分类号_____ 学号 M201472710
学校代码 10487 密级_____

华中科技大学

硕士学位论文

基于区块链的
跨域认证与访问控制的研究

学位申请人： 朱孝兵
学科专业： 网络空间安全
指导教师： 汤学明 副教授
答辩日期： 2017 年 5 月 24 日

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree for the Master of Engineering

Research on Cross-domain Authentication and Access Control Based on Blockchain

Candidate : Zhu Xiaobing

Major : Cyberspace security

Supervisor : Assoc. Prof.Tang Xueming

Huazhong University of Science & Technology

Wuhan 430074, P.R.China

May 24, 2017

独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于　保密□，　　在_____年解密后适用本授权书。
　　不保密□。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

摘 要

随着云计算普及程度逐渐深化，很多企业将应用域环境进行功能切分并迁移至云端使得所有平台共享。不同应用域之间相互访问时需要跨域，目前主流解决方案是通过第三方集中认证或者发放可信票据的形式来实现跨域访问的。第三方机构的存在不仅会导致整个信息流动过程变得复杂，而且第三方是否绝对可信值得怀疑。所以，寻求一种去中心化且安全便利的跨域认证和访问控制机制是一个亟待解决的问题。区块链技术具有去中心化，所有参与节点共同维护数据，不需要一个公认可信第三方节点的特点，这些特征使得区块链在金融领域得到了广泛应用，与此同时众多研究者也在不断的探究区块链在其他行业的应用场景。

基于区块链技术的跨域认证和访问控制机制是在深入分析了区块链的技术原理，并对比传统的基于可信第三方的 Kerberos,passport 等跨域认证协议的此基础上提出的针对可信第三方的去中心化与去信任化，且使得所有参与节点共同决定跨域授权访问的跨域解决方案。该方案采用区块链来存储云环境中所有参与节点之间相互授权的请求授权信息与确认授权信息，同时网络环境中所有参与节点共同维护区块链中的数据信息以防单点作弊来实现信任的去中心化。

基于区块链的跨域访问方案，能够有效的实现在云环境中各应用域节点中用户的跨域认证与访问控制，提供了一种安全便捷的方式保证云环境中各参与节点数据共享的安全性。

关键字：云环境，区块链，跨域，身份认证，访问控制

Abstract

With the popularity of cloud computing many companies cut their application into some segments and migrate them to the cloud for sharing. Visiting different application domains need to cross domain, the current solution is depending on a third party certification or issuing credible paper. The existence of the third party will not only lead to the information flow process is complicated, and that the third party is absolutely credible is questionable. Therefore, seeking a decentralized safe and convenient manner of cross-domain authentication and access control mechanism is a problem. Blockchain is a decentralized technology, all nodes do not need a trusted third party node, these characteristics make blockchain is widely used in the financial field, at the same time many researchers also find use scenarios which blockchain can use in the industries.

The mechanism of cross-domain authentication and access control based on Blockchain is built on the analysis of the technology principle of blockchain and on the comparison of the traditional manners based on the third trusted party,such as the Kerberos and passport. this mechanism is a decentralized mechanism and all participating noes decides cross-domain solutions commonly, because the block chain store all the authorization information between nodes in the cloud environment, at the same time, all participate nodes jointly maintain blocks information in case of a single point cheating to achieve decentralization of trust.

The mechanism which based on blockchain can effectively implement cross-domain authentication and access control, and provides a safe and convenient way to ensure the security of data sharing in the cloud environment.

Key words: Cloud environment, Blockchain, Cross domain, Identity authentication, Access control

目 录

1 绪言

1.1 研究的背景和意义

互联网的快速发展使得人们可以实现信息高效共享和资源便利访问，但是整个社会在信息化的过程中信息的安全问题也变得日益突出，其中在网络环境中如何防范非法访问和严格把控对敏感信息的请求是一个比较重要的安全问题。目前，在网络建设方面安全问题已经是考虑的首要因素[1]。

随着云计算的快速发展，互联网上网络资源的应用域的数量快速增长，客户端与不同应用域以及不同应用域之间的的交互变得更加频繁，对每一个客户而言，如果在不同域之间访问都需要单独登录或单独授权将极大的增加用户的使用负担，为了使得用户可以在不同域之间方便实现切换和获取资源，我们就必须使用跨域的访问控制机制使得在只知道用户的部分信息的情况下，有效的实现在不同应用管理域下的访问控制以及权限的管理，防止网络中的资源受到非法访问。

访问控制策略决定了一个应用域中谁有权访问资源，那些资源可以被访问，以及以什么方式来访问，其核心就是对一个用户的某次访问进行请求合法性判断[2]。因此，在跨域场景中，要解决的核心问题就是一个应用域如何识别来自其他应用域的用户，以及如何根据之前约定的策略对用户在本应用域的操作进行合法性判断[3]。

目前，不论是单个集团内部多个域之间，还是集团与集团的应用域之间，跨域访问协议主要有三种，分别是 Kerberos,Passport ,SAML[4]。这三种协议都采用了基于代理人的策略，使用可信第三方作为中介节点，由可信第三方来维持各个参与应用域之间的信任关系[5]。这样导致的直接问题就是在所有应用域组成的整个网络环境中,第三方的中介节点必须要是所有节点公认绝对可信的。但是在互联网环境中，没有任何节点是可以做到绝对的可信和绝对的安全，同时，由于第三方可信中介的存在，导致各个应用域之间往来信息流变得更加复杂，维护也变得更加困难[6]。

区块链是一种去中心化，去信任化的技术[7]。在区块链网络中，没有可信第三

方存在，所有参与节点共同维护数据，将其可信建立在工作量证明的基础之上。而且基于区块链技术的比特币在金融领域的运用十分成功，运行八年其稳定性和安全性已经得到充分的实践论证[8]。受此启发，基于区块链来实现跨域单点登录和基于角色的授权方案应运而生，基于区块链的单点登录是一种在多个独立的应用域中实现身份认证的一种机制，可以实现一个应用域登录，在相互信任的其他域可以免登陆来访问对应的资源。该机制把原来各自分散开的用户管理集中起来，通过共同建立起来的信任关系来实现用户的自动身份认证。

对比传统的跨域，基于区块链的跨域方案具有如下优点：

1. 完全的做到了去中心化，因为信任关系是基于各个域之间的约定达成的，而且这个信任关系数据会持久化到区块链中，区块链数据是所有参与据点共同维护的，而且不存在传统跨域方案中可信第三方节点。

2. 系统更加的安全，因为信任关系数据是写到区块中的，其合法性得到了网络环境中多数节点的认可，而且区块链是在不断的延长，由于区块是基于 hash 运算产生，如果想要篡改系统中已存在的信任关系数据，则要求攻击者算力必须要能达到组成信任网络中的所有节点算力总和的 50%以上才有可能。无疑，这难度是巨大的，而且随着加入到信任网络中的节点越多，攻击难度呈指数下降[9]。

1.2 研究概况

1.2.1 跨域身份认证

跨域的身份认证主要是指单点登录（SSO），目标是让用户在一个应用域中主动登录，便可以在之后访问相互信任的应用域，而不需要用户在此进行主动的身份认证。通过这种方式，有效的解决了用户在不同应用域之间切换时需要频繁输入用户口令的问题，而且这种方式也有利于各个应用域中用户的管理[10]。

目前，针对跨域的 SSO 的成熟产品主要有三种，微软公司基于.NET 的 Passport，Liberty 的 SAML,以及麻省理工研发的 Kerberos 协议。

其中微软的 Passport 是一个只针对于 Web 的单点登录系统，为所有在 Passpo

rt 上注册的应用域提供身份认证服务，Passport 的主要特点是集中式认证，分布式授权。一个应用域的用户只要在 Passport 上注册，那么当这个用户通过该应用域登录到 Passport 通过身份验证之后，那么该用户就可以直接对所有在 Passport 上注册的合作服务器进行免登陆访问[11]。

Liberty 是 SUN 在 2002 年发起的一个项目，是为互联网上个人实现身份认证管理的一个标准化组织，旨在提供一种开放的与平台无关的单点登录解决方案，它的核心思想是身份联合，多个应用域之间可以在保留各自的身份认证机制的同时建立对应的身份映射关系来达到单点登录的目的[12]。Liberty 是基于 SAML 的，SAML 在对跨域提供良好支持的前提下，对用户的隐私保护保护也十分得当。

麻省理工研发的 Kerberos 协议是一种基于 KDC(Key Distribution Center,秘钥分配中心)的身份认证协议,该协议通过可信的 KDC 并结合对称密钥加密算法为基于 C/S 架构的网络提供统一的身份认证以及单点登录服务，Kerberos V5 协议通过自身验证用户身份的机制来做用户身份的验证，通过安全的打包用户名以及用户身份信任凭证来产生票据，最后通过不同系统间票据的传递并验证票据的有效性实现单点登录[13]。

1.2.2 跨域访问控制

二十世纪九十年代开始，美国国家标准和技术研究院开始对基于角色的访问控制模型（RBAC）进行研究，该模型是一种在企业安全策略中行之有效的访问控制方式，它的核心思想是用户和权限并不直接关联，而是采取了一种中介授权的思想，将用户赋予一定角色，而给角色授权。一旦用户获取的某一角色，就拥有了对应角色所拥有的所有权限。随着网络环境的复杂化,针对多个应用域环境下的访问控制，Kapadia 等人在 RBAC 基础上提出了 IRBAC2000 模型，通过不同应用域之间的角色的动态转换来实现跨域访问控制，但是 IRBAC2000 仅仅只是一个策略框架，并没有考虑到怎么管理对应角色的关联，因此，在 IRBAC2000 的基础上，又提出了管理模型 A-IRBAC2000[14]，这个管理模型使得角色的转换变得动态且易于管理，特别适合角色层次复杂的场景，但是这个管理模型依旧不能避免其可能导致的“隐蔽提升”与“关

联冲突”等安全隐患问题。

在国内，针对 IRBAC2000 的潜在问题，洪帆等人[15]提出了用先决条件和安全约束条件来加强模型的安全性。国内也有其他学者提出通过信誉值和转换因子来实现用户角色的动态调整和域间角色的动态转换来达到保证角色权限安全性的目的[16]。

1.3 论文的主要研究内容

本课题在深入分析了传统集团域不同子域之间以及不同集团域之间的跨域认证以及访问控制方案之后，抽象出简化的跨域认证和基于角色的授权方案。同时，深入分析了区块链技术原理，并在此基础上将传统的跨域认证以及权限控制作为区块链技术的一个应用场景，从跨域身份认证和访问授权两个方面进行分析并分别提出了基于区块链的跨域访问授权与身份认证框架。

1. 基于区块链的跨域访问授权：在该框架中，由各个应用域节点共同组成信任网络，信任网络中请求权限节点（以下称信任域）需要对应权限时，必须发送请求到对应的授权节点（以下称授权信任域）。请求信息中包含本域角色列表，信任域节点还要向授权信任域节点提供自己的应用域信息，授权信任域节点会根据信任域节点提供的应用域信息和角色信息决定授予权限还是拒绝授权，如果决定授权，授权信任域节点会将自己的授权信息广播并写入到区块链中，可信域节点当收到授权节点的肯定回复之后，也会将自己请求权限信息以及被授权信息广播并写入到区块链中。只有当可信域能从区块链网络中成功持有请求授权区块与授权确认区块，才认为授权成功。

2. 基于区块链的跨域身份认证：针对用户跨域访问需求提出了跨域访问框架，在该框架中，基于各信任域之前权限交换为前提，某信任域用户在本域登录之后，信任域会自动根据自己维持的授权信任域列表从区块链中获取给本域的授权数据块，从授权数据块中获取授权信任域开放给本域的访问接口，公钥信息等，然后本域后台采用异步请求的方式发送登录信息到授权信任域。其他授权信任域也会根据自己维护的授权列表的信息从区块链中取得对应的授权区块，并提取区块中的信息验证

权限的正确性等，并根据验证的结果对发送过来的异步请求发送响应应答。

1.4 论文的组织结构

本论文主要由以下六部分组成：

第一章：绪论。介绍了云环境中跨域认证和访问控制的研究背景和意义，以及目前跨域认证和访问控制的发展现状，并讨论了区块链技术，着重分析了结合区块链技术来实现跨域认证和访问控制的过程。

第二章：关键技术分析，对本文涉及到的技术要点进行了详细的阐述，特别是对区块链中的关键概念做了较为深入的分析。

第三章：分析基于传统的跨域访问方案，并结合区块链的关键技术思想，对于基于区块链和 cookie 的单向跨域访问认证方案进行系统的研究。

第四章：基于第三章提出的跨域访问认证研究方案，给出了系统的分层架构，并对主要模块的实现方式进行了详细的设计阐述。

第五章:通过实验数据分析论证了基于区块链和 cookie 的单向跨域访问认证方案的可行性。

第六章：总结与展望，对系统将来可以持续优化的方面进行了总结。

2 关键技术分析

2.1 跨域单点登录与基于角色的访问控制

2.1.1 跨域单点登录

基于技术实现可以将跨域单点登录分为以下两类：

1.以经纪人为核心的单点登录（Broker-Based SSO），这种方式依托一个中央认证服务器来对用户做集中的认证和用户账号管理，这种采用中央认证服务将用户数据集中管理的方式极大的减轻了管理负担，并把它当做一个独立的专门用于登录认证的独立第三方，也就是我们所说的经纪人[17]。该模式的典型应用就是 Kerberos[18]。一个典型的 Broker-Based SSO 主要有需要进行认证的客户端，可以提供认证服务充当经纪人角色的认证服务器，以及支持认证服务器的应用服务器组成，如图 2-1 所示：

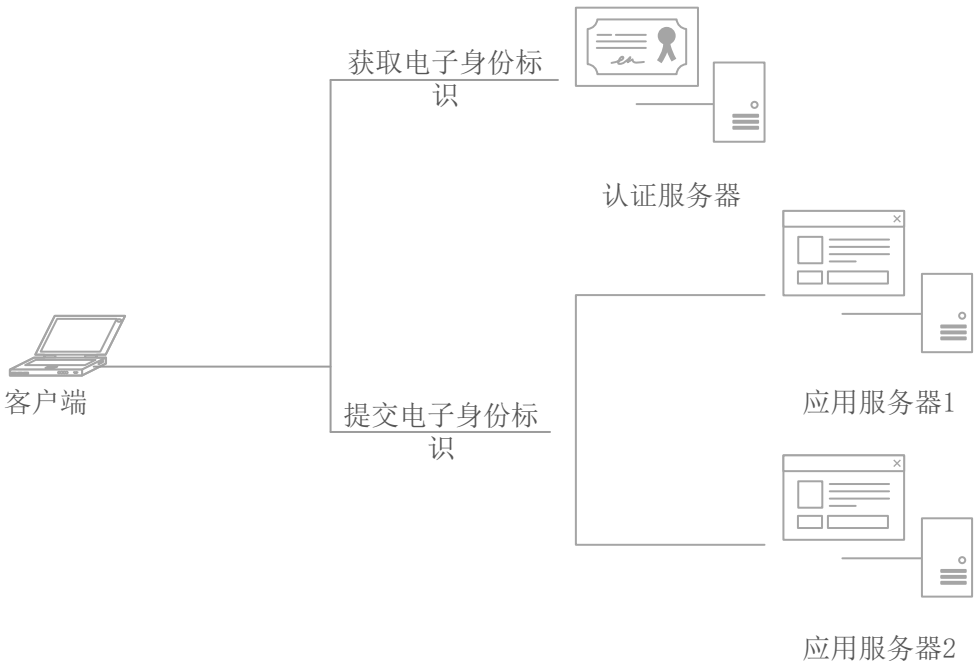

图 2-1　基于经纪人的单点登录方案

基于此方案实现单点登录，首先，在访问资源之前，客户端首先去认证服务器

进行身份认证，为了安全，可以选择客户端和服务器相互认证的方式。其次，用户通过认证之后，中心认证服务器就会发送一个代表了用户身份的点子身份标识。最后，用户通过获得的电子身份标识去访问其他的应用服务器，实现单点登录[19]。

2.基于 Cookie 的单点登录（Cookie-Based SSO），在 Web 模式中，可以采用 Cookie 保存用户登录的凭证信息，以实现单点登录，其实现流程如图 2-2 所示。

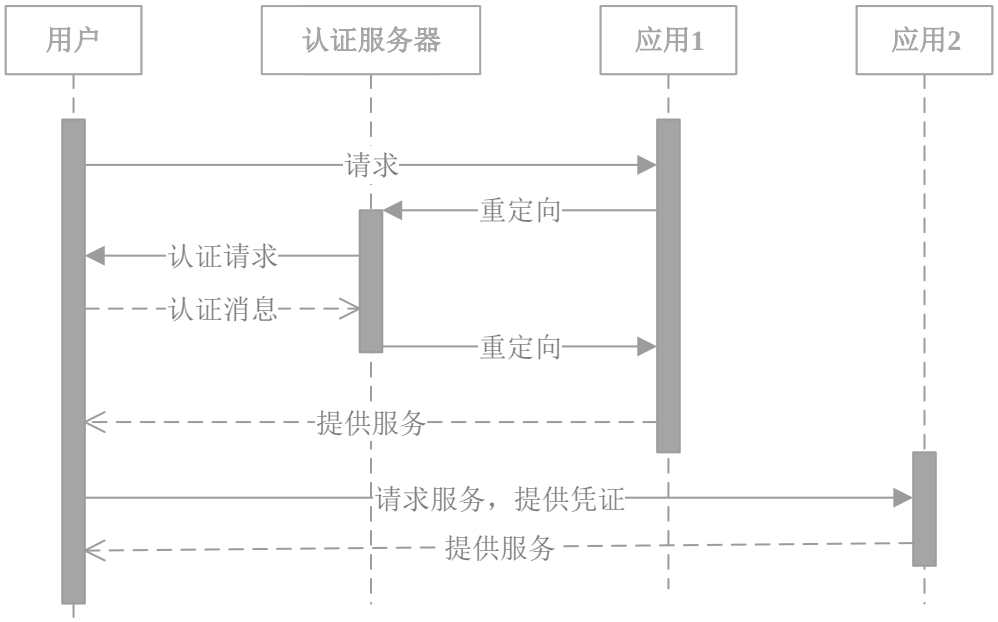

图 2-2　基于 Cookie 的单点登录

首先，客户端向 web 服务器发送请求，请求会被定向到认证服务器，认证服务器对用户进行认证，同时会将认证服务器的 Cookie 写入客户端，认证通过之后，认证服务器将重定向到 web 服务器[20]。其次，用户访问其他 web 服务器的时候也会被重定向到认证服务器，只要之前用户已经在认证服务器上认证过，认证服务器就会采取免认证方式直接重定向到 web 应用服务器从而达到单点登录的效果。

2.1.2 基于角色的访问控制

基于角色的访问控制（RBAC）通过引入角色这一概念，将角色抽象为有权完成一定事务的命名组，事务就是执行一段程序或部分程序的一个过程，角色就是给有权拥有某种能力的人赋予权限的一种抽象。RBAC 的核心就是用户不直接获得某种权

利，而是通过获取某种角色来间接获取对应的权利[21]。RBAC 关注点在于角色与权限之间以及用户与角色之间的关联关系，而且这种关系是一种多对多的关系，也就是说用户可以同时获得多个角色，也可以为某一角色赋予多项权限[22]。其实现方式如图 2-3 所示。

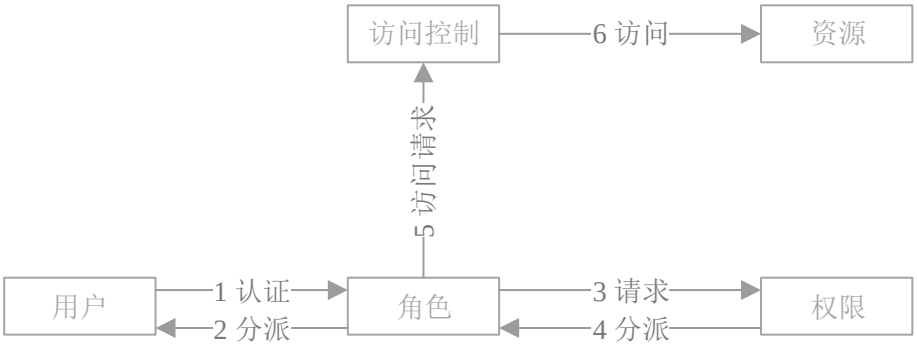

图 2-3 RBAC 实现权限分派方式

RBAC 的特点在于给用户授权的方式，RBAC 从控制主体获取角色的角度出发，通过给主体赋予角色将权限和主体进行绑定，主体和权限不直接关联。在 RBAC 中，角色发挥主体和权限之间的桥梁作用，角色一般有系统管理员直接管理，而且主体没有办法对已获取的角色所绑定的权限进行修改，也不允许用户将自己通过角色获取的权限进行再授权[23]。通过采取这种授权模式，可以满足企业灵活授权的需求，也方便了管理员对权限的管理，是实施企业级安全策略一种行之有效的手段。

RBAC 访问控制具有以下特点：

1. RBAC 通过对一个角色授权可以实现对一组用户授权，避免了对用户单个授权的繁琐，这种灵活性使得整个授权机制得到简化[24]。

2. RBAC 中的角色的层次关系可以很好的映射到组织内部人员的职责和责任关系。

3. 通过 RBAC 可以方便实现最小特权原则，对于组织内任何实体来说，都只应该得到完成工作所需最小的权利。

4. RBAC 还可以很好的执行职责分离的策略，有些权限从安全的角度出发是不能被分配到同一实体的，职责分离是一种保障安全的基本原则[25]。

2.2 非对称加密技术

非对称加密技术于 1976 年由 M.Hellman[26]首先创立，这种公钥密码体制将对数据的加密能力以及对数据的解密能力分解开来，使用私钥加密的数据只能由公钥解密，由公钥加密的数据只能由私钥解密，这种密钥体制的特点是在知道加密算法和加密密钥的条件下，求解解密密钥在算法上是不可行的。

这种公钥密码体制是在数学原理上是基于单向陷门函数[27]，而不是普通加密算法所采取的的替换或者是置换策略[28]。单向陷门函数简单解释就是定义了集合 A 与集合 B 的映射，对于集合 A 中的每一个元素 a 来说，都有唯一的一个原象 b 属于 B，而且对于 b 可以很简单的计算出他的象 a，但是由 a 计算出它的原象 b 在数学上是不可行的。

公钥密码体制是非对称的，基于独立的公钥和私钥来完成加密和解密的功能，公钥密码体制的典型特点如下：

1. 强机密性，通过公钥加密的数据只可以由对应私钥进行解密，可以很好的防止敏感信息的泄露。

2. 实体确认性，可以采取数字签名来保证数据的来源是确定的实体。

3. 数据完整性，基于数字签名技术可以保证数据无法被篡改。

4. 无法抵赖性，也叫不可否认性，通过数据签名技术可以保证数据的拥有者无法否认自己是数据的拥有者。

2.3 区块链相关技术

2.3.1 P2P 网络

P2P 网络也即对等网络，是一种基于分布式在对等的节点直接分配任务以及工作负载的软件架构，在这个对等网络中，所有的参与节点会共享自己的计算能力和硬件资源，在这个网络中对等节点之间的相互访问不需要通过中间实体，在对等网络中，每一个节点充当两种角色，每一个节点既是资源和服务的提供者，也可以是

资源和服务的使用者。

在 P2P 网络中，所有的节点地位均等，没有主从之分，P2P 网络打破了传统的 C/S 网络架构，整个网络不依赖任何集中的第三方服务器，改变了互联网现在以大网站为中心的现状，做到了网络环境的去中心化，此外，P2P 网络本身具有很强的健壮性和较高的可扩展性。

2.3.2 Hash 函数以及工作量证明

Hash 函数也即散列函数，对于任何给定的输入 A,他会计算出一个对应的输出 H(A)。Hash 函数的主要特点是：

1. 输入 A 可以是任意长度的字符串。
2. 计算输出 H（A）的长度是一个固定值。
3. 计算 H(A)的过程效率很高,Hash 函数算法的时间复杂度是 O(输入 A 的长度)。

对于在区块链中使用的 Hash 函数，还需要具备以下几个性质。

1. 免于碰撞，不会出现 A1 不等于 A2，但是 H(A1)等于 H(A2)。
2. 隐蔽性要好，对于给定的 H(A)，想要逆推 A 的值要保证算法上不可行。

区块链中的工作量证明机制与 hash 函数紧密相关,目前在区块链主要应用场景比特币采取的 SHA256 算法。工作量证明的主要特征是客户端需要对于给定的数据自己附加一个随机数，然后对其进行 hash 运算，hash 运算的结果必须小于某一特定值，这个过程对于客户端而言是一个具有相当难度的过程，而验证方可以很容易的验证客户端是否做了对应的工作，工作量证明的核心特征就是不对称性，工作对于请求方具有一定难度，但是对于验证方可以易验证的。工作量证明的流程图 2-4 所示。

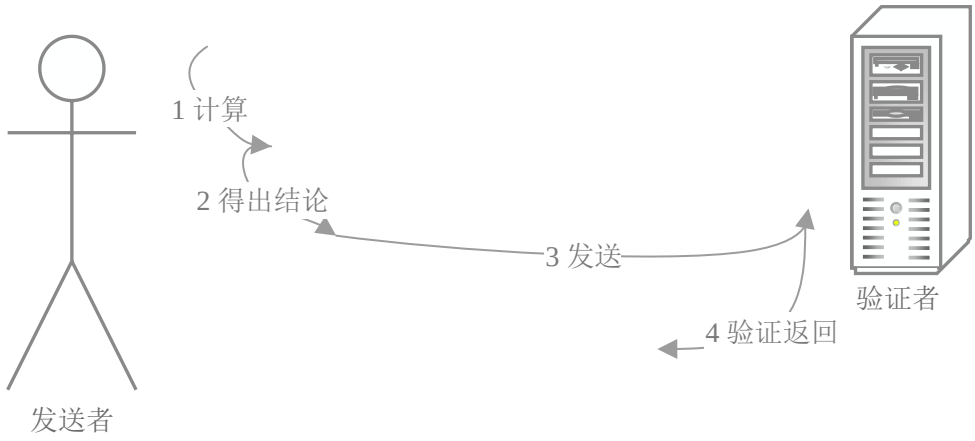

图 2-4　工作量证明机制

2.3.3 区块链以及 Merkle Tree

区块链的由区块组成，每一个区块由区块头和交易数据组成，区块头由 4 字节版本号，4 字节的时间戳，4 字节的随机数，4 字节的难度值，32 字节的梅克尔树根的散列值，32 字节的上一个区块的 hash 值总共 80 个字节大小的数据组成。区块的交易数据则附加在区块头的数据之后[7]。大致结构如图 2-5 所示。

Version	02000
Previous Block hash	134pnva23sn65gnqoiw141mn4gioasfgwfagqef4hasofha24uisfha3sfasgasfa2sgqw
Merkle root	234bnkb42vbuj5k2vhg56k5kjhb5kjvb5h6op23v41gh53n2k5bhj12g4klnb2345kj1
timestamp	3580053
bits	535f0119
nounce	246423432
Transtions count	64
Coinbase transaction	
transaction	
……	

Blockhash

01111102lfnah0falgnalsdff00f7yab96xaf0asdfasdf

图 2-5　区块的数据结构

区块的 80 个字节大小的区块头，就是对于区块进行工作量证明的输入字符串，为了使得区块头的 hash 值能与区块中包含的数据建立关联，采取的方式是对区块中

的交易列表生成 Merkle Tree，并将树根保存到区块头部。

对每个区块中的交易数据建立 Merkle Tree 的过程就是将每笔交易作为叶子节点，分别计算 hash，然后将计算的 hash 值两两合再次计算 hash，直到整个过程只有一个 hash 值产生，这个 hash 值就是 Merkle Tree 的树根。如图 2-6 所示。

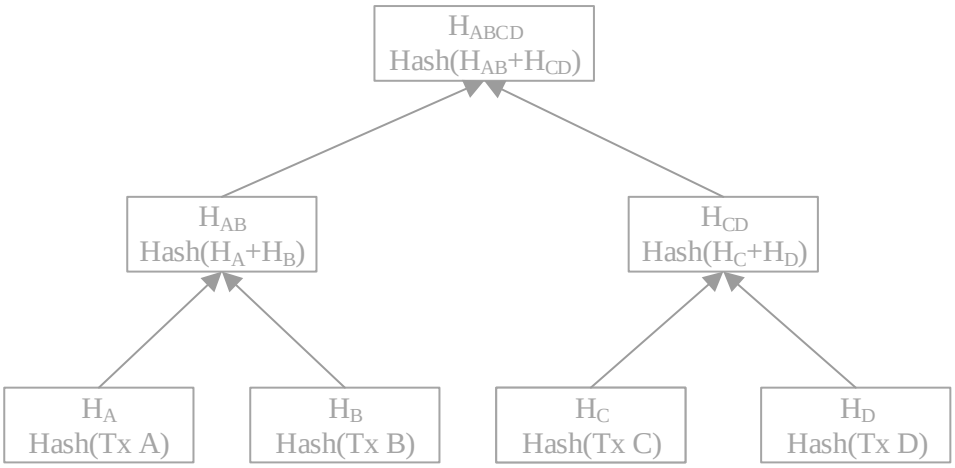

图 2-6　梅克尔树生成过程

区块链中俗称的挖矿就是寻找一个随机数，填入区块头的相应字段，使得整个区块头的 hash 值小于区块头中的难度值指定的标准[29]。这个过程会耗费大量的算力资源。

2.4 本章小结

本章通过对跨域单点登录和基于角色的访问控制进行介绍，了解相关的技术原理和实现的流程，为本文设计的跨域和访问控制机制提供参考，然后分析了非对称加密技术，了解了在访问控制中如何做到信息的安全可靠，最后分析了区块链中的关键技术，着重分析了区块的构成以及区块链网络中挖矿的本质，对本文提出的基于区块链的跨域访问控制机制的设计提供了技术基础。

3 基于区块链的跨域认证与访问控制的方案研究

3.1 已有解决方案的不足

Kerberos,是由 MIT 在 Athena 项目中提出的一个基于可信第三方的认证系统架构，对于两个不同的域 A 和 B，如果想要实现在域 A 中的用户能在域 B 中进行认证，则必须把域 A 的 KDC(密钥分配中心)在域 B 的 KDC 注册成为一个实体[30]。如果在系统中有很多个域，则每一个域都必须在其他所有域中注册自己的 KDC 才可以实现全面的通信，但是当域的个数很多的时候，注册 KDC 的工作量也就会非常庞大。而且，用户域的安全可信完全由可信第三方保证，用户域自身没有任何安全保证机制。此外对于 B/S 网络模式的应用，Kerberos 也很有局限性，因为 Kerberos 要求客户端具有较强的加密解密能力，这会导致客户端的设计变得复杂化。同时，Kerberos 对时间的依赖较大，通过时戳可以很好的防止重放等攻击，但是如果时间差距很大，会直接导致认证的失败[31]。

Passport，是微软推出的基于 Web 应用的统一身份认证系统，系统的架构是由微软提供的 Passport 认证服务器和若干注册的应用服务器组成，基于 Cookie 和重定向的机制实现统一身份认证[32]。该协议的不足之处在于所有的站点共享同一个密钥，如果这个密钥遭到泄露，Cookie 就会处于危险之中。此外，该协议也是基于可信的第三方，如果微软的 Passport 遭到攻击，相应的使用集中权限管理的应用账户也有被泄露的风险。

SAML 协议是 liberty 推出的一套用户认证框架[33]，在该框架中，分为服务提供者和身份提供者，在服务提供者内部是不对用户系统进行管理的。用户需要访问服务提供者提供的资源，会由服务提供者派发一个自动登录到身份提供者的表单，用户在身份提供者处认证之后，身份提供者也会派发一个自动登录到服务提供者但是带有用户认证信息的表单，从而实现用户信息的认证[34][35]。这种方式有两个缺点，首先 SAML 本身是基于 XML 的，在表单中提交的数据冗余度会较高[36]。其次 SMAL 协

议中，用户信息的认证统一由可信的身份提供者进行管理，一旦身份提供者遭受攻击，则整个以该身份提供者来进行认证的服务提供者全部无法对外提供服务。

3.2本方案改进与创新

3.2.1 基于承诺协议

本方案采用承诺（commitment）来达成共识，这里的承诺是一个数字化的过程。类比如下动作，选定一个数字，将数字装进信封，然后将信封放到一个人人可见的地方，这样做完之后，相当于当事人对信封中的数字做出了承诺，信封被打开之前，虽然当事人对数字做出了承诺，但该数字对于其他人依旧是秘密[37]。

本方案承诺协议由如下两部分组成：

com:=commit(msg,nonce)，承诺函数将信息（msg）和一个临时随机数（nonce）作为输入，输出就是一个承诺。

verify（com，msg，nonce），验证函数将某个承诺输出（com），临时随机数（nonce）及信息（msg）作为输入，如果 com==commit(msg,nonce)，则验证通过，否则验证不通过。

约定承诺协议的实施方案采用 hash 运算：

commit(msg,nonce):= H(nonce||msg)

其中，nonce 为固定长度的随机数。为了承诺一个消息，将随机数 nonce 与信息拼接成串并返回其 hash 值来作为承诺的输出，为了便于验证，其他人需要计算拼接之后所得串的 hash 值，来对比输出是否与承诺相同。

该承诺过程要达成安全性，必须要求两个安全特性成立：

隐蔽性：已知 H(nonce||msg)，没有办法找到 msg。当（nonce||msg）满足高阶最小熵（high min-entroy）的概率分布，在给定的 H(nonce||msg)条件下确定（nonce||msg）是不可行的。

约束性：也即免于碰撞，没有可行的办法找到两组（msg,nonce）和（msg’,nonce’），在 msg!=msg 的情况下，而 H(nonce||msg)==H(nonce’||msg’)。

3.2.2 去中心化

Kerberos 对 KDC(密钥分配中心)有依赖，Passport 对微软的 Passport 服务有依赖，以及 SMAL 对 LDP 有依赖。本文从区块链去中心化的思想出发，设计一个基于区块链的跨域认证方式，采取这种方式的直接结果就是授权信息由各个参与到区块链网络中的所有域节点共同维护，而不再是依赖于各个域节点共同信任的可信第三方。

3.2.3 去信任化

在由区块链网络组成的跨域与授权系统中，所有的域信息和授权信息都是加密之后存放在区块链中。在基于区块链的信任网络中，所有域节点会提前协商授权信息，授权信息一旦写入区块，在信任网络中对所有域节点都可见，所有域节点共同维护这些授权信息。对其中某一个域节点而言，由于有密钥加密，他可以读取与自己授权相关的区块记录，同时又无法读取其他域节点的授权信息区块记录，也就无法使用伪造的权限信息去跨域登录其他域节点。

3.2.4 授权数据以区块链形式分散存储

此方案摒弃了 Kerberos,Passport 等协议用户授权信息集中存储或集中认证的缺点。在基于区块链的信任网络中，各个域节点都保存了一份区块数据的副本，任何一个域节点发现自己的区块数据不正确的时候，都可以请求其他域节点同步区块数据[38]。与此同时，任何节点都必须将自己的请求授权信息和被授权信息即时向外广播并最终写入区块，以防止单个域节点被攻击从而造成已授权信息的丢失。基于这种类似备份的机制保证数据的安全可靠[39][40]。

3.3 授权与身份认证系统的研究方案

3.3.1 系统方案设计

本文所提出的基于区块链的跨域访问授权方案，具备去中心化的特点，各域节点授权协商采用的是直接点对点方式进行，协商结果需要写入区块链备份。其概要

如图 3-1 所示。

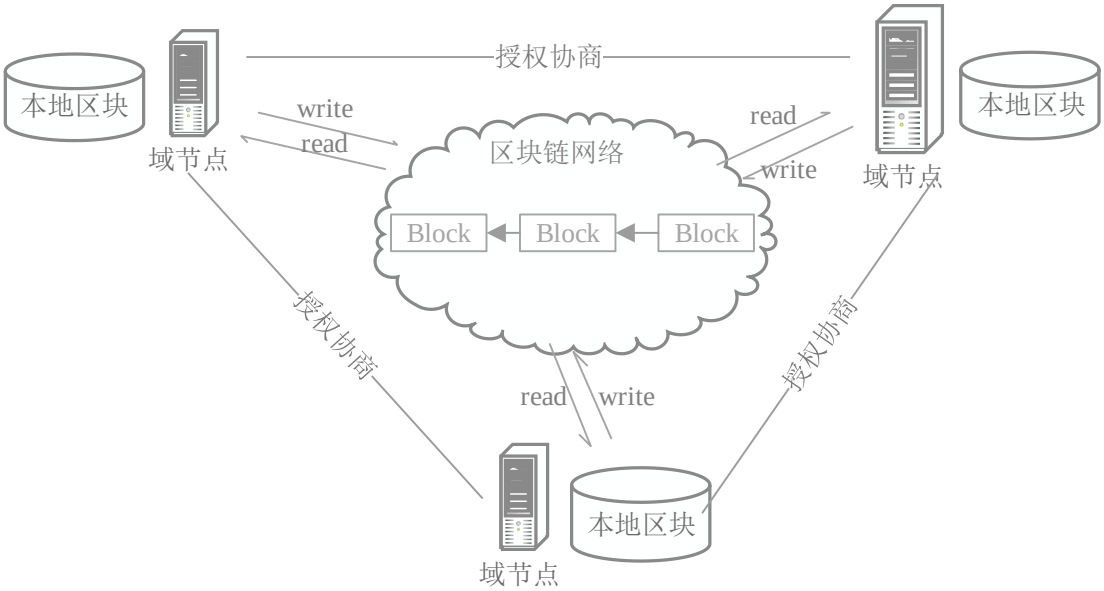

图 3-1　基于区块链的跨域访问授权方案

针对于跨域认证场景，对于每一个域节点而言，必须要完成本域用户的身份认证与授权，域间授权协商，跨域登录，以及区块同步与验证四大功能。本域用户的身份认证与授权主要用来在本域校验用户合法性，域间授权协商主要用来交流域节点之间的请求授权信息，跨域登录主要用来实现单点登录，区块的同步与验证主要用来保证域节点本地区块数据的正确性，这主要通过定时和其他节点通过搭建整个区块链的 Merkle Tree 来校验区块数据副本完整性实现。

为了提高区块链本身被攻击的难度,区块链的长度需要保持一定的速度增长[41]，但是域间授权协商一旦确定，短时间不会更改，所以，本设计方案的区块设计将整个可信网络中的所有节点的相互授权信息与跨域访问的日志信息都统一记录到同一个区块链中，这样只要有用户的登录行为发生且存在跨域行为，跨域日志也会不断的产生，这样区块链中的区块长度就可以保持一定的速率增长。随着区块高度的不断增长，攻击区块链本身的难度成指数级增加。

从系统整体出发，系统中主要存在以下两种角色：

1. 可信域：指需要跨域登录其他域的域节点，也指需要其他域给予跨域权限的域节点。

2. 授权信任域:指被其他域跨域访问的域节点,也指给其他域授予权限的节点。

为了快速的做出权限校验，各域节点本身维护如下三个区块索引列表：

1. authorization_req：该表记录域节点自身的请求权限记录，其中表中字段 requestDomain 记录请求的授权信任域标识，字段 requestIndex 记录请求权限区块记录的区块高度和hash,字段responseIndex记录授权区块记录的区块高度和hash。

2. authorization_res：该表记录域节点授权的记录，其中表中字段 response Domain 记录被授权域的标识信息，字段 responseIndex 记录授权区块记录的区块高度和 hash。

3. blockorder：该表维护域节点所保留区块链副本中区块的顺序关系。

3.3.2 本域用户身份认证与授权

本域用户在本域登录的时候，首先必须通过本域服务器的认证。用户认证完成之后才可以获取到本域的角色信息，也只有通过本域的认证，本域才会将用户信息发送到其他授权信任域进行跨域登录。

用户在本域的权限控制基于 RBAC 模型，用户作为主体登录到域节点之后，域节点会分配给用户一定的角色，访问主体根据域节点预先定义的访问策略授权对客体资源的访问[42]。

其本域用户权限判断过程如下：

1. 设 U,R,P,S 分别表示本域的用户集合，角色集合，权限许可集合以及会话集合。

2. 设 PA :P×R 表示权限许可与角色之间的多对多映射关系。

3. 设 UA :U×R 表示用户和角色之间的多对多映射关系。

4. 设 RH :R×R 表示的是角色之间的偏序关系，即角色之间的等级划分，可以使用≥符号表示。

5. 设 S→U 表示特定会话到某用户 user(si)的映射函数。

6. 设 S→R 表示某次会话所映射的角色子集 roles(si)={r|(r'≥r)[user(si, r')∈UA]}的映射函数。

7. 由 S→R,PA 可以推得某次用户会话的许可 Ur∈roles(si){p|(r''≤r)[(p, r'')∈PA]}并最终决定对特定资源的访问权限。

3.3.3 基于区块链的域间授权协商

域间授权协商是整个跨域系统的基础，所有加入到网络中的域节点都只有通过预先的跨域协商之后，才可以让本域的用户跨域登录其他授权信任域。

域间授权协商首先需要每个域节点相互交换自己的公钥信息，为保证信息的安全，所有的跨域权限请求和跨域授权应答都是基于密钥加密传送。整个过程分授权协商和区块发现两个阶段。

算法 3-1 ： 发送授权请求算法

输入： 协商决策

输出： 写入区块链的授权信息

1： FUNCTION sendAuthReq

2： $TR_{X\to Y}$:= createTR($Rmap_{XY}$,$info_Y$);//$Rmap_{XY}$是域 X 和域 Y 实现约定的角色映射

3： IF(trin($TR_{X\to Y}$));//判断 $TR_{X\to Y}$是否已存在于 $Rlist_X$中

4： THEN RETRUN;

5： ELSE $Rlist_X$。insert($TR_{X\to Y}$);//将新增条目添加到 $Rlist_X$。

6： nonce:= generateRandom();//$info_X$为 X 域标识，R_X为域 X 角色表

7： str := pk_Y(sk_X(R_X||$info_X$||nonce)); //异步发送授权请求并获取应答

8： pk_X（sk_Y($I_{Y\to X}$||T||$R_{Y\to X}$||nonce)）: = sendNogitiation($info_Y$,str);

9： str2:=sk_X（pk_Y($I_{Y\to X}$||T||$R_{Y\to X}$||nonce)）;//解密域 Y 的授权应答数据

10： s_X:= (TimeStamp||$info_X$||sk_X($I_{Y\to X}$||T|$R_{Y\to X}$||pk_Y));

11： writeBlock(s_X); //授权结果写入区块链。

12： END IF

13： END FUNCTION

在授权协商阶段，可信域发送授权请求，请求时提交本域角色信息，域信息，并等待授权信任域的授权响应。当授权响应到达时，可信域需要将授权结果广播至区块链存储。

算法 3-2 ：响应授权请求

输入：请求授权域信息与协商决策

输出：写入区块链的授权信息

```
1:    FUNCTION sendAuthRes
2:       rolemap;//rolemap 用来临时存放授权角色映射关系
3:       //从域 X 发送的信息解密角色信息和域标识
4:       info_X, R_X := sk_Y(pk_X(R_X||info_X||nonce));
5:       FOR all R' ∈R_X DO
6:         //依据约定的角色映射关系 Rmap_Y-X 为 R' 匹配跨域角色
7:         role := matchRole(R',Rmap_Y-X);
8:         rolemap.insert(R',role);
9:       END FOR
10:        TS_Y→X := createTS(rolemap,info_X);//生成授权记录
11:        Slist_Y。insert(TS_Y→X);//将新增条目添加到 Slist_Y。
12:        //I_X表示开放给域 X 的跨域接口，T 表示授权过期时间
13:        str:= pk_X(sk_Y(I_X||T||rolemap||nonce));
14:        s_Y:= sk_Y（T|| rolemap ||info_X）;
15:        writeBlock(s_Y);//授权结果写入区块链。
16:        RETURN sendNogitiation(info_x,str);//将授权结果告知域 X。
17:    END FUNCTION
```

在响应授权请求阶段，授权信任域依据可信域提交的角色信息以及域信息，并结合事先约定的角色映射关系表对可信域中相应角色授予权限。并将授权结果通过区块链广播存储。

在区块发现阶段，授权双方需要在本域请求权限列表与授权列表中记录已经持久化到区块链网络中的授权区块数据标识。

如果是域节点主动发现记录授权区块，当域节点获取到一个最新区块数据并且确定授权区块属主为本域时，需要判定是本域发出的请求授权信息块，还是授权信息块，并依据判定结果做出不同处理。

算法 3-3 ：域节点区块发现响应算法

输入：被区块链网络成功打包区块

输出：区块高度与区块 hash

```
1：  FUNCTION noteAuthBlock
2：    //membership 函数用于判断授权区块中的记录数据属主是否为域节点
3：    IF(membership(blockdata))
4：      THEN
5：        //取得区块中的记录并采用私钥解密数据
6：        itemInfo := sk(figureOut(blockdata));
7：        IF(isAuthReq(itemInfo))//如果是请求授权块
8：          THEN
9：            TR:=createTR(itemInfo);//依据 itemInfo 构造 TR
10：           IF(trin(TR))
11：             //如果记录存在，将请求授权区块信息记录在 TR 中。
12：             TR.requestIndex := [Height(blockdata):H(blockdata)]
13：           END IF;
14：          ELSE
15：            TS:=createTS(itemInfo);//依据 itemInfo 构造 TR
16：            IF(tsin(TS))
17：               //如果记录存在，将请求授权区块信息记录在 TR 中。
18：               //Height，H 函数分别用于计算区块的高度与 hash 值
19：               TS.responseIndex := [Height(blockdata):H(blockdata)];
20：    sendAuthRes([Height(blockdata):H(blockdata)],itemInfo.info);
21：           END IF;
22：       END IF;
23：      ELSE RETUIRN;
24：     END IF;
25：    END FUNCTION;
```

在域节点被动更新授权区块阶段中，当域节点被动发现记录授权信息的区块时，该区块是本域从其他域收到一个定向传播至本域的授权区块信息，需要依据本域保存的区块副本对区块信息进行校验并存储。这个区块信息可以被本域将来用作权限校验和权限审计。对于本域中的某一项权限而言，只有当本域被动收到与该权限相关的授权区块时，才能确认该项授权是否完整。

算法 3-4 ：域节点被动更新授权区块算法

输入：授权信息区块高度与区块 hash

输出：

```
1： FUNCTION noteAuthBlockRes
2：     //根据区块高度和区块 hash 从区块副本中提取记录并采用私钥解密数据
3：     //如果本地提取的区块副本所计算的 hash 值与参数 hash 不等，则返回。
4：     itemInfo := figureOutDomain([height,hash]));
5：     TR:=createTR(itemInfo);//依据 itemInfo 构造 TR
6：     IF(trin(TR))//判断本域节点是否曾对 TR 关联的域发起过授权请求
7：         THEN
8：         //持久化记录区块数据
9：         TR.responseIndex:=[height:hash];
10：       ELSE RETURN;
11：     END IF
12：   END FUNCTION
```

对于 X 域的请求授权记录表中某一项授权记录 TR 而言，只有 requestIndex 以及 responseIndex 两字段均指向合法的本地区块副本才算做授权完整。

如图 3-2 所示为当可信域（假设为 A）请求授权信任域（假设为 B）给予跨域登录权限并完成授权之后，A 域和 B 域两个授权相关列表中各自维护的授权区块记录状态数据图。

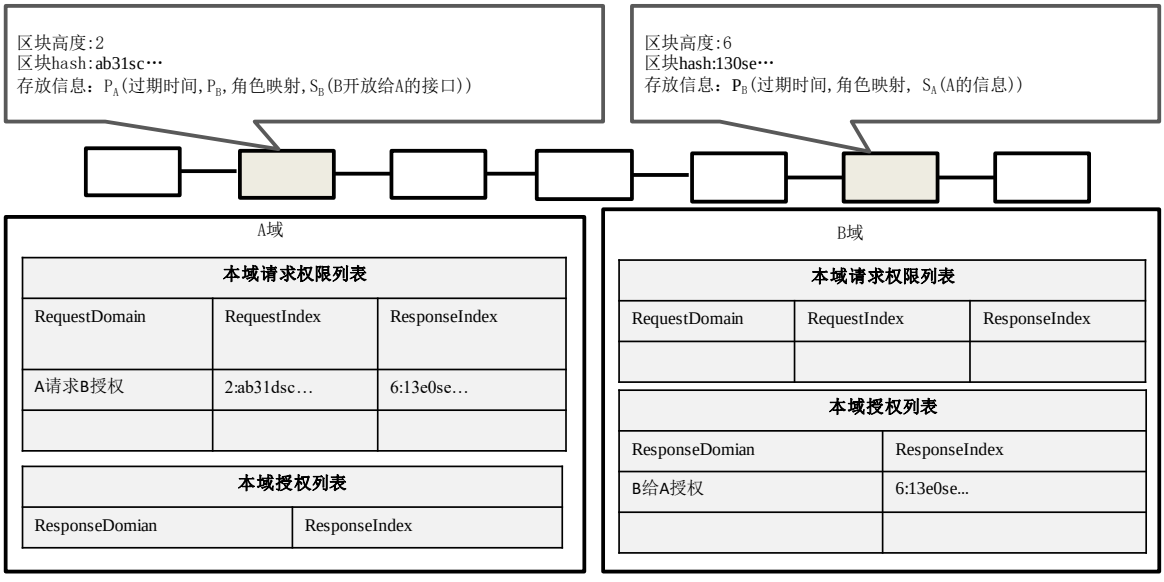

图 3-2　A 域与 B 域的授权示例图

3.3.4 基于区块链的跨域认证

采用区块链去中心化的方式实现跨域认证的过程实质是一个权限过滤的过程。首先需要依据本域请求权限列表过滤所有已授权完成的记录，然后根据已授权完成的记录获取并解密相关区块数据并提取相关目标域的公钥、授权过期时间、目标跨域接口以及角色映射等信息，最后依据授权过期时间以及当前登录用户的角色对当前用户有权跨域访问的目标域进行过滤并拼接 JS 跨域串。

算法 3-5 ：跨域权限过滤算法

输入：本地请求授权列表

输出：跨域 js 串

```
13:    FUNCTION crossDomain
14:        K:={TR|TR∈Rlist};
15:        Uname,Urole1; blockItems;crossDomains;
16:        FOR all k IN K
17:           Kblock := readBlock (k.requestIndex); //读取授权区块数据
18:            IF(H(Kblock)==k.requestIndex.hash)//校验区块是否受损
19:               THEN  blockItems.add(sk(Kblock.item));
20:               ELSE  syncSingleBlock (k.requestIndex.height);
21:            END IF;
22:        END FOR;
23:       FOR all bi IN blockItems
24:        time:= bi.T; rolemap := bi.Rmap;pk:=bi.pk;
25:        IF(time > Tnow & rolemap[Urole1]!=null)//Tnow表当前时间
26:          THEN   crossDomains.add([bi.I,rolemap[Urole1],infot,pk);
27:        END
28:       END FOR;
29:      FOR all item IN crossDomains
30:          str: =
31:         (from=info&username=Uname&roleFrom=Urole1&roleTo=item[1]);
32:         vender(str);//将跨域信息串通过 JS 渲染在用户登录成功页面
33:    END FOR
34:    END FUNCTION;
```

以可信域 A(www.a.com)跨域登录到授权信任域 B(www.b.com)为例,表 3-1 说明了可信域 A 内部形成的最终待跨域列表结构。

表 3-1　可信域 A 过滤权限之后得到列表结构样例

可信域	本域用户名	可信域角色	授权信任域	授权信任域角色	登录接口
www.a.com	ausername	role_a1	www.b.com	role_b1	www.b.com/interface_a

可信域根据过滤所得列表生成 javascript 异步请求代码段,其中请求参数使用授权信任域公钥加密形成加密串 ST=www.b.com/interface_a?pk(from=www.a.com&username=ausername&roleFrom=role_a1&roleTo=role_b1)传送以保证安全性。

生成的请求代码嵌套在用户在本域登录成功之后的返回页中，由于 javascript 在浏览器加载之后会自动执行，以此实现将登录信息发送到授权信任域节点。其代码如下所示：

```
<html>
<head>
……
  <!—其中，src 请求的参数需要用 B 域公钥加密，此处已解码-->
<script type=”javascript” src=”www.b.com/interface_a?from=www.a.com&username=ausername&roleFrom=role_a1&roleTo=role_b1”></script>
……
<head>
<body>
    …… <!—页面主体代码-->
</body>
</html>
```

权信任域 Y 收到跨域请求，需要依据跨域参数查找本域授权列表，并根据授权列表中指向的授权区块获取授权详情，然后依据授权详情中指定的授权过期时间以及角色映射信息对当前用户的跨域合法性做出判断,如果跨域判定为合法,需要采取

P3P 协议将 cookie 回传到客户端[43]。

算法 3-6 ：跨域权限校验算法

输入：通过网络传输的跨域请求数据

输出：cookie 信息

```
1： FUNCTION varifyCrossDomain
2：   //使用私钥解密跨域数据，从中提取出跨域源，用户名，源角色和目标角色
3：   [info,uname,urole1,urole2]:=
4：   sk(from=info&username=uname&roleFrom=Urole1&roleTo=Urole2);
5：   TS := createTS(info);//根据 info 构造 TS 条目
6：   IF(tsin(TS))//tsin 函数查看 TS 条目是否存在于本地授权列表中
7：     THEN
8：       Kblock := readBlock（TS.responseIndex）;//读取授权区块数据
9：       IF(H(Kblock)==TS.responseIndex.hash)//校验区块 hash
10：        THEN
11：          //从授权区块数据记录提出授权过期时间以及角色映射信息
12：          T,Rmap := figureOut(sk(Kblock.item));
13：        ELSE
14：          //区块数据 hash 校验出错，请求单区块同步
15：          syncSingleBlock (TS.requestIndex);
16：      END IF;
17：    IF(T>Tnow & Rmap(urole1)==urole2) //Tnow表示当前时间
18：       //会话有效，生成会话信息并用 P3P 协议回传 Cookie
19：    END IF
20：    END FUNCTION;
```

3.3.5 区块同步与验证

区块同步分两种情况，一种是各个域节点利用空闲时间通过 Merkle Tree 校验本域所有区块的完整性，定时同步本域的所有区块。另外一种情况就是区块校验出错的时候，请求特定区块数据即时同步。

全区块同步发生在本域区块同步水平落后于平均水平或本域区块出错时，本域需要进行完整区块链副本更新。

算法 3-7 ： 区块数据副本全链校验算法

输入： 本地区块数据副本

输出： Merkle Hash 及 区块高度

```
1:    FUNCTION syncBlockChain
2:      merkleRoot:=figureOutMerkleRoot();//计算本域节点梅克尔树根 hash
3:      blockHeight;ress:=syncGlobalBlockChain(blockHeight,merkleRoot);
4:      sortBaseHash(ress); //将 ress 根据 hash 排序。
5:      IF（ress[0]==ress[50]）//判断响应结果是否存在占比 50%以上的响应
6:     THEN
7:      IF(blockHeight<ress[0].blockHeight)
8:        syncBlock(ress[0].blockHeight,blockHeight)
9:      ELSEIF(blockHeight==ress[0].blockHeight&
10:           merkleRoot!= ress[0].merkleRoot)
11:           syncBlock(0,ress[0].blockHeight)
12:         END IF
13:       ELSE //不存在占比 50%的响应，说明网络较差，稍后在同步。
14:      END IF;
15:    END FUNCTION;
```

单区块校验多发生在权限校验过程中发现区块数据 hash 值与预期不匹配时。临时请求更新当前区块数据。

算法 3-8 ： 单区块校验算法

输入： 本地单区块数据副本

输出： Merkle Hash 及 区块高度

```
1:     FUNCTION syncSingleBlock
2:       blockHeight; lockHash;
3:       ress:=sync SingleBlock(blockHeight); sortBaseHash(ress);
4:       IF（ress[0]==ress[50]）//判断响应结果是否存在占比 50%以上的响应
5:          THEN
6:           IF(blockHash!=H(ress[0]))
7:            THEN UpdateBlock(ress[0]); //采用 ress[0]更新区块数据副本。
8:         END IF
9:       ELSE //不存在占比 50%的响应，说明网络较差，稍后在同步。
10:     END IF;
11:     END FUNCTION;
```

3.4 安全性分析

3.4.1 节点主动欺骗安全性分析

由于每一项授权信息都由可信域和授权信任域分别记载授权记录，而且这个授权记录是广播至区块链来保存的，可以做到无法篡改。所以，两方均无法篡改权限的过期时间，当可信域修改权限有效期企图非法跨域登录到授权信任域，授权信任域在处理跨域登录请求时可以实时发现错误，并及时清除授权记录并告知可信域原授权已过期要求其重新发起授权请求。如果是授权信任域故意丢失本地授权列表中授权记录，此时可信域可以提供在域间授权协商时授权信任域发送给自己的授权确认信息区块编号和区块的 hash 值向授权信任域追责，而且授权信任域无法抵赖[44]。

3.4.2 节点区块数据被篡改安全性分析

定理：如果在授权信息被写入到区块链网络中并得到多个区块确认，理论上，授权信息被攻击者修改难度较大，而且其难度系数随着确认区块数的增长呈指数级增加。

证明：因为所有的授权信息全部存放区块链中，攻击者需要攻击区块链，从理论上讲，必须要伪造长度大于实际区块链长度的新链来抵消实际区块链条才可以。假设实际区块链和攻击者企图伪造的假链都是基于一定概率在随机延长，攻击者成功攻击与否涉及到和实际区块链网络中的诚实节点比拼算力的过程，这个过程充满随机性。另外，定义成功事件表示实际区块链网络中的诚实节点计算出一个区块，与失败节点拉开一个单位差距。而失败事件表示被攻击者计算出一个区块，与成功节点所持有链的差距减少一个单位。攻击者想要不断努力弥补与成功节点所持有链的差距的过程可以看做是一个赌徒破产问题，我们可以依据概率分析计算出攻击者想要成功攻击实际区块链的概率值。

假设定义 p 标识为诚实节点成功计算出下个区块的概率，定义 q 标识为攻击者成功计算出下个区块的概率，定义 q_x 标识为攻击者最后计算出 X 个区块后并赶超诚实区块链的概率。那么有：

$$q_x = \begin{cases} 1 & p \le q \\ \left(\frac{q}{p}\right)^x & p > q \end{cases} \tag{3-1}$$

基于实际算力的考量，我们假定 p>q,我们可以知道攻击者成功概率随着实际区块链区块数的增长是呈指数下降的，所以，随着时间的推移，攻击者成功的概率会越来越小。

我们假设可信域作为攻击者想单方面对授权信息进行攻击，比如延长授权信任域给自己的授权有效期，可信域必须不断伪造新的区块。从攻击者开始伪造节点开始，当实际区块链上已经延长了 x 个区块时，我们不能肯定攻击者已成功计算出多少个假区块数据，假定诚实节点基于实际区块链每计算出一个区块数据需要耗费一个平均时间，那么攻击者伪造的新链长度将会符合泊松分布[45]，其期望着为：

$$\lambda = \mathrm{x}\frac{q}{p} \tag{3-2}$$

基于此种假设，为了计算攻击者成功攻击的概率，我们需要取得攻击者新链中区块新增数量的泊松分布的概率密度，乘以在该数量下攻击者依旧可能攻击成功的概率。设：

$$\mathrm{Q}(\mathrm{x},\lambda) = \sum_{k=0}^{x-1}\frac{\lambda^k}{k!}e^{-\lambda} \tag{3-3}$$

现在判断期望λ可能取值的三种情况。

$$\begin{cases} 1-\mathrm{Q}(\mathrm{x},\lambda\mathrm{x}) \sim \frac{1}{1-\lambda}\frac{1}{\sqrt{2x\pi}}e^{-x(\lambda-1-\log\lambda)} \text{ 且 } \mathrm{Q}(\mathrm{x},\lambda\mathrm{x}) \to 1 & 0<\lambda<1 \text{ 时} \\ 1/2-\mathrm{Q}(\mathrm{x},\mathrm{x}) \sim \frac{1}{3}\frac{1}{\sqrt{2x\pi}} \text{且 } \mathrm{Q}(\mathrm{x},\mathrm{x}) \to 1/2 & \lambda=1 \text{ 时} \\ \mathrm{Q}(\mathrm{x},\lambda\mathrm{x}) \sim \frac{1}{1-\lambda}\frac{1}{\sqrt{2x\pi}}e^{-x(\lambda-1-\log\lambda)} & \lambda>1 \text{ 时} \end{cases} \tag{3-4}$$

我们可以推算出当攻击者落后 x 个区块，并结合概率密度 k 可以得出赶超的概率是：

$$\mathrm{P}(\mathrm{x},\mathrm{k}) = 1-\mathrm{Q}\left(\mathrm{x},\frac{\mathrm{kxq}}{\mathrm{p}}\right)+\left(\frac{\mathrm{q}}{\mathrm{p}}\right)^x e^{kz\frac{p-q}{q}}Q(x,\mathrm{kx}) \tag{3-5}$$

当概率密度 k=1 时，攻击者一定可以成功的概率为：

$$P(x,1) = 1 - Q\left(x, \frac{xq}{p}\right) + \left(\frac{q}{p}\right)^{x} e^{z\frac{p-q}{q}} Q(x,x) \quad (3\text{-}6)$$

当$\lambda > 0$时，$Q(x, \lambda x)$随着$x \to +\infty$，是会逐渐渐进于一个固定值。

假设 q=0.1，攻击成功的概率随 x 的变化趋势如表 3-2 所示。

表 3-2　q=0.1 时攻击成功的概率（变量 x）

攻击者落后区块数	攻击者攻击成功概率
X=1	P=0.2035893
X=2	P=0.0535729
X=3	P=0.0148723
X=4	P=0.0023415
X=5	P=0.0005092

假设 q=0.2，攻击成功的概率随 x 的变化趋势如表 3-3 所示。

表 3-3　q=0.2 时攻击成功的概率（变量 x）

攻击者落后区块数	攻击者攻击成功概率
X=1	P=1.0000000
X=4	P=0.1934151
X=8	P=0.0194534
X=12	P=0.0034915
X=16	P=0.0025362
X=20	P=0.0000513

假设必须使得攻击成功的概率 P<0.1%，那么安全的领先区块个数统计如表 3-4

所示。

表 3-4 P<0.1% 时安全区块数统计（变量 q）

攻击者产生下个区块概率	为保证安全攻击者需落后区块数
q=0.10	X=5
q=0.14	X=9
q=0.18	X=12
q=0.22	X=17
q=0.26	X=24
q=0.30	X=51
q=0.34	X=92

结论：通过计算数据发现，当 q=0.1 时，只要攻击者试图修改的节点落后于当前区块高度数为 5 个区块时，攻击成功的概率就已经小于 0.1%。当 q=0.2 时，要使得攻击成功的概率小于 0.1%，当前区块高度值必须领先攻击者攻击节点大约 24 个区块才可以。虽然攻击者产生下一个区块的概率仅仅提升 0.1%，但是对于某一个区块的区块确认数却提升了大约 20 个区块。所以要保证整个区块链的安全性，必须避免潜在攻击者算力过大。但在特定网络环境中，当 q 取一个固定的概率时，随着 x 的增大，攻击者攻击成功的概率呈指数下降。

3.5 本章小结

本章首先分析了现有跨域解决方案的不足，针对这些不足并结合区块链技术提出了基于区块量的跨域访问方案,并分析了使用区块链解决跨域问题的优势。然后，提出了该改进后的跨域方案的设计思想，并对主要的功能模块进行了阐述。最后，对该方案涉及的安全问题进行了分析并对该方案如何做到数据安全做了形式化证明。

4 基于区块链的跨域认证与访问控制系统的设计

本章在第三章对基于区块链和 cookie 的跨域方案研究的基础之上,对系统从整体架构上进行了分层设计并采用 Java 语言设计实现了系统的各个模块。

4.1 系统的总体架构

系统的总体架构如图 4-1 所示。

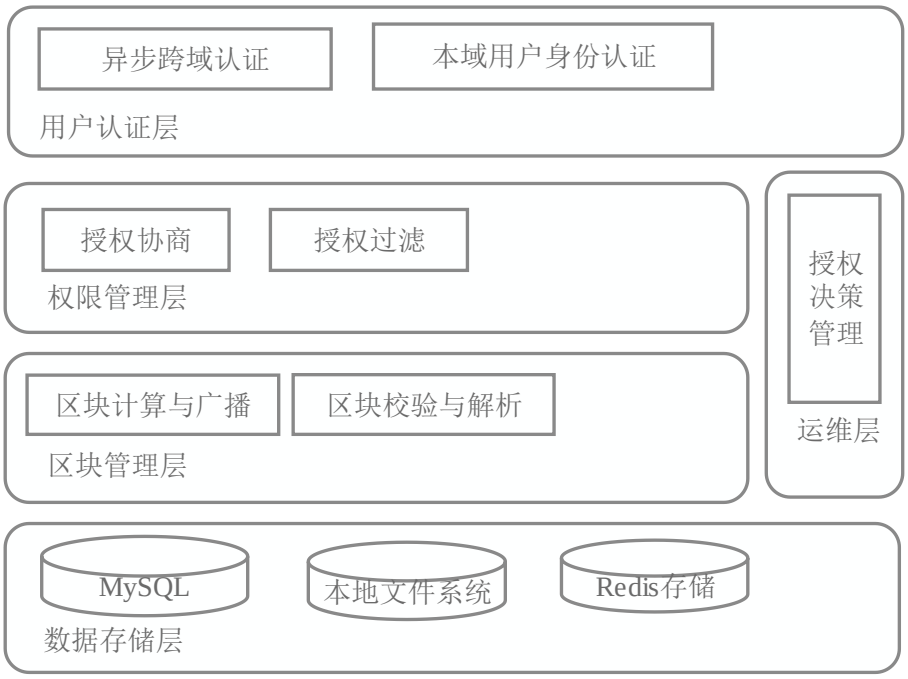

图 4-1　系统设计架构图

系统核心分为四个层次。其中用户认证层完成用户认证，由两个模块构成，本域用户身份认证模块完成域内用户认证，异步跨域认证模块协助本域用户完成单点登录；权限校验层主要完协助域间用户实现单点登录，其中授权协商模块用于域间跨域权限协商，其决策由运维层管理员决定，授权过滤主要用于单点登录时的权限校验；区块管理层主要用于保证本域区块数据的完整性，其中区块计算与广播模块主要用于域节点挖矿，区块同步模块主要用于域节点主动校验区块完整性并请求同

步不正确的区块数据，区块解析模块主要用于验证接收到的广播区块数据的正确性以及用于解析并提取区块数据用于权限校验使用；数据存储层则用于数据维护，其中 Redis 主要用于存放待打包的数据，本地文件系统用于存放已经序列化的区块数据，而通过 MySQL 数据库来维护本地区块链副本中区块的顺序性以及为授权区块建立相关索引。

下面依据软件架构图，首先介绍本系统涉及到的主要交互流程，然后介绍本系统采取的区块数据结构，最后采取自底向上的方式介绍软件主要模块的设计。

4.2 认证与授权流程分析

本节所有流程分析基于可信域 A(www.a.com)和授权信任域 B(www.b.com)来分析，跨域采取的流程是假设用户首先登录可信域 A，然后跨域登录授权信任域 B。

4.2.1 域间授权流程分析

不同域之间跨域授权流程图如图 4-2 所示。

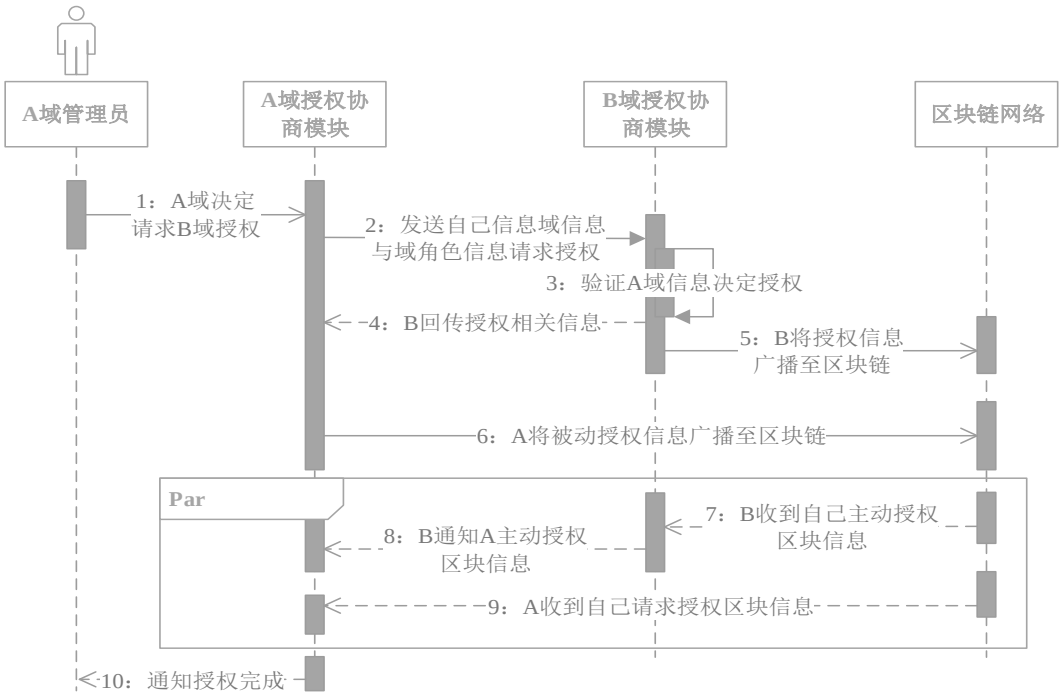

图 4-2　不同域之间的跨域授权协商流程

域间授权协商具体过程描述为：

1. 域A的管理员后台发起一条请求跨域权限指令，指明希望域B给域A授权，使得登录域 A 的用户可以免登陆直接登录域 B。

2. 域 A 将自己的域信息，域内角色信息，通过自己的私钥加密后拼接一个临时生成的随机数，然后将整个信息先用域 B 的公钥加密，在用自己的私钥加密发送给域 B,请求域 B 的授权。同时域 A 在本地请求授权列表中生成一条请求授权记录，记录的 requestDomian 为域 B 的域名：www.b.com，requestIndex、responseIndex 两字段暂时置空。

3. 域 B 收到域 A 的授权请求信息之后，首先用域 A 的公钥以及域 B 的私钥解密授权请求信息，验证域 A 信息的合法性，并对比域 A 的角色信息决定将本域的角色选择性配对做角色映射来给域 A 的角色授权。同时域 B 在本地授权列表中生成一条授权记录，记录的 responseDomian 为域 A 的域名：www.a.com，responseIndex 暂时置空。

4. 域 B 首先将自己开放给域 A 的跨域登录接口信息，角色映射信息用自己的私钥加密，然后附加本次授权过期时间以及域 A 请求授权的随机数形成请求授权响应信息并采用域 A 的公钥加密该响应信息回传给域 A。

5. 域 B 将自己的授权信息广播至区块链网络中，希望持久化存储本次授权确认信息，存入到区块链中的授权确认信息需要经由域 B 公钥加密，其中包含域 B 给域 A 授权的过期时间，域 A 和域 B 之间的角色映射关系，域 A 的节点信息。域 B 广播本次授权确认信息时，标记记录属主为域 B 自身域名 www.b.com，标记记录类型为 authorization。

6. 域 A 收到域 B 给自己的授权信息之后，首先解密信息，验证随机数来判断授权来源是否是域 B，若验证通过，则将域 B 私钥加密过的域 A 登录接口信息以及角色映射信息，域 B 的公钥信息，本次授权过期时间以及域 B 的域信息等几项数据用域 A 的公钥加密发送至区块链网络持久化存储。域 A 广播本次授权信息记录时，标记记录属主为域 A 自身域名 www.a.com，标记记录类型为 authorization。

7. 域 B 从区块链网络中感知到给域 A 的授权确认信息被持久化以后，更新自己维护的本域授权列表中 responseDomain 为域 A 域名的记录，在记录的 responseIndex 字段记录授权确认信息区块的区块高度和区块的 hash，随后将本次授权确认信息区块的区块高度和区块 hash 发送给域 A。

8. 域 A 收到域 B 发送的授权确认信息区块相关数据之后，首先会根据自己保存的区块链数据副本校验数据的正确性，如果校验通过，则更新自己维护的本域请求授权列表中 requestDomain 为域 B 域名的记录，在记录的 responseIndex 字段记录主动授权记录区块的高度和区块的 hash。

9. 域 A 从区块链网络中感知到本域向域 B 的授权信息记录被持久化以后，更新自己维护的本域请求授权列表中 requestDomain 为域 B 域名的记录，在记录的 requestIndex 字段记录授权信息记录区块的区块高度和区块的 hash。

10. 通知管理员，域 B 给域 A 的授权完成。

4.2.2 跨域身份认证流程分析

跨域登录流程图如图 4-3 所示。

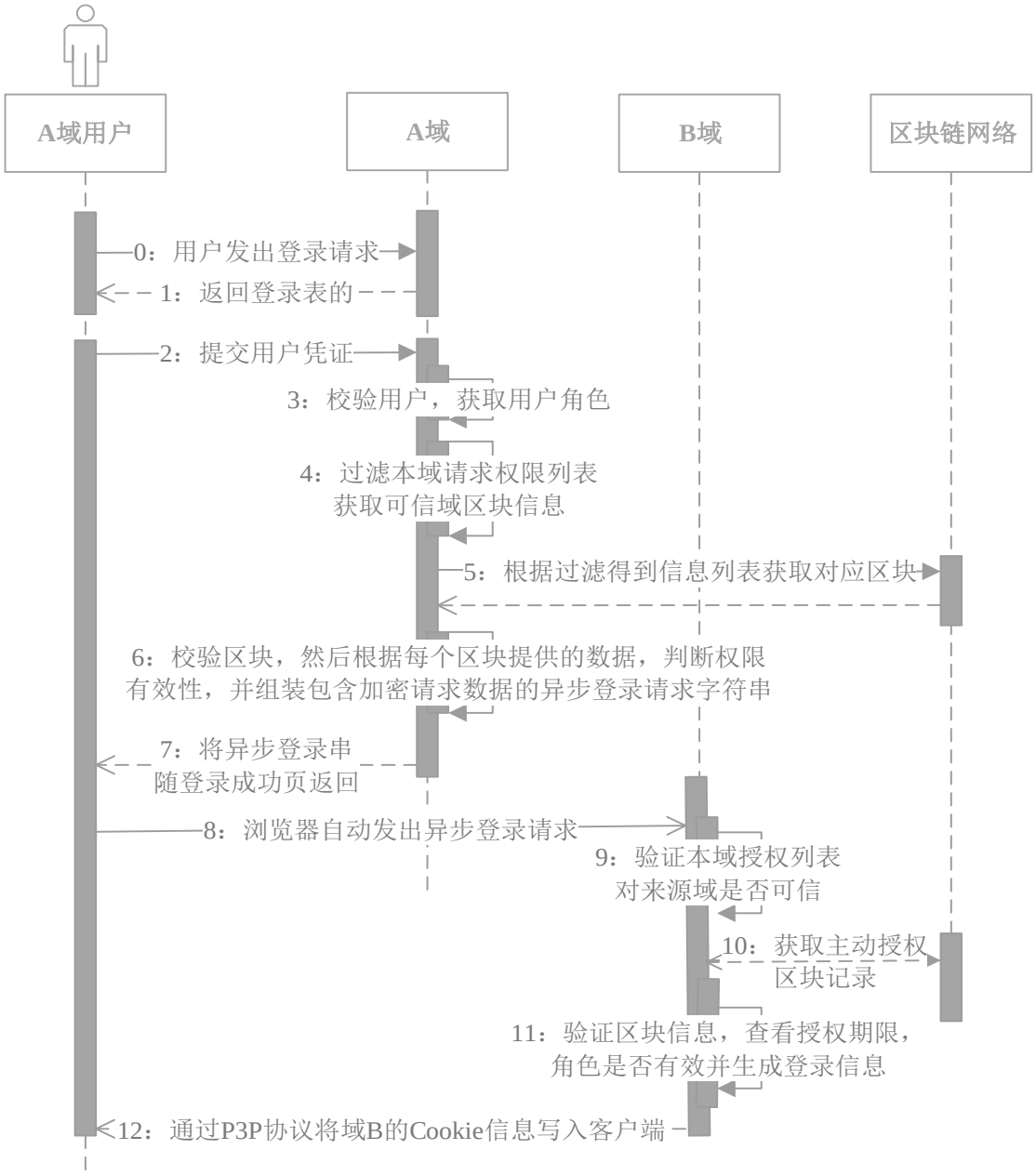

图 4-3　跨域登录流程图

假设只有授权信任域 B 给可信域 A 授权，用户跨域登录的具体过程描述为：

1. 域 A 用户向域 A 发出登录请求，域 A 返回给用户登录的入口表单。
2. 用户提交用登录口令。

3. 域 A 用户根据用户提交的口令，校验用户身份是否合法，校验通过之后，获取用户在域 A 的角色信息。

4. 域 A 过滤本域请求权限列表，得到给本域授权的授权信任域列表。过滤本域请求权限列表的依据是：对于本域请求权限列表中每一个请求权限条目来说，条目中的 responseIndex 和 requestIndex 字段均不为空且都指向本地区块链副本中有效的区块。

5. 根据过滤之后得到的授权信任域列表，获取域 A 请求域 B 授权的列表项中 requestIndex 字段所指向的区块数据。

6. 获取到对域 B 授权信息进行记录的区块数据之后，域 A 首先校验区块的 hash 是否和本域请求权限列表中对域 B 请求跨域权限的记录中 requestIndex 字段中保存的 hash 值一致，校验通过之后，域 A 采用私钥解密数据，并通过解密后的数据判断权限是否过期，如果没有过期，则通过 B 的公钥提取出授权信息从而获得域 B 给域 A 的登录接口信息，角色映射信息，并附加登录用户在域 A 中的登录信息并拼接成异步登录文本串并采用域 B 的公钥加密。文本串明文格式如下图 4-4（1）所示：

```
<script  type=”javascript”
src=”www.b.com/interface_a?from=www.a.com&username=ausername&roleFrom=role
_a1&roleTo=role_b1”>
</script>
```

图 4-4（1） 明文跨域串格式

文本串中登录到域 B 的参数为了保证安全，会采用域 B 的公钥加密传送，如图 4-4（2）所示。

```
<script  type=”javascript”
src=”www.b.com/interface_a?Nfc345DG9SXGEWSAFG0SGBXVA6SDFsfa87S7FS9
A7FA.....”>
</script>
```

图 4-4（2） 密文跨域串格式

7. 将异步登录文本串嵌入用户在域 A 登录成功的页面返回给客户端。

8. 用户浏览器自动向域 B 发送登录请求。

9. 域 B 收到登录请求之后，先用私钥解密数据获取来源域域 A，然后通过检索本域授权列表来确认曾经是否对来源域 A 进行过授权行为。

10. 若验证通过，通过本域授权列表中给域 A 授权的条目中的 responseIndex 字段所指向的区块高度从本地区块链副本取得授权确认信息区块。

11. 域 B 首先依据 responseIndex 字段中的 hash 校验区块 hash，以验证区块有效性，验证通过之后，采用自己的私钥解密数据。对于解密之后的信息，首先查看域信息是否和域 A 匹配，然后验证授权过期时间，如果权限没有过期，则最后对比域 A 用户提交过来的角色映射是否和区块中保存的角色映射是否匹配，如果匹配则验证通过，域 B 依据跨域提交的数据为用户生成登录信息。

12. 采用 P3P 协议将用户在域 B 的登录信息以 Cookie 形式写入用户浏览器。

4.2.3 域间授权失效鉴定流程分析

授权失效分主动授权失效和被动授权失效两种。

主动授权失效就是域 A 在跨域登录时发现授权过期，主动要求在再次授权。流程图如 4-5 所示。

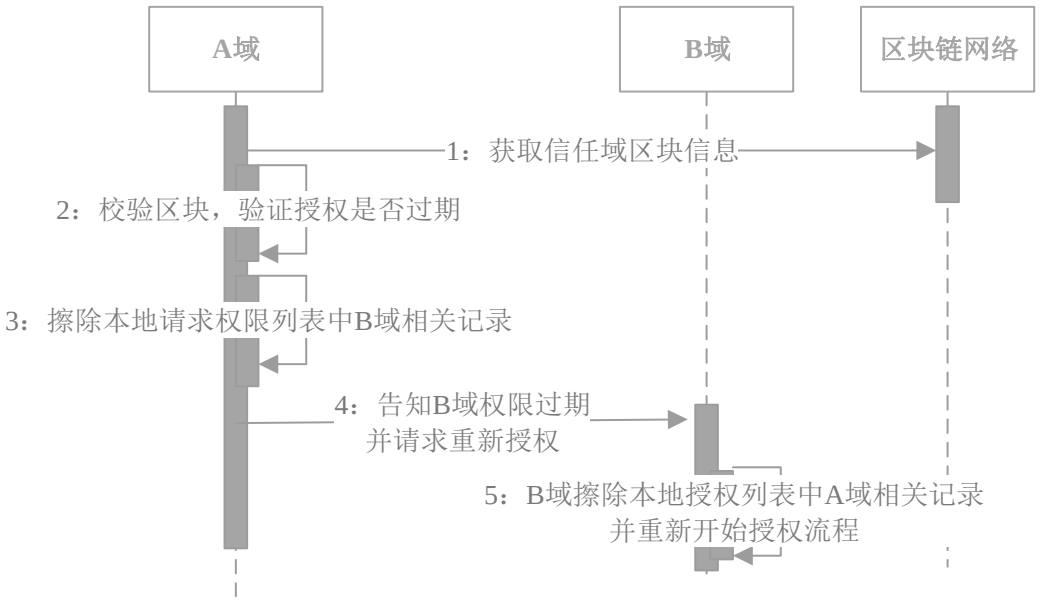

图 4-5 主动授权失效流程

主动授权失效具体过程描述为：

1. 域A发生跨域登录需求时，从区块链中获取到授权信任域B的授权区块信息。

2. 域A验证区块信息，发现域B的该授权已经过期。

3. 域A查找本地请求权限列表中的requestDomain为www.b.com的授权记录，并将该记录的requestIndex和responseIndex字段清空，表明域A单方面清除授权记录。

4. 域A告知域B授权过期，并发起再次授权请求。

5. 域B收到域A的通知，查找本地授权列表项中responseDomain为www.a.com的授权记录，并将其responseIndex字段清空，表明域B清除授权，然后域B重新审核域A的授权请求决定是否授权。

被动授权失效就是域A使用过期权限跨域登录域B时，由域B发现权限过期让域A重新请求授权。流程图如4-6所示。

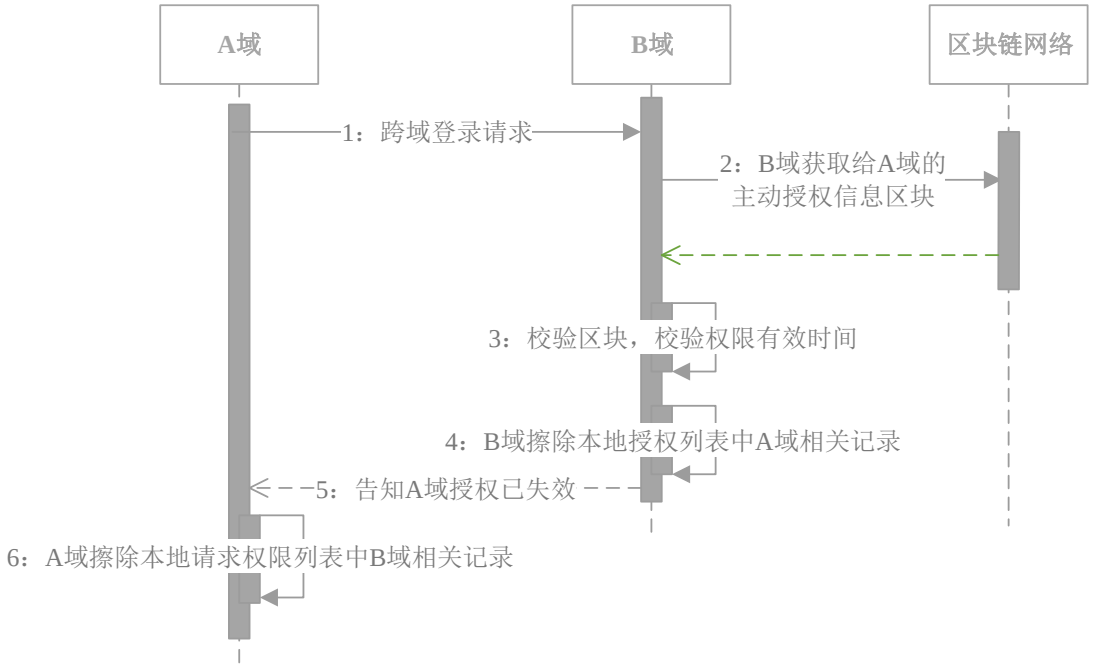

图4-6　被动授权失效流程

被动授权失效具体过程描述为：

1. 域A向域B发出跨域登录请求

2. 域 B 查找本地授权列表获取其 responseDomain 为 www.a.com 的记录，根据记录的 responseIndex 字段找到域 B 与给域 A 授权的区块。

3. 域 B 校验区块是否正确，校验通过后查看授权过期时间是否过期。

4. B 确认授权已过期，域 B 首先擦除本地授权列表中 responseDomain 为 www.a.com 的记录中 responseIndex 字段的值，表示域 B 单方面取消授权。

5. 域 B 通知域 A 原授权已失效。

6. 域 A 查找本地请求权限列表中的 requestDomain 为 www.b.com 的授权记录，并将该记录的 requestIndex 和 responseIndex 字段清空,表明域 A 方面清除授权记录。如果需要需可再次发起授权请求。

4.3 区块相关数据结构

本文提出的跨域设计方案要求区块结构兼容域间授权协商记录和域间跨域日志记录两种类型数据。每一个区块分区块头和数据域组成。

对于区块头，分别记录了前一个区块的 hash 值，区块高度，所有记录的梅克尔树根，区块类型标记，随机数，区块打包节点信息，记录总数等字段信息。各个区块通过记录前一个区块的 hash 来前后关联,区块头通过所有记录的梅克尔树根来关联本区块中的所有记录。通过区块的类型来区别该区块记录的数据是域间授权协商记录还是域间跨域日志记录，如果区块类型确定为授权区块，则该区块中的记录总数为 1,只包含一条域间授权协商记录。其中时戳用来排序指明该区块的打包时间，这样可以从全局保证所有区块的顺序性。

对于区块记录而言，每一个记录分四个字段。字段 owner 指定该记录的属主是谁；字段 item_type 标识记录的类型，方便节点挖矿时区别记录是域间授权协商记录还是域间跨域日志记录；字段 item_info 指定记录的详细信息，由于授权信息具有私密性，如果区块类型是授权区块，则区块记录中的记录详情在通过 Java 序列化为字符串之后需要用属主的公钥加密存储；字段 timestamp 记录初始节点最开始广播这条记录的时间戳。

区块数据结构如表 4-1 所示。

表 4-1　区块数据结构

区块头字段标识	数据类型	区块头字段说明
prev_hash	String	前一个区块的 hash
block_type	String	区块的类型 [authorization|log]
block_height	Bigint	区块高度
merkle_tree	String	梅克尔树根
random_num	Bigint	随机数
timestamp	String	区块时间戳
packer	String	打包数据库的域节点标识
item_count	Int	记录的条数。如区块类型为 authorization,则固定值为 1
记录字段标识	记录字段类型	记录字段说明
owner	String	标识记录的属主
timestamp	String	标识属主广播此记录的时间
item_type	String	标识记录的类型 [authorization]|log]
item_info	ItemInfo （自定义类型）	记录详情。若为授权信息，需加密

item_info 为记录详情，属于系统自定义 Java 类（ItemInfo.java）。不论是日

志信息类记录还是授权信息类记录，在系统中都表现为 ItemInfo 类的实体。该类的主要字段如表 4-2 所示。

表 4-2 ItemInfo 关键字段说明

字段	类型	说明
itemType	String	标识记录的类型[authorization|log]
loginUrl	String	开放给可信域的登录接口
roleMap	HashMap<String,String>	角色映射信息
publicKey	String	授权信任域的公钥信息
expireTime	Date	授权过期时间
domainInfo	String	域的标识信息
logInfo	String	日志信息

对于日志信息类记录(即 itemType=log)而言，ItemInfo 对象只要求 logInfo 字段非空即可。对于可信域用来维护请求授权信息的 ItemInfo 对象而言，其 logInfo 字段必须为 null，其他字段都必须有合法值才能是一个合法的请求授权记录。对于授权信任域用来维护授权信息的 ItemInfo 对象而言，其 logInfo，publicKey，loginUrl 字段数据必须为 null，其他字段都必须具有合法值才是一个合法的授权记录。

4.4 数据存储层设计

对于一个域节点而言，需要存储整个网络中广播的信息、区块数据信息、授权记录相关信息三方面的数据。

4.4.1 Redis 广播信息存储

从网络接受到的广播信息会随着加入到区块链网络中节点的增多而呈现线性增长，而且广播的信息需要被打包进入区块。同时，每当一个域节点收到到一个区块数据之后，还需要快速校验区块中的信息，以防止的重复打包相关广播信息。基于

广播信息需要快速读取的需求，采用了读取速度非常占优势的 Redis 存储。

对每一个域节点而言，系统内部通过 Redis 维护两个散列表 authorization 以及 log。其中 autoration 散列主要用来存放从广播报文中收到的与授权相关的信息，而 log 散列主要用来存放从广播报文中收到的与域间跨域登录日志相关的信息。对于节点从广播报文中收到的每一条记录信息，域节点会判断记录信息中的 Item_type 标识，如果标识是 authoration，则将记录信息存入 authoration 散列表中，否则存入 log 散列表中存储。

对于本系统中散列表而言，其 key 和 value 的格式设计如表 4-3 所示。

表 4-3　散列表字段格式设计

字段	格式	说明
key	timestamp#owner	owner:说明这条授权信息有哪个域节点发出来的。 timestamp:说明这条信息的属主发出这条信息的时间。
value	信息详情	如果是授权信息，信息是密文，如果是日志则是明文。

4.4.2 本地区块数据存储

采用本地文件系统实现区块数据序列化存储，域节点每次接收到一个区块数据，当通过验证确认区块数据合法之后，系统通过调用 fastjson 程序模块将区块数据序列化为 json 串保存至本地文件系统中，采取的存储策略是每一个区块对应于一个文本文件的存储方式。区块文件的命名采用“区块数据的 hash.txt”的格式。

4.4.3 区块索引与授权索引存储

因为区块数据序列化之后保存在本地文件系统中，而且由于区块数据的命名策略是基于区块数据的 hash 值来实现，从而无法直接通过文件名表达区块数据文件之间的顺序关系。而且如果通过解析区块文件中的 timestamp 字段来排序区块数据文件代价十分巨大。为了维护授权信息以快速检索区块数据，系统采用 MySQL 数据库为本地文件系统保存的区块数据文件建立索引表。本地维护一个 blockorder 表以维

护区块顺序。

表 4-5 MySQL blockorder 结构设计

字段	类型	说明
id	bigint	主键
block_height	bigint	区块高度
block_hash	varchar	区块的 hash 值
pre_block_hash	varchar	记录前一个区块的 hash
block_path	varchar	区块序列化文件存储路径

基于第三章提出的研究方案，为了快速的做出权限校验，每个域节点同时需要在 MySQL 数据库中维护两张表 authorization_req 以及 authorization_res。其中本域请求权限列表 authorization_req 记录域节点的请求授权信息，本地授权表 authorization_res 记录域节点的授权信息，其表的详细设计如下。

表 4-6 MySQL authorization_req 结构设计

字段	类型	说明
id	bigint	主键
requestDomain	varchar	表明域节点请求哪个域节点授权，值为域名
resqustIndex	varchar	说明授权请求信息所在区块的区块高度和区块 hash。格式[height:hashcode]
responseIndex	varchar	说明授权应答信息所在区块的区块高度和区块 hash。格式[height:hashcode]

表 4-7 MySQL authorization_res 结构设计

字段	类型	说明
id	int	主键
requestDomain	varchar	表明是哪个域节点请求本域授权，值为域名
responseIndex	varchar	说明授权应答信息所在区块的区块高度和区块 hash。格式[height:hashcode]

4.5 区块管理层的设计

区块管理层是整个系统的核心，主要有区块打包与广播、区块解析、区块同步三个模块组成。

4.5.1 区块打包与广播

区块打包与广播模块是域节点在区块链网络中挖矿的核心模块，域节点会将收到的权限广播信息和跨域日志广播信息分别记录到系统中的散列表 authorization 以及散列表 log 中,这两个 hash 表相当于是域节点维护广播数据报文的“数据池”。当域节点从区块链网络中收到一个区块的时候，需要首先校验区块的完整性，然后域节点将根据区块类型对散列表 authorization 和散列表 log 这两个数据池中的数据进行过滤，从中剔除已经打包到区块链中的记录，避免同一个记录被打包两次，过滤采取的策略是根据区块中的每一条记录的属主 owner 和发出这条记录的时间戳 timeStamp 字段组成查找键”timestamp#owner”，然后依依据此查找键和记录的类型去 authorization 散列或 log 散列中去查找数据，看是否存在和查找键匹配的记录，如果有则从对应的散列表中删除。

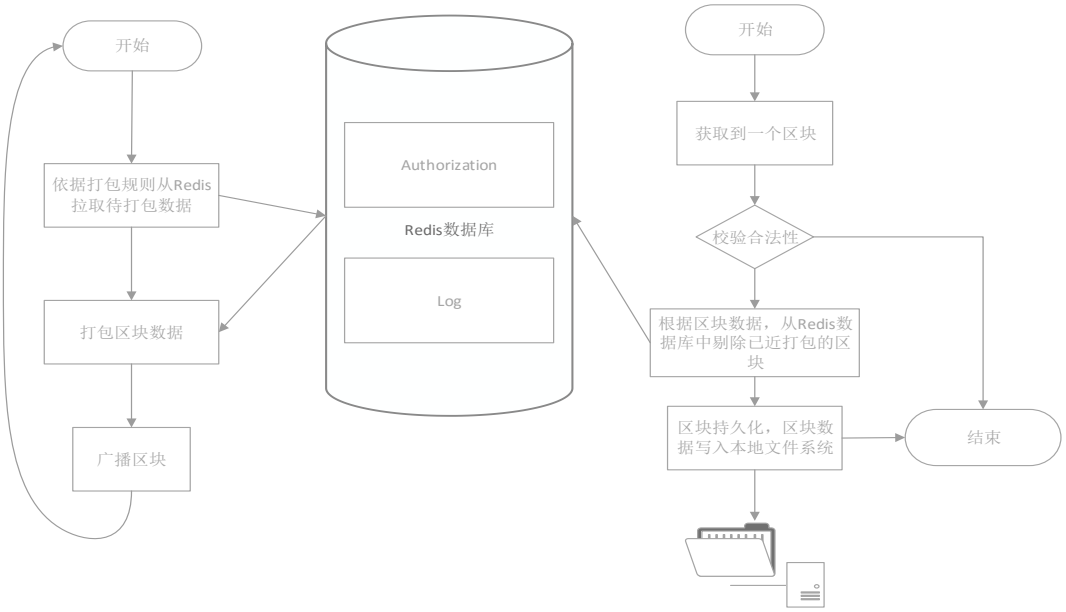

图 4-7　区块广播域打包

对每一个域节点而言，打包数据进入区块链需要遵循四个原则：

原则一是采取优先打包授权区块后打包日志区块的原则，因为对于区块链网络而言，最核心的信息就是授权信息，而跨域日志信息存放到区块链网络中并没有实质的作用，但是日志区块却可以达到比特币网络中的“区块确认”的功能以保证授权区块数据的安全性和确定性，使其篡改更加困难。一个域节点当发现自己维护的 authoration 散列不为空时，都需要优先对其中的记录进行打包，这样才可以使得授权信息尽快写入区块链，如果发现自己维护的 log 散列中数量超过指定阈值，也需要积极将日志信息打包进入区块。

原则二是如果打包授权区块，区块中只能包含一条区块记录，这条记录要么是授权请求记录，要么是授权响应记录。这样的好处是在跨域的流程中，跨域相关域节点可以快速的校验区块数据的完整性并可快速提取区块中跨域相关授权信息，同时也避免了处理不必要的其他域节点授权信息。

原则三是约定日志区块中的日志记录按每 256 条记录组成一个区块。目前比特币中每区块保存四千条交易记录，但是比特币网络中节点是近似等于全网络节点，而在基于区块链技术设计的本跨域系统中，参与节点较少，日志信息的单位时间增量约等于所有域节点当前登录用户数与域节点的乘积，而且打包区块涉及到计算所有记录的梅克尔树树根 hash，为了使得满二叉树的叶子节点都是唯一记录，避免满二叉树叶子节点的复制，所以设定每日志区块打包 256 条记录，随着该区块链网络中加入的域节点数的增多，这个阀值可以调大。

原则四是记录打包顺序依据时间排序，不论是授权相关记录还是日志记录。每个域节点在广播记录时都会带有该记录生成的时间戳以及记录的属主，当收到该记录的域节点将记录存放到本地 hash 表中时，会采用“时戳#属主”的格式作为主键来存储记录，这样，保证节点打包数据的时候每次都提取时间最早的记录来打包到区块中，这样避免某些域节点的记录迟迟无法记录到区块链中的情况而显得公平合理。

区块的打包挖矿中，参与挖矿的节点只有参与到此区块链网络中的域节点，且域节点较少，同时由于我们区块链网络中不涉及到比特中需要控制比特币总量的问

题，所以，我们将难度值固定为 6 个零，使得在待打包记录充足的情况下平均每十分钟可以产生一个区块。

区块打包的过程就是一个比拼算力的过程，首先需要对本地待打包记录数据计算 Merkle 树根并写入区块头部，然后不断尝试生成随机数，使得对区块的头部进行 sha-256 运算得到的 hash 结果必须小于给定的难度值。每当计算出一个区块，通过组播传递给所有的其他域节点。

4.5.2 区块校验与解析

区块校验发生在三个时间节点上，每当节点收到一个广播区块时、授权节点将授权区块发送给请求授权节点时以及区块权限解析时。

对于授权相关区块的校验只需要来两步即可完成，第一步计算区块记录信息的 Merkle Tree 根 hash 值，因为授权相关区块只有一条授权相关信息，所以只需要计算记录本身的 hash 即可。第二部是将计算的到的 hash 值置于区块头部的 merkle tree 字段，然后计算整个区块头的 hash 值。依据计算的 hash 结果校验区块是否符合可信网络中约定的难度值。如果计算的结果不满足约定难度值，则直接丢弃相关区块。如果满足难度值则将其纳入本地区块链中。

对于日志相关区块的校验与授权相关的区块的校验区别仅仅在第一步，因为每个日志区块中的日志记录条数是 256，所以搭建梅克尔树的时候，约定根据区块里面记录的次序将记录作为满二叉树的叶子节点来计算其 Merkle Tree 值。其后续步骤与校验授权区块时类似。

每一个域节点同时运行两个线程，分别完成跨域时区块数据解析以及权限校验时区块数据解析。

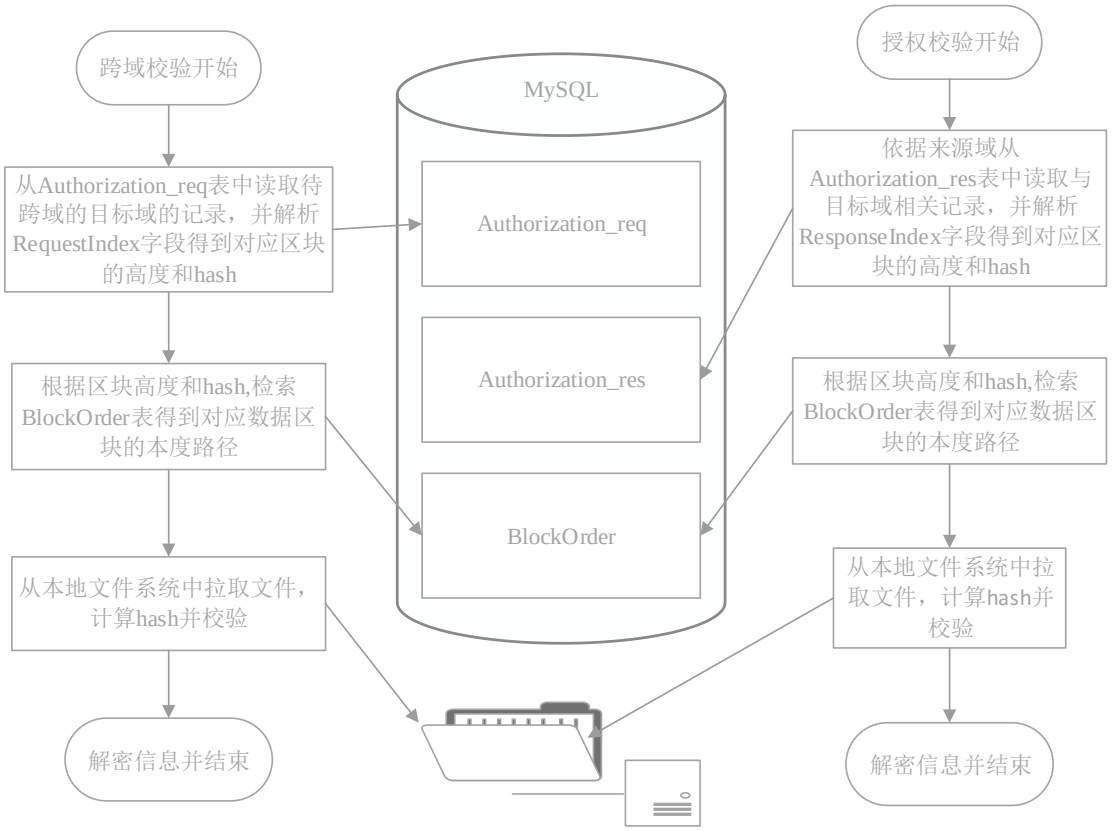

图 4-8　单域节点区块数据解析与校验

区块解析发生在可信域跨域登录以及授权信任域验证权限这两个时间节点上。当可信域跨域登录时，可信域通过检索本地 Mysql 数据库的 authorization_req 表以查找有权登录的授权信任域记录列表,解析记录中的 resqustIndex 字段获取到相关授权信任域的授权区块高度和区块 hash 值。随后查找 MySQL 数据库 blockorder 表,根据区块高度和区块的 hash 值检索对应的区块数据文件的本地文件系统存放路径 block_path。然后依据 block_path 加载文件，并执行区块数据 hash 运算校验。通过校验之后，可信域使用私钥解密区块数据文件，得到授权信任域给自己的授权信息并组装跨域登录请求即可。

对于授权信任域而言，授权信任域通过检索 MySQL 数据库 authorization_res 表以查找对相关请求域进行授权的记录，并通过记录的 responseIndex 字段获取到对应授权区块高度和区块 hash，随后查找 MySQL 数据库 bolockorder 表，得到对应

授权区块文件的本地文件系统路径 block_path，依据此路径加载文件，并执行区块 hash 校验。通过校验之后，授权信任域使用自己的私钥解密数据，从中获取对可信域授权的相关信息，并对可信域的本次访问请求是否在权限限定范围内做出判断。

4.5.3 区块同步处理

区块同步用来使区块链网络中的所有域节点最大可能维护关键数据区块的一致性，区块同步发生在区块链全链完整性校验以及临时区块同步两个时间点。

区块链的全链完整性校验是指在某一特定时刻，某域节点向全网申请区块链全链校验，以维护本地区块链数据文件的完整性。如图 4-9 所示：

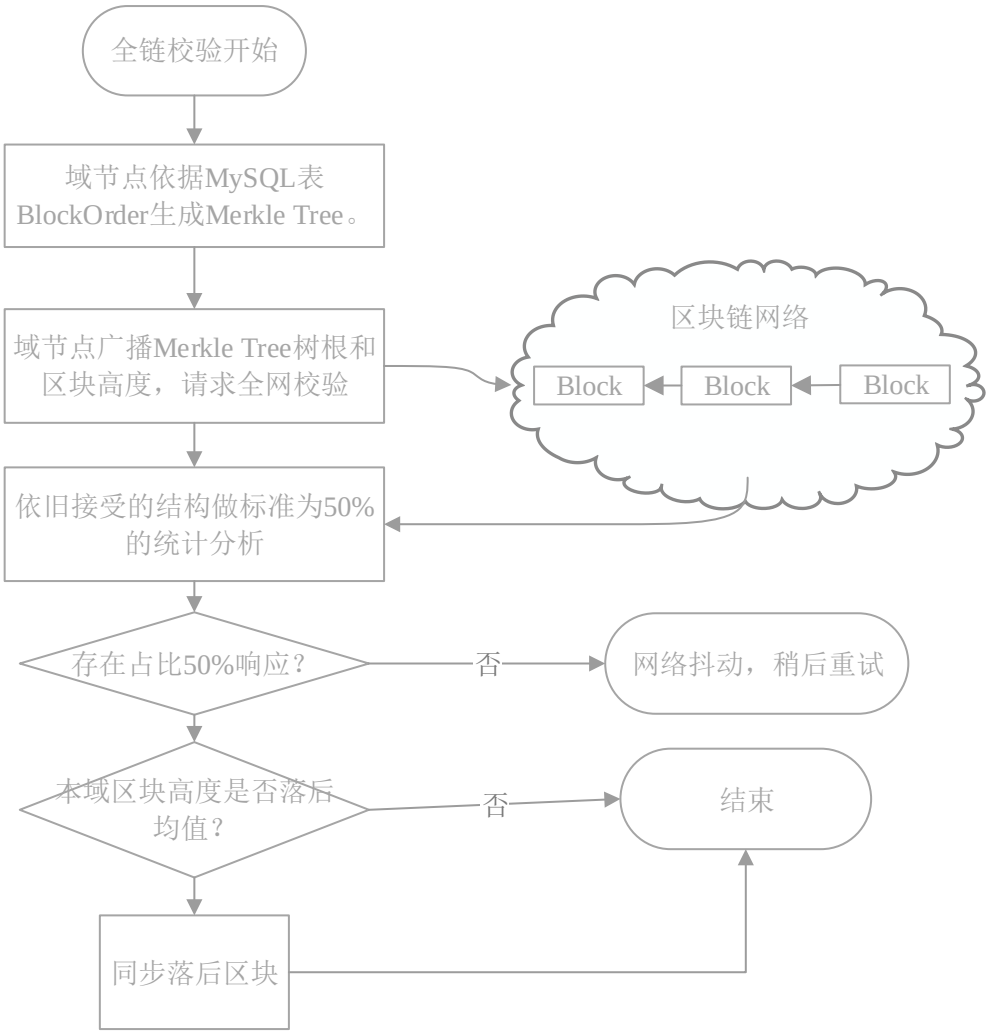

图 4-9　区块链全链校验流程

当一个域节点有全链校验需求的时，首先依据 MySQL 数据库中的 blockorder

表对区块数据文件排序，依据排序结果计算本地完整区块链的梅克尔树树根MTvalue，随后向全网广播区块链数据文件全链完整性校验请求，在请求中域节点声明自己本地计算的区块链梅克尔树根 MTvalue 和区块总高度 H。其他节点均以本域节点的区块链梅克尔树根 Mtvalue’和区块链高度值 H’以响应。请求域节点保留前100 个响应，当其中某一个响应值占比 50%以上，则说明当前环境下全网络区块整体同步情况较好，发起请求的域节点就可以根据这个占比 50%的响应对比本地结果。当对比结果中显示 Mtvalue=Mtvalue’且 H=H’，表明本域节点在平均水平和全网维持了区块同步关系；如果对比结果不等且区块高度落后于占比 50%的响应值中指明的高度，即 Mtvalue!=Mtvalue’且 H<H’，则说明本域节点区块同步水平在全网维持区块同步的平均水平之下，这个时候域节点就需要请求其他节点将自己缺少的区块同步到本域节点；如果对比结果不等且区块高度高于占比 50%的响应值中指明的高度，即 Mtvalue!=Mtvalue’且 H>H’， 则说明本域节点区块同步水平在全网维持区块同步的平均水平之上，这个时候域节点直接终止同步即可；如果对比结果显示 Mtvalue！=Mtvalue’且 H=H’，这表明本地区块链副本存在受损区块，此时需要发起全链同步请求。

另外需要特别需要说明的是当网络抖动厉害，会出现整个网络中各域节点维护的区块的区块高度差异很大，全链完整性校验时没有任何响应占比 50%之上，此时请求全链校验的域节点需要暂时中止全链校验过程，并在间隔固定时间之后待全网数据基本稳定再次发起校验请求。

临时区块同步发生在可信域与授权信任域之间依据区块做权限校验的过程中，其流程类似于全链校验。

对于可信域而言，当域节点通过检索 MySQL 数据库请求权限列表 authorization_req 得知本域确实已经得到某授权信任域的授权之后，并依据查找 MySQL 数据库中 blockorder 表得到相关区块数据文件在本地文件系统中的路径并依此获取到相关授权记录信息区块文件，但是在区块校验过程中如果出现授权记录信息区块文件的 hash 值和本地请求权限列表 authorization_req 中 requestIndex 字段中记录的 hash 值不一致，则极有可能是出现了本地文件被攻击或故障导致的文件

损毁的情况，此时，可信域需要临时发起单区块数据同步请求。且域节点在广播的单区块同步请求信息中需要指明需同步区块的高度值和区块的 hash 值,其他节点收到信息之后将指定区块高度的区块数据发送给请求域，请求域节点保留保留前 100 个响应,并采纳其中占比 50%以上的响应数据,如果不存在占比 50%以上的响应数据，认为网络不稳定，直接中止本次区块同步并同时中止本域对目标授权信任域的跨域请求。

对于授权信任域而言，区块同步过程和可信域基本类似，只有两点不同，第一是获取到本域授权列表中指定的授权确认信息区块之后，校验时通过检索 MySQL 数据库中本域授权列表 authorization_res，对比相关记录的 response_res 字段指定的 hash 来判断区块数据文件的完整性，另外当对授权确认信息区块进行同步时，如果前 100 个响应中得不到占比 50%的响应数据，则可直接拒绝可信域本次的跨域请求。

4.5.3 区块分叉处理

虽然在数据打包成区块的过程中采取了区别域比特币的数据打包策略，但是依旧无法避免比特币网络中的区块链分叉问题。该系统在运行过程中，依旧会存在某些域节点同时收到两个 hash 值不同的区块数据,而且这两个区块指向同一个前置区块的可能。对于这种情况，类似于比特币，域节点不能主观丢弃其中任何一个数据区块。否则随着时间的推移，会造成一个严重后果就是区块链网络分裂成多个小网络，各个小网络不断在自认为合法的区块链条上不断延展数据区块。

本系统设计中，对此问题采取的策略是，如果同时收到以区块 N 为前置区块的两个数据区块 A 和 B,则在 MySQL 数据库 blockorder 表中会同时记录下这两个区块的数据,包括各个区块的前导区块所对应的 Hash 值。假设到某一个时刻,blockorder 表中的区块记录分叉成两条链，标识其中一条为:“N←A←C”即区块 C 以 A 为前导，区块 A 以 N 为前导。另外一条分叉表示为:“N←B←D”。由于 hash 算力的影响，每个域节点不能永远都同时接收到两个数据区块，所以，当某一时刻区块收到了一个以区块 C 为前导区块的区块 M，而且在较长一段时间类都没有收到以区块 D 为前导

区块的区块数据，这个时候就说明“N←B←D”这条区块链已经被丢弃，所以，系统对MySQL数据库中的blockorder表中进行数据重整,删除其中的被丢弃的区块记录。如图 4-10 所示。

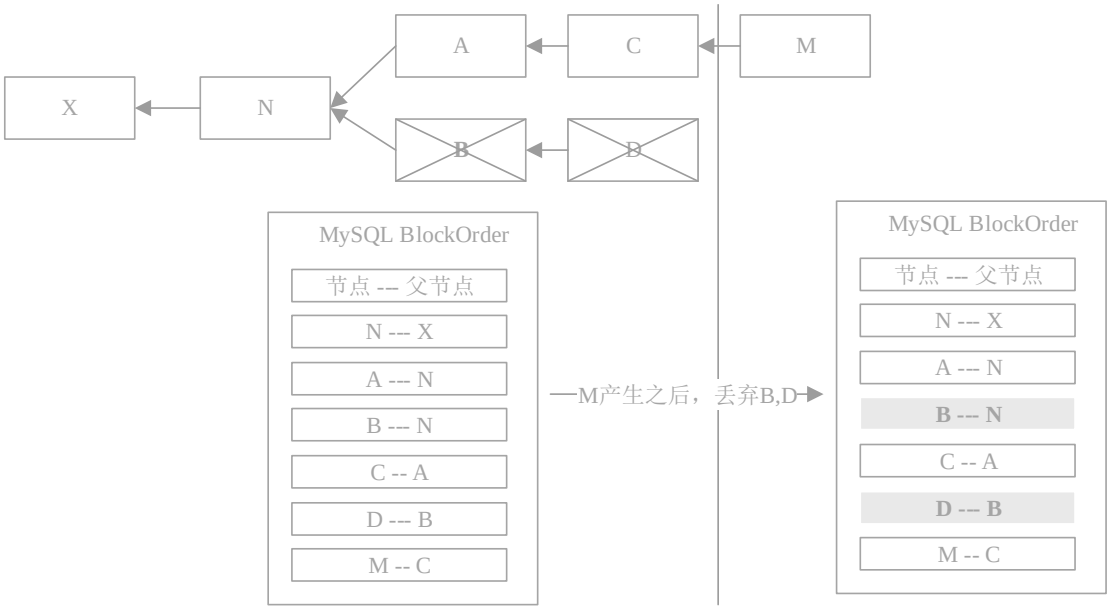

图 4-10　区块分叉的处理策略

值得说明的是为了防止数据丢失，删除记录也需采取相应的策略。以区块分叉“N←A←C←M”和“N←B←D”为例，当需要丢弃区块 B 和区块 D 的时候，此时需要做集合运算 K= ((B∪D)-((B∪D)∩(A∪C∪M))),其中集合 K 中的元素是需要根据其类型重新放入 Redis 缓存数据库中重新打包的。

4.6 权限管理层的设计

权限管理层是该系统的主要功能，主要包含授权协商和授权校验两个模块。这两个功能涉及域节点本地 MySQL 数据库中的本地请求授权列表 authorization_req 以及本域授权列表 authorization_res。每个域节点必须保证这两个数据库表中数据的正确性，尤其是本域请求权限列表 authorization_req 中的数据的丢失会直接导致域节点丢失其他域给本域的授权。

4.6.1 授权协商

授权协商做为用户可跨域访问其他域的基础，基于第三章跨域授权协商相关部分的研究分析，我们得知参与到授权协商模块中的域节点需要完成角色的选择性映射，授权数据区块信息的持久化两种最主要功能。如图 4-11 所示。

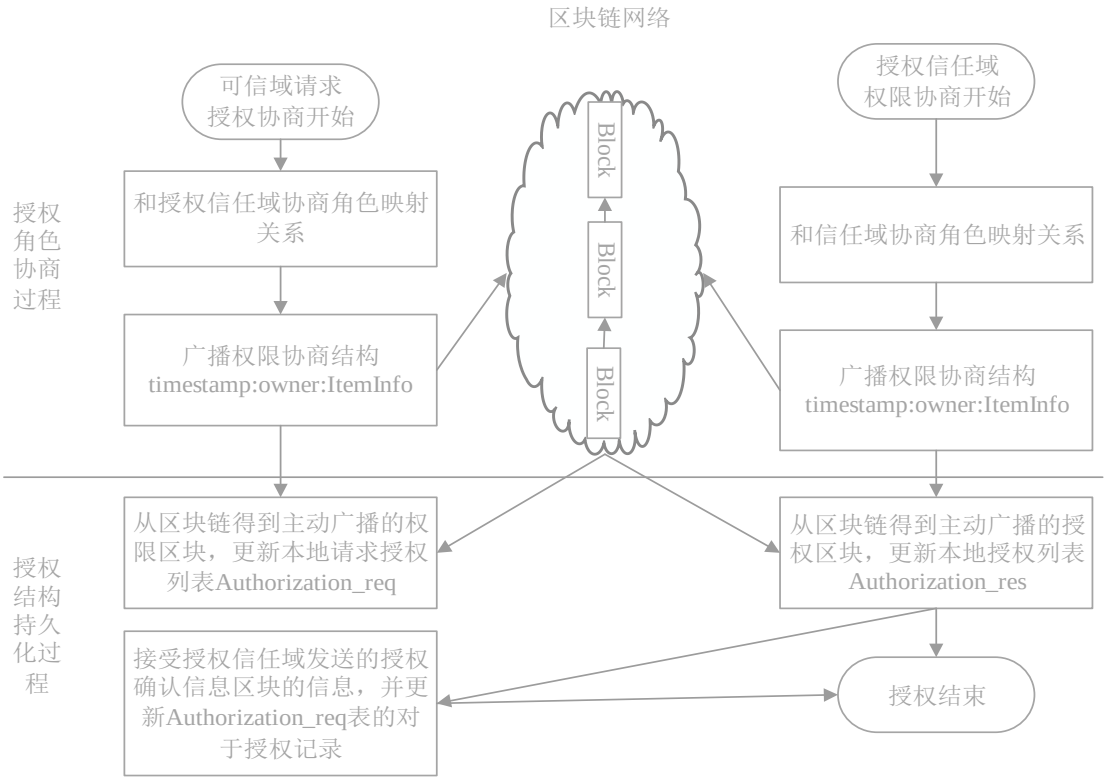

图 4-11　授权协商策略

角色的选择性映射由各个域管理员事先协商，并通过运维层的授权决策管理进行操作。当可信域管理员请求授权信任域授权时，首先将本域角色列表[roleA、roleB、 roleC、……]传输给授权信任域，授权信任域管理员根据事先的约定决定给那些角色授权,并返回一个授权的角色映射列表[roleA’、null、roleC’、……]，其映射关系依据次序和请求授权的角色列表对应,其中为 null 的选项表明授权信任域拒绝对可信域中某种某角色进行授权。

授权信息的持久化由授权信息写入区块链以及本地记录授权信息两部分组成，当可信域与授权信任域角色映射完成之后，可信域和授权信任域需要根据系统

ItemInfo 类组装权限广播信息，组装的过程就是根据双方授权交流的结果信息填充 ItemInfo 中的必要字段。参考 ItemInfo 类的定义，具体而言，可信域对于的 ItemInfo 对象必须填充 itemType，loginUrl，roleMap,publicKey,expireTime,domainInfo 等字段信息。授权信任域必须填充 itemType,roleMap,expireTiem,domainInfo 等字段信息。因为授权信息是属于敏感信息，在整个可信网络中，为了保证授权信息只能由域节点自己解析，各个节点无法从本地区块链副本中感知其他节点之间的授权行为，当双方序列化 ItemInfo 对象准备将其广播之前需采用自己的公钥加密。

本地记录授权信息是为授权协商的区块在域节点维护的 MySQL 数据库中建立索引，任何域节点当接收到一个区块时，解析区块发现其区块头部字段 block_type=authorization 且区块中的授权记录信息的 owner 字段为本域节点标识，则表明这是本域节点广播出去的授权记录，此时域节点根据私钥解密其中区块中数据记录的 item_info 字段并反序列化为 ItemInfo 类的对象。如果类对象的 loginUrl 字段不为 null,则表明这是本域的请求授权信息，则在本地请求授权列表 authorization_req 表中新增一条请求授权记录，其中记录的 requestIndex 字段记录该区块的区块高度和区块 hash，其 requestDomain 字段记录请求的是哪个授权信任域的授权。如果反序列化的 ItemInfo 对象的 publicKey 字段不为 null，则表明该记录是本域的授权信息,则在本域授权列表 authorization_res 中新增一条记录，其中 responseIndex 字段记录该区块的区块高度和区块 hash，其 requestDomain 字段表明此记录是对哪个域授权，与此同时，将该区块信息发送给对应的可信域使其更新其本地请求授权列表 authorization_req 表中的 responseIndex 字段以完善本次授权。

4.6.2 权限校验

权限校验模块主要需要完成授权过滤，参数加密两个核心功能。本系统中实现流程如图 4-12 所示。

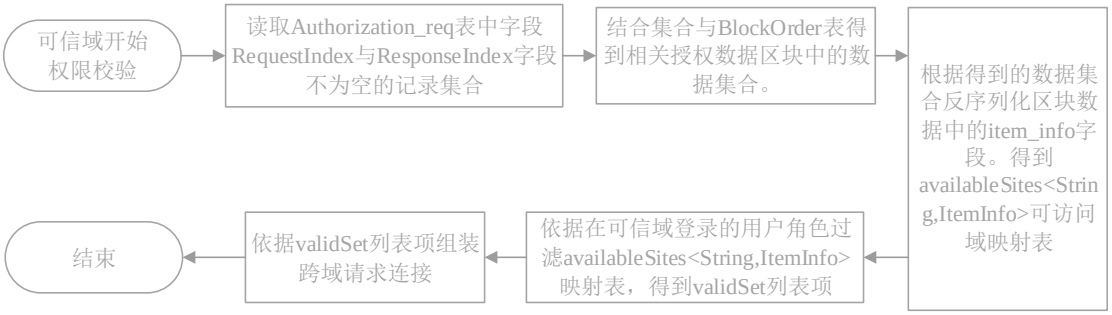

图 4-12　授权校验策略图示

权限过滤功能用来完成本域所有有权跨域访问的授权信任域的过滤操作，在本系统设计中，对于可信域而言，在内存中维护着一个实时 hash 映射表 availableSites<String,ItemInfo>，因为存在于 authorization_req 表中的记录，只有其 requestIndex 和 responseIndex 字段均有有效值才认为授权完成。所以 availableSites<String,ItemInfo>中记录了在 MySQL 数据库中本域请求权限列表 authorization_req 中 requestIndex 与 responseIndex 两字段都不为空的记录所对应的授权区块信息，其中 availableSites 的 key 对应于相关授权信任域的标识，value 为对应的授权区块的数据记录中的 item_info 字段反序列化得到的 ItemInfo 对象。同时，可信域当发现某一项授权过期之后，要从 availableSites 中删除相关记录。所以，当某一个用户登录到可信域之后，可信域获取到用户的角色 roleN，并根据 roleN 对 hash 映射表 availableSites 进行过滤,从中找到与本域角色 roleN 有映射关系的授权信任域信息集合 validSet 即完成权限过滤。

在权限过滤完成得到与 roleN 角色有映射关系的授权信任域集合 validSet 之后，系统需要使用参数加密功能完成跨域信息的拼接，对于 validSet 中的每一项，我们可以从 ItemInfo 类对象中提取的信息包括:授权信任域公钥 P=ItemInfo.publickey、授权信任域开放给可信域的接口 I=ItemInfo.loginUrl、本域角色 roleN 对应的角色 R=ItemInfo.roleMap.get(“roleN”)、授权信任域标识 S=ItemInfo.domainInfo。同时跨域请求需要传递本用户的用户名 U,用户本域角色 R’以及本域标识 S’。跨域请求字符串拼接的格式如下：

S/I?from=S’&username=U&roleFrom=R’&roleTo=R

拼接完成之后，为了信息保密性，请求串“from=S’&username=U&roleFrom=R’&roleTo=R”还需要采用 P 加密。最终的跨域串格式如下：

S/I?P[from=S’&username=U&roleFrom=R’&roleTo=R]

4.7 运维层的设计

系统中的运维层只有一个授权决策管理模块，通过该模块，可信域管理员可以实现对本域可跨域节点进行管理、申请跨域权限、管理本域给其他域的跨域授权、角色管理等功能。

其中，对本域可跨域节点的管理和管理本域给其他域的跨域授权实现方式是通过对本域请求权限列表(authorization_req)和本域授权列表(authorization_res)进行管理来实现。值得说明的是，当通过角色管理功能变动了域节点系统中的角色信息时，会造成与该角色关联的权限产生变动，也可能造成部分授权的失效。

其运维层后端授权页如图 4-12（1）所示，域管理员可以决定对某域的跨域请求授权或拒绝授权，同时也可以对角色映射加以控制。

图 4-12（1） 授权管理页

其运维层跨域权限申请管理页如图 4-12（2）所示，用户可以对授权信任域发起授权请求，并可以决定对本域哪些角色进行授权。

图 4-12（2） 权限申请管理页

其运维层的区块管理页如图 4-12（3）所示，管理员可以主动触发全链校验动作。

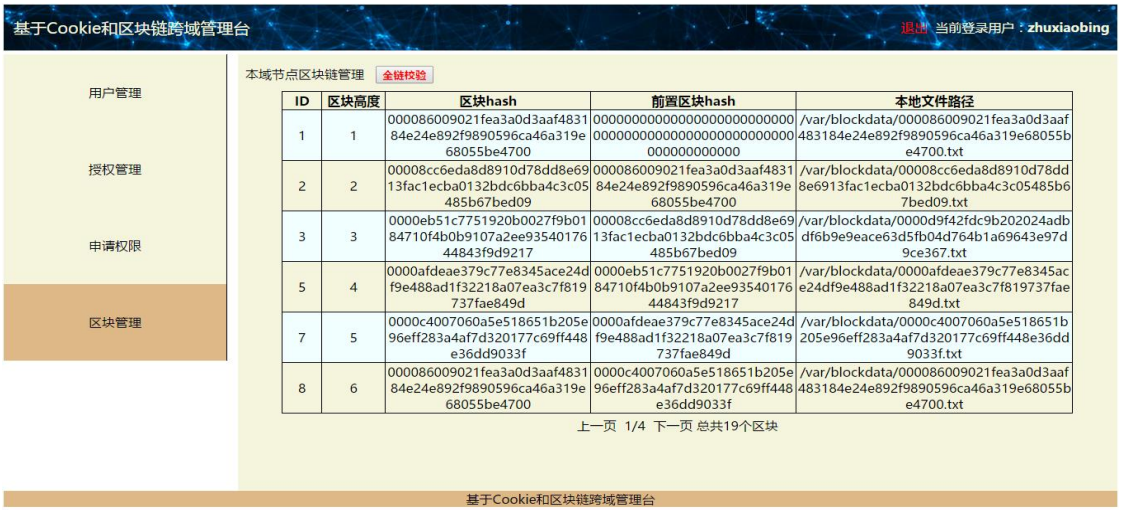

图 4-12（3） 区块管理页

4.8 本章小结

本章在上一章提出的基于区块链的跨域认证和授权系统方案研究的基础上，提出了一个系统设计的总体架构，并对系统各模块的功能和设计思路做了详细说明，包括流程图设计和主要数据结构设计等。

5 实验测试

5.1 实验环境

操作系统：系统运行平台 CentOS5.6

开发语言：Java

持久化数据库：MySQL

内存数据库：Redis

服务器：Tomcat8

5.2 实验系统部署

系统部署了五台主机，通过 Socket 通信模拟区块链网络通信，同时，将广播报文存储在 Redis 中，区块数据存储在本地文件系统中，采用 MySQL 数据库为区块数据建立索引。测试系统部署如图 5-1 所示：

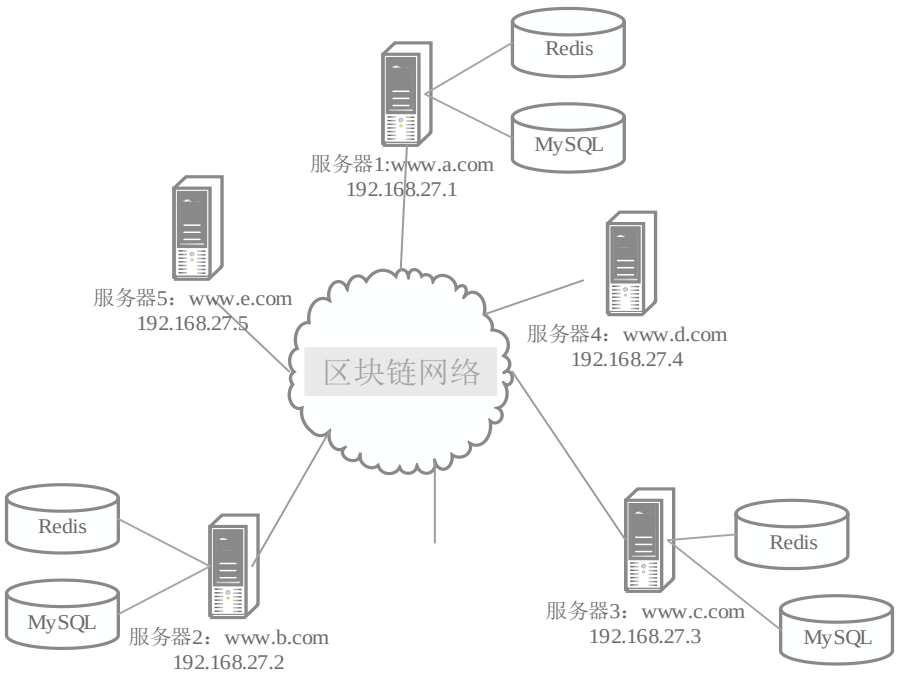

图 5-1　测试系统部署图

为了模拟真实环境，进而收集结果数据，本实验采取的策略是：各个域节点之间事先定义好彼此之间的授权规则。各个域节点之间授权请求间隔 30 分钟，即 150 分钟之内所有节点跨域权限请求全部发送完毕，对于授权信息何时可以得到区块链的确认并不做假设。每当一个域节点的某次授权请求完成，域节点就会每隔一分钟对有跨域权限的域发起一次跨域登录请求，并标识各个跨域登录的会话超时时间是 30s，系统持续运行，统计前六十个区块信息。本实验统计分析 hash 难度前导为分别设置为 6 个 0，7 个 0 的数据。

5.2 实验结果与分析

测试采样前 60 个区块，其中包括 40 个授权区块，21 个日志区块，其中各个区块产生的耗时统计如图 5-2 所示。

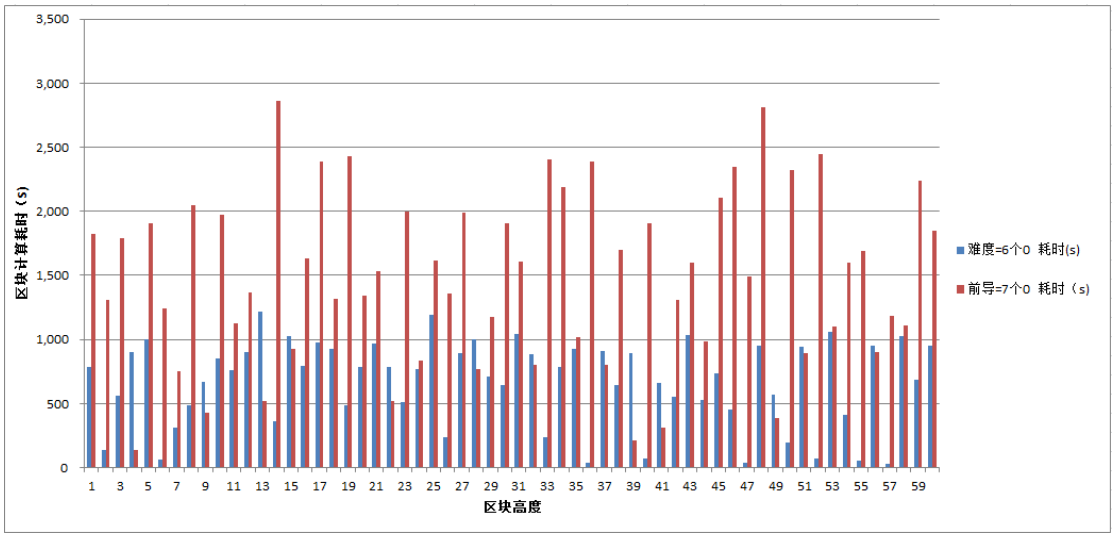

图 5-2　各个区块计算耗时统计分析图

从统计分析结果发现，对于区块链区块计算的耗时完全是随机分布的，而且 hash 难度值设置为 7 个前导 0 时随机性比 hash 难度值为 6 个前导 0 时随机性更大。依据统计数据发现 hash 难度值设置为前导为 7 个 0 时，区块计算平均耗时为 18 分钟，hash 难度值设置为前导 6 个 0 的平均耗时约为 12 分钟。对于本实验系统，设置 hash 难度值前导 6 个 0 更加满足每十分钟产生一个区块的需求。

另外本实验在所有域节点授权交互完成之后，从 www.a.com 域节点随机跨域访

问其他域节点 www.b.com,www.c.com,www.d.com,www.e.com 各十次，得到的的耗时时间统计如图 5-3 所示。

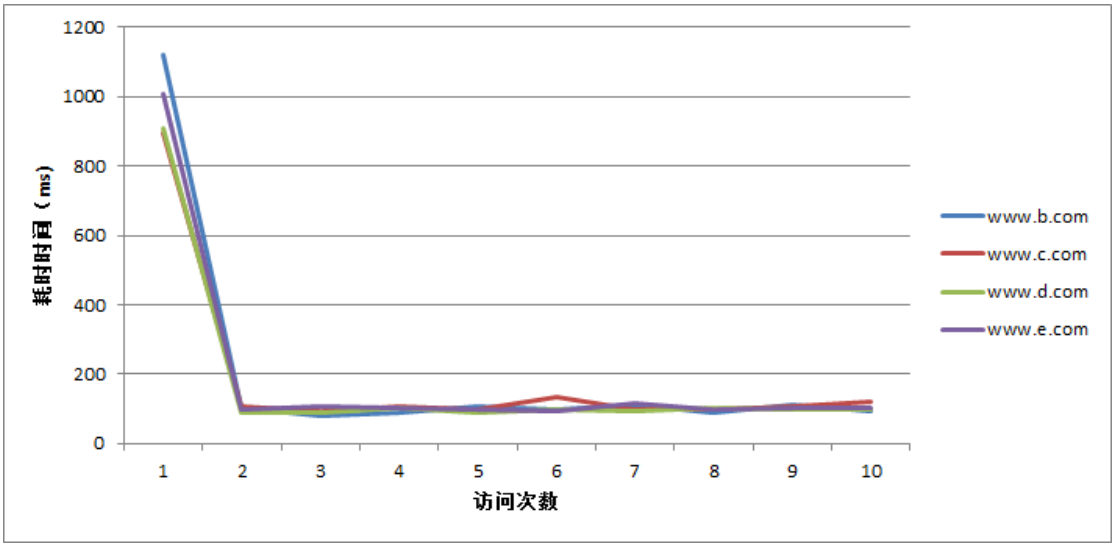

图 5-3　基于区块链的跨域耗时统计分析图

可以很明显看出，当第一次跨域访问的时候，因为涉及到从磁盘 IO 读取区块文件数据，耗时较长，后续访问都是基于内存中的 availableSites 列表来直接跨域访问，耗时均维持在 100ms 左右。相比传统的跨域访问实现方式，效率有显著提升。

5.3 实验小结

本章主要通过在同一个网段的五个域节点模拟区块链网络，并使其相互交换跨域授权信息，并对区块的计算耗时与跨域访问耗时进行了统计分析。从实验的数据分析可以得出结论：如果跨域授权行为具有时间敏感性，采用区块链网络来实现跨域的这种去中心化和跨域授权信息的持久化方案并不适用，但是如果在对跨域授权行为时间不敏感的环境中，采用该方案可以显著的提升系统的效率。

6 总结与展望

6.1 全文总结

伴随着云计算的发展，分布式数据存储得到广泛的应用。区块链网络作为一种综合分布式数据存储、点对点传输、共识机制、加密算法等计算机技术的新型应用模式在各领域逐渐得到应用，其分布式部署数据存储将单点数据故障造成的影响降到最低，其点对点的数据传输保证数据高效流通，共识机制使得节点之间工作量证明可以得到快速校验，加密算法保证了数据存储的安全性。所有这些特征使得区块链当前在信息安全保密程度要求较高的金融领域大放异彩，基于区块链技术的比特币网络运行七年，从未出现故障，其安全性，稳定性得到了实践的充分证明。

本文基于区块链去中心化的技术思想，分析了目前的主流跨域技术，并提出了一种利用区块链实现去中心化认证的跨域方案。本文主要做了如下工作：

1. 对目前国内外的主流跨域技术进行了分析，对比 Kerbose，passport，SAML 等协议的实现原理，指出目前主流跨域实现技术严重依靠于可信第三方，如果第三方故障或不可信将直接导致跨域认证授权失败的不足。

2. 基于当前跨域技术的不足，对基于区块链去中心化技术的跨域方案进行了研究，并对该方案涉及到的主要技术进行了分析。

3. 基于提出的方案，进行了详细的系统设计，并对该方案的主要关注点进行了详细的阐述和具体实现。其中着重设计和实现的模块包括：区块结构的设计，各域之间的授权协商，跨域时的授权校验以及区块的同步。

4. 最后，本文基于系统实现给出了实验分析，并指出了采用区块链网络来实现跨域的构想相对于传统跨域方案的利与弊。

6.2 课题展望

本文提出的基于 Cookie 和区块链的跨域方案，解决了传统跨域方案严重依赖于

可信第三方的不足，做到了认证的完全去中心化，去信任化，而且实现认证数据的分布式存储，避免了单点数据故障对全局造成的影响。但是，本系统的设计仍然有以下一些方面需要完善。

本系统基于区块链技术思想来实现，区块链网络为了维护共识机制，任何成功挖矿行为都需要提供工作量证明，这不但需要消耗大量的算力资源，而且对整个网络系统来说都是一个比较耗时的操作，目前采用和比特币网络一样的策略，控制在平均每十分钟产生一个区块数据。这导致的直接后果是授权行为平均每十分钟才能成功。后续可以采用 IBM 的开源框架 hyperledger fabric 进行架构，以此提升效率。

在效率问题之外的另外一个问题是本系统虽然采取了分布式区块存储来做到授权与认证的去中心化，但是每一个节点的授权和被授权信息与区块链中数据区块的映射关系依旧是存在本地数据库中。虽然单个域节点的区块丢失可以通过全网同步得到恢复，但是通过完整的区块数据重新构建本地的授权信息映射表将会遍历整个区块链，当区块链长度一定时花费的代价将十分巨大。所以，一旦一个节点的数据库中维护的区块映射数据丢失，将直接导致授权信息和被授权信息丢失而且很难恢复。

本系统目前采用的难度值是固定不变的（难度固定为 6 个 0）。随着加入到本跨域系统的域节的增多，这个难度值需要采用比特币类似机制来自适应调整难度值，这在后期需要不断试验以找出合适的难度控制策略。

最后存在的问题是本系统区别于比特币系统的关键点在于本系统节点挖矿是没有奖励措施的，不存在成功挖出区块可以获取相关激励的过程。所以采取一种有效措施使得可以激励所有节点积极挖矿并防止某些节点作弊是一个需要解决的问题。

致谢

时光如驹，三年时间就这么一晃而过，对于即将离开熟悉的校园，开启人生的另外一个篇章的我，三年的时间，得到了许多也错过了许多，但现在更多的是感恩与不舍，不舍华科美丽的校园，不舍伴青春三年的实验室，不舍敦敦教诲的老师，不舍共同奋斗的同学。借此机会向所有人表达我的感谢。

首先要感谢的是授业恩师汤学明老师，从毕业设计选题到完成，汤老师给予了我耐心的指导。研究生生涯中，汤老师严肃的科学态度，严谨的治学精神以及对工作精益求精的工作作风对我产生深刻影响，这是我人生的宝贵财富。感谢实验室的崔永泉老师、骆婷老师、付小青老师，谢谢老师们为我们提供了优良的学习环境，营造了良好的学习氛围，让我们能够在三年的时间里专业素养得到极大提升。

其次要感谢实验室的兄弟姐妹，感谢我的师兄师姐平时对我在工作和学习上的指导和建议，感谢同届的伙伴以及我的师弟师妹们，如果没有与你们的学习探讨，我不能取得今天的进步，能和大家一起构建实验室和谐奋进的氛围，我感到十分荣幸。最后感谢我的室友和在华科认识的朋友们，是你们让我在繁忙的研究生生涯中得到片刻的放松，祝大家前途似锦。

最后要感谢我的父母，我的妹妹，我的女朋友，你们的信任是我不断坚持奋斗的动力，在未来，我们风雨同舟，荣辱与共。

参考文献

[1] Gentry C. Fully homomorphic encryption using ideallattices. Association for Computing Machinery, 2009, 42(2): 24~36

[2] 林利，石文昌．构建云计算平台的开源软件综述．计算机科学，2012，27(2): 108~107

[3] 张建伟，李鑫．基于 MD5 算法的身份鉴别技术的研究域实现．计算机工程，2003, 29(4): 118~119

[4] Liang Zhigang. A New Kind of Single Sign-on Model Base on Mobile Agent. The 2nd International Conference on Information Engineering and Computer Science, 2010

[5] 樊蕊. 跨域身份认证系统的研究域实现: [硕士学位论文]. 西安电子科技大学, 2007.

[6] 马荣飞．统一身份认证系统的研究和实现．计算机工程与科学，2009，31(2): 145~149

[7] NAKAMOTO,S. Bitcoin: A Peer-to-Peer Electronic Cash System. www.bitcoin. org/bit- coin.pdf，2009

[8] Arvind, Narayanan, Joseph, Bonneau. BITCOIN AND CRYPTOCURRENCY TECHNOLOGIES A comprehensive Introduction. CITIC Publishing Group, 2016: 24~26

[9] Rosenfeld,M. Analysis of hashrate-based double spending. 2014: 35~37

[10] Khan A R. Access control in cloud computing environment. ARPN Journal of Engineering and Applied Sciences. 2012, 7(5): 613~615

[11] Oppliger R. Microsoft.net passport and identify management. Information Security Technical Report, 2004, 9(1): 24~30

[12] Pfitzmann B, Waidner M. Analysis of liberty single-sign-on with enabled clilents. IEEE Internet Computing, 2003, 7(6): 38~44

[13] Newnan B Ts'o T. Kerberos: an authentication service for computer networks. IEEE Communications, 1994, 32(9): 32~40

[14] Jalal Al-Muhtadi, Apu Kapadia, Roy Campbell, et al. The A-IRBAC 2000 Model: Administrative Interoperable Role-Based Access Control. Technical Report, UIUCDCS-R-2000-2163, University of Illinois, 2000.

[15] Liao Junguo, Hong Fan, Zhang Zhaoli. Management of Association in The IRBAC2000 Model. Wuhan University Journal of Nature Science, 2006, 11(5):1262~1266

[16] 陈颖, 杨寿保, 郭磊涛. 网络环境下的一种动态跨域访问控制策略. 计算机研究与发展, 2006, 43(11): 1863~1869

[17] 刘润达, 诸云强, 宋佳等. 一种简单跨域单点登陆舰系统的实现. 计算机应用, 2007, 27(2): 288~291

[18] 丁立新, 赵曦滨, 顾明. 基于 Kerberos 的 Web 单点登录研究. 计算机工程域应用, 2005.10

[19] Bellovin S.M., Merritt M. Limitations of the Kerberos authentication system Computer Comm.Review. 1990, 20(5): 119~132

[20] Deng Y, Cheng X H. System design of mobile cross-domain single sign-on. Computer Engineering and Design, 2010, 31(8): 1667~1672

[21] Subashini S, Kavitha V. A survey on security issues in service delivery models of cloud computing. Journal of network and computer applications, 2011, 34(1):1~11

[22] Tu S S, Niu S Z, Li M J. An Efficient Access Control Schema for Cloud Environment. Cybernetics and Information Technologies, 2013, 13(3): 77~90

[23] 李凤华, 苏锟, 史国振等. 访问控制模型研究进展及发展趋势. 电子学报, 2012, 40(4): 805~813

[24] Yang Z, Wang J X, Yang L, et al. The RBAC model and implementation architecture in multi-domain environment. Electronic Commerce Research, 2013, 13(3): 273~289

[25] Bertino E. RBAC models—concepts and trends. Computers and Security, 2003, 22(6): 511~514

[26] W.Diffie, M.Hellman. New Directions in Cryptography. IEEE Trans Inform TheoryIT, 1976, 22(6): 644~654

[27] M.E.Hellman. The Mathmatics of Public-key Cryptography.Scienctific American. 1979:146~157

[28] Bruce Schneier. 应用密码协议算法与 C 源程序. 北京:机械工业出版社, 1999:15~19

[29] 袁勇, 王跃飞. 区块链技术发展现状与展望. 自动化学报, 2016, 42(4):481~494

[30] ELDOS. “Kerberos vs. SSL/TLS. What’s the Buzz?”. Software Compomnents for Data Protection-Secure Storage and Transfer, 2016

[31] T. Y. S. H. R. C. Neuman. RFC 4120: The Kerberos Network Authentication Service(V5) , 2005.5

[32] 陈晓东. 基于微软护照协议的单点登录系统的研究: [硕士学位论文]. 华中科技大学, 2004.

[33] Duan B, Sun L, el al. SAML Based Intelligent Logging-on System in Electric Enterprise Integration Services. Automation of Electric Power Systems, 2006, 30(15): 30~32

[34] 王洪生, 袁捷, 曹春山等. 基于 SAML 的身份联盟建立的研究. 计算机工程与设计, 2007, 28(21): 5122~5125

[35] 倪力舜. 基于联邦的跨域身份认证平台的研究. 电脑知识与技术, 2011, 07(1): 53~55

[36] Anderson A. Hierarchical resource profile of XACML v2.0. Oasis Standard, 2005, 58(11): 1701~1703

[37] 林华, 王勇等. 区块链技术驱动金融. 北京:中信出版集团, 2016.10: 43~45

[38] M. Swan. Blockchain Thinking: The Brain as a DAO(Decentralized Autonomous Organization. Texas Bitcoin Commerce, 2015, 13(3): 27~29

[39] A.saxena, J.Misra, A.Dhar. Increasing annoymity in bitcoin. Financial Cryptography and Data Security, 2014, 30(15): 122~139

[40] Nasdaq. Nasdaq Linq Enables First-Ever Private Securities Issuance Documented With Blockchain Technology. http://ir.nasdaq.com /releasedetail.cfm?releaseID=948326, 2015

[41] J.Garay, A.Kiayias, N.leonardos. The bitcoin backbone protocal:Analysis and applications. 2015, 6(3): 10~315

[42] 李金库, 张德运. 身份认证机制研究及其安全性分析. 计算机应用研究, 2001, 18(2): 126~128

[43] 孙井封. 一种基于 Cookie 票据的网络用户身份认证系统的设计与实现: [硕士学位论文]. 北京邮电大学, 2007.

[44] Indranil Nath. Data Exchange Platform to fight Insurance Fraud On Blockchain. IEEE Internet Computing, 2016, 7(6): 12~16

[45] Cyril Grunspan, Ricardo Perez-Marco. DOUBLE SPEND RACES. 2015:23~25