem 服务屏蔽了区块链与系统逻辑层之间的差异，简化了与区块链相关的操作。

4.2.4 非功能性需求分析

本系统的主要目标是基于 Diem 区块链平台提供具有一定通用性的区块链数据溯源系统以满足不同场景下的数据溯源需求。因此，系统在性能、易用性、可扩展性、可维护性方面存在以下需求。

(1) 性能

为提升系统使用的友好性，系统在不涉及区块链数据读写的功能模块响应时间应小于 1 秒。在涉及区块链数据读写功能模块如数据上链，数据溯源模块，响应时间应小于 5 分钟。

(2) 易用性

在完成系统前端页面设计时，对于相同或类似的功能，应使其在页面上相对集中，减少用户移动鼠标的距离。同时，对于每个页面应具有响应的标题和合理的功能模块划分，做到界面整齐美观。

(3) 可扩展性

系统基于 Diem 区块链平台实现数据溯源，同时 Diem 区块链作为较新的一种区块链平台也在不断发展，支持更多新的特性。为支持 Diem 区块链的更新以及系统功能的扩展，系统在设计时应具有一定的可扩展性，减少再次开发的难度，做到增量开发。

(4) 可维护性

系统开发完成后的维护往往需要花费较大的人力和物力，为减少后期系统的维护成本，开发时应使系统具有一定的可维护性。比如，使用相对成熟先进的开发工具，使用面向对象，模块化的软件设计思想。

(5) 可靠性

系统应满足不能频繁出现宕机、重启的可靠性要求。在满足使用要求的前提下，尽量使用成熟的技术，提高系统的可靠性。

(6) 可移植性

系统采用 B/S 架构，服务端使用 Ubuntu 平台，浏览器端应支持在 Chrome 96.0.4664.45、Edge 96.0.1054.53、Firefox90.0 版本的主流浏览器上运行。

4.3 总体设计

本节将围绕系统架构和 4+1 视图两方面对系统总体设计进行介绍。在系统架构上，围绕系统架构图说明系统整体设计及各模块之间存在的关联。之后，全面介绍系统逻辑视图、系统开发视图、系统进程视图以及系统物理视图。

4.3.1 系统架构

图 4.2: 系统架构图

图 4.2所示为系统架构图。该系统为 B/S 架构下的应用，采用 MVC 的设计模式，自上而下将系统分为前端、服务端、数据存储三个层次。相邻层之间，下层为上层提供服务，上层调用下层的服务。系统在数据存储层搭建 Diem 本地链、使用 IPFS 星际文件系统、MySQL 数据库作为存储数据实体的载体，在服务层采用 SpringBoot 框架实现系统内部复杂的业务逻辑和多种服务、在前端展示层使用 Vue 框架和 Axios 完成数据展示和系统交互。

系统前端主要由三大界面组成。分别是对象模板管理界面、溯源模板管理界面以及溯源任务管理界面。其中对象模板管理界面主要包含对象模板新建、对象模板修改、对象模板详情查看以及对象模板删除。在溯源模板管理界面，用户需要逐个确定每个阶段的字段属性以及字段之间的前驱关系并由此生成字段关系图。在溯源任务管理界面，用户需要首先进行数据采集，采集完成之后逐字段进行数据上链。待数据采集和数据上链完成后，用户可进行数据溯源。系统前端采用 Vue 框架实现并基于 Axios 组件进行前后端异步通信，并使用模块化和组件化的思想设计并实现前端代码，便于后续代码的维护和修改，提升系统的可维护性。

系统服务端基于 SpringBoot 实现，主要包括对象模板服务、任务模板服务、Diem 服务、溯源任务服务以及数据溯源服务。对象模板服务与任务模板服务的实现类似，各自负责对象模板和任务模板的新建、修改、详情以及删除功能的实现。溯源任务服务是系统核心服务之一，包含溯源任务创建和溯源任务详情

模块，溯源任务创建负责选择任务模板，并基于任务模板收集用户在平台中输入的真实数据，之后通过 Diem 服务将数据摘要存储于区块链中，等待后续数据溯源的相关操作；溯源任务详情负责提供当前溯源任务的任务名称、使用模板、时间等基本信息并提供数据溯源的操作入口。数据溯源服务在溯源任务新建完成之后即可使用，数据溯源包括字段关系图展示以及原始数据获取两大功能点，字段关系图提供当前溯源字段的所有前驱依赖节点和关系，原始数据获取通过区块链以及 IPFS 将原始数据取出，保障溯源数据的真实性和可靠性。Diem 服务负责区块链相关的操作，具体如数据上链、数据下载、区块链账户创建、区块链交易查询、账户查询等操作，其中数据下载与数据上链对应区块链数据的读写；区块链账户创建与查询负责区块链对账户的基本管理；数据上链触发区块链交易信息的产生，交易信息包括发送方、交易序号、接收方、交易哈希、最大 Gas 数、Gas 价格、脚本类型等各类型数据，区块链交易查询则负责搜索链上所发生的全部交易信息。

系统数据存储层是系统数据实体的存储介质。系统中的对象模板信息、任务模板信息、溯源任务信息以及数据上链时产生的交易信息存储于 MySql 数据库中。在数据上链时，考虑大文件比较消耗存储空间，所以将文件先上传到本地 IPFS 系统得到 IPFS_HASH，然后把得到的 IPFS_HASH 保存至 Diem 本地链上进行持久化存储，防止原始数据遭到破坏。在数据溯源需要时，则根据链上存储的 IPFS_HASH 将真实数据取出，满足数据溯源的需求。在实际的溯源任务中，当前待溯源字段有可能存在阶段间的依赖关系，因此，在具体溯源某一字段时，系统将以树状图的方式描述与当前待溯源字段存在前驱关系的所有字段。同时，系统也将提供获取这些字段链上真实数据的方式。系统整体使用 Docker 进行部署，便于系统的管理与扩展。

4.3.2 4+1 视图

为准确描述软件架构，本文结合“4+1 视图”从场景视图、逻辑视图、开发视图、进程视图以及物理视图五个角度对系统设计进行讨论。其中，场景视图从用户角度出发描述系统设计，与图 4.1系统用例图作用相同。本节将从逻辑视图、开发视图、进程视图以及物理视图四个角度进行描述。

(1) 逻辑视图

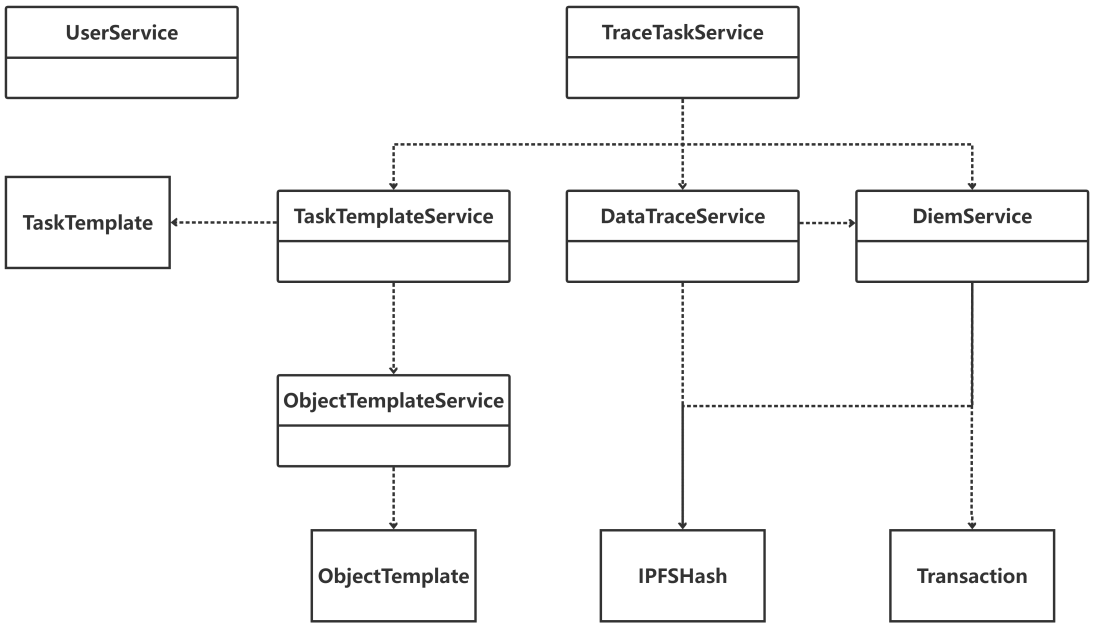

图 4.3: 逻辑视图

图 4.3所示为系统逻辑视图。逻辑视图描述系统的功能性需求，包含了系统中关键的对象模型。系统中的数据实体包含对象模板 (ObjectTemplate)、任务模板 (TaskTemplate)、区块链交易 (Transaction)、哈希值 (IPFSHash) 等。系统中的基础服务包含用户服务 (UserService)、溯源任务服务 (TraceTaskService)、任务模板服务 (TaskTemplateService)、数据溯源服务 (DataTraceService)、Diem 服务 (DiemService)、对象模板服务 (ObjectTemplateService) 等。UserService 负责用户的注册及登录。ObjectTemplateService 负责对象模板的新建、删除、查看以及修改。TaskTemplateService 负责任务模板的新建、删除、查看、以及修改，TaskTemplateService 也可调用 ObjectTemplateService 服务使用已经创建的对象模板。TraceTaskService 通过调用 TaskTemplateService 服务使用已经创建完成的任务模板完成溯源任务创建，配置完成后可查看溯源任务的详情。DataTraceService 负责完成系统的核心服务数据溯源，数据溯源包含字段关系图的展示以及原始数据的获取。DiemService 提供区块链相关的数据上链、数据下载、账户创建、交易查询等功能。

(2) 开发视图

系统开发视图如图 4.4所示。开发视图是从开发者角度出发所看到的系统设计与架构，是系统开发人员的基础视图。系统分为展示层、业务逻辑层与数据存储层并采用前后端分离架构和 MVC 设计模式实现，下面结合开发视图进行详细

阐述。

展示层是用户与系统的桥梁，用户操作系统前端界面完成系统使用需求，服务端为前端提供业务服务。前端代码使用 Vue 框架实现，依赖 Ant-Design 前端组件。前端代码主要由 assets、components、router、pages、config 包组成。assets 存放图片等系统静态资源；components 存放前端公共组件，便于实现代码复用；router 存放前端路由文件，用于设置前端路由；pages 则是前端页面的具体代码实现，每个文件通常代表前端的一个页面；config 用于保存前端的一些配置。

业务逻辑层使用 SpringBoot 框架开发，主要由 Controller、Service、Dao、Util 以及 Model 五个包组成。Controller 是控制器，用于调用 Service 层服务，响应前端页面发送的 http 请求，是前后端数据传输的桥梁。其中，ObjectTemplateController 和 TaskTemplateController 分别接收对象模板和任务模板的相关请求，TaskController 负责溯源任务相关服务，DiemController 则用于调用区块链相关服务。Service 包是系统业务逻辑的核心，包含 ObjectTemplateService、TaskTemplateService、TaskService、DiemService 以及 DataTraceService。Service 包可调用 Util、Model 以及 Dao 包，是对 Dao 中数据的封装加工，主要用于具体业务逻辑的实现。Dao 实现对 Model 包中相关实体的增、删、改、查。Model 包存放了系统的对象实体，与数据存储层紧密相关。Util 包是封装了系统的常用工具类，是对代码的进一步抽象，实现代码复用。

数据存储层包含 JDBC 与 Diem，分别负责与传统数据库和区块链平台的交互，数据存储层封装了与数据库和区块链的交互接口，屏蔽了系统与存储平台之间的差异，提高数据库以及区块链操作的效率。

(3) 进程视图

进程视图用于描述系统各进程之间的关联和时序关系。系统进程视图如图 4.5所示。系统从主线程输入模板信息创建对象/任务模板开始。之后，主线程再次调用溯源任务服务完成溯源任务中的数据采集流程，采集完成之后溯源任务调用区块链服务经共识后完成数据上链并将数据上链时产生的交易信息返回给溯源任务进程。在完成模板创建、数据采集、数据上链之后，主线程调用数据溯源线程开始数据溯源，数据溯源进程针对待溯源字段调用区块链服务获取链上数据、生成字段关系图、解析原始真实数据、查询相关交易信息，最终将溯源信息一并返回给主线程。

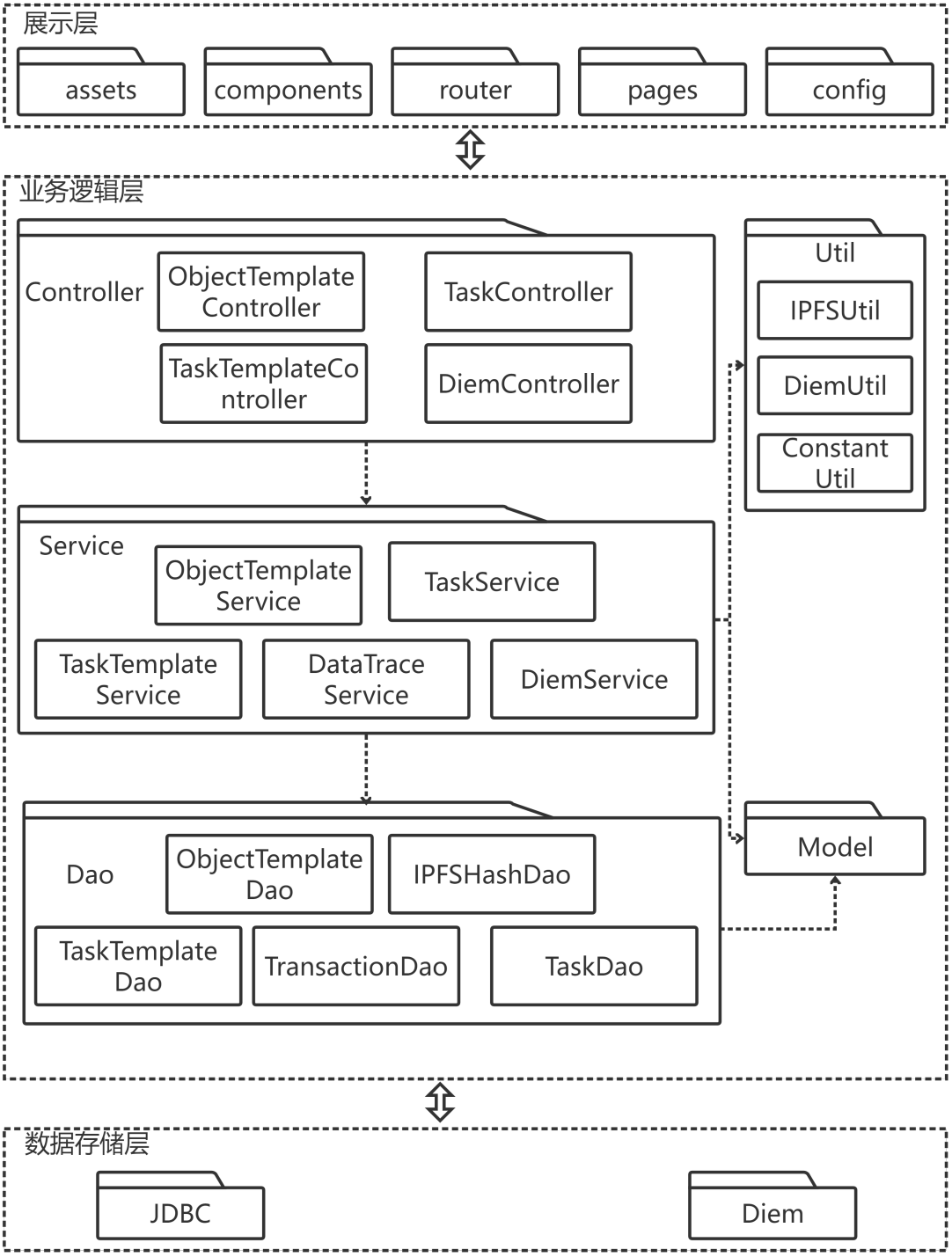
展示层
assets
components
router
pages
config
业务逻辑层
Controller
ObjectTemplate Controller
TaskController
TaskTemplateCo ntroller
DiemController
Util
IPFSUtil
DiemUtil
Constant Util
Service
ObjectTemplate Service
TaskService
TaskTemplate Service
DataTrace Service
DiemService
Dao
ObjectTemplate Dao
IPFSHashDao
TaskTemplate Dao
TransactionDao
TaskDao
Model
数据存储层
JDBC
Diem

图 4.4: 开发视图

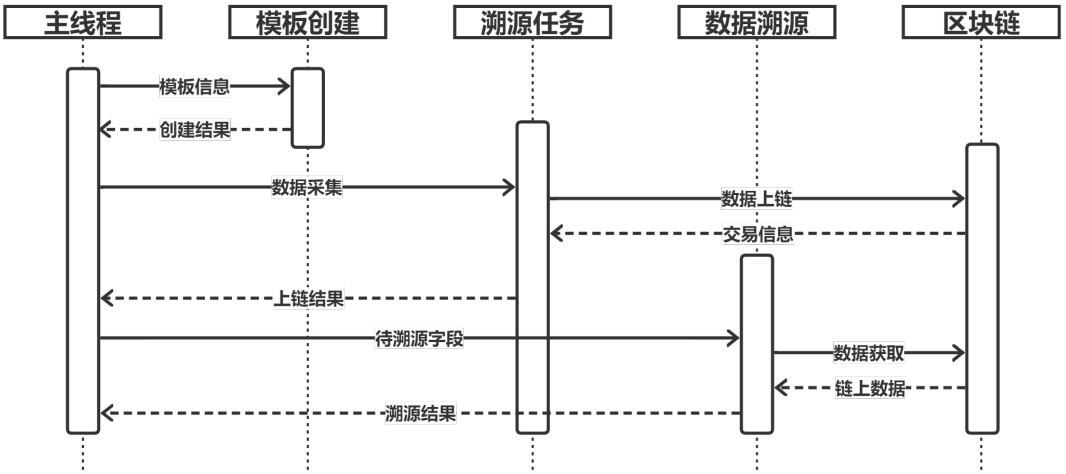

图 4.5: 进程视图

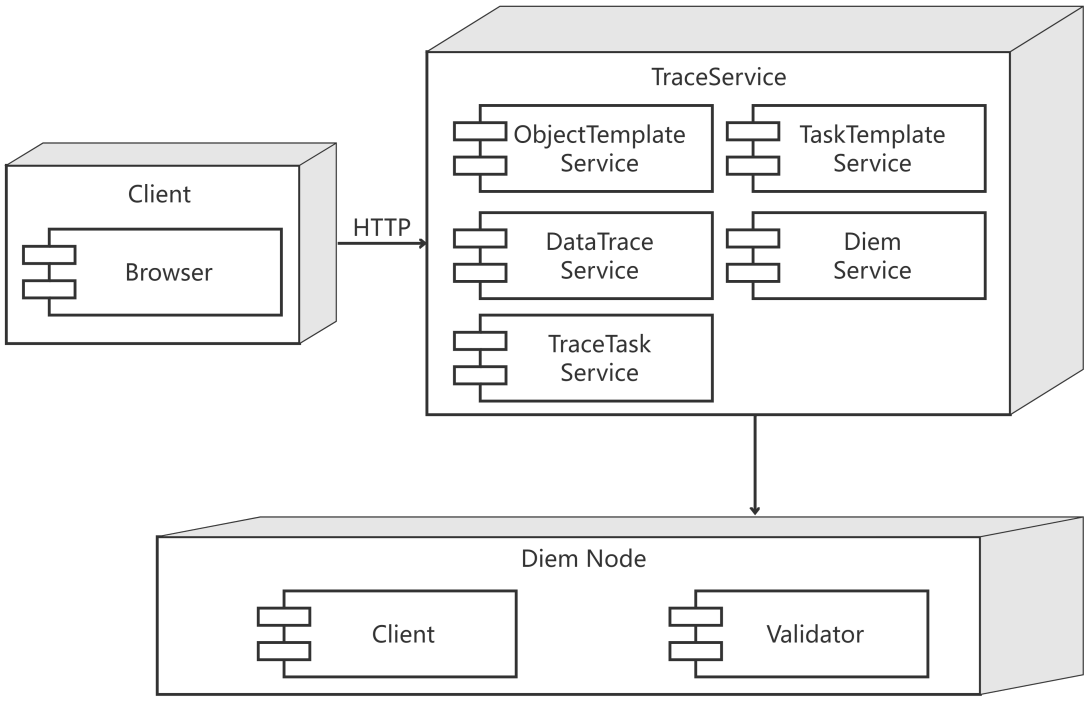

图 4.6: 物理视图

(4) 物理视图

图 4.6所示为系统物理视图。物理视图描述了系统使用软硬件资源的具体情况，是从部署运维人员角度出发看到的系统全貌。用户在客户端使用浏览器访问系统主页，通过 Http 请求访问溯源系统服务端。溯源系统服务端的对象模板服务 ObjectTemplateService、数据溯源服务 DataTraceService、任务模板服务 TaskTemplateService、Diem 服务 DiemService、溯源任务服务 TraceTaskService 分别响应来自系统前端的不同请求。Diem 基础平台以串行化的方式与溯源系统服务端建立通信完成数据交互，Diem 基础平台包含 Client 和 Validator 两种节点，其中，Client 用于提交查询交易，Validator 负责交易执行和区块链账本的维护。

4.3.3 持久化模型设计

为满足系统设计需要，系统共设计七种持久化数据模型。分别是账户信息数据、对象模板数据、溯源任务数据、任务模板数据、交易信息数据、用户数据以及 DiemIPFS 数据。下面将详细介绍每种持久化数据模型。

账户信息属性表记录了 Diem 账户的信息，账户信息包括账户索引、账户地址、账户余额、账户交易序列号。记录账户信息表是为了追踪系统中 Diem 账户的改变。表 4.6列举了账户信息属性表在数据库中的具体设计。

表 4.6: 账户信息属性表

字段	含义	类型
IndexNum	账户索引	long
Address	账户地址	string
Amount	账户余额	long
SequenceNumber	账户交易序列号	long

对象模板属性表记录了系统内创建的所有的对象模板，对象模板的信息包括用户名、对象模板名、对象模板结构数据、创建时间、对象模板概述及哈希值。表 4.7列举了对象模板属性表在数据库中的具体设计。

表 4.7: 对象模板属性表

字段	含义	类型
UserName	用户名	string
ObjectTemplateName	对象模板名	string
DataSource	对象模板结构数据	string
DataTime	创建时间	string
Desc	对象模板概述	string
Hash	哈希值	string

任务属性表记录了系统内创建的所有的溯源任务，任务的信息包括用户名、溯源任务名、任务模板名、阶段名、字段名、哈希值、创建时间及类型。表 4.8列举了任务属性表在数据库中的具体设计。

表 4.8: 溯源任务属性表

字段	含义	类型
UserName	用户名	string
TaskName	溯源任务名	string
TaskTemplateName	任务模板名	string
PeriodName	阶段名	string
FieldName	字段名	string
Hash	哈希值	string
Date	创建时间	string
Type	类型	string

任务模板属性表记录了系统内创建的所有的溯源任务模板，任务模板的信息包括用户名、任务模板名、任务模板结构数据、创建时间、任务模板概述及哈希值。表 4.9列举了任务模板属性表在数据库中的具体设计。

表 4.9: 任务模板属性表

字段	含义	类型
UserName	用户名	string
TaskTemplateName	任务模板名	string
DataSource	任务模板结构数据	string
DataTime	创建时间	string
Desc	任务模板概述	string
Hash	哈希值	string

交易信息属性表记录了系统内所有交易行为的交易信息，交易信息包括发送方、交易序号、接收方、交易哈希、最大 Gas 数、Gas 价格、脚本类型等数据。表 4.10列举了交易信息表在数据库中的具体设计。

表 4.10: 交易信息属性表

字段	含义	类型
sender	发送方	string
signature	签名	string
SequenceNumber	交易序列号	long
MaxGasAmount	最大 Gas 数	long
GasUnitPrice	Gas 价格	string
ScriptType	脚本类型	string
Receiver	接收方	string
TxHash	交易哈希	string

用户属性表记录了用户名、密码、电话、真实名称、地址和描述。用户属性表服务于用户的注册和登录活动，保障用户系统使用的安全性。表 4.11列举了用户表在数据库中的具体设计。

表 4.11: 用户属性表

字段	含义	类型
UserName	用户名	string
Password	密码	string
phone	电话	string
Name	真实名称	string
Address	地址	string
Desc	描述	string

DiemIPFS 属性表记录了区块链账户地址到 IPFS_Hash 的映射。其中 Hash_Diem 是区块链账户地址，Hash_IPFS 是 IPFS 的哈希值。建立本表的目的是记录账户地址和 IPFS 哈希值之间的映射关系，便于完成数据溯源服务中原始数据的获取。表 4.12列举了 DiemIPFS 属性表在数据库中的具体设计。

表 4.12: DiemIPFS 属性表

字段	含义	类型
Hash_Diem	区块链账户地址	string
Hash_IPFS	IPFS 哈希	string

4.4 本章小结

本章对系统进行了完整的需求分析和总体设计。首先，从全局角度给出系统概述。之后，分析了系统中所涉及到的关键角色，从用例图出发，详细描述了系统涉及的主要用例。然后，以需求分析结果为基础，进行了系统架构设计和持久化模型设计，并结合 4+1 视图描述系统全貌。

第五章　溯源系统详细设计与实现

本章主要介绍系统的详细设计与实现。本文将系统分为溯源模板管理、数据采集、数据上链、溯源任务管理以及数据溯源五个模块。针对系统的各个模块，分别从时序图、关键类图、关键代码三个角度描述模块功能的实现。

5.1　溯源模板管理模块

5.1.1　溯源模板设计与实现

溯源模板模块主要包括溯源模板的新建、删除、查找以及修改功能。其中，溯源模板的删除和查找功能与对象模板管理模块类似。溯源模板的新建/修改功能相较于对象模板的新建/修改较为复杂。溯源模板的新建/修改包含模板阶段信息填写、阶段间字段填写、模板关系图查看、模板创建与上链四个步骤。其中，模板阶段信息填写的目的是使用户在新建和修改溯源模板时，对阶段信息进行填写和修改。其中，阶段信息包括模板名称、阶段个数、各阶段名称和模板概述。阶段间字段填写提供各阶段字段的新增删除，字段类型支持字符串、数字、文件、对象模板、时间五种类型。在模板关系图查看步骤中，用户在新建或修改溯源模板字段后，可生成该溯源模板的字段间关系图。模板创建与上链使用户创建溯源模板，保存创建结果，并实现溯源模板的上链操作，通过单独页面向用户反馈创建与上链完成提示。为用户提供了两个接口：溯源模板新建，溯源模板库管理。溯源模板新建接口目的为使用户继续新建其他溯源模板；溯源模板库管理接口目的为使用户管理该用户创建的所有溯源模板。其中，提示的内容包括创建结果、上链结果及溯源任务模板概述。

溯源模板管理模块核心类图如图 5.1所示。TaskTemplateController 是控制器，负责根据前端请求返回不同的请求结果。控制器调用 TaskTemplateService 实现溯源模板相关的创建、修改、查看以及上链操作。控制器依赖于 ObjectTemplateMapper 实现对象模板的嵌套，提高模板的通用性。FileService 用于实现文件相关的服务，如溯源模板文件下载。TaskTemplateServiceImpl 是 TaskTemplateService 的实现类。TaskTemplateServiceImpl 通过 TaskTemplateMapper 保存溯源模板，通过 ObjectTemplateService 获取对象模板相关服务。此外，TaskTemplateServiceImpl 实现类调用 DiemService 实现溯源模板上链并通过 TXMapper 保存上链时发生的交易信息。

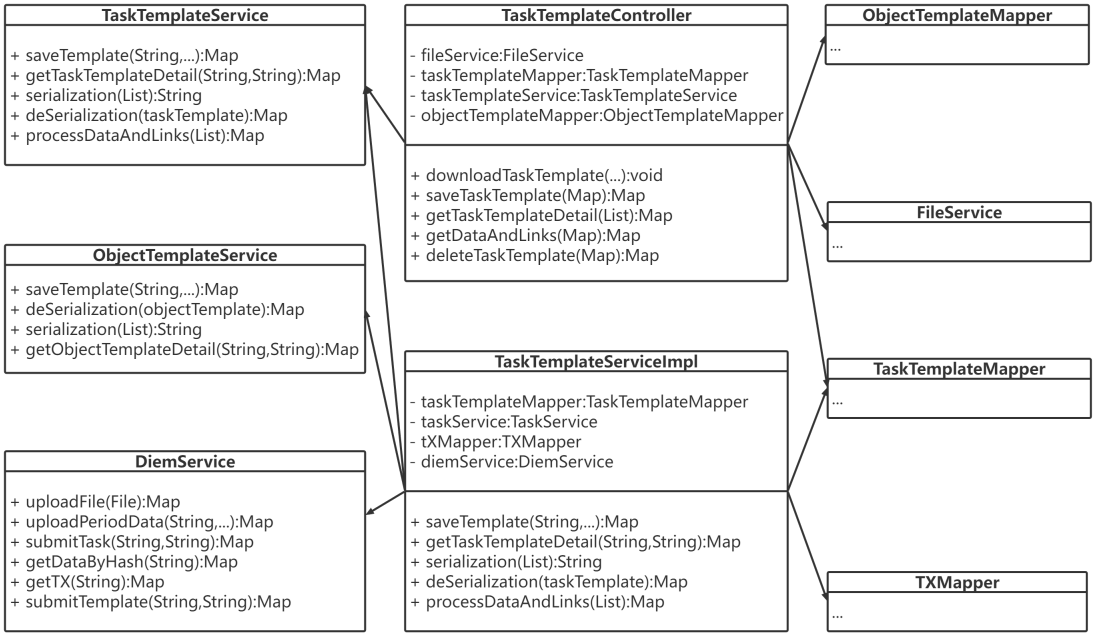

图 5.1: 任务模板类图

图 5.2展示了溯源模板管理模块中模板上链的调用顺序。TaskTemplateController 首先向 TaskTemplateService 发出 saveTaskTemplate() 调用信息。之后溯源模板服务调用对象模板服务查看是否需要保存对象模板。若溯源模板中不含有对象模板，则溯源模板直接向 DiemService 中的 upload() 方法发起调用。否则需要先完成对象模板的上链，再完成溯源模板的上链。最后，将溯源模板的上链信息如上链状态，交易信息等进行封装一并返回给溯源模板控制器。

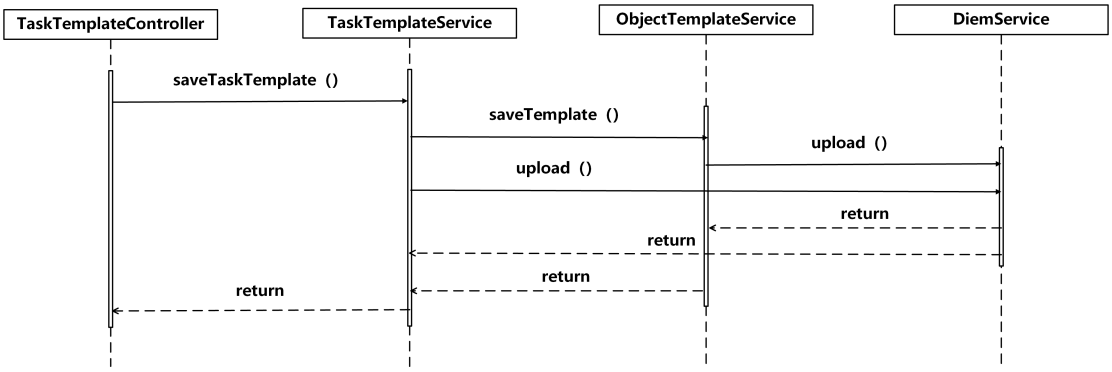

图 5.2: 溯源模板时序图

```
public Map saveTemplate(String username, List<List<Map<Object, Object>>>
    dataSourceList, String dateTime, String templateName, List<String> periodNames,
                            boolean flag, String desc) {
        HashMap<Object, Object> map = new HashMap<>();
        String dataSource = serialization(dataSourceList, periodNames);
        String hash = "";
        try {
            ArrayList<String> list = new ArrayList<>();
            list.add(ConstantUtil.TASK_TEMPLATE_TYPE);
            list.add(dataSource);
            ...
            File file = xxxService.generateFile(list);
            Map<Object, Object> objectObjectMap = xxxService.uploadFile(file);
            hash = ((String) objectObjectMap.get("certificate"));
        } catch (IOException e) {
            log.error("溯源模板上链失败!");
            e.printStackTrace();
        }
        TaskTemplate taskTemplate = new TaskTemplate(username, templateName,
            dataSource, dateTime, desc, hash);
        return map;
    }
```

图 5.3: 溯源模板管理关键代码

图 5.3所示为 saveTemplate 方法的关键代码。saveTemplate 方法用于保存溯源模板。由于溯源模板存在复杂的层级化数据，难以直接保存至关系型数据库中。本文将溯源模板的层级化数据在存储时进行序列化的操作，读取的时候进行解序列化即可。在保存溯源模板时，系统会将模板数据上链存储，以达到防篡改的目的。saveTemplate 方法首先将获取的溯源模板数据序列化。之后，将序列化信息保存至临时文件。最后，调用 uploadFile 方法获取临时文件的 IPFS_HASH，并将此哈希值存储至区块链中。

5.2 溯源数据管理模块

溯源数据管理负责管理原始数据的获取和存储。数据提供方通过该模块完成原始数据的采集和溯源数据的链上存储。下文将详细介绍数据采集和数据上链的具体实现。

5.2.1 数据采集设计与实现

数据采集用于收集一次数据溯源活动中的原始数据。在溯源模板管理模块中，溯源模板确定了具体场景的步骤阶段以及各阶段中所涉及到的具体字段。数据采集则以溯源模板为根据，收集不同阶段各个字段具体的值。字段值可以是数字、字符串、时间、对象以及文件五种类型。

图 5.4所示为数据采集核心类图。ProvDataController 接收前端发出的数据采集请求，获取经 Http 传输得到的原始数据并检查其是否符合当前溯源模板的格式要求。ProvDataService 是 ProvDataController 的成员变量。ProvDataController 以组合的方式实现 ProvDataService 类的代码复用。ProvDataServiceImpl 类是 ProvDataService 的具体实现。在 ProvDataServiceImpl 中基于 TaskTemplateService 服务加工 ProvDataController 所获取的原始数据，区分简单数据类型和复杂数据类型，使其便于后续原始数据的存储和溯源。

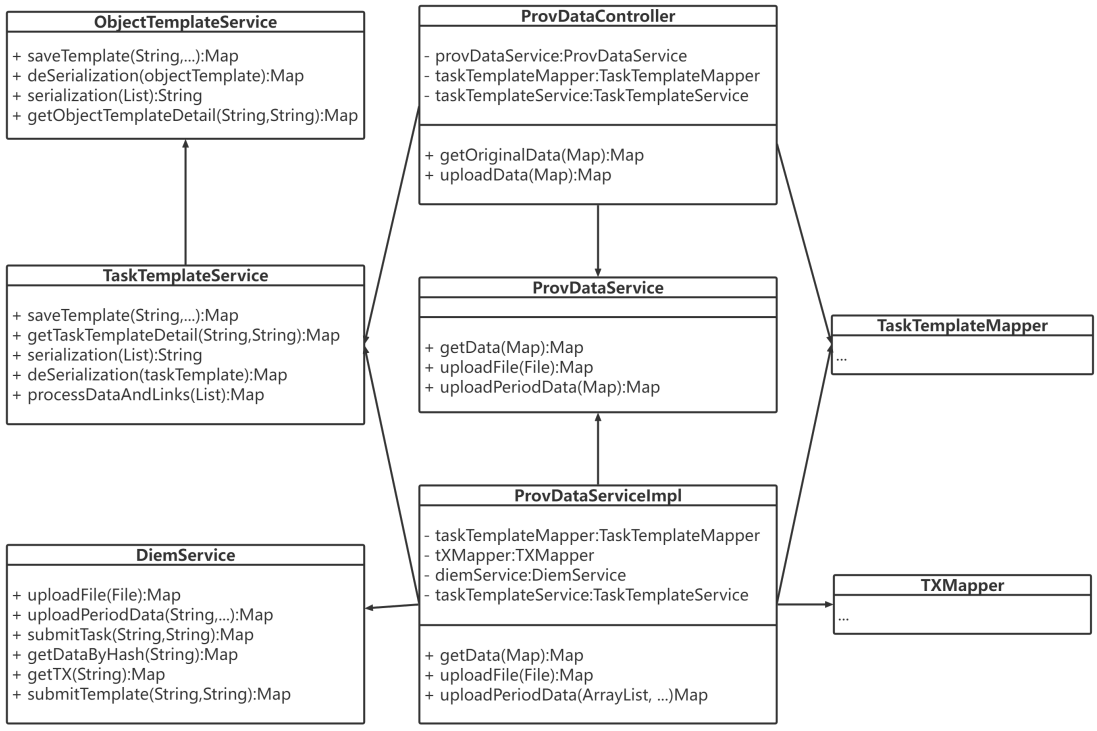

图 5.4: 数据采集类图

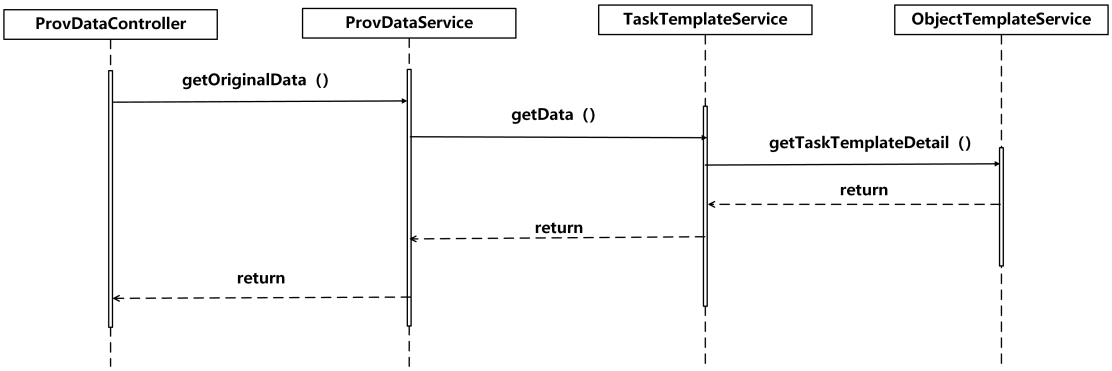

图 5.5: 数据采集时序图

```
public Map<Object, Object> getData(Map data) {
        ...
        if (!objectMap.containsKey("child")) {
            // 基本对象
            String objectValue = String.valueOf(objectMap.get("info"));
            Object info = objectMap.get("info");
            System.out.println(info);
            line = objectName + ":" + objectValue;
        } else {
            // 复杂对象
            List<Map<Object, Object>> child = (List<Map<Object, Object>>)
                objectMap.get("child");
            StringBuilder stringBuilder = new StringBuilder();
            stringBuilder.append(objectName + "&[");
            for (Map<Object, Object> objectObjectMap : child) {
                ...
                stringBuilder.append(objectObjectMap.get("info"));
            }
            stringBuilder.deleteCharAt(stringBuilder.length() - 1);
            stringBuilder.append("]");
            line = stringBuilder.toString();
        }
                ...
    }
```

图 5.6: 数据采集关键代码

数据采集时序图如图 5.5所示。数据采集请求首先被 ProvDataController 中的 getOriginalData 方法拦截。之后，在 getOriginalData 方法中调用 ProvDataService 接口中的 getData 方法。getData 方法基于 getTaskTemplateDetail 方法获取溯源模板的详细信息。若溯源模板中包含对象模板，需要进一步确定对象模板所含属性。最终，依据溯源模板信息，采集到的多类型原始数据被加工为数据存储指定的格式。

图 5.6所示为 getData 方法中部分关于数据采集的核心代码。代码根据 objectMap 中是否含有 child 属性区分基本对象和复杂对象。根据 info 属性获取采集到的原始数据。若采集的数据中包含复杂对象，则需要对复杂对象进行嵌套处理，提取出复杂对象中各字段的真实数据。最终，将获取到原始数据结构与溯源模板比较，确定其是否满足溯源模板的要求。

5.2.2 数据上链设计与实现

数据上链是系统进行可信溯源的基础，用户在完成数据采集的同时需要将数据存储至区块链中。由于采集到的数据类型复杂多样，在数据上链操作中需要首先将原始数据转化为 Ipfs 哈希值。数据上链存储的值为转化后的哈希值。在完成数据上链的同时需要记录区块链中发生的交易信息，用于后续数据溯源中交易信息的展示。在数据成功存储至区块链中后，系统将返回数据存储的地址值。在后续数据溯源的操作中，用户可根据地址值完成原始数据的获取。

图 5.7所示为数据上链核心类图，DimeController 将数据上链请求转发至 DiemService 服务。DiemService 接口中定义了 uploadPeriodData、submitTemplate 等与数据上链相关的方法。其中 uploadPeriodData 方法用于上传原始数据，submitTemplate 用于上传模板数据。DiemServiceImpl 作为 DiemService 的实现类实现了接口中定义的相关方法。此外，DiemServiceImpl 通过 TaskTemplateService 和 TaskService 获取数据上链采用的溯源模板信息。DiemUtils 是用于操作 Diem 区块链的工具类。DiemUtils 中封装了 queryTx 方法用于查询交易信息、accountCreate 用于在 Diem 中创建账户、upload 用于完成数据上链、download 用于获取链上数据、mintCoin 用于操作 Coin 相关、updateAccountInfo 用于更新账户信息。

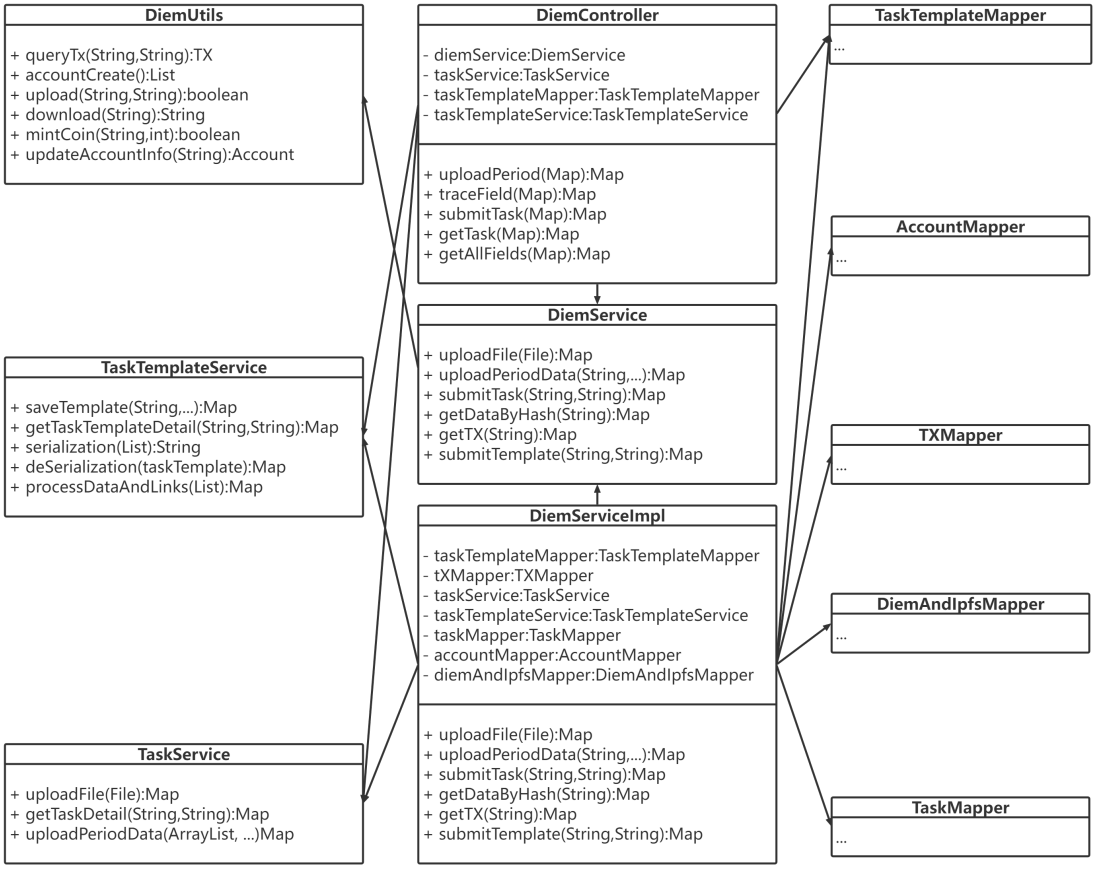

图 5.7: 数据上链类图

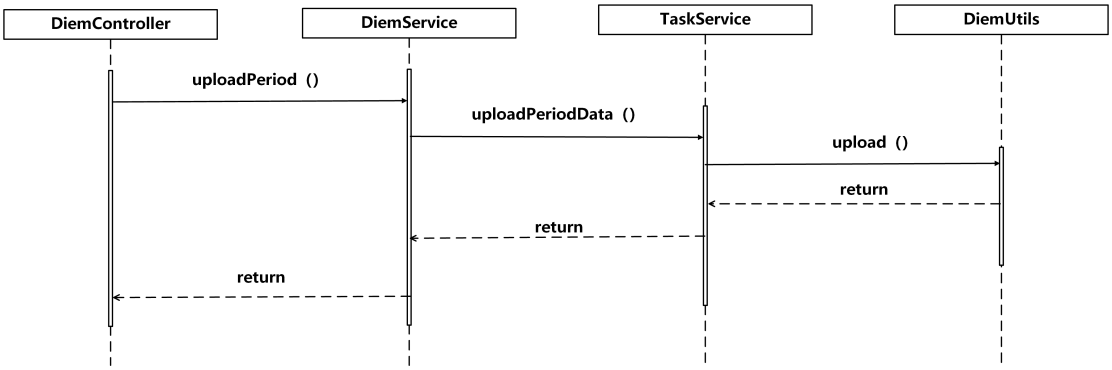

图 5.8: 数据上链时序图

数据上链时序图如图 5.8所示。DimeController 拦截前端数据上链请求，调用 uploadPeriod 方法将上链任务下发至 DiemService 中。之后，DiemService 通过

TaskService 获取溯源模板相关信息并将上链数据进行封装。最后，通过 DiemUtil 中的工具类中 upload 方法完成数据上链操作。数据上链完成之后，依次将上链结果传递至前端页面进行提示。

```
public Map<Object, Object> uploadFile(File file) throws IOException {
        //待存储哈希值
        String hash = IpfsUtil.add(file.getAbsolutePath());
        file.delete();
        // 创建账户
        List<String> strings = LibraUtils.accountCreate();
        ...
        //数据上链
        boolean upload = LibraUtils.upload(address, hash);
        //更新账户
        Account account = LibraUtils.updateAccountInfo(index);
        //记录交易
        int sequenceNum = Integer.parseInt(account.getSequenceNumber());
        for (int i = 1; i <= 5; i++) {
            TX tx = LibraUtils.queryTx(index, String.valueOf(sequenceNum - i));
            if (tx != null) {
                txMapper.insert(tx);
            }
        }
        //更新映射关系
        xxxAndIpfsMapper.insert(new XXXAndIpfs(address, hash));
        //返回链上存储地址
        map.put("certificate", address);
        return map;
    }
```

图 5.9: 数据上链关键代码

如上所示代码为数据上链方法 uploadFile 的部分核心代码。数据上链具体过程描述如下。首先，获取待上链字段的哈希值，哈希值由 Ipfs 提供。其次，在 Diem 中创建账户用于数据上链。账户创建完成之后即可调用 upload 方法完成数据上链。数据上链完成之后需要更新账户信息并将上链过程中产生的交易信息记录至关系型数据库中。最后，完成 Ipfs 哈希值和 Diem 地址的映射并将 Diem 地址返回。

5.3 溯源任务管理模块

5.3.1 溯源任务管理设计与实现

系统面向数据溯源需求方提供溯源任务管理功能。用户进行数据溯源操作之前需要完成溯源任务的新建。具体包括溯源任务名称确定、溯源模板的选择、数据采集完成验证以及数据上链验证。只有当前溯源模板已经完成数据采集和数据上链动作后，用户才可创建数据溯源任务。此外，创建溯源任务的同时，系统也将此次溯源任务的相关信息存储至区块链中。

如图 5.10所示为溯源溯源任务管理类图。TaskService 是溯源任务服务的核心，TaskService 提供了溯源任务创建并上链的方法和获取溯源任务详细信息的方法。DiemService 为 TaskService 提供区块链相关服务。TaskTemplateService 提供溯源模板服务，TaskService 调用 TaskTemplateService 服务获取溯源模板信息。TaskMapper、TaskTemplateMapper 以及 TXMapper 分别用于溯源任务数据、溯源模板数据以及交易数据的读取和存储。

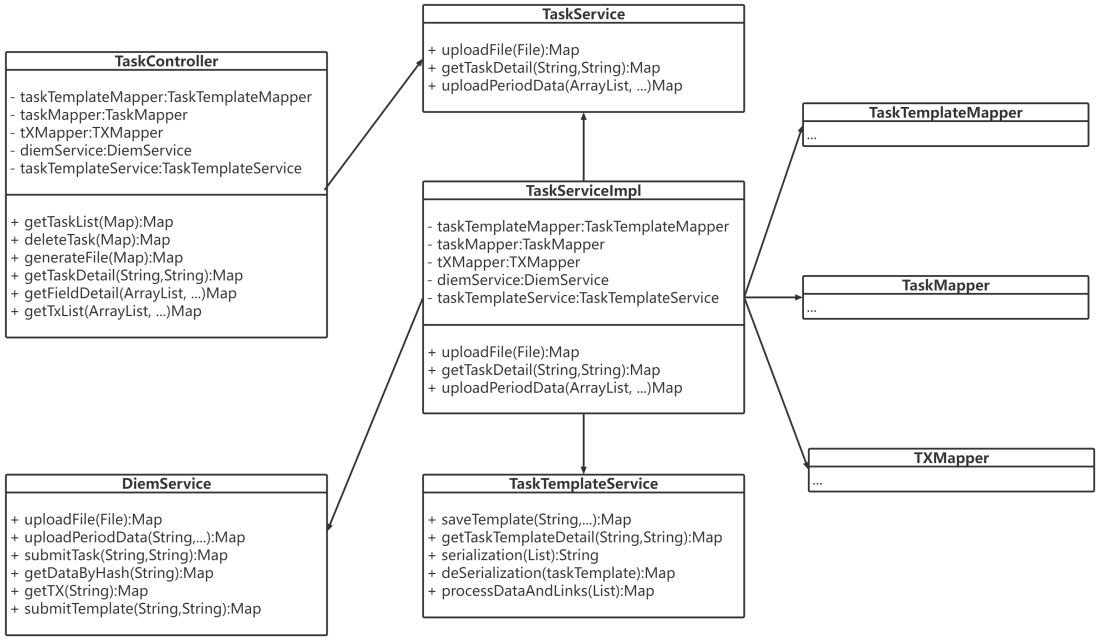

图 5.10: 溯源任务管理类图

图 5.11所示为溯源任务管理时序图。本图描述了一次溯源任务创建成功后获取溯源任务相关信息的过程。控制器 TaskController 调用 getTaskDetail 方法发

起获取任务详情信息的请求。TaskService 在被调用后，通过 DiemService 服务调用 getTX 方法获取溯源任务上链时产生的交易信息。此外，TaskService 也将通过 TaskMapper 类获取存储于关系型数据库中的溯源任务信息。最终，TaskService 将获取到的交易信息和溯源任务信息包装后一并返回给 TaskController。

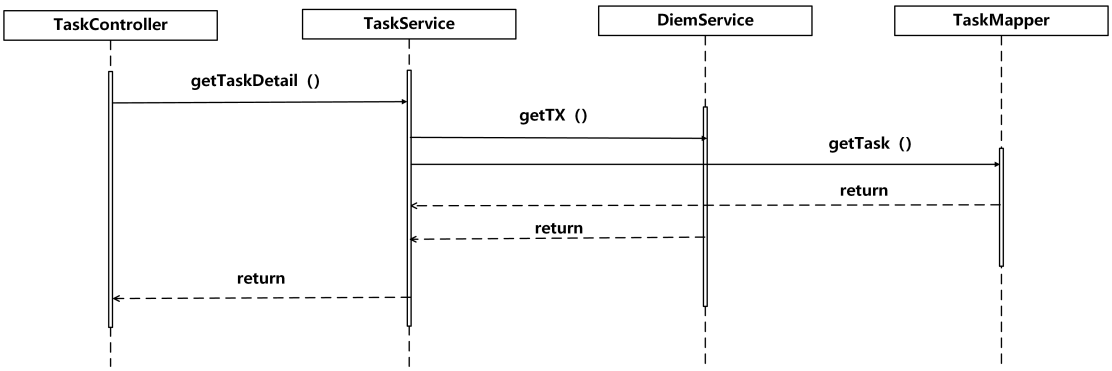

图 5.11: 溯源任务管理时序图

```
1  @Override
2      public Map<Object, Object> getTaskDetail(String username, String taskName) {
3          ...
4          Task task = taskMapper.getEntireTaskByUsernameAndTaskName(username,
               taskName);
5          Map<Object, Object> basicInfo = getBasicInfo(task);
6          Map<Object, Object> taskUploadInfo = getTaskUploadInfo(task);
7          Map<Object, Object> fieldInfo = getFieldInfo(task);
8          ...
9      }
```

图 5.12: 溯源任务管理关键代码

图 5.12所示为溯源任务管理模块中溯源任务信息获取的关键代码。TaskService 接口提供了 getTaskDetail 方法用于获取溯源任务信息。溯源任务详情由任务基本信息、任务上链信息、字段信息三种信息组成。任务基本信息描述了溯源任务的创建时间、任务名称、所使用的溯源模板。任务上链信息是指将溯源任务进行上链时所产生的交易信息。字段信息由溯源模板中的各个字段组成。

5.4 数据溯源模块

5.4.1 数据溯源设计与实现

用户可在成功创建溯源任务的前提下进行数据溯源。用户通过数据溯源功能可获取到溯源模板中定义的全部字段的溯源信息。溯源信息包括链上原始数据哈希、溯源目标数据的前驱关系以及目标数据上链存储时产生的交易信息。下面将详细介绍数据溯源的设计与实现。

图 5.13所示为数据溯源类图。数据溯源类图以 DataProvService 为核心，通过调用其他服务获取多种溯源信息。DataProvService 提供了 traceField、getRealData、getTxBySender 三种方法。traceField 完成字段关系的确定，提供了有向无环图中的边和节点的信息。getRealData 从区块链中获取原始数据哈希并将哈希值转化为原始数据。getTxBySender 用于获取原始数据哈希上链产生的交易信息。DiemService 为 DataProvService 提供区块链服务，通过 DiemService 获取区块链中存储的数据。TaskService 将一次数据溯源活动与溯源任务绑定。TaskTemplateService 服务将溯源模板与溯源任务绑定，提供了当前数据溯源活动可供选择的目标溯源字段。DataProvController 将溯源信息返回至前端页面。

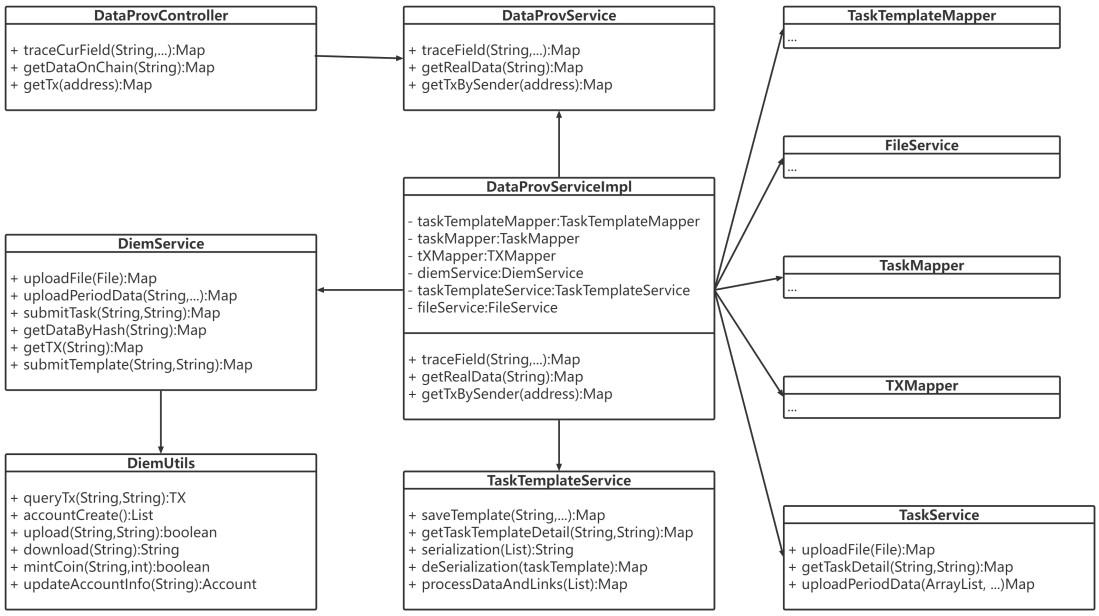

图 5.13: 数据溯源类图

数据溯源时序图如图 5.14所示。DataProvController 首先调用 traceField 等方法向 DataProvService 发出数据获取的指令。之后，DataProvService 通过 get-

TaskDetail 方法从 TaskService 中获取到溯源任务信息，通过 getTaskDetail 方法从 TaskTemplateService 中获取溯源模板相关信息。随后，DataProvService 利用溯源任务信息和溯源模板信息得到全部待溯源目标字段，并通过向 DiemService 调用 getDataByHash 等方法获取链上原始数据哈希和交易信息。最终，DataProvService 将溯源信息返回至控制器。

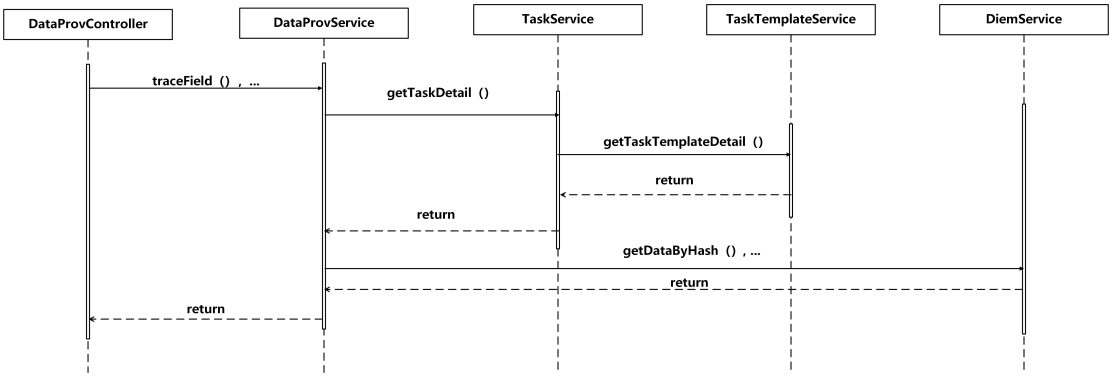

图 5.14: 数据溯源时序图

```
1  public Map<Object, Object> traceField(String username, String taskName, String
       fieldName) {
2          ...
3          // 解析出字段名和对应的前驱
4          Map<String, List<String>> fieldNameAndPreCursor =
               getFieldNameAndPreCursor(templateMap);
5          // 阶段名和字段
6          Map<String, List<String>> itsPreCursor =
               getItsPreCursor(fieldNameAndPreCursor, fieldName, templateMap);
7          // 获取两者之间的关系
8          List<Map<Object, Object>> relation = generateRelation(itsPreCursor,
               templateMap, username, taskName);
9          Map<String, List<Map<Object, Object>>> dataAndHashXY =
               generateXYAndHash(itsPreCursor, taskName, username);
10         retMap.put("data", dataAndHashXY);
11         ...
12     }
```

图 5.15: 数据溯源关键代码

图 5.15所示代码描述了有向无环图信息的生成过程。首先，获取到当前目标溯源字段的字段名和前驱字段。之后，通过宽度优先搜索算法找出目前溯源字段的所有前驱字段。然后，将获取到的前驱字段信息封装为有向无环图中的边和节点。最后，获取有向无环图中所有节点在链上存储的哈希值。通过哈希值即可获取目标溯源字段的原始数据。

```
public Map<Object, Object> getDataByHash(String certificate) {
        String hash = LibraUtils.download(certificate);
        ...
        hash = hash.substring(0, 68);
        File file = new File("/home/usr/Files/download/target.txt");
        ...
        file.createNewFile();
        IpfsUtil.download(hash, file.getAbsolutePath());
        ...
        String line = bufferedReader.readLine();
        if (line.equals(ConstantUtil.OBJECT_TEMPLATE_TYPE)) {
                // 对象模板
                ...
            } else if (line.equals(ConstantUtil.TASK_TEMPLATE_TYPE)) {
                // 任务模板
                ...
            } else if (line.equals(ConstantUtil.SIMPLE_DATA_TYPE)) {
                // 基本数据类型
                ...
            } else if (line.equals(ConstantUtil.COMPLEX_OBJECT_TYPE)) {
                // 复杂对象类型
                ...
            }
    }
```

图 5.16: 原始数据获取关键代码

如图 5.16所示为根据 Diem 账户地址获取原始数据的方法。首先，通过 Diem 账户地址读取区块链数据，获取原始数据哈希。之后，根据原始数据哈希从 IPFS 中获取原始数据。原始数据类型可分为对象模板、任务模板、基本数据以及复杂数据四种类型。针对不同的原始数据类型选取不同的处理方式，最终将处理结果返回。

5.5 本章小结

本章分为溯源模板管理、溯源数据管理、溯源任务管理、数据溯源四个部分详细介绍了溯源系统的设计与实现。在每一部分，结合关键类图、时序图、关键代码对模块的实现过程做进一步阐述。

第六章　系统测试与案例分析

本章主要完成系统的测试与案例分析。首先设置系统测试环境和详细的评价指标。之后，完成系统功能测试设、性能测试设计和安全测试设计。随后，对三种测试结果进行分析。最后，结合数据溯源具体案例进行分析。

6.1　测试环境

表 6.1给出了系统的硬件环境。选用 Intel(R)Core(TM)i9 10900CPU 高性能处理器、16GB 高速内存以及 512GB 硬盘。在操作系统上，使用 Ubuntu18.04.5 完成功能测试和性能测试。

表 6.1: 硬件环境

设备	配置
处理器	Intel(R)Core(TM)i9 10900CPU
内存	16GB DDR4
硬盘	ST1000DM003 512GB 硬盘
操作系统	Ubuntu18.04.5

表 6.2所示为系统软件环境。前端基于 Vue3.0 构建，使用 Axios0.21.1 组件完成前后端异步通信，在 hrome88.0(64 位) 浏览器进行页面展示。服务端基于 SpringBoot2.0.2 框架实现，使用 JDK1.8 作为 Java 运行环境，MySQL5.7.30 提供数据存储服务，并选用 Docker19.03.9 容器平台。区块链版本选择 Diem e7ecab683bf37465d3a99cb297c22f326316d908。

表 6.2: 软件环境

模块	软件
前端	Vue3.0, Axios0.21.1, Chrome88.0(64 位)
服务端	JDK1.8, SpringBoot2.0.2, MySQL5.7.30, Docker19.03.9
区块链	Diem e7ecab683bf37465d3a99cb297c22f326316d908

6.2 测试指标

表 6.3: 测试指标

测试指标	详情
响应时间	客户端发出请求，到接收到服务端响应的时间
内存占用率	系统运行所需的内存开销
CPU 占用率	系统运行所占用的 CPU 资源
并发量	单位时间内客户端向服务端发出请求的数量
吞吐量	单位时间内服务端处理请求的数量

表 6.3所示为测试指标，本系统测试指标包括响应时间、内存占用率、CPU 占用率、并发量和吞吐量。响应时间是指客户端从发出请求到接收到服务端回应所经历的时间，响应时间的大小直接影响用户的使用体验。内存占用率代表系统运行时所需要的内存空间。系统设计的优劣和系统复杂程度都会影响系统运行时内存占用率。CPU 占用率是指系统运行所需的计算资源。并发量指一定时间段内用户向服务端发出请求的数量。吞吐量的大小代表着系统单位时间内处理请求任务的能力。

6.3 测试设计

6.3.1 功能测试设计

功能测试关注点在于系统功能是否正常满足需求，其测试过程通常是黑盒的，不关心具体的代码实现。根据系统需求分析，本节共设计溯源模板管理、溯源任务新建、数据采集、数据上链以及数据溯源五个测试用例完成系统功能测试。

表 6.4为溯源模板管理测试用例。用例模拟平台方进行溯源模板管理。用户点击“模板管理”，进入模板管理页面。点击新建菜单栏，可填写溯源模板所需信息并将其保存至区块链中。溯源模板创建成功后，正常情况下用户可查看溯源模板的详细信息以及溯源模板的上链信息。对于未上链的溯源模板，用户可进行溯源模板信息的修改。此外，用户还可删除已创建的溯源模板。这里的模板删除并不是指从区块链中删除本模板相关数据，而是指该溯源模板在后续溯源流程中不可使用。

表 6.4: 溯源模板管理测试用例

测试编号	**TC1**
测试功能	模板管理
测试目标	用户可正常完成模板管理
测试步骤	1. 点击“模板管理”按钮; 2. 输入模板信息完成模板创建; 3. 点击“修改”按钮进行模板的修改; 4. 点击“删除”按钮删除模板; 5. 点击“查看”按钮查看模板详情。
期望结果	1. 显示模板新建页面; 2. 模板信息正常输入; 3. 完成模板的修改; 4. 成功删除模板; 5. 显示模板详情信息。

表 6.5: 溯源任务创建测试用例

测试编号	**TC2**
测试功能	溯源任务创建
测试目标	用户可正常完成溯源任务创建
测试步骤	1. 点击“新建溯源任务”按钮; 2. 在“选择溯源模板”页面中的“溯源任务名称”文本框内输入溯源任务的名称; 3. 在“选择溯源模板”页面中的“选择模板”下拉框中选择溯源模板/修改; 4. 点击“创建”/“删除”/“修改”/“查看”按钮;
期望结果	1. 显示溯源任务新建页面; 2. 溯源任务名称正常输入; 3. 溯源模板正常选择。

表 6.5所示为溯源任务创建测试用例。溯源任务创建与溯源模板绑定。一个溯源任务只能使用一个唯一的溯源模板。溯源模板创建时需要确定溯源任务的

名称并选择溯源模板。当该溯源模板已完成数据采集和数据上链时，用户点击“创建”按钮即可成功创建溯源任务。

数据采集测试用例如表 6.6所示。用例模拟溯源数据提供方完成溯源数据采集。用户点击“数据采集”按钮进入数据采集页面。在该页面下，用户可根据阶段名称选择数据采集所属阶段。之后，在页面“采集信息”下的文本框内输入阶段内字段信息，包括数字、字符串、时间、自定义对象、文件等类型数据完成数据采集。

表 6.6: 数据采集测试用例

测试编号	**TC3**
测试功能	数据采集
测试目标	用户可正常完成数据采集
测试步骤	1. 点击“数据采集”按钮，进行数据采集； 2. 点击“阶段”按钮，选择阶段名称； 3. 在“采集信息”下的文本框内输入阶段内字段信息，包括数字、字符串、时间、自定义对象、文件等。
期望结果	1. 进入数据采集页面； 2. 阶段名称正常选择； 3. 字段信息正常输入。

表 6.7所示为数据上链测试用例。用例模拟用户完成数据上链功能。数据采集成功后，点击“字段数据上链”发起数据上链请求。等待数据上链成功后，系统进行弹窗提示并将本次上链的交易发起方地址返回给用户。最后，点击“下一步”按钮，可以查看本次数据上链时所产生的交易信息。

表 6.7: 数据上链测试用例

测试编号	**TC4**
测试功能	数据上链
测试目标	用户可正常完成数据上链
测试步骤	1. 字段数据输入后，点击“字段数据上链”； 2. 数据上链成功，查看“发送方地址”信息； 3. 数据上链成功后，点击“下一步”查看交易信息。
期望结果	1. 弹窗提示“上链成功”； 2. 页面正常显示“发送方地址”信息； 3. 成功进入数据上链交易信息查看页面。

表 6.8所示为数据溯源测试用例。溯源任务为数据溯源提供了前端页面的入口。用户点击溯源任务的“详情”按钮进入数据溯源页面。数据溯源页面展示本次用户可进行数据溯源的所有目标字段。用户可选择其中任意一个字段进行数据溯源。用户选择字段后，点击“字段溯源”按钮，页面以列表形式展示溯源任务的交易行为。点击“追踪溯源”按钮，页面将以有向无环图形式展示各字段之间的关系。此外，有向无环图中悬浮鼠标至节点上方时，将展示该数据链上存储的地址信息。在页面右方，以表格形式对有向无环图进行描述，详细展示有向无环图中各字段信息。

表 6.8: 数据溯源测试用例

测试编号	**TC5**
测试功能	数据溯源
测试目标	用户可正常完成数据溯源
测试步骤	1. 点击溯源任务的“详情”按钮； 2. 点击“字段溯源”按钮； 3. 点击“追踪溯源”按钮。
期望结果	1. 显示溯源任务的上链信息； 2. 以列表形式展示溯源任务的交易行为; 3. 以列表形式展示各字段信息; 4. 以有向无环图形式展示各字段之间关系。

6.3.2 性能测试设计

系统针对服务端接口进行性能测试。服务端接口性能测试采用 JMeter[1]工具开展。JMeter 是一种常用的 Web 接口性能测试工具。

表 6.9所示为系统服务端关键接口。系统针对表中所有接口进行性能测试。在 JMeter 性能测试工具中创建线程组用于模拟用户对接口发起请求。本次性能测试设置线程组中用户数为 40 并循环两次。之后通过构造 Http 请求路径对系统接口发起测试。这里 Http 请求路径为系统对应接口 API 路径。其中，每个用户需要在两秒内对服务端发起请求。

表 6.9: 服务端关键接口

接口名称	作用描述
查询对象模板	根据对象模板名查询对象模板信息
查询任务模板	根据任务模板名查询任务模板信息
查询溯源任务	根据溯源任务名查询溯源任务信息
查询交易信息	根据交易发送方地址查询交易信息
查询 Diem 账户信息	查询系统所有 Diem 账户信息
查询 DiemIpfs 映射信息	查询系统所有 DiemIpfs 映射关系

6.4 测试结果与分析

6.4.1 功能测试结果分析

表 6.10: 功能测试用例结果

需求用例	测试用例	测试结果
UC1	TC1	通过
UC2	TC2	通过
UC3	TC3	通过
UC4	TC4	通过
UC5	TC5	通过

根据功能测试用例表，测试方按照操作步骤对所有测试用例开展测试。测试过程中将测试结果归档记录。最终得到系统测试结果如表 6.10所示。根据表

[1]JMeter. https://jmeter.apache.org/

格记录可知，功能需求与测试用例一一对应且所有功能测试用例全部通过。证明本系统已根据系统用例图实现所有功能需求。

6.4.2 性能测试结果分析

表 6.11: 接口性能测试结果

接口名称	请求数	平均值	中位数	最小值	最大值	成功率	吞吐量
查询对象模板	200	534	521	469	568	100%	12.3/s
查询任务模板	200	480	511	471	534	100%	13.6/s
查询溯源任务	200	499	503	454	519	100%	14.7/s
查询交易信息	200	472	477	467	499	100%	15.9/s
查询 Diem 账户	200	506	492	477	535	100%	17.3/s
查询哈希映射	200	511	491	434	525	100%	15.6/s
总计	1200	502	491	434	535	100%	15.9/s

表 6.11所示为服务端接口性能测试结果。在性能测试中，针对每个接口均发起 200 次请求。测试结果表明，所有接口均可成功接收请求并回应请求，请求成功率为 100%。系统整体查询时间开销平均为 502ms，中位数为 491ms。在吞吐量方面，系统平均每秒处理的交易次数为 16 次。综上所述，系统各接口功能正常，响应时间整体表现良好。

6.5 案例分析

为验证溯源系统的真实有效，本文将以众包测试场景为例，描述系统完成数据溯源的完整流程。众包测试是指一种通过互联网招募大量专业测试人员共同来完成的软件测试活动 [47]。如图 6.1所示，一次众包测试活动主要分为需求提交阶段、报告提交阶段、报告审核阶段、报告融合阶段四个阶段。其中，需求方在需求提交阶段将众测需求提交至众测平台；工人方在报告提交阶段针对测试需求对待测软件开展软件测试活动，并将测试报告提交至众测平台；平台方在报告审核阶段对工人方提交的每份测试报告进行打分形成对应的报告审核结果；最后，平台方在报告融合阶段将融合信息返回给需求方。接下来，本节将先后从对象模板创建、任务模板创建、数据采集与上链、数据溯源、原始数据获取五个步骤详细描述众包测试场景下的溯源系统使用。

(1) 对象模板创建：针对以上众包测试场景，本文拟在溯源系统建立众测需求、测试报告、报告审核结果、融合信息四个对象模板。以测试报告为例，对象

模板的详细信息如图 6.2所示。测试报告对象模板主要包括测试报告名称、众测工人、提交报告者、报告文件、报告提交时间五种数据类型。

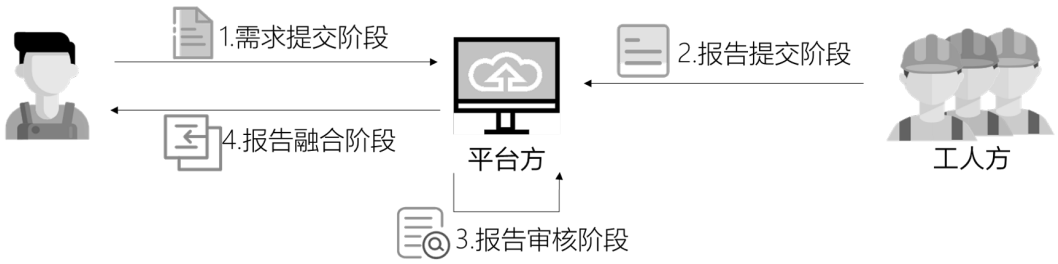

图 6.1: 众包测试流程图

图 6.2: 测试报告对象模板

图 6.3: 众测溯源任务模板

(2) 任务模板创建：基于以上四种对象模板和四个阶段，结合众包测试场景，本文建立的众测溯源任务模板如图 6.3所示。模板由需求提交阶段、报告提交阶

段、报告审核阶段、报告融合阶段四个阶段组成。以报告提交阶段为例，报告提交阶段分别由测试报告 1、测试报告 2 两个测试报告字段类型组成。此外，测试报告字段均依赖于测试需求字段。

(3) 数据采集与上链：以任务模板为依据，数据提供方（众包测试中的需求方、工人方、平台方）分别在不同阶段提交相关数据，并将原始数据上链存储。图 6.4描述了报告提交阶段数据采集界面的详细信息。图片上方是数据采集的进度条。在进度条的右侧是数据上链按钮。完成数据采集后点击数据上链按钮，系统将逐个字段完成数据上链。页面下方是数据采集区域，众测工人按照溯源模板完成各个字段的数据上传。数据采集区域右侧按钮为字段详情，点击查看按钮将展示当前采集字段的详细描述信息。数据采集区域中间交易发起方地址一栏，在未完成数据上链时不显示数据；在完成数据上链后，显示交易发起方地址数据。

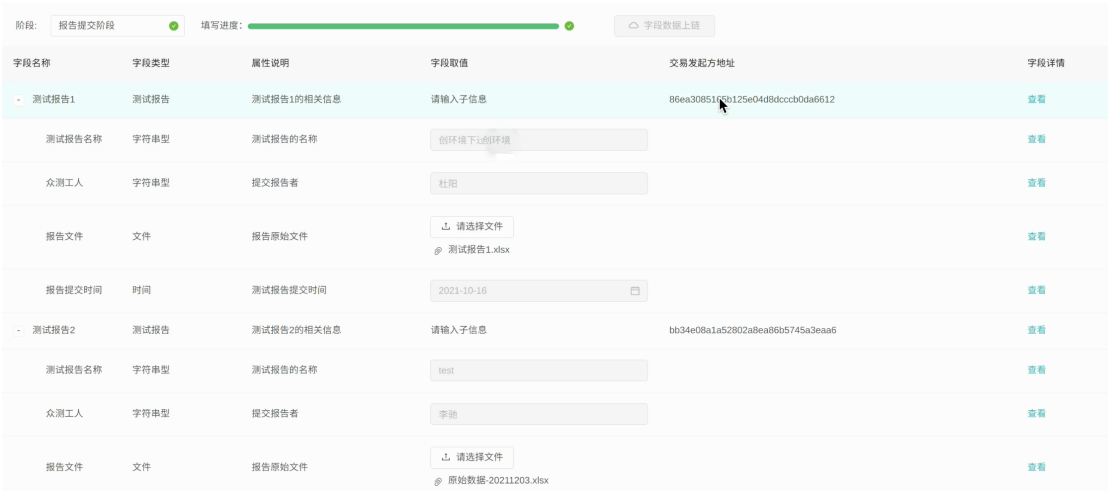

图 6.4: 数据采集与上链

(4) 数据溯源：图 6.5为针对融合结果进行溯源的结果图。界面的左侧为针对融合信息进行数据溯源得到的有向无环图。有向无环图中的节点是字段的名称，节点之间的边代表两个字段之间存在关系，箭头的开始节点是箭头结尾节点的前驱。最上层测试需求代表数据的最初起源，最下层的融合结果代表当前的溯源字段。界面的右侧为一个折叠表格，自上而下表格的每一行与有向无环图的每一行相对应。表格中每一行展开后是有向无环图中节点的详细信息，用户在表格中点击对应按钮可获得原始数据。

图 6.5: 溯源结果

(5) 原始数据获取：图 6.6所示为融合结果的链上原始数据。用户可在字段取值部分查看链上原始数据。融合结果的原始数据由融合者和融合报告两种信息组成。点击融合报告可将原始的融合报告进行下载。

图 6.6: 原始数据

6.6 本章小结

本章对系统整体分别进行功能测试和性能测试。并给出系统测试结果分析。功能测试结果表明，系统满足所有功能需求。测试结果表明，系统接口均可正常满足 Http 请求，响应时间表现良好。之后，结合众包测试场景详细描述了溯源系统的使用和溯源结果的分析。综上所述，系统满足功能完整性与系统可用性的需求。

第七章　总结与展望

7.1　总结

数据溯源作为一种追溯数据起源的技术已在工作流、数据库等多个领域取得应用。采用区块链技术存储溯源信息可提升数据溯源的可信性。当前，区块链技术高速发展，已有各种区块链平台如以太坊、超级账本、Diem 等。Diem 相较于其他区块链平台的特性之一是采用全新的 Move 编程语言进行智能合约编程。Move 语言将数字资产和真实资源均视为一等公民。其中，Move 语言中的资源是指在智能合约中定义的变量，资源具有唯一的所有者，且只能花费一次。Move 语言的这一特性从程序语言的角度进一步提高了区块链智能合约编程的安全性。本文基于 PROV 数据溯源模型并结合抽象化模板的思想解决数据溯源通用性较低的问题，以 PROV 溯源模型为基础，设计相应的数据溯源方法和流程。本文采用区块链作为溯源数据存储平台，解决了中心化数据库存储数据安全性低，数据不透明的问题。同时，选用 Diem 区块链实现数据溯源存储相较于以太坊、超级账本区块链平台具有更高的安全性。

基于 PROV 模型的 Diem 安全溯源系统主要包括溯源模板管理、溯源数据管理、溯源任务管理以及数据溯源四个模块。溯源模板包括对象模板和任务模板两种。对象模板用于支持更多的溯源数据类型，任务模板则用于定制不同场景下数据溯源的阶段和数据字段。溯源模板的引入使得系统支持更多场景下的数据溯源，同时支持更多数据类型的溯源。溯源数据管理用于管理系统中所设计的原始数据。在溯源数据管理中，数据提供方需要将原始数据分阶段逐字段完成采集。并在采集结束后向 Diem 区块链发起交易请求，区块链中各节点同意交易提案并调用事先定义的智能合约将数据存储于区块链中。溯源任务管理用于向溯源需求方提供数据溯源入口，溯源需求方需要选择对应场景下的溯源模板创建溯源任务。溯源任务创建完成后，系统可向溯源需求方提供数据溯源服务。数据溯源支持用户选择模板中的任一字段作为溯源目标数据进行溯源，溯源服务将最终溯源结果以有向无环图的形式返回给用户。图中的节点代表了与目标溯源字段相关的数据，图中的边代表了数据之间的前驱后继关系。此外，溯源服务还将数据提供方存储于区块链中的数据返回给溯源需求方。以此验证数据的真实性与可信性。

在技术实现方面，前端采用 Vue 框架，服务端采用 SpringBoot 框架，存储层选用 Diem 框架和 Mysql 关系型数据库。此外，系统采用 Docker 容器部署系

统服务，进一步提升系统的可用性和可维护性。本文最后对系统分别进行了功能测试和性能测试。测试结果表明，本文实现了所有用例的功能性需求。针对查询接口，系统能够以较高效率处理用户请求，具有良好的可用性。可稳定提供可信数据溯源服务。

7.2 展望

本文还存在着一定的不足之处和可改进空间。后期可针对系统做进一步完善。不足之处主要有以下三个方面：

第一：支持更加丰富的数据采集方案。当前系统的数据采集主要依赖于用户对原始数据的手工输入和文件上传，数据采集过程较为繁琐。后续可考虑增加 Http 链接形式的数据采集方式，可分别从不同的外部数据库，外部数据提供地址自动采集数据，简化数据采集流程。

第二：提供更加丰富的区块链交易数据展示界面。当前系统中数据上链产生的数据交易信息，只有在新建溯源任务后，进行数据溯源时才可查看。对于数据提供方而言，并不能轻易获取到数据上链所产生的交易信息。对于平台方而言，不能实时清晰地了解系统中所发生的交易规模和交易详情。后续，可考虑增加区块链信息统计展示页面，统计汇总链上的所有交易信息。

第三：进一步提升系统数据溯源的效率。本文采用有向无环图的方式组织溯源关系，并在数据溯源时，使用宽度优先搜索算法搜索有向无环图。然而，当图中节点和边的数量巨大时，宽度优先搜索算法不能做到高效查询。后续可考虑通过增加索引的方式提升数据搜索效率。

参考文献

[1] OLUFOWOBI H, ENGEL R, BARACALDO N, et al. Data provenance model for internet of things systems[C] // International Conference on Service-Oriented Computing. 2016 : 85 – 91.

[2] 明华, 张勇, 符小辉. 数据溯源技术综述 [J]. 小型微型计算机系统, 2012, 33(9) : 1917 – 1923.

[3] 焦通, 申德荣, 聂铁铮. 区块链数据库: 一种可查询且防篡改的数据库 [J]. 软件学报, 2019, 30(9) : 2671 – 2685.

[4] 沈鑫, 裴庆祺, 刘雪峰. 区块链技术综述 [J]. 网络与信息安全学报, 2016, 2(11) : 11 – 20.

[5] WOOD G, OTHERS. Ethereum: A secure decentralised generalised transaction ledger[J]. Ethereum project yellow paper, 2014, 151(2014) : 1 – 32.

[6] ANDROULAKI E, BARGER A, BORTNIKOV V, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains[C] // Proceedings of the thirteenth EuroSys conference. 2018 : 1 – 15.

[7] LATT C N Z, RAHMADIKA S, RHEE K. A Data Provenance System for Myanmar Rice Cycle Based on Ethereum Blockchain[J]. Journal of Multimedia Information System, 2021, 8(1) : 35 – 44.

[8] GOURU N, VADLAMANI N L. DistProv-Data Provenance in Distributed Cloud for Secure Transfer of Digital Assets with Ethereum Blockchain using ZKP[M]. USA : Cyber Warfare and Terrorism, 2020.

[9] DEMICHEV A, KRYUKOV A, PRIKHODKO N. The approach to managing provenance metadata and data access rights in distributed storage using the hyperledger blockchain platform[C] // Ivannikov Ispras Open Conference. 2018 : 131 – 136.

[10] BLACKSHEAR. Move: A language with programmable resources[EB]. 2020 (2020/5/26) [2022/5/20].

[11] MEHAR M I, SHIER C L, GIAMBATTISTA A, et al. Understanding a revolutionary and flawed grand experiment in blockchain: the DAO attack[J]. Journal of Cases on Information Technology, 2019, 21(1): 19–32.

[12] SAYEED S, MARCO-GISBERT H, CAIRA T. Smart contract: Attacks and protections[J]. IEEE Access, 2020, 8: 24416–24427.

[13] MOREAU L, GROTH P. PROV-Overview: An Overview of the PROV Family of Documents[J]. W3C Note, 2013: 1–9.

[14] BOWERS S, MCPHILLIPS T, RIDDLE S, et al. Kepler/pPOD: Scientific workflow and provenance support for assembling the tree of life[C] // International Provenance and Annotation Workshop. 2008: 70–77.

[15] MOREAU L, CLIFFORD B, FREIRE J, et al. The open provenance model core specification[J]. Future generation computer systems, 2011, 27(6): 743–756.

[16] HASAN R, SION R, WINSLETT M. The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance.[C] // USENIX Conference on File and Storage Technologies. 2009: 1–14.

[17] SIMMHAN Y L, PLALE B, GANNON D. A survey of data provenance in e-science[J]. ACM Sigmod Record, 2005, 34(3): 31–36.

[18] BUNEMAN P, KHANNA S, TAN W-C. Data provenance: Some basic issues[C] // International Conference on Foundations of Software Technology and Theoretical Computer Science. 2000: 87–93.

[19] 戴超凡, 王涛, 张鹏程. 数据起源技术发展研究综述 [J]. 计算机应用研究, 2010, 27(9): 3215–3221.

[20] LANTER D P. Design of a lineage-based meta-data base for GIS[J]. Cartography and Geographic Information Systems, 1991, 18(4): 255–261.

[21] GAO Y, CHEN X, DU X. A big data provenance model for data security supervision based on PROV-DM model[J]. IEEE Access, 2020, 8: 38742–38752.

[22] CHITICARIU L, TAN W-C, VIJAYVARGIYA G. DBNotes: a post-it system for relational databases based on provenance[C] // Proceedings of the 2005 ACM SIGMOD international conference on Management of data. 2005: 942–944.

[23] FAN H. Tracing data lineage using automed schema transformation pathways[C] // British National Conference on Databases. 2002 : 50 – 53.

[24] 徐飞, 高济. 一种通用的数据追踪系统的实现 [J]. 计算机工程与应用, 2003, 39(34) : 3.

[25] WANG L-W, HUANG Z-Q, LUO M, et al. Data provenance in a scientific workflow service framework integrated with object deputy database[J]. Chinese Journal of Computers, 2008, 31(5) : 721 – 732.

[26] GADANG S S, PANDA B, HOAG J E. Provenance tracking with bit vectors[C] // The Fourth International Conference on Information Assurance and Security. 2008 : 132 – 137.

[27] WIDOM J. Trio: A system for integrated management of data, accuracy, and lineage[R]. USA : Stanford InfoLab, 2004.

[28] ANAND M K, BOWERS S, LUDÄSCHER B. Techniques for efficiently querying scientific workflow provenance graphs.[C] // International Conference on Extending Database Technology. 2010 : 287 – 298.

[29] RUAN P, CHEN G, DINH T T A, et al. Fine-grained, secure and efficient data provenance on blockchain systems[J]. Proceedings of the VLDB Endowment, 2019, 12(9) : 975 – 988.

[30] CARO M P, ALI M S, VECCHIO M, et al. Blockchain-based traceability in Agri-Food supply chain management: A practical implementation[C] // IoT Vertical and Topical Summit on Agriculture-Tuscany. 2018 : 1 – 4.

[31] NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system[J]. Decentralized Business Review, 2008 : 21260.

[32] HASAN S S, SULTAN N H, BARBHUIYA F A. Cloud data provenance using IPFS and blockchain technology[C] // Proceedings of the Seventh International Workshop on Security in Cloud Computing. 2019 : 5 – 12.

[33] FANNING K, CENTERS D P. Blockchain and Its Coming Impact on Financial Services[J]. Journal of Corporate Accounting Finance, 2016, 27(5) : 53 – 57.

[34] NIZAMUDDIN N, HASAN H, SALAH K, et al. Blockchain-based framework for protecting author royalty of digital assets[J]. Arabian Journal for Science and Engineering, 2019, 44(4): 3849–3866.

[35] LIANG X, SHETTY S, TOSH D, et al. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability[C] // IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. 2017: 468–477.

[36] 刘耀宗, 刘云恒. 基于区块链的 RFID 大数据安全溯源模型 [J]. 计算机科学, 2018, 45(11A): 367–368.

[37] 张国英, 毛燕琴. 一种基于区块链的去中心化数据溯源方法 [J]. 南京邮电大学学报: 自然科学版, 2019, 39(2): 8.

[38] NUGENT T, UPTON D, CIMPOESU M. Improving data transparency in clinical trials using blockchain smart contracts[J]. F1000Research, 2016, 5.

[39] MONTECCHI M, PLANGGER K, ETTER M. It’s real, trust me! Establishing supply chain provenance using blockchain[J]. Business Horizons, 2019, 62(3): 283–293.

[40] 任浩方. 基于区块链的物联网数据溯源研究与实现 [D]. 北京: 北京工业大学, 2020.

[41] RAM S, LIU J, OTHERS. A New Perspective on Semantics of Data Provenance.[J]. SWPM, 2009, 526.

[42] MISSIER P, BELHAJJAME K, CHENEY J. The W3C PROV family of specifications for modelling provenance metadata[C] // Proceedings of the 16th International Conference on Extending Database Technology. 2013: 773–776.

[43] GERVAIS A, KARAME G O, WÜST K, et al. On the security and performance of proof of work blockchains[C] // Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016: 3–16.

[44] KING S, NADAL S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake[J]. self-published paper, 2012, 19(1): 1–6.

[45] CASTRO M, LISKOV B, OTHERS. Practical byzantine fault tolerance[C] // Operating Systems Design and Implementation. 1999 : 173 – 186.

[46] BAUDET M, CHING A, CHURSIN A, et al. State machine replication in the Diem blockchain[EB]. 2020 (2021/8/17) [2022/5/20].

[47] 冯剑红, 李国良, 冯建华. 众包技术研究综述 [J]. 计算机学报, 2015, 38(9) : 1713 – 1726.

简历与科研成果

基本情况

常家鑫，男，汉族，1999 年 9 月出生，河南平顶山人。

教育背景

2020 年 9 月 ~ 2022 年 6 月 南京大学软件学院 硕士

2016 年 9 月 ~ 2020 年 6 月 中国矿业大学计算机科学与技术学院 本科

致　　谢

时光荏苒，美好且充实的研究生时光到了必须说再见的时刻。在此，衷心感谢两年来帮助我的老师、同学和朋友。

首先，我要感谢 iSE 实验室和我的导师陈振宇教授，陈老师从论文的选题到最终的完成给予了许多帮助和指导，在这里我由衷地感谢陈老师对我的帮助和指导。其次，我要感谢王兴亚老师，在完成论文写作的过程中，您们总能给予细致的指导意见和论文写作方法，促使我能按计划完成论文写作。

其次，我要感谢所有帮助我的学长学姐和同学们。感谢大家两年以来的陪伴和帮助，祝愿大家前程似锦。

感谢南京大学，在这里我不仅学到了专业知识，更认识到许多良师益友，尤其要感谢软件学院的各位老师，谢谢您们的无私奉献与辛勤栽培。

最后，我要感谢我的父母和家人，两年来，快乐的事情因为有你们的分享而更快乐，失意的日子因为有你们的关怀能忘却伤痛，坚强前行。无论我成功与否，你们总以鼓励的言语告诉我很棒，谢谢你们，我会继续努力。