北京大学

硕士研究生学位论文

题目：基于区块链的 PKI 域名鉴权方案设计与实现

姓　　名：李安然

学　　号：1501214389

院　　系：信息科学技术学院

专　　业：计算机软件与理论

研究方向：网络与信息安全

导　　师：关志　副研究员

二〇一八年五月

版权声明

摘要

公钥基础设施（PKI）作为保证数据机密性、完整性、不可抵赖性以及身份鉴别的重要方式，已经被广泛应用在电子邮件、电子商务、网上银行及资源发布等日常网络应用。经过近二十年的发展，PKI 相关技术和标准逐渐完善，但由于复杂的应用环境和中心化的系统架构，仍然存在着各种问题。这些问题主要表现在系统的信任构建和签发证书的可信度两方面，其中签发证书的可信度问题可以通过一系列外在的改进措施，规范证书操作的流程来解决；系统的信任构建问题需要对现有 PKI 系统组件和架构进行调整，均衡各个实体之间的信任和权利分配，近年来一直是 PKI 相关研究所关注的问题。

本文对现有的 PKI 系统架构进行介绍和分析，依照证书的生命周期透视该系统中可能存在的问题，并对现有的解决方案进行分析和归类。其后对具有去中心化特性的区块链技术进行介绍，包括共识机制和智能合约技术，并调研了与区块链相关的 PKI 解决方案。发现现有 PKI 系统中的信任构建问题，其存在的主要原因是该系统赋予中心化的授权机构（CA）权力过高。为了削弱 CA 在 PKI 系统中对证书的控制力，需要均衡各个实体之间的权利分配，提升证书申请者对证书签发的控制权。本文基于区块链技术提出了一种域名身份认证的方案，域名拥有者可以在分布式、无需可信第三方的网络环境下完成对信任 CA 的控制；该方案作为现有 PKI 系统的辅助工具，可以帮助 PKI 用户提升对证书签发的控制权，削弱授权中心的绝对权力，增强 PKI 系统的安全性。

与现有的类似方案 DANE 相比，本方案利用区块链技术完成身份认证，无需可信第三方，具有更好的安全性；另外，本方案中的所有操作在区块链上进行，提供了公开透明的审计特性。本文就给出方案的安全性进行了分析，对可能存在的攻击方式进行了讨论，并给出其免受这些攻击的说明。最后，对本文提出的方案进行了系统设计，详细描述了相关模块的功能和接口；依据设计给出相应的系统实现，对实现的各个模块进行功能性测试，系统能够良好稳定地运行；并对未来能够开展的工作和适用的应用场景进行了分析。

关键词：PKI，区块链，身份认证，密码学

Blockchain-based Authentication of Domain Name Design and Implementation

Anran Li (Software and Theory of Computer)
Directed by Prof. Zhi Guan

ABSTRACT

Public Key Infrastructure (PKI) has been widely used in daily network applications, such as e-mail, e-commerce, online banking, and resource publishing, which is an important way to ensure data confidentiality, integrity, non-repudiation, and identity authentication. After nearly two decades of development, PKI-related technologies and standards have gradually improved. But due to the complex application environment and centralized system architecture, there are still various problems. These problems are mainly manifested in the trust building of the system and the credibility of the issuance certificate. The credibility of the issuance certificate can be solved by some external improvement measures and the process of standardizing the certificate operations. The trust construction problem of this system needs to adjust existing system components and architecture to balance the trust and rights allocation between entities, and it has been a concern of PKI-related research institutes in recent years.

This paper introduces and analyzes the existing architecture of PKI system, analyzes the possible problems in the system according to the life cycle of the certificate, and then analyzes and categorizes the existing solutions. Then it introduces the blockchain technology with decentralized characteristics, including consensus mechanism and smart contract technology, and investigates the PKI solution related to blockchain. The main reason for the existence of the trust-building problem in the existing PKI system is that the system gives the Certificate Authority (CA) power to be too high. In order to weaken the CA's control over certificates, it is necessary to balance the distribution of rights among entities and improve the control of certificate issuers for certificate owner. This paper proposes a domain name authentication scheme based on blockchain technology. The domain owner can complete the control of the trusted CA in a distributed, trusted third party network environment; this solution is used as an auxiliary tool for the existing PKI system，which can help PKI users to improve the control over the issuance of certificates, weaken the absolute authority of the authorized center, and

enhance the security of the PKI system.

Compared with the existing solutions like DANE, this scheme uses blockchain technology to identity authentication without a trusted third party. Its security does not depend on other application entities; in addition, all operations are completed through the blockchain, which provides open and transparent auditing features. This paper analyzes the possible attacks to our scheme, and gives explanation for away from these attacks. Furtherly, the solution proposed in this paper is implemented. The functions and interfaces of related modules are described in detail. According to the design, the functional testing of each module is performed, and the system can operate in a good and stable manner. At Last, future work and applicable scenarios are discussed.

KEYWORDS: PKI, blockchain, identity authentication, cryptography

目录

第一章　绪论

1.1　背景知识和研究意义

随着网路技术的迅猛发展，大量的信息需要通过网络进行交互，在其给人们带来巨大便利的同时，也存在着巨大的安全隐患。在网络中传输的数据可能会被恶意的攻击方非法的窃听，导致传输信息的泄露，如何保证数据在传输过程中的机秘密性、完整性和不可否认性一直是网络安全技术所关注的问题，而公钥基础设施 PKI(Public Key Infrastructure) 技术是解决这一问题的重要手段。

在现有的 PKI 系统中，授权机构和用户之间的权利是不对等的，用户只能发起证书签发请求，而不能对授权机构是否签发证书起到制约性。由于信任都集中在中心化的授权机构之上，恶意的授权机构或者被攻破的授权机构将可以签发任意用户的证书，从而对相应用户发起中间人攻击，对整个系统带来恶劣的影响。

在 2011 年，DigiNotar 遭到入侵，签发的虚假证书涉及到 20 多个网域的 200 多个 SSL 证书[1]，对用户造成了巨大影响，最终导致自身被吊销授权机构资格的后果。2013 年 TURKTRUST 创建了两个具有欺诈性质的数字证书，其中携带了 CA 证书的所有授权内容，可以创建任意域名的认证证书，从而执行钓鱼攻击或者中间人攻击，并且影响所有的 windows 版本。Mozilla 在 2015 发现 CNNIC 颁发存在可以发起中间人攻击的证书，被用于部署到网络防火墙中，劫持所有的 HTTPS 通信，从而绕过游览器警告。GlobalSign 在 2016 年由于自身操作失误，吊销了错误的中间证书，导致证书信任链被破坏，很多网站无法使用正确的证书进行安全连接的建立。2017 PROCERT 由于不遵守规则，签发低安全性的证书和不合法的证书被 Mozilla 移除信任列表；同年，由于自身的不端行为，WoSign 和 StartCom 被 Mozilla 从信任列表中移除。

从以上的例子中可以看出，授权机构以中心化的形式存在于 PKI 系统中，并受到了所有实体的绝对信任，一旦其出现问题那么将会瓦解整个信任体系。为了防止授权中心的不端行为，如何去平衡 PKI 系统中实体的权利一直都是大家说关注的问题。如果授权中心在签发证书前也需要用户同意，那么将会大大减小授权中心被攻破或有不端行为所带来的影响。

区块链作为比特币底层的核心技术，具有去中心化、防篡改的特性，使得比特币与传统货币相比无需信任中心，不依赖第三方来保证其上交易过程的安全性。比特币由于不需要中心化的发行机构，不会被组织或政府操控的特性，受到大家的广泛关注。

依据区块链的准入机制，可以将区块链分为公有链、联盟链和私有链。公有链是

公开透明的，任何实体都可以加入到公有链的区块链网络中，在其上可以发送交易并获得交易的确认；在公有链中每个人都可以参与到区块的扩展当中，根据共识算法来完成下一区块的生产。联盟链是半公开的，按照准入规则只有特定的个人、群体或者组织才可以加入到该网络中；该网络中区块的维护者可能是预先选取的实体，每个区块的产生由他们共同决定，其它准入的实体可以在该网络中完成交易。私有链是完全封闭的，供一个实体或者公司所完全控制，仅用区块链技术进行记账，记账权不公开，而且只记录内部的交易。在不同的区块链类型中，将会使用到不同的共识机制，来完成对区块链的扩展。

随着区块链技术的不断发展，区块链的类型已经从比特币最初公有链的形式扩展到了上述的联盟链和私有链。在不同类型的区块链之中，将由不同的共识算法来完成对区块链网络的支撑。同时，智能合约在区块链中不断的被完善，可以支持更加复杂的逻辑操作，从而使用在很多传统的应用场景之下。区块链技术在各个领域和相关应用的结合，旨在利用区块链的去中心化和防篡改的特性去解决原有系统或者场景下中心化所带来的各种问题。

如前面所说的那样，PKI 中授权中心和用户之间权利不对等，如何去减弱用户对授权中心的依赖十分重要。而区块链技术作为应用在 P2P 网络中的底层技术，可以在无需中心的情况下，完成对网络中事务的状态统一，保证该网络中实体的各自权益对等。区块链的这些特性可以借鉴并应用到 PKI 系统中，解决 PKI 中实体之间的权利不对等所带来的安全隐患，巩固 PKI 系统的安全。

1.2 国内外研究现状

1.2.1 PKI 的研究现状

1976 年，美国密码学专家 Diffie 和 Hellman 提出了著名的 D-H 密码分发体制，从而解决了不使用秘密信道进行密钥分发的问题，允许通信双方可以在不安全的通信线路上完成密钥信息交换。在 1978 年，CA 认证中心由 Kohnfelder 提出，在他给出的方案中，CA 集中式的管理公钥，以 CA 证书的形式公布于目录库中，私钥继续使用秘密信道分发。1996 年 PKI 的解决方案被正式提出，PKI 设立 CA 认证中心，以第三方权威机构的身份完成对公钥和标识的绑定。

经过二十多年的发展，PKI 作为提供信息安全服务的基础性普适性设施，无论在理论上还是技术上，都已经日趋成熟。在理论方面，以公钥密码学作为基础的密码算法、协议以及认证方法都得到了全面的发展，为 PKI 技术提供了安全性保障；在技术上，数字证书的申请、签发和管理，数字证书的相关应用格式，以及 PKI 实体之的通

行协议都被相继提出并制定，使得 PKI 技术可以按照相应的标准使用在一系列的场景之下。

由于公钥基础设施 PKI 的存在，我们可以安全地与仅拥有对方公钥的人进行安全通信，通过信任第三方机构 CA 签发的证书完成公钥与身份的绑定。从安全的角度去审视现有的 Internet PKI 生态系统，主要存在以下几个方面的问题：首先，最为严重的问题是 CA 拥有绝对的权利，在未经实体允许的情况下就可以给其签发证书；其次，对 CA 的信任是没有灵活度的，只存在信任与否的抉择；其三是在域名证书的验证过于薄弱，在传统的方法中只是进过简单的邮箱确认方式，并不是特别安全；最后是证书的吊销不能正常工作，在很多实际使用的情况下，对证书吊销的检查并不完整[2]。

如上所提及的那样，对 CA 的信任是绝对的，而且 CA 拥有绝对有的权利，单个恶意或者被攻破的 CA 可以给任意一个域名签发证书[3]; 同时对于该类假冒的证书，将会需要很大一段时间才能被发现。我们知道全球存在很多很多的 CA，不同 CA 的安全防护能力各不相同，攻击者只需要攻破单个 CA，即可签发任意域名的假冒证书，从而对该域名发起中间人攻击，达到攻击相关域名的目的。

为了检测通信过程中的证书是否可靠，一系列方案通过依赖方在浏览器端询问不同的证书提供者，对比得到的多个证书信息的一致性来确保证书的有效性。Perspectives[4] 在 2008 年被 Wendlant 等人提出，并通过插件的方式应用于 FireFox 之上，该方案通过在建立安全连接前询问不同的公证服务器来检测该域名的公钥是否合法，一旦发现得到的密钥不相同，将检测出是否有人发起了中间人攻击 (MITM)。然而由于证书的传播需要时间，对于新签发的证书可能需要一段时间才可以使用以上的服务，二次验证[5] 的方案在 09 年被提出，该方案的主要思路是向目标服务器请求两次证书：一次通过 TLS 连接，另外一次使用 Tor[5]；该方法在一定程度上同时解决了浏览器泄露用户隐私的问题。另外一个 Perspectives 存在的问题是用户向其请求证书的同时，暴露了自己访问的历史记录，泄露了自身的隐私。为了解决这一问题，Convergence[6] 的方案被提出，在用户向公证服务器请求证书的时，将不是直接向某一公证服务器请求数据，而是随机的选择一个公证服务器将其请求转发给其它公证服务器，类似洋葱路由的机制。

由于证书的有效期一般相对比较长，当一个新的证书被使用时，很有可能它是攻击者伪造的证书并发起了攻击，基于这一点 Certificate Patrol[7] 的方法在检测到新证书时会向用户发起提示；同时，一般情况下证书的申请也会向自己国家的授权机构发起，当得到一个其它国家的签发的证书时，同样很有可能是虚假的证书，CertLock[8] 就是基于这个想法而提出的方案。

一些方案希望通过域名或其它的限制来削弱 CA 的权利，HPKP[10] 就是其中的一种。HPKP 技术给予域名主动选择信任 CA 的权利，它的工作原理是通过响应头或者

分类	存在的方案
传统	基于 CA 的证书管理系统
证书验证	Perspectives (’08)[4]; 二次验证 (’09)[5];
	Convergence(’11)[6]; Certificate Patrol(’11)[7]; CertLock(’11)[8]; TACK(’12)[9]
签发限制	HPKP(’11)[10]; DANE(’11)[11]; CAGe(’13)[12]
签发审计	Sovereign Keys(’12)[13];Certificate Transparency(’12)[14];
	AKI(’13)[15]; CIRT(’14)[16]; ARPKI(’14)[17]; DTKI(’14)[18]

表 1.1　存在的解决方案

<meta> 标签告诉浏览器当前网站的证书指纹，以及过期时间等其它信息。DANE[11] 方案则希望借助 DNSSEC 来发布域名相关的公钥信息；也就是说，域名可以将其公钥保存在 DNS 记录中，当浏览器发起 DNS 解析服务的时候，就能够获取到相应域名所使用的公钥信息，并用于验证收到证书的正确性当中。CAge 方案则和 CertLock 拥有类似的出发点，通过在客户端浏览器上给各个 CA 设定允许签发服务器证书的顶级域名范围，来保证 CA 不能随意的去签发证书，一定发现其签发了范围之外的服务器证书，则会向用户发起警告提示。CAge 方案通过在客户端浏览器上给各 CA 设定所允许签发服务器证书的顶级域名范围; 一旦发现有 CA 签发了设定范围之外的服务器证书, 则警告提示用户

另外一些方案旨在让 CA 的行为变得更加透明，基本的想法是使用公共的日志来完成对证书签发的记录，从而利益相关方可以检查这些日志，查看是否存在错误签发的证书。Soveregin Keys(SK)[13] 是第一个使用公共日子服务器的 PKI 方案。在该方案中，允许域名拥有者宣称一个长期的主权密钥，该密钥存储在一个公共的只增服务器上，并提供多个镜像服务器供公众访问。当浏览器在建立安全连接之前，需要检查使用的证书是否被主权公钥签名，验证失败的情况下将不会建立连接。

Certificate Transparency(CT)[14] 是由 Goole 提出的另外一种方案，旨在帮助域名拥有者检测错误签发的证书。该方案中要求 CA 签发证书的操作都要记录在公开可审计的日志服务器上，未被记录的证书将得不到认可。证书透明的方案并没有考虑到证书吊销的问题，Certificate Issuance and Revocation Transparency(CIRT)[16] 方案在证书透明的基础上加入了证书吊销相关的审计服务，在日志服务器能够上维护可审计的证书撤销状态信息。更进一步，Distributed Transparent Key Infrastructure(DTKI)[18] 方案作为 CIRT

的改进，通过设置两种不同类型的日志服务器，来减弱对 CIRT 中单类型日子服务器的信任，使得其具有更高的安全性。

与 SK 方案相似，Accountable Key Infrastructure(AKI)[15] 也允许域名拥有者去申明证书相关的规则，比如哪些 CA 或者日志维护者可以为其提供服务、证书中至少需要包含多少个签名等的策略。这样更加进一步限制了 PKI 系统中单个实体的权利，不会因为系统中的单个 CA 的疏忽或者不端行为造成虚假证书的流通。Attack Resilient Public-Key Infrastructure（ARPKI）[17] 作为 AKI 的改进方案被随后提出，相比于 AKI，该方案在建模过程中加入了安全性验证，理论上具有更好的安全特性。

1.2.2　区块链技术的应用

区块链的概念最初在虚拟数字货币 Bitcoin 中提出的[19]，其后很多利用区块链的数字货币相继被提出。区块链技术应用在 P2P 对等网络中，拥有去中心化、防篡改等特性，解决了传统方案中依赖于中心化信任的问题。在比特币的简单记账的基础上，以太坊[20] 加入了图灵完备的脚本语言，使其可以运行智能合约，在区块链上可以完成更加复杂的逻辑操作。随着区块链技术的发展，其衍生出来的应用不再局限于数字货币，已经涉及到金融、物联网、公共社会服务、声誉系统及安全和隐私等方面[21]，如图1.1所示。

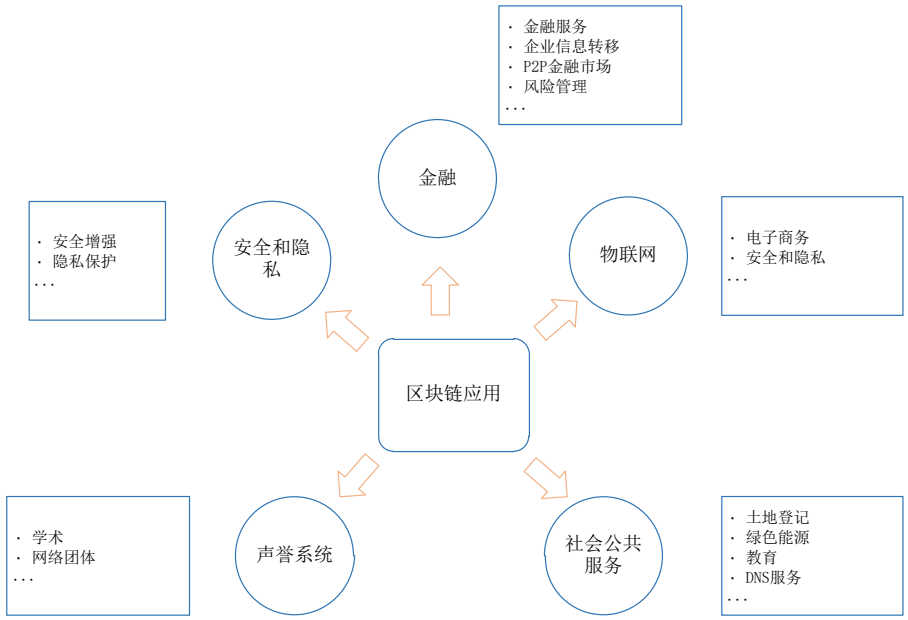

图 1.1　区块链典型的应用领域

金融

区块链技术最受关注的应该是金融方面的应用，其中很大一部分原因是因为区块链作为比特币的底层技术，其最开始衍生的相关应用都和密码货币相关。在 2015 年，已经有相关研究者讨论将区块链应用在银行当中[22]。在 16 年微软和 IBM 也相继推出了自己的区块链平台，希望作为一种服务提供给传统的金融机构。同时，区块链在企业资产转移、P2p 金融市场和分享管理上，都做出了相应的尝试。

物联网

物联网通过传感器将大量的设备连接在一起，其间通过传感器获取数据并自动的完成数据的交换，可以很适宜的将区块链应用到其中。如智能汽车、记录运输药物过程中的温度[23]、食品运输信息[24] 以及感知病人数据都是物联网和区块链相结合的一些尝试，不仅减少了制造商的成本，也让客户对相关商品信息更加放心，两方都能从中获益。一些研究者也对物联网和区块链的结合进行了探讨，[25] 中提出了一个全新的物联网电子商务模型，基于区块链上智能合约来完成交易的自动化；同时使用区块链完成无需可信第三方认证的物联网构建，提升了物联网的安全和隐私。

社会公共服务

区块链也被广泛的应用在各种社会公共服务当中，其中最为典型的例子是土地登记，将土地的信息如物理状态和所有权放置在区块链上[26]。区块链也被应用与加固互联网基础设施上面，如域名和身份的管理，Blockstack[27] 和 NameCoin[28] 就是将 DNS 服务建立在区块链上的尝试。同时，区块链也被用在如教育、能源、婚姻登记及税务系统构建[29] 等公共服务。

声誉系统

声誉是作为一个社区信任的衡量标准，可以通过其之前的行为和与社会的交互进行评判。Sharples 等人提出了基于区块链的分布式教育信息声誉系统[30]，用于记录各个教育机构和相关个人的声誉情况，任何相关的改变都可以很轻易的被检查到。[31] 中设计了一系列的投票写，将用户的反馈行为记录到区块链上，进而对服务供应商完成声誉评估。

安全和隐私

区块链技术同时也被用来巩固传统计算机相关服务的安全和隐私。Charles 在 2016 年提出了一个名为 BitAV 的反恶意软件环境，用户在区块链上发布和查询病毒的相关

信息，其改善了扫描速度并增强了系统的鲁棒性。就本文所述的 PKI 系统而言，也有很多方案被提出用于增强其安全性，如在 [32] 中，将撤销的证书存储在区块链上，提供更加可靠的证书验证服务，可以避免传统 OSCP 中单点失效的问题。IKP[33] 方案利用以太坊智能合约，将 PKI 中证书相关的操作都转移到区块链上，记录证书的申请、签发以及吊销过程，让 PKI 中的各个步骤都是公开可审计的，此时区块链相当于是公开且不可篡改的日志服务器，供大家审计 PKI 系统中的所有操作。MIT Media Lab 等人提出了名为 MedRec[34] 的项目，利用区块链去中心化的特性来提升病人对自己医疗数据的管理权，完成对数据访问和共享的控制，使得用户隐私得到了进一步保障。

1.3　本文研究工作和章节安排

通过上述分析和说明，可以得知现有的 PKI 构建在一个中心化的层级模型之上，其中存在着一系列问题，其中最为核心的问题是现有的 PKI 系统需要对 CA 给予绝对的信任。本文将围绕这一问题对 PKI 系统进行分析和讨论，并对区块链技术进行介绍，给出一种基于区块链技术的解决方案。

在本文提出方案中，域名实体可以在区块链网络中完成身份鉴权，之后可以记录自己信任的 CA 列表到区块链上，供依赖方在使用证书的时候对证书签发的合法性进行二次检查。本方案作为现有 PKI 系统的辅助手段，可以在不借助可信第三方的情况下，提升域名实体对证书签发的控制权，削弱对中心化 CA 的绝对信任，保证恶意的 CA 或者被攻破的 CA 签发的虚假证书不能用于中间人攻击，提升现有 PKI 系统的安全性。

本文结构如图1.2所示：

第一章：论文绪论，介绍本文的研究背景和研究意义，给出 PKI 的研究现状以及区块链的应用现状，提出研究的问题，并简要说明本文的组织结构。

第二章：介绍 PKI 的相关知识，介绍 PKI 系统和证书，给出 PKI 系统中存在的问题和已有的改善措施；其后介绍区块链技术的基本知识，包括比特币、共识和智能合约等概念，并介绍区块链的应用场景

第三章：介绍一种基于区块链的命名实体鉴权方案的设计，首先给出方案的概述，描述方案设计到角色和工作流程，然后对方案进行详细阐述，并对方案的安全性进行分析。

第四章：阐述本文给出方案的一种系统实现，给出系统设计和模块设计，根据设计对系统进行实现，并进行系统功能性测试和相关分析。

第五章：对本文提出的方案进行总结，并对未来的工作提出展望。

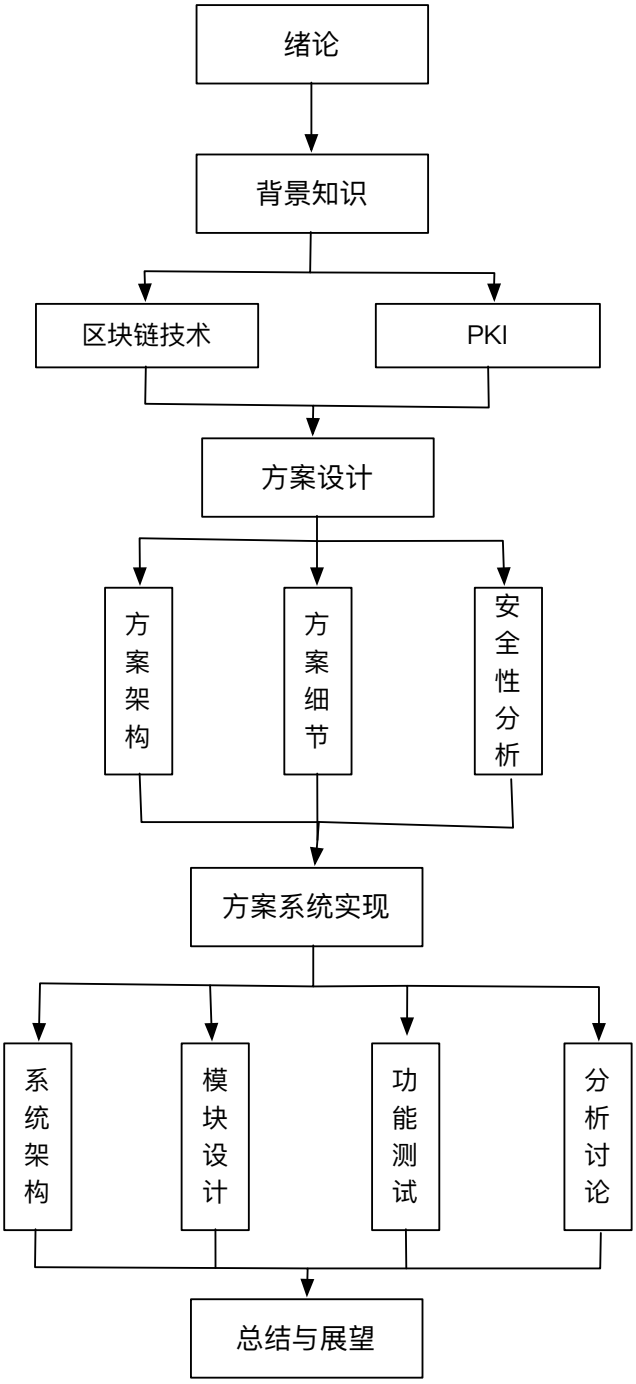
绪论
背景知识
区块链技术
PKI
方案设计
方案架构
方案细节
安全性分析
方案系统实现
系统架构
模块设计
功能测试
分析讨论
总结与展望

图 1.2　文章结构

第二章　PKI 与区块链技术

2.1　PKI 背景知识

本小节对 PKI 系统进行简要介绍，首先从系统架构对 PKI 进行解析，阐述本系统中包含的组件和各自功能；然后对证书进行介绍，包括证书的结构以及证书的生命周期。

2.1.1　系统架构

PKI 是由安全和操作策略、安全服务以及支持公钥密钥和证书管理的交互式协议构成的。公钥和对应证书的生成、分发和管理将通过授权机构（CAs）、注册机构（RAs）和目录服务来完成[35]，它们将会建立等级信任或者说信任链。以上提到 CAs、RAs 和目录服务可以将数字证书用于鉴定不同实体的身份，而 PKI 拥有如此架构的目的是为了能支持并完成数据、凭证在各种不安全环境下的安全交换。

在 PKI 中主要涉及的实体和相关组件如图2.1所示。

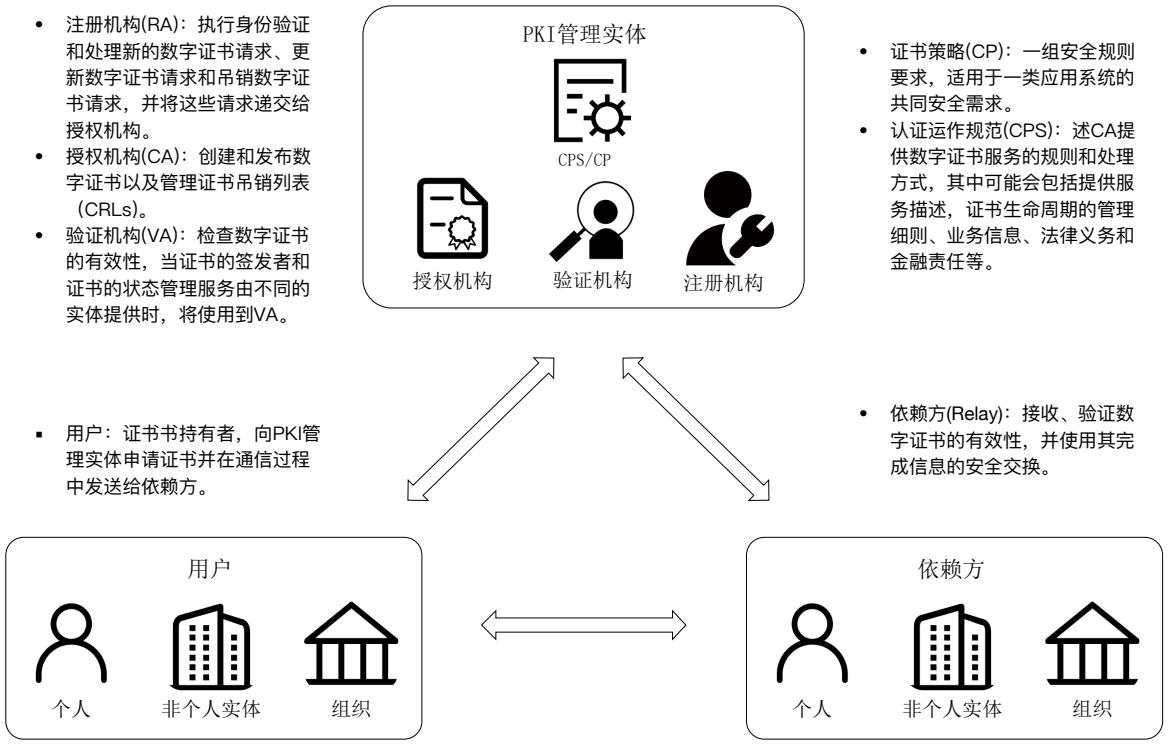

图 2.1　PKI 实体和组件

2.1.2 证书

证书的结构

在 PKI 的发展历程中，存在着多种数字证书的类型，每种类型的证书应用在不同的场景之中，因此也拥有各自独有的格式。目前最为通用的证书标准为 X.509，该数字证书标准由国际电信联盟 (InternationalTelecommunication Union, ITU) 制定，于 1988 年公布最初版本。此标准的证书最为核心的部分是公钥和用户标识符，同时，X.509 公钥证书中也可以包含证书版本号、序列号、签名算法标识、签发者名称、证书持有人名称和证书有效期等信息。X.509 证书标准的最新版本是 X.509 v3，在该版本中定义了数字证书的扩展信息，使得数字证书的功能可以进一步扩展，具有更大的灵活性；当数字证书在特殊应用环境下使用时，该扩展信息还可以起到传递附加信息的作用。

RFC3280[36] 对 X.509 中包含的字段和各个字段的含义进行了详细描述，该类证书中主要包含三个部分：待签名的证书、签名算法和签名值。

```
Certificate  ::=  SEQUENCE  {
    tbsCertificate          TBSCertificate, \\待签名的证书
    signatureAlgorithm      AlgorithmIdentifier, \\签名算饭
    signatureValue          BIT STRING   \\签名值}
```

Listing 2.1 X.509 证书结构

待签名的证书中包含了需要纳入的的相关信息，如2.2所示。

```
TBSCertificate  ::=  SEQUENCE  {
    version          [0]  EXPLICIT Version DEFAULT v1, \\证书版本号
    serialNumber          CertificateSerialNumber, \\证书序列号
    signature             AlgorithmIdentifier,  \\签名表示算法
    issuer                Name,  \\证书签发者
    validity              Validity, \\证书有效期
    subject               Name, \\证书持有者
    subjectPublicKeyInfo  SubjectPublicKeyInfo,  \\证书持有者公钥信息
    issuerUniqueID   [1]  IMPLICIT UniqueIdentifier OPTIONAL, \\证书签发者唯一标识
                          -- If present, version MUST be v2 or v3
    subjectUniqueID  [2]  IMPLICIT UniqueIdentifier OPTIONAL, \\证书持有者唯一标识
                          -- If present, version MUST be v2 or v3
    extensions       [3]  EXPLICIT Extensions OPTIONAL  \\扩展
                          -- If present, version MUST be v3
    }

Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }

CertificateSerialNumber  ::=  INTEGER
```

```
Validity ::= SEQUENCE {
    notBefore       Time, \\开始时间
    notAfter        Time  \\截止时间}

Time ::= CHOICE {
    utcTime         UTCTime,
    generalTime     GeneralizedTime }

UniqueIdentifier  ::=  BIT STRING

SubjectPublicKeyInfo  ::=  SEQUENCE  {
    algorithm             AlgorithmIdentifier, \\标识算法
    subjectPublicKey      BIT STRING  \\公钥}

Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
    extnID        OBJECT IDENTIFIER,  \\扩展ID
    critical      BOOLEAN DEFAULT FALSE,  \\级别
    extnValue     OCTET STRING  \\ 扩展值}
```

Listing 2.2　X.509 证书包含字段

证书的生命周期

证书作为包含公钥、数字签名以及一些其它附带信息的数字文档，在 PKI 系统中充当着公钥交换、存储和使用的介质。了解证书的申请、签发和使用流程，可以明白 PKI 系统的是如何运作的，并从中发现可能存在的问题。

证书的声明周期从用户提交准备的证书签发请求 (CSR) 并提交给其选择的 CA 开始。CSR 中包含了用户的公钥和需要纳入的信息，并通过签名的方式表明对相应私钥的所有权。同时，CSR 可以携带额外的信息元，但在实际使用过程中并没有全部使用。授权机构可以对 CSR 中的内容进行重写，放置一些其它的信息在证书中。

其后 CA 遵循验证流程，对用户进行身份验证。待成功完成验证之后，CA 将签发证书，同时提供验证至根证书的所有中间证书。得到证书后，用户就可在证书过期之前使用证书。如果证书对应的私钥泄露，证书将可以被吊销，该过程和证书签发的过程类似。

对于 Internet PKI 系统而言，根据以上证书流转流程，证书的生命周期如图2.2所示.

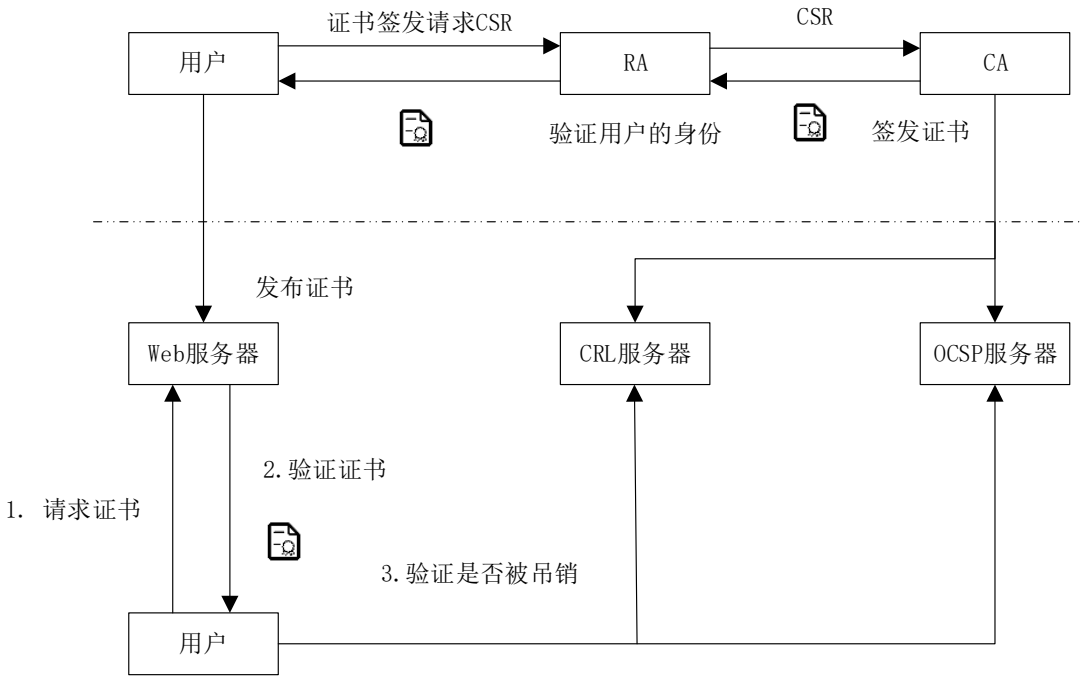

图 2.2　证书的生命周期

2.2　PKI 系统的问题与解决方案

从安全的角度来看现有 Internet PKI 系统，其中存在着大大小小的各种问题，在本小节中，将对这些存在的问题进行简要概述，并列出一些已有的解决方案。

2.2.1　主要问题分析

当 Web 安全在 1995 年刚开始被谈及时，当时的互联网和现在是具有很大差异的，它的重要性并没有现在这么强。随着互联网的蓬勃发展，加密安全成为了商业中必备的一部分，关乎这一个企业的生死。现有的 PKI 系统和当初设计的目的是一致的，即为上电子商务操作提供足够的安全保障，更进一步可以说 PKI 系统希望提供商务安全。这种安全可以通较少的资金、更快的网页获取速度、可接受的不安全操作和不对用户做过多的限制来完成。本系统由 CAs、寻利的商业实体和希望扩大市场份额的浏览器供应商一起来组成。

CA 在本系统中充当着信任的基础，其每年签发了数百万计的证书，这些证书将在互联网中流通并使用。虽然这个系统正在正常的运转着，但是证书的安全性并没有想象中那么好。同时，在不那么完美的条件下，有时候用户还需要对申请的证书进行付费，然而并不是所有用户都愿意为之付出代价的，很简单的一个原因就是他们在付费

之后希望得到完美的安全性保障。

在该系统中，存在着以下几个方面的问题[2]：

- **域名权利过弱**

  在 PKI 系统中最大的问题是任意 CA 在未经域名同意的情况下就可以对其签发证书，导致这个问题的主要原因是系统需要对 CA 给予相当高的信任，但是没有相关的技术策略去避免 CA 的疏漏和安全隐患。当 CA 数量比较少的时候，这个问题并没有那么严重，但是当下有数以百计的 CA 存在，很难保证每一个授权结构都不会存在不端行为或者不当的安全配置。一个系统的安全性取决于该系统最薄弱的环节，而 PKI 系统中存在着各种潜在的薄弱环节。虽然所有的授权机构都会接收到审计，但是审计的质量却各不相同。例如在 2011 年 DigiNotar 由于自身安全性问题就被黑客攻陷[1]，签发了数百个虚假证书，造成极其恶劣的影响，最后导致自身倒闭。

  同时，另外一个存在的问题是 CA 是否可以给予信任，它们能否在不需要监督的情况下为了公众的利益去做好自己的本质工作。这些被信任的 CA 可能在面对商业利益的时候放弃公众所需要的安全。例如在 2012 年 Trustwave 承认其签发了低级别的假冒证书用于流量检查。虽然 Trustwave 是唯一公开承认自己做过类似事情的 CA，但大家相信这样的事情肯定大量存在。

  政府也可能会滥用 PKI 系统签发的虚假证书，完成对任意域名的假冒。公众无法确保 CA 不会作为政府的前线，即使不是也无法保证这些 CA 不会被迫签发虚假证书。

- **没有信任灵活度**

  另外一个重要的问题是本系统缺乏信任的灵活度。依赖方将会存储一系列信任的根证书，一个 CA 只存在信任与否，并不存在中间地带。理论上，依赖方可以移除任何对任意 CA 的信任，实际上这种情况只会在 CA 很小或者其已经被攻陷的时候发生。一旦一个 CA 签发了大量的证书，其将由于自身的大体量不会被撤销。一些小的改进措施在逐渐被提出，例如对具有过失行为的 CA 不再信任，但是其之前签发的证书仍然可以被使用。

- **域名验证过于简单**

  DV 证书的签发是基于域名的 WHOIS 协议查询域名拥有者信息来完成的，也就是说大部分验证是通过邮件来完成的，而其本身的安全性就存在问题。如果域名被黑掉或者相应的邮箱密码被获取，那么就可以得到给域名的 DV 证书。同时通过拦截 CA 端验证信息也可以发起攻击。

- **客户端验证不完整**

在一般情况下，对吊销证书的检查并没有那么严格，大多数情况都不能正常的工作。在 2011 年中有很多这样的例子，依赖方不得不将被泄露的证书通过特殊通信方式下载并存储在黑名单中，来保证吊销查询是可靠的。

这样做的原因主要包括以下两点，首先将吊销信息发送各个系统需要一定延时，在基准规则中允许 CRL 和 OCSP 的信息在 10 天是有效的，也就是说至少需要 10 天才能保证吊销信息被完全扩散出去；其次软失败机制在所有的浏览器中被使用，当依赖方在查询吊销信息时，如果没有收到查询回复时将任务证书未被吊销，一个主动的网络攻击者将可以轻易的拦截 OCSP 的请求，保证被吊销的证书可以完美的被使用。

由于以上的原因，Chrome 开发者取消证书吊销的检查，除非是 EV 类型的证书。对于重要的证书，例如中间证书，其将依赖于 CRL 信息的吊销通道查询相关信息。一种可能的解决方案是使用 Must-stale 的方案来保证证书的有效性。

2.2.2 现有的解决方案

为了解决 X.509 PKI 中的安全问题并削弱对 CAs 的信任，一系列的方案被提出来，本小节将对这些方案进行分类并简要叙述。

根据 PKI 中存在的主要三方实体域名、CA 和客户端，可以对现有的方案进行分类，如图2.3所示。

以客户端为中心的方案

这类方案希望在客户端接受证书之前，可以更加准确的验证证书的有效性。Policy engine[37] 的方案允许客户端在本地制定信任决策，比如支持的密码算法，证书的一致性性等。

有一些其它的方案希望构建一个公共的域名证书存储厂库，使得客户端可以在接受到域名证书之后与厂库中的证书进行对比，确定证书的正确性，Perspectives[4] 和 Convergence[6] 就属于这类方案。

以客户端为中心的方案并不需要对服务端进行的改动，但是其需要客户端建立额外的连接去查询资源厂库，这将在一定程度上降低建立 HTTPS 的速度。

以 CA 为中心的方案

X.509 PKI 包含了证书吊销列表（CRL) 标准[38]，希望防止客户端与一个使用已经被吊销证书的域名之间建立 TLS 连接。但是这需要保证客户端需要能随时的访问到 CRLs。为了进一步解决在线验证的问题，在线证书状态协议（OCSP）[39] 允许客户端通

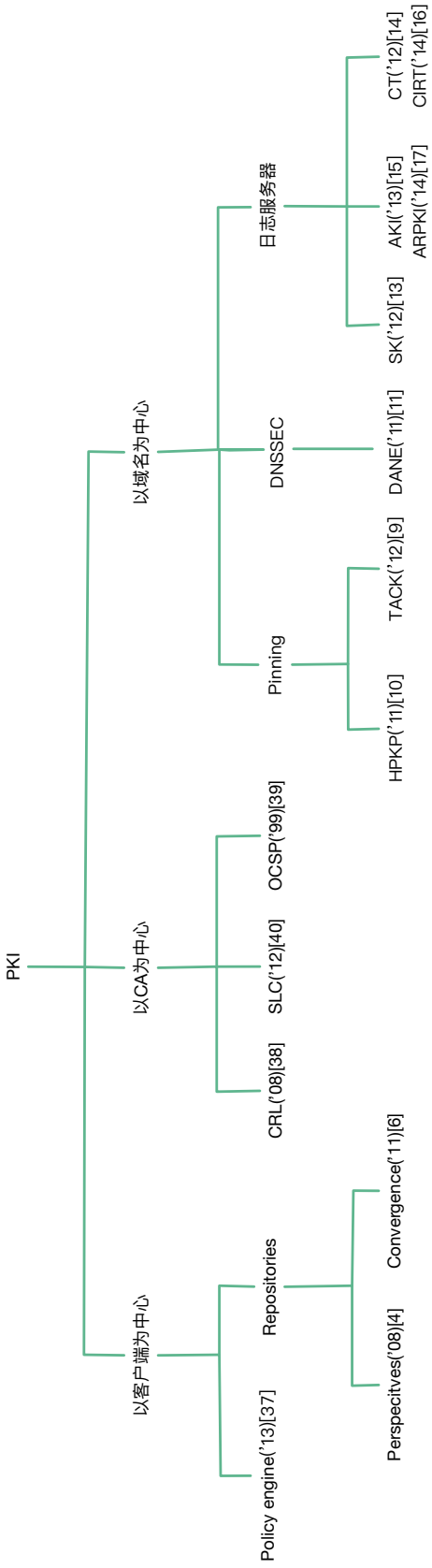
PKI
以域名为中心
以CA为中心
以客户端为中心
日志服务器
DNSSEC
Pinning
OCSP('99)[39]
SLC('12)[40]
CRL('08)[38]
Repositories
Policy engine('13)[37]
CT('12)[14]
CIRT('14)[16]
AKI('13)[15]
ARPKI('14)[17]
SK('12)[13]
DANE('11)[11]
TACK('12)[9]
HPKP('11)[10]
Convergence('11)[6]
Perspecitves('08)[4]

图 2.3　PKI 方案分类

过 CA 的 OCSP 服务器去检查域名证书的状态。但是 OCSP 也拥有如安全、隐私、性能方案的顾虑。另外一个解决方案是短期证书（SLC）[40]，该方案希望签发短生命周期的证书，让域名定期的更换证书。SLC 希望带来和 OCSP 类似的好处，但是不需要在线验证。以证书为中心的方案严重依赖于浏览器去检测和拉黑被攻陷 CA 签发的证书，这是这类方案的最大缺陷。

以域名为中心的方案

第三类方案允许域名拥有者积极控制和保护他们的证书而不被 CA 端的潜在问题所威胁。这些方案又可以分为以下三类：pining、DNSSEC、日志服务器。

1. Pinning-based（公钥固定）

如 Public Key Pining(PKP)[10] 和 Trust Assertions for Certificate Keys(TACK)[9] 等 Pinning 的方案希望域名宣称自己使用的密钥，以便客户端收到的密钥是否正确。但是这类方法有自己的安全缺陷，比如在第一次访问域名的时候无法给予安全保护。

2. DNSSEC-based

这类方案的全称叫基于 DNS 的命名实体鉴权（DANE）[11]，希望域名的所有者可以在 DNSSEC 实体上放置证书相关的特殊声明，例如可以为其签发证书的 CA 名单、声明接受的证书或者是声明验证证书有效性的站点，但是 DANE 的安全性严重的依赖于 DNS 操作的安全性。

3. 日志服务器

另外一种使用比较多的方法是通过日志服务器的方案来记录 CA 的行为，为域名拥有者提供公共的、可审计的 CA 行为日志，监督 CA 的行为。例如 Sovereign Keys (SK)[13] 要求域名拥有者生成一个主密钥对来完成对 TLS 公钥的签名，并且将其主公钥以只读和只可追加的方式记录在时间服务器上。然而，SK 需要客户端查询服务器，增加了延迟并牺牲了隐私。

证书透明（CT）[14] 的方案允许每个域名将自己的证书注册记录到一个公共日志服务器上，该服务器使用默克尔哈希树的结构来存储这些证书，并保证只能追加的性质。该服务器将返回一个不可否认的证书审计证明给域名，域名使用证书和该证明来完成与客户端 TLS 连接的建立。但是，本方案并不能防止当一个攻击者攻破了 CA 并创建并注册一个虚假的证书的情况，CT 并不能阻止客户端接受这类证书。由于证书透明在设计过程中并没有强调证书的吊销，相应的证书吊销透明（RT）[41] 也被提出。为了提高 CT 对证书吊销的效率，证书签发吊销透明（CIRT)[16] 方案被提出。

另外一种基于日志服务器的方案称作可审计密钥设施（AKI）[15]，该方案希望保护域名和客户端遭受到单点失败攻击，比如某个 CA 的根密钥被泄露，通过制衡该系统

中各个实体，AKI 在保持高效处理证书操作的情况下，成功的将信任分散到了多个实体，并可以检测出实体的恶意行为。

2.3　区块链技术

区块链技术作为当下作为最火热的技术之一，从 09 年以比特币的形式出现在人们的视野中，就一直被大家关注和讨论。本章节将从比特币出发，给出区块链技术的整体概况，其后将对区块链技术中的核心部分共识机制和智能合约进行简要介绍，最后简单列举区块链技术典型应用场景。

2.3.1　比特币与区块链技术

比特币作为迄今为止最为成功的数字货币，是由中本聪在 08 年中的一篇论文中提出[19]，整个网络如下图2.4所示。

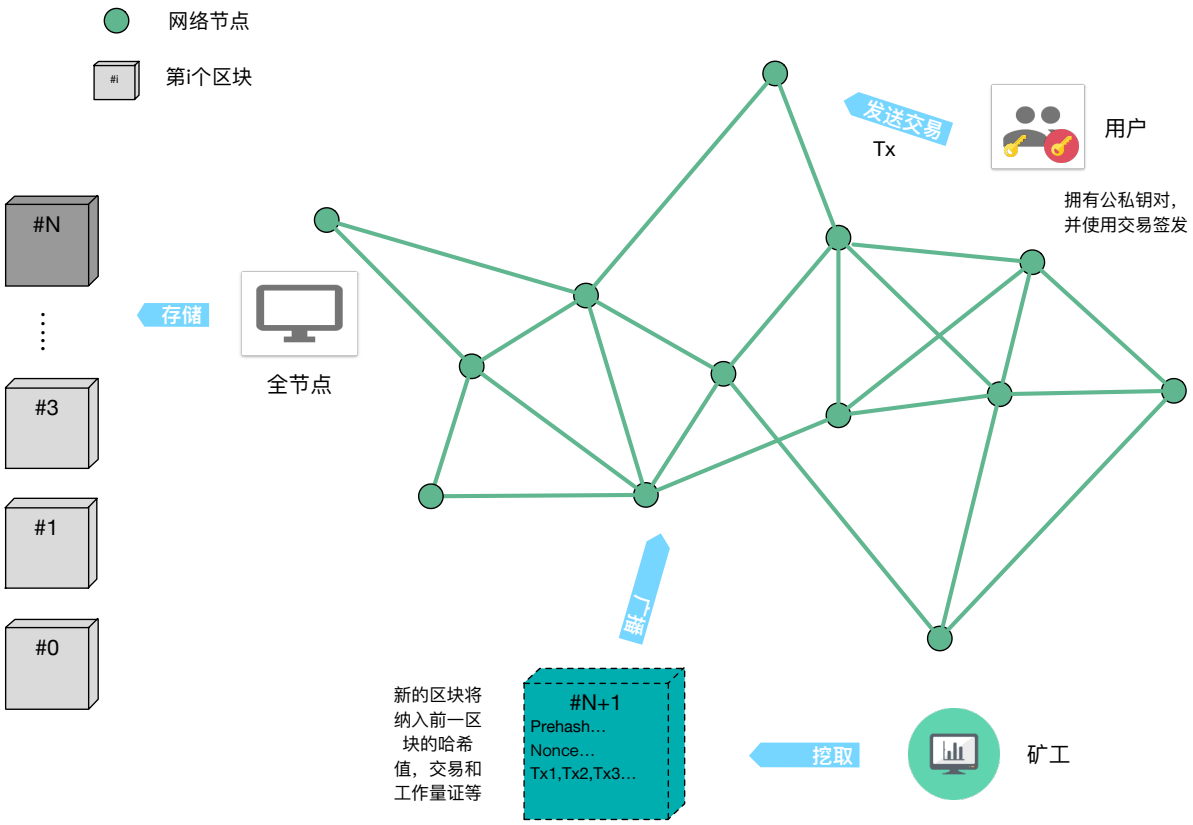

图 2.4　bitcoin 网络

比特币是构建在一个 P2P 网络上的数字货币系统，由拥有公私钥对的用户、网络中传播的交易和通过计算竞速的生成块的矿工共同组成。在该系统中，用户使用私钥完成对已有资产（比特币）的转移，形成有效的交易递交到比特币构建的去中心化网络

中；而矿工作为网络中区块的生成者，为了获得建块的奖励和纳入交易后的交易费，将有效的交易纳入新的区块，并使用自己的计算资源不断尝试下一个区块的构建；当某个矿工率先完成下一区块的创建时，会将新的区块立即广播给网络中的其它矿工，待他们验证通过后会在被接收的区块上进行一下区块产生的竞速过程。

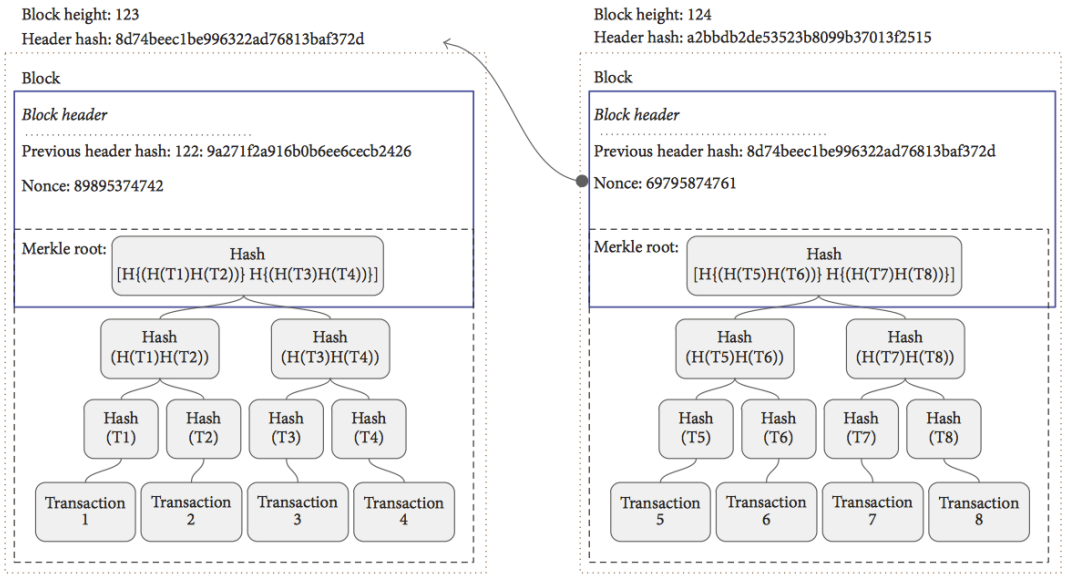

图 2.5　区块结构

区块作为比特币中交易记录的载体，通过纳入上一区块哈希的方式保持所有区块的连贯性，其结构如图2.5所示。一个区块由区块头和交易两部分组成，交易部分包含了本区块的所有交易的详细内容。在区块头中主要包含了上一区块的哈希值，用于指明上一区块；以及本区块中所有交易组建 Merkl 树的根和通过大量计算得到一个工作量证明的 nonce 值。区块创建过程即是矿工在区块链中所谓的挖矿，而挖矿的本质其实就是计算区块当中的 nonce 值。

2.3.2　共识机制

区块链网络中的节点需要对新生成块中的交易及顺序达成共识，从而达到相同的状态，否则任何节点都可以产生分支而导致分叉。如果网络中的节点各自拥有不同的状态，那么该网络就无法对所有操作做出相同的判断，这个系统就不能协同运行下去。在一个分布式的 p2p 网络中，在没有第三方协调分歧的情况下，就需要有一种机制来保证每个节点可以达到相同的状态，而区块链上的共识机制就解决这一问题的，保证网络中各个节点的一致性。

在理想的情况下，所有的验证节点对下一个区块中的交易顺序进行投票，根据大多数人的选择来确定下一区块如何产生。但是在一个公开且没有身份管控的开放网络

中，任何人都可以自由的加入，将会遭受到女巫攻击[42]: 单一节点可以拥有多个身份表示，通过控制系统中的大部分节点来左右网络向对自己有利的方向进行。也就是说，少数的个体可以通过女巫攻击完成对整个网络的控制。

比特币通过提高产生块的计算量来解决这一问题，使得在网络中添加多个实体并不能发起女巫攻击，因为对于单个实体而言，其计算资源是有限的。更具体一点，网络中的任意节点都可以去产生下一个块，但是其需要找到一个正确的随机值 (nonce) 填入到区块的头部，使得产生块的哈希拥有至少指定数量的前导零[43]。任何节点都可以对这个问题进行尝试求解，也就是所谓的工作量证明（proof of work)，来完成对下一区块的构建[19]。在该过程中使用的是单向哈希函数，当一个节点宣称自己找到一个解的时候，其它节点很容易的去验证是否符合要求，但在给出目标的时候却无法直接确定输入应该是什么，只能通过不断尝试来玩寻求答案。

依照上述方案，当网络中的节点同时计算出下一个块时，依旧有可能存在分叉的情况。但这种分叉并不会带来影响，因为比特币中的工作量证明机制规定节点应该在工作量最大的链上进行下一区块的产生，而两个分叉的区块基本上不可能同时产出下一个块，先产生块的分支将会成为网络中被网络中节点选中的链，会有更大的几率成为更长的链，这使得网络又恢复到了统一的状态。

工作量证明的方式虽然简单有效，但是其需要很多计算资源才能进行区块的产生，并且大量的计算导致能源浪费。权益证明（proof-of-stake)[44] 是另外一种替代 PoW 的共识算法，根据节点的账户余额来决定下一块的生成权；DPOS 则对 PoS 进行了一定的改进，让区块链上的用户通过自己持有的账户余额进行信任节点选取，选择自己信任的节点进行区块链的构建。这两种共识算法与 PoW 相比，都能在开放网络下工作，同时还可以降低能源消耗。另外一些典型的共识算法可以参见表2.1。

	PoW[19]	PoS[44]	PBFT[45]	DPOS[46]	Ripple[47]	PoC[48]
网络类型	开放	开放	准入	开放	开放	开放
能耗	高	一般	低	一般	低	低
容错能力	< 33.3% 算力[49]	< 51% 股份	< 33.3% 恶意节点	< 51% 验证节点	< 20%UNL 恶意节点	< 51% 容量
实例	Bitcoin	Peercoin	Hyperledger Fabric	Bitshares	Ripple	Burstcoin

表 2.1　典型的共识算法对比

空间和时间一直是两个对立面，很多算法为了降低时间复杂度不得不牺牲大量空间，最近另外一种名为容量证明共识机制也被很多研究者讨论。容量证明[48] 的原理是需要大量的空间去寻求一个谜题的答案，如果能够给出给定问题的解，那么证明其拥有住够多的空间。具体而言，容量证明会基于矿工的公钥进行初始化，这将耗费矿工大量的磁盘空间，并将初始化的相关内容存储在矿工的计算机中；其后在挖矿的过程中，需要对区块链上的谜题进行求解，每个矿工根据本地存储相关内容，通过对磁盘的扫描获得对应的解，并完成区块的生成。以上两个过程分别是初始化过程和挖矿过程，值得注意的是，初始化过程是一个很慢的过程，单个矿工可能需要耗费数天来完成初始化，这样做的目的为了防止提过快速的计算硬件来替代存储；对于挖矿过程而言，是一个极快的过程，只需要对磁盘进行扫描即可，并不需要像工作量证明那样耗费大量的能源。随着硬件的运算能力越来越高，工作量证明的方式从原有的 CPU 挖矿已经转变成 ASIC 挖矿，使用容量证明的方式可以抗 ASIC 挖矿，这也是广大研究者对其感兴趣的原因。

2.3.3 智能合约

智能合约的概念于 1994 年被 Nick Szabo 提出[50]，是一种旨在以信息化方式传播、验证或执行合同的计算机协议，允许在没有可信第三方的情况下进行交易，完成的交易将是可追溯且不可逆的。Szabo 建议将合同的条款转换成代码，并将代码放置在特定的软件或者硬件环境中运行，减少交易方对可信中介的依赖，以及第三方的避免不端行为或者意外操作。由于一般网络中缺少可信的执行方，使得智能合约在提出之后并没有被广泛应用，而区块链提供了可信的执行环境，因此智能合约在区块链出现之后逐渐被火热起来。

在区块链的环境下，智能合约是存储在区块链上的脚本。由于智能合约存在于链上，每个智能合约会拥有独一无二的地址，并通过发起对相应地址的交易来实现对智能合约的调用。收到出发交易后，智能合约将会在每个节点上按照合约的设定，自动独立地执行合约内容。事实上在每个节点上都运行着一个虚拟机 (VM) 来执行合约代码，区块链网络充当着分散式 VM 的角色。

区块链技术最开始仅仅是用于转账交易的记录，而智能合约的应用使得在区块链上可以开发出功能更加丰富的应用。用户根据自身的需求，使用智能合约语言将需要的逻辑操作编写成合约代码，并以交易的形式发布到区块链上，该步骤即为智能合约的书写和部署。当智能合约存在于区块链上之后，用户就可以向合约的地址发送交易来调用合约，完成合约内容的执行，即智能合约的调用。

不同区块链平台在实现智能合约功能时，选取的合约执行引擎和合约语言将直接

平台名称	合约引擎	合约语言	是否可编程	是否图灵完备
Bitcoin[19]	Bitcoin script	Ivy-lang, Balzac	否	否
Ethereum[20]	EVM	Solidaty	是	是
Byteball[51]	Byteball	JSON	是	否
hyperledger[52]	Sandbox	Go, Java, node.js	是	是
EOS	EVM/eWASM	C/C++	是	是

表 2.2　典型智能合约平台

决定智能合约能够实现的操作，比较典型的平台如表2.2所示。比特币上使用了基于堆栈式的脚本语言，指令集小并且不支持循环等复杂运算，是非图灵完备的；同时，可以认为比特币上的智能合约是脚本指令的组合，是不可编程的。以太坊之后的区块链平台，基本上都拥有图灵完备的合约语言，各个平台都拥有相应的合约引擎以完成合约的执行；与比特币相比，其还具有可编程的特性。通过智能合约的可编程特性，在这些平台上可以完成更多的应用逻辑实现，极大的扩展了区块链本身的功能，可以应用在数字货币之外的场景。特别是对于金融领域、智能资产、托管支付等许多场景都能通过智能合约来实现。区块链为智能合约的执行提供了可信的环境，而智能合约则给区块链赋予了更加宽广的应用场景。

2.3.4　区块链在 PKI 中的应用

由于 PKI 系统中的信任是基于第三方中心化的 CA，当单个 CA 被攻击之后，将会导致整个系统的处于危险之中，同事该系统中提供的服务都存在着单点失效（POF）的风险。一些研究希望通过将区块链技术应用在 PKI 系统中，弥补现有体系中的一些缺陷，例如在 [32] 中，将撤销的证书存储在区块链上，提供更加可靠的证书验证服务，可以避免传统 OSCP 中单点失效的问题。IKP[33] 方案利用以太坊智能合约，将 PKI 中证书相关的操作都转移到区块链上，记录证书的申请、签发以及吊销过程，让 PKI 中的各个步骤都是公开可审计的，此时区块链相当于是公开且不可篡改的日志服务器，供大家审计 PKI 系统中的所有操作。

Certcoin[53] 借鉴 Namecoin 实现了一个基于区块链的 PKI 系统，其通过默克尔哈希树来完成对身份信息的存储并通过 Kademlia DHT 完成快速查询。Nidaba[54] 则从分布式 PKI 的可扩展性和证书操作的代价出发，基于区块链提出了一套完整的架构；SCPKI[55]

利用 web-of-trust 模型以及以太坊上的智能合约，提出了一种去中心化且透明的方案，保证异常证书被签发时能够及时的被检测到。

第三章　域名鉴权方案设计

为了解决现有 Web PKI 系统中信任不平等构建问题，削弱给予中心化授权机构 (CA) 过高的信任和权利，提高域名实体对自身证书的管控权，本章将提出基于区块链的域名鉴权方案；首先给出方案的相关概述，对其中的角色和流程进行描述；然后对设计的域名鉴权方案进行详细描述，并对设计的方案进行安全性分析。

3.1　方案概述

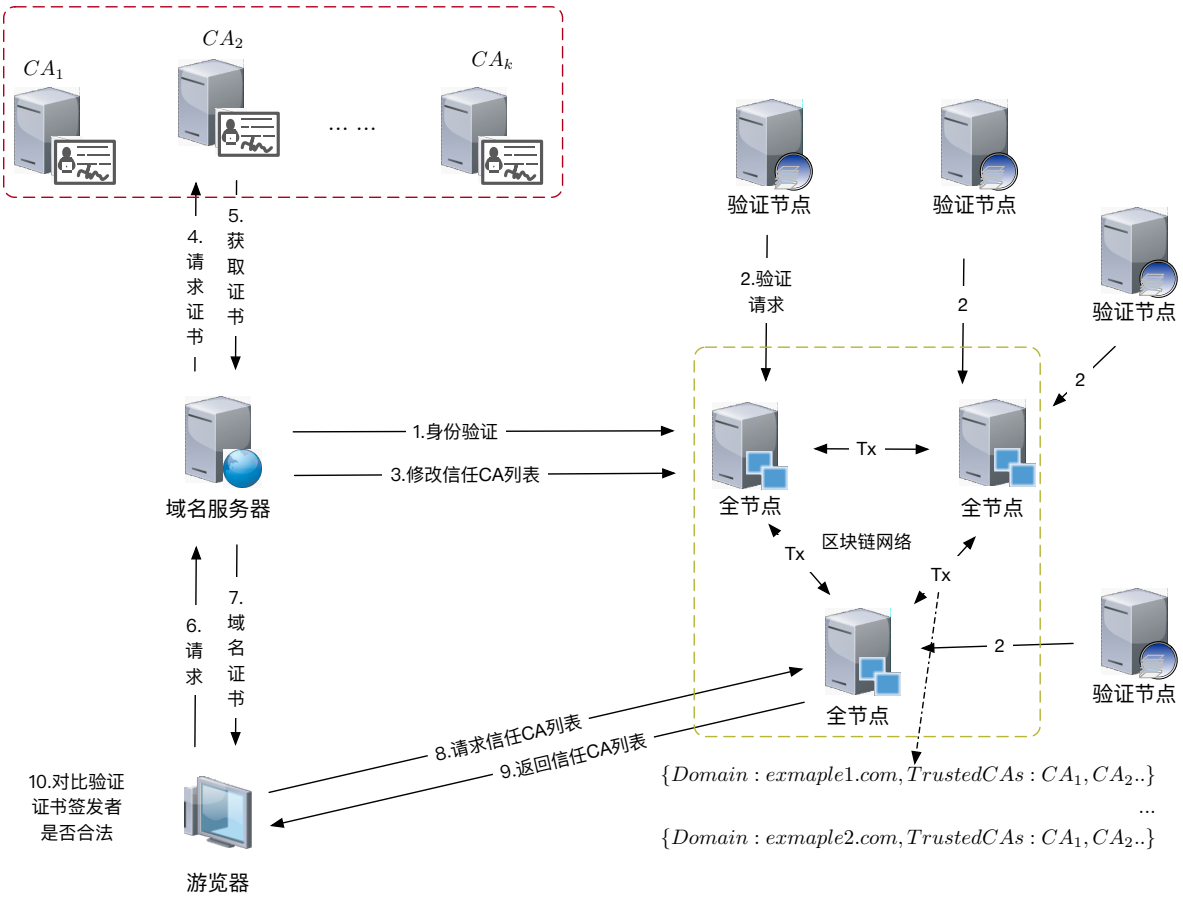

图 3.1　域名鉴权方案

在对身份鉴权方案进行详细描述之前，首先对本系统从上层进行透视，了解系统的工作原理和方式，明白设计方案的目的。在本方案中包含的主要实体包括一下几类：

- **授权机构（CA）**：作为证书的管理者，负责对发起证书申请的实体进行证书颁发。

- **证书申请者（域名服务器）**：证书的持有者，依照本文设计的方案进行身份绑定，并将自己信任的 CA 列表发布在区块链上。
- **证书验证者（游览器）**：与域名完成 TLS 建立。
- **验证节点**：依据设计的方案完成对域名身份的绑定。

域名作为证书的申请者和持有者，在原有的 PKI 系统中需要向授权机构进行证书的申请操作，以及向与通信的实体提供持有的证书已证明身份。在本方案中，域名将作为核心角色来完成对证书签发相关的控制，方案的整体概况如图3.1所示。

在域名进行证书申请前，需要通过区块链完成信任列表的存储：首先其向区块链网络发起身份绑定请求，待交易被区块链网络接收后，验证节点对这些请求进行验证；当域名的身份绑定完成之后，域名发送自己信任的 CA 列表到区块链网络中，完成对可以为自己签发证书 CA 的控制。

其后域名作为 PKI 系统中的证书申请者，与在之前的系统中一样向授权机构发起证书申请，授权机构对域名进行身份核实和一系列检查之后完成证书的签发。在该过程中，CA 作为证书的签发方，也可以通过区块链来进行证书申请者的信任 CA 列表查询，来确认自己是否被该申请者纳入到了信任列表之中。但是这样做将对原有 PKI 系统的 CA 进行一定的更改，并且这样做并不会对后面的检查造成实质性的影响，只是确保了自身签发证书的合理性，不会影响到证书相关的验证。所以在本方案中并不要求授权机构在完成证书签发的时候通过区块链来做进一步的检查。

当依赖方通过浏览器向域名发起安全连接时，将会获得域名持有的证书；首先和在原有 PKI 系统中一样，对证书进行签名相关的验证，确认证书是否由可信的授权机构签发。其后，浏览器会向区块链请求该域名的信任 CA 列表，并将得到的信息与证书进行对比，查看是否由可信的 CA 签发。

3.2 域名鉴权方案设计

在本方案中，域名需要将自己信任的 CA 列表记录到区块链上，但是区块链具有去中心化和匿名的特性，域名直接将自己的信任 CA 放置到区块链上是不具有可信性度，也就是说在区块链上任何人都可以对任何域名进行声明。因此，之前需要完成区块链地址（或者公钥）和域名本身的绑定，实现地址与域名之间的实体认证，完成对域名的鉴权。本文将提供基于验证时间和基于验证次数的鉴权方案。

身份绑定过程大致分为 3 个步骤：

1. 域名发送认证请求到区块链上；
2. 待交易确认后，根据交易和区块在服务端放置验证；
3. 验证节点对其进行验证，达到验证期限或者次数后，完成身份绑定。

3.2.1　基于验证时间的鉴权方案

基于验证时间的方案旨在让绑定者提供验证信息持续一段时间，以供所有节点对其进行验证，当提供的验证达到指定时间后，即认为其对该域名服务器的所有权，完成指定地址和域名的绑定。

假设

假设绑定者 A 拥有公钥 Pk_A、私钥 Sk_A 以及域名 $example.com$，其需要完成 $(Pk_A, example.com)$ 的绑定；某一验证者 B 拥有公钥 Pk_B 和私钥 Sk_B。

交易类型

在本方案中，包含以下两种类型的交易：

1. 绑定请求交易 Tx_{req}：交易请求者发布绑定请求，包含了自己的公钥 Pk_A 和域名 $examle.com$。
2. 检举交易 Tx_{rep}：在绑定完成之前，验证节点发现绑定者的验证信息放置不准确，可以提出检举交易，驳回身份绑定。
3. 验证交易 Tx_{vfy}：在检举交易发起后，验证者对检举交易进行的验证。

流程

绑定的具体流程如图3.2所示，包含以下几步：

1. A 发布绑定交易 $Tx_{req}(Pk_A, example.com)$ 到区块链上;
2. 根据需要绑定的信息 $(Pk_A, example.com)$,计算 $(Path, Chal) = F(Pk_A, example.com)$ 放置在自己控制的域名下,即访问 $example.com/Path$ 即可获取到 $Vcode = Sign_{Sk_A}(Chal)$ 值;
3. 验证者 B 访问域名验证 A 是否正确操作,如果不,发送检举交易 $Tx_{rep}(Pk_B, Pk_A, example.com)$ 到区块链;
4. A 保持验证时长 T 后，可停止该验证服务，完成绑定。

在该绑定过程中,首先需要绑定者将绑定信息自己的 Pk_A 和自己的域名 $example.com$ 发布到区块链上，将绑定身份信息展现给所有的验证节点，并根据函数 F 生成相关验证内容。我们选择哈希函数 $sha256$ 作为函数原型，将 Pk_A 和 $example.com$ 直接拼接后作为输入，将计算得到的 256 位哈希值前 128 位赋值给 $Path$、后 128 位赋值给 $Chal$，并且使用自己的私钥对 $Chal$ 进行签名：

$$Path_{128}||Chal_{128} = sha256(Pk_A||example.com) \tag{3.1}$$

$$Vcode = Sign_{Sk_A}(Chal) \tag{3.2}$$

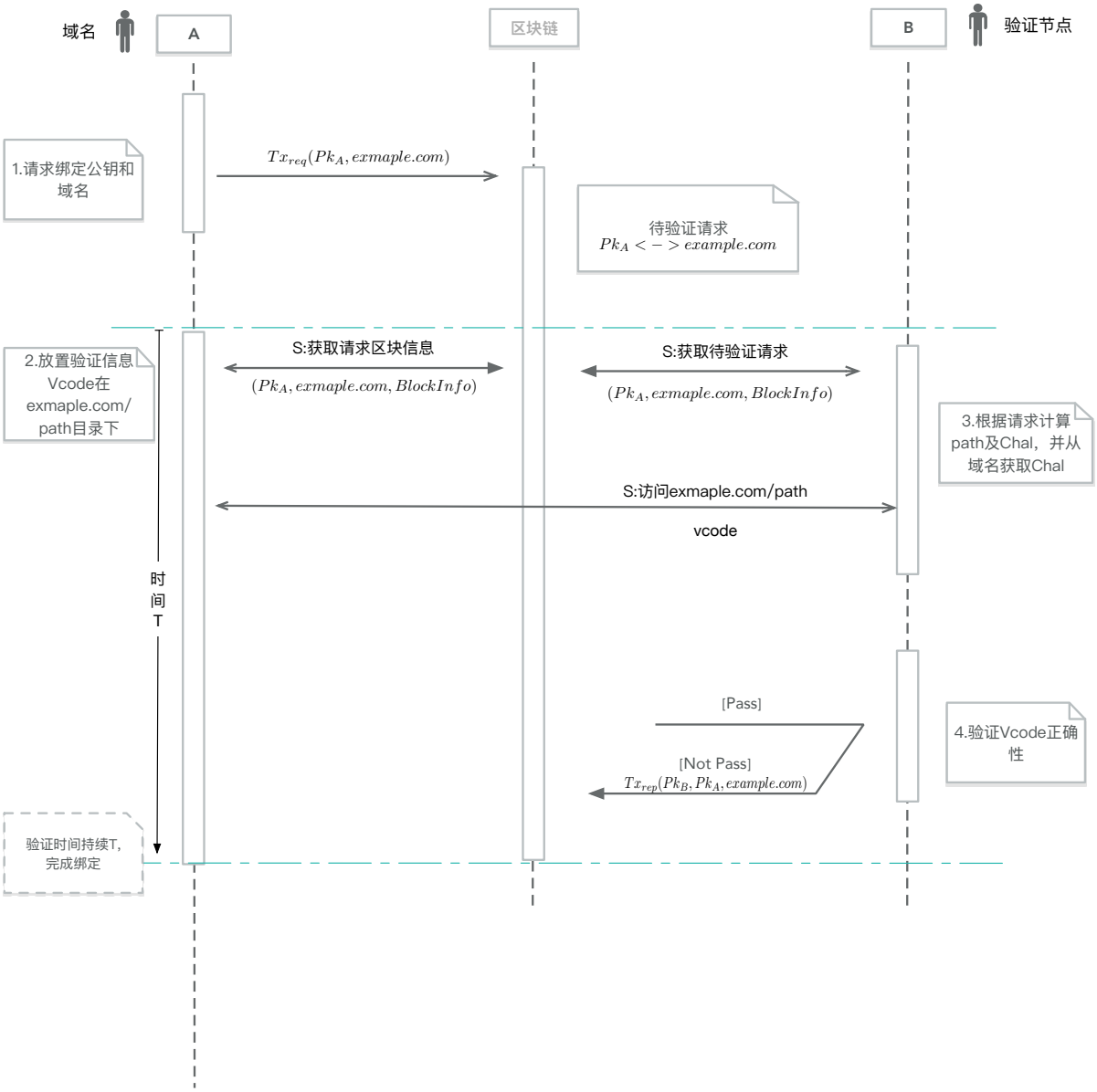

图 3.2　基于时间的方案流程图

绑定者将以上得到的 $Vcode$ 值放置在网站的 $example.com/Path$ 目录下，作为身份验证的内容，供所有验证节点验证。如果验证节点在验证过程中发现其并没有正确的放置验证信息，将可以通过发送检举信息 Tx_{rep} 对身份绑定消息进行驳回。检举的相关机制会在后面进行详细叙述。

3.2.2　基于验证次数的鉴权方案

基于验证次数的方案需要在绑定者发布绑定信息后，选择出合适的验证者，对绑定者的验证信息进行确认，当验证的次数达到规定次数后即可完成身份绑定。

假设

假设绑定者 A 拥有公钥 Pk_A，私钥 Sk_A 以及域名 $example.com$，其需要完成 $(Pk_A, example.com)$ 的绑定；某一验证者 B 拥有公钥 Pk_B 和私钥 Sk_B。

交易类型

在本方案由于需要特定验证者完成验证信息的确认，相比于基于时间的验证方案，增加了验证交易类型，包含一下三类交易类型：

1. 绑定请求交易 Tx_{req}：交易请求者发布绑定请求，包含了自己的公钥 Pk_A 和域名 $examle.com$。
2. 检举交易 Tx_{rep}：在绑定完成之前，验证节点发现绑定者的验证信息放置不准确，可以提出检举交易，驳回身份绑定。
3. 验证交易 Tx_{vfy}：验证者在完成验证信息的对比后，在验证通过的情况下发送验证通过的交易。

流程

基于验证次数的鉴权方案相比于基于验证时间的方案需要验证者更多的和区块链进行交互，在验证过程中需要将验证是否通过的信息提交到区块链上去，绑定流程如图3.3所示，详细流程如下：

1. A 发布绑定交易 $Tx_{req}(Pk_A, example.com)$ 到区块链上;
2. A 获取 Tx_{req} 所在位置区块信息 $Info_{block}$，并根据其计算值 $(Path, Chal) = F(Info_{rcvBlock})$ 放置在自己控制的域名下，及访问 $example.com/Path$ 即可获取到 $Vcode = Sign_{Sk_A}(Chal)$ 值；
3. 根据交易 Tx_{req} 计算符合该条请求的验证者：$v_1, v_2, ..., v_k$，假设 B 为其中一个验证者
4. B 从 $example.com/Path$ 获取验证内容，提交交易 $Tx_{vfy}(Pk_B, Pk_A, Vcode)$ 完成验证。
5. 在经过 K 个验证过后，即完成绑定。

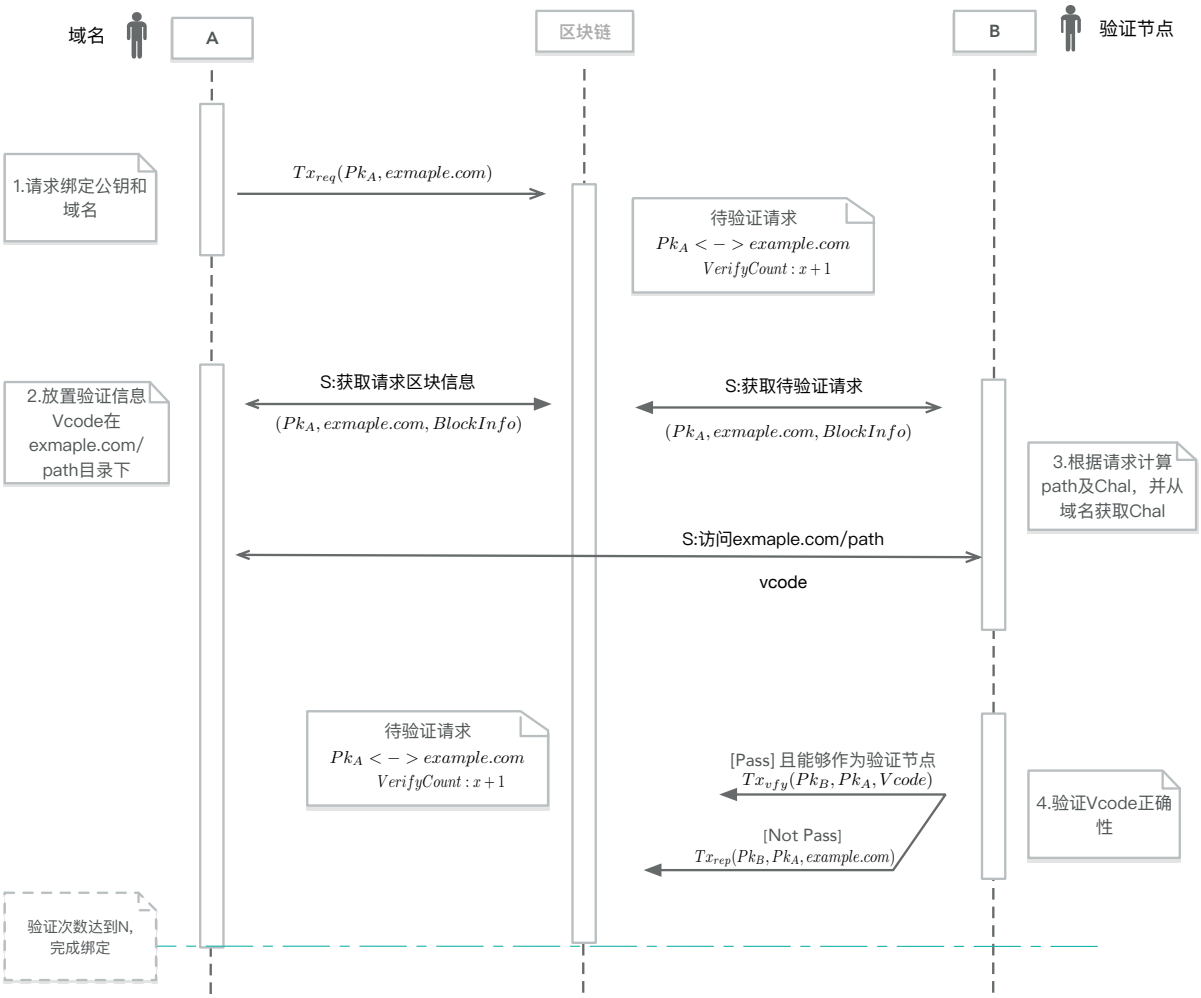

图 3.3　基于验证次数的方案流程图

整体流程和基于时间的方案类似，需要绑定者将自己的公钥 Pk_A 和自己的域名 $example.com$ 发布到区块链上，然后通过公式(3.1)和(3.2)计算得到 $Path$ 和 $Vcode$ 的值放置到自己的服务器上；验证者通过(3.1)计算获得 $Path$、$Chal$ 和 Pk_A 对验证信息进行确认，确认通过后发送验证交易 Tx_{vry} 到区块链上完成一次验证，待达到验证次数 K 后，即完成身份绑定。

3.2.3　检举的工作机制

在一个 p2p 的网络中或者一个区块链的网络中，每个用户使用都是一个公钥（即地址），和自身的身份是没有任何关系的，这就是为什么需要设计公钥与域名绑定方案的原因。但是在这样的网络中，任何人都可以发起对任意域名的绑定，为了排除一些错误的绑定或者假冒的绑定，在以上给出两种不同的身份绑定方案中，都是用到了检举交易来完成对这类情况的处理。

当一个诚实的验证节点 C 发现一个虚假的绑定时，他就可以向区块链发起检举交易，和验证交易提交的内容基本一致，只是将最后一项 $Vcode$ 替换为需要检举绑定的域名地址，所以检举交易包含的内容为自身的公钥 Pk_C 和绑定者公钥 Pk_A 和域名 $examle.com$。

但是对于一个恶意的验证节点，可能也会对一个真实有效的绑定发起检举交易。为了确认一条检举交易的有效性，需要区块链上验证节点对该条交易给出判断。此时，提交的检举交易和域名提交的绑定交易一样，都是由 P2P 网络中单一实体提出的待验证的请求，需要网络中的其它实体来进行核实。当检举交易被提交到网络中并被区块所确认，将如同绑定请求一样，随机的选着 n 个验证者对该请求进行验证，决定本次检举的有效性。

3.2.4　验证者的选取

基于时间的方案中，需要一种选取验证者的规则，避免节点为了自己的利益而不纳入或者滞后其它节点的验证交易；更为重要的是，如果验证节点不是经过一定规则筛选得出的，而是每个节点都可以对绑定身份进行验证的话，在本方案中恶意的绑定者可以使用不同的公钥作为验证节点，然后对恶意绑定进行确认，即发起女巫攻击；同时，正常的验证者为了获得更多的奖励，也会申请尽量多的账户对交易进行验证。

为了避免以上的情况出现，本方案中设计了依据绑定信息随机选择验证节点的方法，保证验证可以有效的进行。当绑定信息被发布到区块链上后，根据绑定信息 Pk_A、$example.com$ 和该交易所在位置的区块信息 $Info_{rcvBlock}$，将其转换为基准值 $hash_{cmp}$:

$$hash_{cmp} = sha_{512}(Pk_A||example.com||Info_{rcvBlock}) \tag{3.3}$$

得到基准值之后，验证节点将通过对纳入该区块之后的区块信息进行签名，来判断自己是否具有成为验证本次请求的权力，判断的方法如3.4式所示：

$$dist_{ham}(Sha_{512}(Sig_{Pr_C}(Info_{curBlock})), hash_{cmp}) <= d + \Delta_t/t_{adjust} \tag{3.4}$$

其中 $dist_{ham}(a,b)$ 表示两个数 a 和 b 之间的海明距离；Pr_C 表示验证节点的私钥；d 表示初始时的海明距离要求；Δ_t 表示从开始收到验证请求已经过去的时间；t_{adjust} 调整海明距离的时间；$Info_{curBlock}$ 表示最新块的相关信息；Sig 为一个 unique signature 算法。

以上判断方法使得一个验证者可以在每产生一个区块就可以进行一次尝试，判断自己是否能够成为验证节点，同时，随着时间的推移，成为验证节点的难度也在不断减小，保证一定会有验证者被选出。

3.2.5 验证交易的附带信息

验证交易在本方案中起着重要的作用，是验证节点参与到本系统中的重要途径之一。验证节点随时监控区块链上提交的身份绑定请求以及相对应的检举信息，当这两种消息公布到区块链上时，意味着需要验证节点参与到事件的确认当中，每个验证节点都会有机会成为验证该事件的节点。

对于身份绑定请求和检举信息的确认，同时都要提交自身的公钥 Pk_C 和绑定者的公钥 Pk_A，不同的是一个需要提供验证内容 $Vcode$，另外一个需要提供域名 $example.com$。在此基础上，还需要验证者提供一个基于容量证明的内容，保证身份不是随意产生的，而是付出一定代价维护的。假设需要提供的容量证明内容为 prf，其需要满足以下条件：

$$Verify(Pk_C, prf) = Sha_{512}(Pk_A||Vcode/example.com||Info_{curBlock}) \tag{3.5}$$

其中 $Verify$ 是容量证明的验证函数。

3.2.6 激励机制

为了促使本方案有足够多的验证节点加入，保证身份绑定能够完整有效的进行，需要对验证节点给予一定的奖励。在验证过程中，如果发现存在未合理放置验证信息的域名，将可以发起检举交易到区块链上，待交易确认后，其可以得到相应的奖励; 为保

证本系统中的奖励平衡，发起绑定者需要付出一定的代价，而验证者发现错误时，将可以获取得到相应的奖励。

由于这些实体都存在与区块链的网络当中，所有的身份都是公钥（或者公钥转换得到的地址）来表示的；根据前面方案描述的场景，假设域名拥有 Pk_A，验证者拥有 Pk_B，以及检举者拥有 Pk_C，在区块链上上各自都拥有相应的账户余额。

在这些实体在与区块链网络交互的过程中，涉及到对区块链记录进行修改的操作将都会有伴随着余额的变化，相关操作的惩罚和奖励机制如下表3.1所示：

交易名称	发起对象	公钥	交易含义	交易费用	后续奖励	请求确认后余额
Tx_{req}	域名	Pk_A	完成身份绑定	$-P_1$	0	$Blance(Pk_A)-P_1$
Tx_{rep}	检举者	Pk_C	对身份绑定进行检举	$-P_2$	$R_1=\frac{P_1}{2}$	$Blance(Pk_C)+\frac{P_1}{2}$
Tx_{vfy1}	验证者	Pk_B	对身份绑定进行验证	$-P_3$	$R_2=P_3+\frac{P_1}{K}$	$Blance(Pk_B)+\frac{P_1}{K}$
Tx_{vfy2}	验证者	Pk_B	对检举进行确认	$-P_3$	$R_3=P_3+\frac{P_1}{2n}$	$Blance(Pk_B)+\frac{P_1}{2n}$

表 3.1　激励机制

第一条规则的设计让身份绑定者需要余额，而不是无限制的随意的在本方案中发起身份绑定操作，从而避免了恶意的破坏者无限度的去发起无效的身份绑定，从而影响本系统中正常身份绑定者的顺利进行；第二条规则是为了防止网络中的验证节点随意的发起检举交易来扰乱正常绑定的完成；第三条规则是对付出验证操作的节点给予奖励，作为吸引更多节点加入的激励机制，但是其在验证之前需要付出一定的余额，防止一些节点对验证过程进行作弊操作；第四条规则和第三条规则类似，属于激励规则，促使更多的节点加入到本系统中。

结合交易代价和后续奖励，可以知道该系统中的所有奖励来源主要来源于身份绑定者域名，而且整个系统在工作的过程中不会凭空产生价值，总的收支是平衡的。

3.3 分析与讨论

3.3.1 安全分析

安全模型

我们假设区块链网络中存在 M 个全节点，其中有 p 占比的全节点被恶意攻击者控制，另外 $1-p$ 的节点为诚实的全节点；验证节点的数量为 N，其中有 q 占比验证节点被恶意攻击者所控制，另外 $1-q$ 占比数量的验证节点为诚实的节点。

51% 攻击

显然，当 $p > 1/2$ 时，无论底层区块链使用的是 PoS 还是 PoW 共识机制，攻击者都可以完成对区块链网络的控制。换而言之，其可以完成对区块中纳入交易的控制，对任意域名发起身份绑定之后，将检举交易不纳入到区块当中，进而通过自己手中持有的验证节点对该绑定进行验证，最终完成对虚假身份的绑定。

恶意绑定分析

当一个验证请求被发起后，需要从网络中选取的验证节点进行验证。根据前面验证者的选取过程可以知道，每次进行验证者选取的过程其实一个随机的过程，这里假设我们设计的随机算法是完全随机的，那么选择出 K 节点全部都为恶意节点的概率为：

$$Pr_1 = \prod_{i=0}^{K-1} \frac{qN-i}{N-i} \tag{3.6}$$

也就是说，将会有 $1-Pr_1$ 的概率会产生检举交易。根据方案设计，当检举交易产生时，会根据检举交易的内容进行验证者的选取，将会对本次检举进行半数确认，确认通过的概率为：

$$Pr_2 = \prod_{i=0}^{n/2} \frac{qN-i}{N-i} \tag{3.7}$$

所以当攻击者拥有全网 q 占比的验证节点时，完成一次恶意绑定的概率如式3.8所示。

$$Pr_{attack} = Pr_1 + (1 - Pr_1) * Pr_2 \tag{3.8}$$

$$= Pr_1 + Pr_2 - Pr_1 * Pr_2 \tag{3.9}$$

$$= \prod_{i=0}^{K-1} \frac{qN - i}{N - i} + \prod_{i=0}^{n/2} \frac{qN - i}{N - i} - \prod_{i=0}^{K-1} \frac{qN - i}{N - i} * \prod_{i=0}^{n/2} \frac{qN - i}{N - i} \tag{3.10}$$

拒绝绑定分析

当一个验证请求被发起后，域名正确的放置了验证信息，等待验证者完成验证即可完成对身份绑定。但是，恶意的攻击者可能会对任何验证请求提交检举请求，扰乱绑定的正常进行。在发起恶意的检举之后，能够成功的扰乱本次请求的概率如式3.11所示。

$$Pr_{denial} = \prod_{i=0}^{n/2} \frac{qN - i}{N - i} \tag{3.11}$$

攻破域名

当一个恶意的攻击者将域名攻破时，其可以完成对域名的完全控制，那么在本方案中，就可以完成对信任 CA 的完全控制。为了 ji 降低本状况带来的影响，本文建议域名在不同的服务器上放置信任列表变动监测程序，当信任列表发生变化时及时的给出提醒，供域名拥有者进行确认。

3.3.2　其它讨论

站点对域名服务器的控制权

为了保证绑定者对域名服务器拥有控制权，在以上方案中，使用提交的 Tx_{req} 来生成验证内容 $Path$ 和 $Chal$，保证了验证内容和提交请求的相关性；同时在(3.1)中使用 $sha256$ 作为随机函数，将提交内容转换为验证内容，该过程是一个不可逆的过程，从验证内容到提交内容的生成是很难完成的，保证了很难通过验证内容去构造提交内容。在放置的验证内容中，$Vcode$ 是使用绑定公钥对应私钥对 $Chal$ 进行签名得到的，保证了绑定者对私钥的拥有权。

验证交易的有效性

基于时间的验证方案中不需要提交者发起验证提交，在发现错误的时候需要发送检举交易；对于基于验证次数的方案而言，需要提交验证内容即 $Vcode$，因为有签名的

存在，验证节点不能凭空的发送验证交易，需要通过服务器提供的验证信息才能正确的完成验证。

绑定者如果未能正确的放置验证内容，或者通过其它方式公布验证内容，导致验证过程依旧执行，那么其他验证节点可以通过检举交易对其发起检举。

女巫攻击

本方案建立在一个 P2P 的开放式网络中，任何人都可以生成公钥加入到本系统中，成为一个合法验证节点。在第二章中我们知道，一个公开且没有身份管控的开放网络很容易受到女巫攻击；更具体一点，恶意的攻击者可以生成任意多的公私钥对，从而拥有本系统的多个身份，然后在验证过程中增大自己被选为验证节点的可能性；或者恶意攻击者通过在发起绑定前，不断尝试生成公私钥对并挑选出可以成为验证节点的公私钥，然后在发起绑定交易后瞬间完成对绑定请求的确认。

对于上述的第一种情况，在本方案中的任何节点都可以生成任意多的身份，以增大自己的作为验证节点的几率。根据验证者选取公式3.3可知，不仅依赖于交易的绑定信息，而且还和交易被确认在区块链上的位置密切相关，所以一个公私钥是否能成为验证者并不能事先决定。对于第二种攻击方式是一样的道理，攻击者无法事先确认成为验证者的要求。

同时，如果有人希望通过拥有多个公私钥对来发起女巫攻击，在发起验证交易还需要提供一个附加信息，其中包含了一个可验证的工作量证明。这意味着如果一个节点想拥有多个身份，就需要投入更多的空间来提供工作量证明所需的内容，很大程度上限制了一个节点所拥能拥有的身份数量。

防定点攻击

在本方案中，一个公私钥拥有者是否可以成为一个请求的验证者，是通过式3.3和式3.4来决定。当一个请求发起之后，大家都可以得到共同的基准值 $hash_{cmp}$，其无法决定一个公钥拥有者 Pk 与该基准值之间的关系，只有拥有 Pk 对应私钥 Pr 的节点才可以去计算式3.4，判断自己是否能够为这一次请求进行确认操作。这意味着对于每次请求发出之后，网络中没有人能够去判断谁能为此次请求进行确认；否者，攻击者可以通过对这些确认的验证节点发起比如 DDoS 攻击或者劫持服务来阻止它们完成验证，进而增大自己成为验证节点的几率。

第四章　域名鉴权系统设计与实现

本章将给出本文提出方案的一种实现方法，首先对方案实现的系统设计进行阐述，叙述实现本方案的方法以及本文选择的实现方式；其后给出系统的模块设计，实现并完成对系统的功能性测试及分析。

4.1　系统设计

本小节将对方案实现的整体系统进行描述，涉及到系统中的各个角色要求和功能；然后给出系统实现的架构，从上层透视系统中的各个模块的功能和信息交互；最后讨论实现本系统的方式以及本文所采用的方法和原因。

4.1.1　系统角色

通过上一小节中对方案的概述可以知道，在实现过程中涉及的实体主要包括以下四类：作为证书申请者的用户 (即域名）、依赖方、参与到系统中的验证节点、以及区块链的全节点。

用户

域名作为本方案的用户，其希望通过本系统完成对自己可信 CA 的控制，达到防止未经允许的恶意 CA 对其签发证书的目的。

在本系统中，所有的数据都是通过区块链来存储的，域名和所有参与者一样在区块链上的身份都是不具有代表性。为了建立域名在区块链上的身份 (公钥或者地址) 和自身的联系，需要通过上一章中设计的身份鉴权方案进行身份的绑定；在其完公钥和自身身份的绑定后，就可以通过区块链上的身份来完成对域名信任 CA 的控制。所以，在域名在本系统需要完成两个操作，其一是在区块链上完成身份绑定操作；其二是完成可信 CA 列表的管理操作。

依赖方

依赖方通过浏览器获取证书并建立安全通信时，需要对证书的有效性进行检查。在本方案中，需要通过区块链进一步确认证书是否被合理的签发，提供证书的进一步检查，达到不被恶意签发证书所欺骗的作用。

在本系统中，依赖方所需要完成的操作相对比较简单，只需要从区块链网络中获取通信实体的可信 CA 列表，然后与收到证书的签发方进行对比，验证其是否存在与可信 CA 列表之中，如果不存在则需要向依赖方发出警告提示。

验证节点

验证节点加入到本系统中的目的是为了获取验证过程中产生的奖励。

验证节点在本系统起着至关重要的作用，是否有住够多的验证节点加入到本系统中，决定着本系统是否能够有效的运行下去。在本系统中，验证节点需要对域名发起的身份验证请求进行验证，帮助它们完成区块链上身份与域名真实身份的绑定操作。根据方案设计可知，验证节点为了自身利益，需要实时的监听区块链上发起的验证交易，判断自己是否能够成为验证节点并获得相应的奖励，所以验证节点在本方案中需要完成操作包括监听区块链上的交易和对交易的验证。

全节点

全节点加入到本系统的目的是为了完成对区块链的维护，从而获得相应的奖励。

全节点作为区块链系统中的完整节点，实时的监控整个网络中的所有交易，并尝试将这些交易打包到最新的区块中。每当完成一个区块被创建，创建者将会获得挖矿所对应的奖励。对于本系统而言，它们是底层区块链平台的维护者，为上面的所有实体提供了基础服务。

在系统中的单个实体可以扮演以上的单个或者多个角色，比如域名作为本系统中的用户，可以成为本系统中的验证节点或者全节点，为本系统正常运行做出贡献的同时，也可以从中获得相应的奖励。普通的依赖者，通过浏览器进行网页游览并发起对域名证书的二次验证，确保自身的权益，同时也可以成为本系统中的验证节点和全节点。

4.1.2 系统架构

本系统中涉及的角色包括域名、依赖方、验证节点以及全节点，通过区块链网络连接到一起，共同参与到本系统之中，系统的整体架构如图4.1所示。

在本系统中，区块链是整个系统的重心，所有参与到系统中的角色都要通过其进行关联。不同的角色之间需要使用不同的工具来完成与区块链网络的交互，依赖方在传统 PKI 系统中使用的是浏览器来完成对证书的检验；同样，在本系统也依赖与浏览器中的插件来完成对证书的二次验证。对于域名而言，需要有相应的客户端来与区块链进行交互，完成验证请求的发布和相关验证信息的部署；验证节点同样需要有相应的客户端来获取区块链上所提交的验证请求，并通过客户端来判断自己是否能够完成验证请求，对验证请求进行响应；作为维护区块链的全节点，则直接通过区块链的客户端进行区块链维护工作即可。

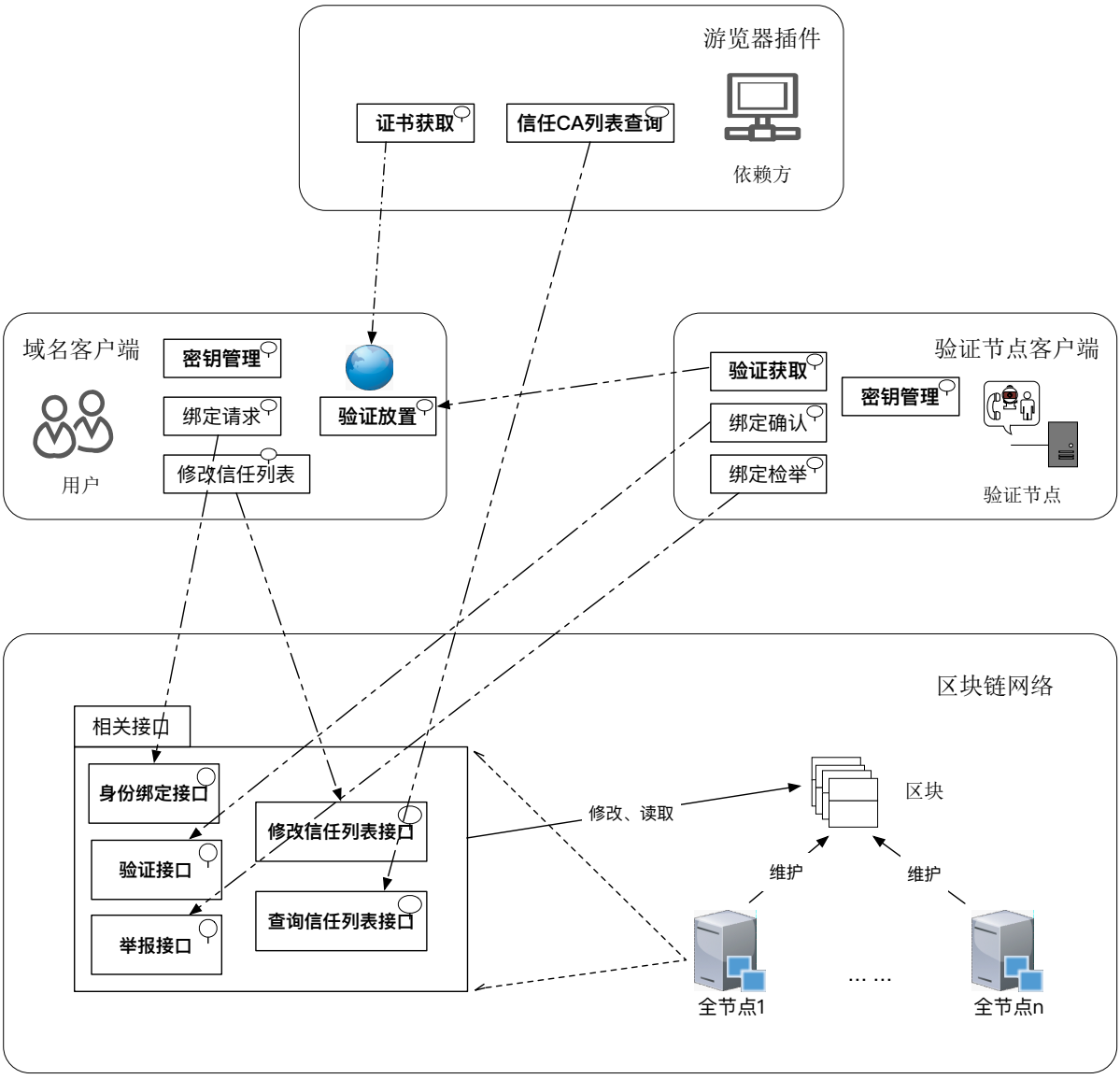
游览器插件
证书获取
信任CA列表查询
依赖方
域名客户端
密钥管理
绑定请求
验证放置
修改信任列表
用户
验证节点客户端
验证获取
密钥管理
绑定确认
绑定检举
验证节点
区块链网络
相关接口
身份绑定接口
修改信任列表接口
验证接口
查询信任列表接口
举报接口
修改、读取
区块
维护
维护
全节点1
……
全节点n

图 4.1　系统架构

4.1.3 实现方式

本文所提出的方案是依托于区块链及其上面的用户实体的，所以必须需要一个区块链平台才能完成上述系统的构建，实现本文给出方案。实现以上设计的系统有两种方式：一种是利用已有的区块链平台，通过其上的智能合约来实现本文给出方案的相关逻辑和操作；另外一种方式是重新构建一个全新的区块链平台，依据本文中给出方案去设计区块链平台的相关功能和交易，完成上述实体之间的信息交互。

以上提到的两种方式都可以完成本系统的构建，且只是影响该系统中的底层区块链的构建，对上层如域名、依赖方以及验证节点并没有实质性的影响，仅仅是通过不同的方式去访问不同的区块链网络。但是两种不同的实现方式各有利弊额，如表4.1所示。

	难度	工程量	网络建立速度	性能	独立性	全节点维护代价
基于智能合约	低	小	快	低	低	高
创建新链	高	大	慢	高	高	小

表 4.1 实现方式对比

基于智能合约的方式相比于创建新链而言，实现起来更加容易，不需要对区块链的底层如交易结构、协议等进行设计和修改，仅在智能合约层面完成各个实体的交互逻辑即可，因此实现难度更小；同时，由于不需要从底层对一个区块链平台进行设计或者修改，从智能合约层面出发来完成也意味着工程量更低。从区块链网络构建速度来考虑，重新构建一个链需要时间去推广并纳入足够多的用户，保证其平台的稳定性；而基于智能合约则可以选择用户较多已有平台 (如以太坊)，构建起整个区块链底层网络更加迅速。

基于智能合约方式的缺点主要表现在性能、系统的独立性和全节点维护代价方面。现有支持智能合约的公链性能都很低，比如以太坊每秒只能处理大概 30 笔交易；而创建一条新链，可以根据本系统的需求去设计并改进区块链底层，提升自己的性能。另外一个问题就是整个系统的独立性并不好，严重依赖于底层的区块链平台；如果该平台出现任何问题，那么将直接影响到本系统的正常工作。最后是全节点付出的代价问题，基于智能合约需要维护额外的区块内容，而新建的链的方式将只需要维护本系统中的相关交易，极大的减少全节点付出的代价。

本文将选择基于智能合约的方式实现，选择的区块链平台为以太坊，身份验证方案选择基于验证次数的方案。

4.2　模块设计与实现

本方案以区块链作为底层架构，需要在其上需要上述方案中的相关逻辑，让 PKI 中的相关实体与区块链进行交互，在其上完成身份认证并进行相关数据的存储，实现对域名可信 CA 列表的访问与控制，并最终完成证书有效性的进一步确认。

根据上一小节的论述，本文中的实现将基于以太坊来完成，在智能合约中编码方案中的相关逻辑。在本小节中，我们将给出系统实现的相关模块，并对模块的功能和实现细节进行描述。我们知道，在本系统中进行信息交互的实体包括域名、验证节点、依赖方以及区块链上的全节点；依据在前面对这些实体的相关功能描述，本系统如4.2图包含四个模块进行实现：智能合约、域名客户端、验证节点客户端、浏览器验证插件。

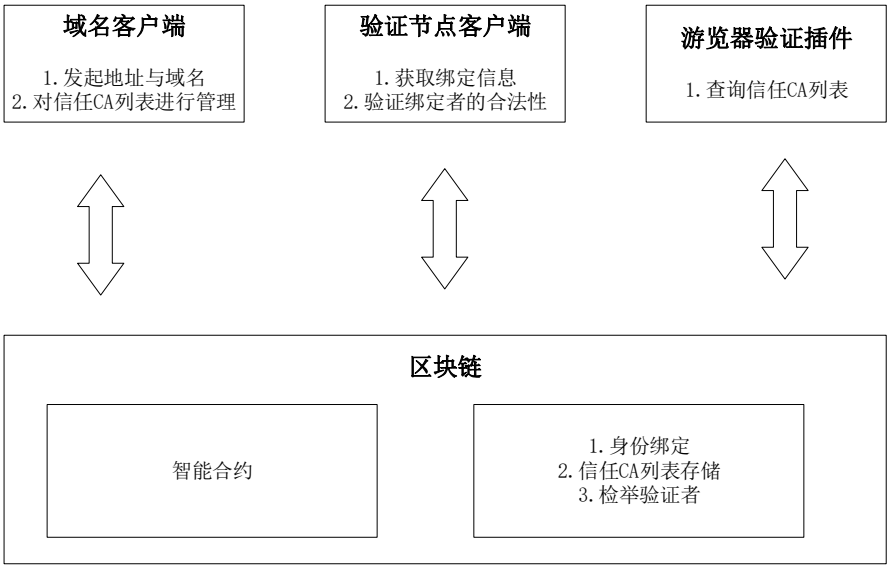

图 4.2　系统模块设计

4.2.1　智能合约

模块功能

本系统中的所有实体都需要通过区块链进行连接，而在本文的实现方式中，所有的数据都要通过智能合约来完成交互。在智能合约中，不仅要完成对域名信任 CA 的相关信息存储，还需要完成域名认证的操作，同时，还要向其他的实体提供查询操作。智能合约整个模块需要提供给的功能如下：

- 域名认证

  域名认证功能的服务对象是域名，其作用是让域名可以在区块链网络中完成身份的绑定。该功能中的参与者不仅仅包括域名本身，还包括本系统中的验证节点。

当域名调用该功能的时候，将通过在上一章节中设计的验证节点选取方法，选取一系列验证节点来完成验证；在验证信息存在错误的时候，本系统的中其它节点还可以对认证过程发起检举，并通过相关的机制来完成对检举确认。

- 信任列表存储
  信任列表存储是该模块中的另外一个功能，其面向的对象是已经通过身份验证的域名节点，通过本功能可以更改智能合约中存储的数据，完成对自身可信 CA 的控制。
- 信任列表查询
  信任列表查询功能面向的对象是 PKI 系统中的依赖方，通过该功能完成对特定域名可信 CA 列表的查询。

模块接口

根据本模块的功能，设计相关的接口如表4.2所示。

接口名称	面向对象	参数	返回值	接口功能
register	域名	绑定地址、域名名称	无	发起身份绑定请求
report	验证节点	域名绑定地址、域名名称	无	发起绑定检举
verify	验证节点	待绑定地址、域名名称、验证内容	无	发起验证确认
modifyTrustedCAs	域名	域名名称、可信 CA 列表	无	修改域名自身的可信 CA 列表
queryTrustedCAs	任意实体	域名名称	可信 CA 列表	查询给定域名的可信 CA 列表

表 4.2　智能合约接口设计

模块实现

本模块的实现依托于以太坊上的智能合约，使用 Solidity 语言编程完成上述接口的功能。在智能合约中，用户的身份是以地址表示的，需要完成与域名实体的认证与绑定，并在其上进行信任列表的存储；所以在智能合约中设计了存储域名的数据结构，

将存储域名自身相关的信息和认证相关的基本信息，并使用 Solidity 中的 *mapping* 数据类型完成以太坊地址与域名之间的对应；同时，在方案中还涉及到一些例如验证次数、验证时长等系统参数，也需要在智能合约中以 *constant* 类型进行申明；以上提及的数据结构和重要变量如下4.1所示。

```
struct Domain {
    string name; //域名名称
    uint count;  //验证次数
    string trustCAs;  //信任CA
    uint stBlock;  //验证开始的区块
    address[] validator;  //验证者列表
    address addr;  //域名地址
    bool isEntity; // flag
}
mapping(address => Domain) reg_domain; //待认证域名
mapping(string => Domain)  auth_domain; //已认证域名
//some constant
uint constant auth_times = 10; //目标验证次数
uint constant limit_blocks = 100; //验证期限
```

Listing 4.1　域名数据结构和重要变量

register 接口需要接收用户传入的绑定地址和域名，其后在智能合约中将其实例化为 *Domain* 变量，并在 *reg_domain* 中完成地址和域名实例的关联。根据上一章的方案设计，绑定请求被接收的区块信息需要被用于验证信息的生成；而以太坊智能合约中可以通过 *block* 的变量来获取交易执行区块的相关信息，这里将 *block.number* 即区块号记录赋值给域名变量的验证开始区块；代码实现如4.2所示。

```
function register(address addr, string domainName) {
    reg_domain[addr].name = domainName; //域名名称
    reg_domain[addr].count = 0; //初始化认证次数为0
    reg_domain[addr].trustCAs = ""; //初始化信任CA为空
    reg_domain[addr].stBlock = block.number; //设置区块为当前区块号
    reg_domain[addr].addr = addr; //设置绑定的地址
    reg_domain[addr].isEntity = true; //flag

    ... //some other operations
}
```

Listing 4.2　register 接口部分实现

按照上述实例，依据各个接口设计的要求和功能，在智能合约中完成对相关变量的存储和修改，实现本模块的所有功能。

4.2.2 域名客户端

模块功能

域名作为证书的申请者，需要将自己的可信任 CA 列表存储到区块链上，以供证书受用者在使用证书的过程中对证书的真伪进行进一步检查。

域名客户端需要与区块链进行交互，实现域名和公钥的绑定以及对自己可信 CA 列表的控制，该模块包括的功能如下：

- 密钥管理

  区块链上身份是通过公私钥来管理的，所以本模块需要提供密钥管理的功能，完成对相关交易的签名；同时，在提供验证信息的过程中也需要使用到相关私钥的签名功能。

- 发起认证请求

  客户端的另外一个重要功能就是发起认证请求，向区块链网络公布自己拥有的域名和响应的密钥。在区块链对请求进行纳入后，根据交易生成相关的验证内容。

- 修改信任列表

  信任列表的修改功能是在域名完成认证后才可使用的功能，向区块链网络提交自己的信任 CA 列表，供网络中搞定其它节点查询。

模块逻辑

域名客户端模块需要完成以上功能，为了能够对自身信任列表的控制，其模块运行逻辑如图4.3所示。

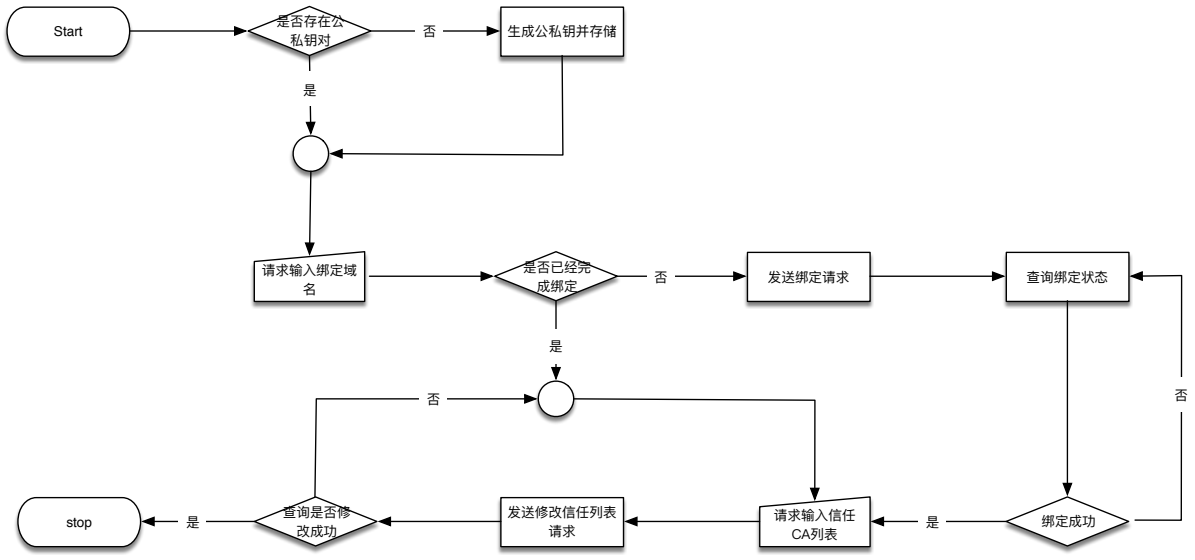

图 4.3 域名客户端模块流程

模块实现

在本模块实现过程中，最主要的任务是要完成与区块链智能合约的交互。在上一模块中已经将智能合约中的接口实现，这里将给出如何使用 Python 脚本完成对智能合约的调用，达到客户端与区块链的信息交换。

向智能合约发起的请求是通过 rpc 完成的，在实现过程中，需要使用到 Python 中的 Web3 库。根据书写好的智能合约，通过 Remix 工具生成相关的 abi 接口文件，作为调用智能合约的配置文件，申明了智能合约接口的相关参数，*register* 接口的 abi 内容如4.3所示。

```
var abi =[
...
{
  "constant": false,
  "inputs": [
    {
      "name": "addr",
      "type": "address"
    },
    {
      "name": "domainName",
      "type": "string"
    }
  ],
  "name": "register",
  "outputs": [],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function"
},
...
]
```

Listing 4.3　register 接口的 abi

其后，将 abi 内容实例化到 Web3 的相关变量中，并通过其提供的 *eth* 接口，就可以完成对智能合约 *register* 接口的调用，并在 Python 对该接口进一步封装，方面后面使用过程中的调用，实现代码如4.4所示。

```
def register(address, name):
    nonce = w3.eth.getTransactionCount(wallet_address)
    txn_dict = contract.functions.register(address, name).buildTransaction({
        'chainId': 5777,
```

```
        'gas': 300000,
        'gasPrice': w3.toWei('40', 'gwei'),
        'nonce': nonce,
    })

    signed_txn = w3.eth.account.signTransaction(txn_dict, private_key=
        wallet_private_key)
    result = w3.eth.sendRawTransaction(signed_txn.rawTransaction)
    tx_receipt = w3.eth.getTransactionReceipt(result)

    count = 0
    while tx_receipt is None and (count < 30):
        time.sleep(10)
        tx_receipt = w3.eth.getTransactionReceipt(result)
        print(tx_receipt)

    if tx_receipt is None:
        return {'status': 'failed', 'error': 'timeout'}

    return {'status': 'added', "txn_receipt": tx_receipt}
```

Listing 4.4 register 接口封装

对所有接口封装之后，按照上述程序逻辑，完成 python 脚本书写，并在与区块链交互的时候调用相关接口，完成对本模块的实现。

4.2.3 验证节点客户端

模块功能

验证节点是本系统中身份鉴权的重要组成部分，是身份绑定有效性的保证，只有住够多验证节点存在的情况下，才可以保证身份确认的有效。

验证节点的主要工作是实时的查询验证请求并对验证请求进行验证，并根据其验证请求计算自己是否符合验证节点的要求，如果符合的话将进行验证确认操作，如果不是，也可以进行验证，对不符的验证提交检举交易。所以本模块包含的功能如下：

- 密钥管理

  通域名客户端一样，验证节点在区块链上身份是由公私钥来完成控制的，所以同样需要密码管理功能，完成对自身身份的控制。
- 请求验证 验证节点客户端将对区块链上发起的请求进行监控，包括身份绑定请求和检举信息，对所有的请求都会验证，并将验证的结果呈现在客户端中。
- 请求处理

本客户端会对所有的验证请求进行验证，当验证请求通过的且自己能够成为该请求的验证者时，将会发起交易确认，从而获得相应的奖励。当请求存在问题是，如果是绑定请求，那么发送检举交易到区块链上；如果是检举确认，将根据确认情况进行确认。

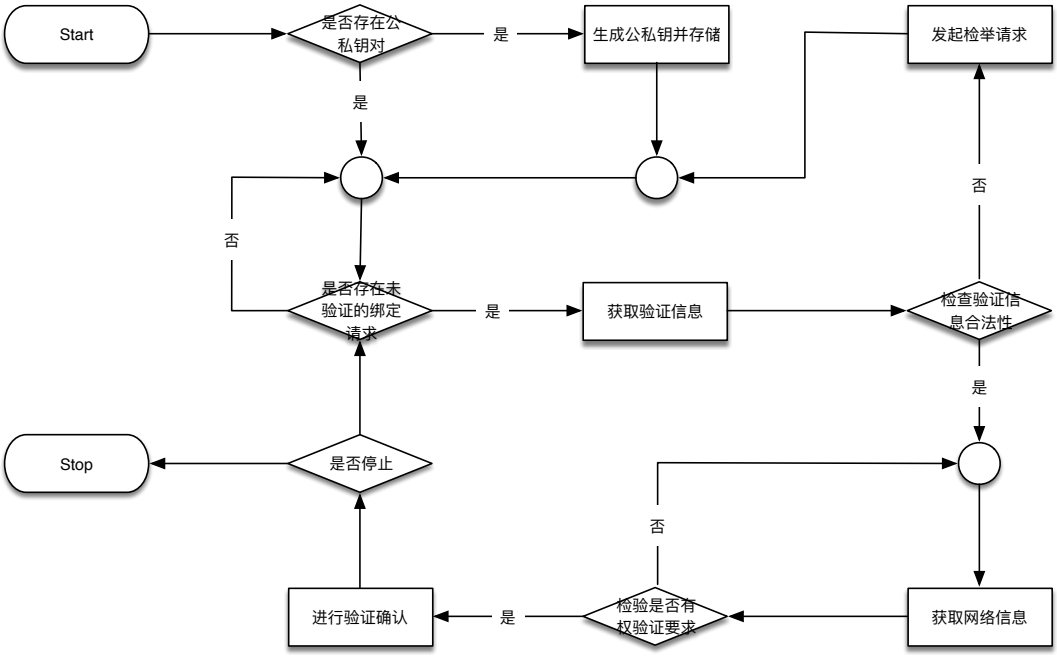

图 4.4　验证客户端模块流程

模块逻辑

验证节点客户端需要实时的获取区块链网络中的绑定请求，并查询相关请求的验证内容，对请求做出回应，本模块的运行逻辑如下如4.4所示。

模块实现

本模块的实现的技术难点和域名模块一样，需要完成与智能合约的交互；同样，使用 python 脚本来完成本模块的功能，使用前一模块的相同的方式与区块链进行交互。根据模块逻辑，按照上一模块的方法即可实现本模块。

4.2.4　浏览器验证插件

模块功能

浏览器是 Web PKI 中证书受用者的客户端，在原有体系结构中充当着检查证书真伪的角色。在本方案中，为了保证被恶意 CA 私自签发的证书可以被识别出来，需要在

客户端进行对证书进行额外的检查，也就是需要浏览器需要有区块链进行交互，查询收到的证书是否由证书申请者信任的 CA 所签发。

本模块相比于前面的模块较为简单，只需要通过调用智能合约中的信任列表查询模块，获取域名可信 CA 列表与证书进行对比，在发现错误的时候向用户发出警告。

模块流程

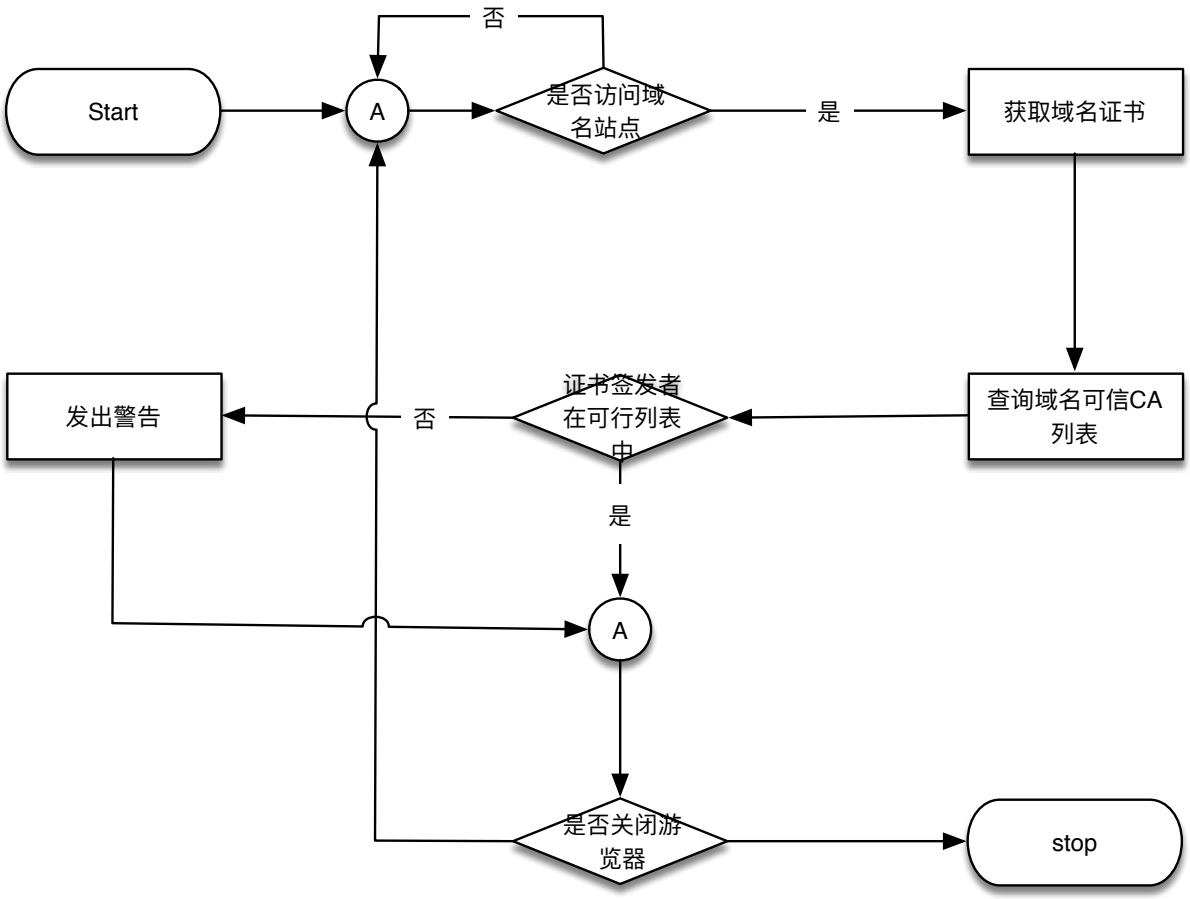

图 4.5　浏览器验证插件模块流程

浏览器验证插件需要在游览器访问网址时获得域名的证书，同时通过智能合约的相关接口获取域名设置的信任 CA 列表，并与证书签发者对比，具体的流程如图4.5所示。

模块实现

由于 Chrome 游览器本身对插件的支持很完善，同时本身也是日常生活中常用的游览器之一，将选着它作为插件实现的目标平台。

Chrome 插件作为一个由 HTML、CSS、JS、图片等资源组成的功能软件，其上可以完成如书签控制、下载控制、窗口控制以及网络请求控制、各类事件监听等等操作。在本插件中主要使用到的操作将包括事件监听和窗口控制，完成对访问站点的监听，并获得该域名的证书；同时，获取该域名在区块链的信任 CA 列表。

对于证书获取，通过调研发现已经存在获取正在访问网页证书的插件：certificate-info[56]。该插件通过监听 Chrome 游览器访问网页的事件，获得网页中正在访问的网址，再通过 JS 进行证书获取，并将证书的相关验证信息动态显示在该插件的 html 页面当中。

在该插件之上，使用以太坊提供的的 web3 JS 库完成对智能合约中信任列表查询接口的调用。在通过 JS 调用智能合约之前，和在 python 中类似，需要将智能合约的 abi 形式初始化到 JS 的智能合约对象当中，为智能合约的调用提供相关接口和参数依据。同时，在调动智能合约之前，也需要设置访问以太坊网络节点、智能合约地址等信息，该函数的实现如4.5所示。

```
function fetchTrustedCAList(nodeIp, nodePort, hostname){
    var account = '0x67EC76aD62A83A32ecE695D5F7E9580c2E625b80';
    var contractAddress = '0x7dd6f5b21cba0dfc0ea1de7169cedd99993e4387';

    var web3 = new Web3();
    web3.setProvider(new Web3.providers.HttpProvider("http://" + nodeIp + ":" +"
        nodePort"));

    web3.eth.defaultAccount = account;

    var Mycontract = web3.eth.contract(abiArray);
    var contract = Mycontract.at(contractAddress);
    console.log("calling the smart contract...");

    var ret = contract.queryTrustedCAs(hostname);

    return ret;
}
```

Listing 4.5　获取信任 CA 函数实现

通过调用以上函数就可以完成信任 CA 列表的获取，之后对证书中的签发者进行提取，并对比是否存在信任 CA 列表之中，并通过 JS 将最终结果显示在插件的页面之上。

4.3 系统测试

4.3.1 环境搭建

实验所需的区块链网络搭建在阿里云服务中，如图4.6所示。区块链网络中包含 5 台阿里云服务器，作为全节点完成对区块链网络的维护；另外 5 台服务器作为验证节点，每台服务器上启动 10 个验证客户端，共计拥有 50 个验证节点；1 台阿里云部署 web 服务，作为域名服务器并运行域名客户端；本地 PC 作为依赖方，模拟完成对域名的访问，在过程中完成对信任 CA 列表的获取。

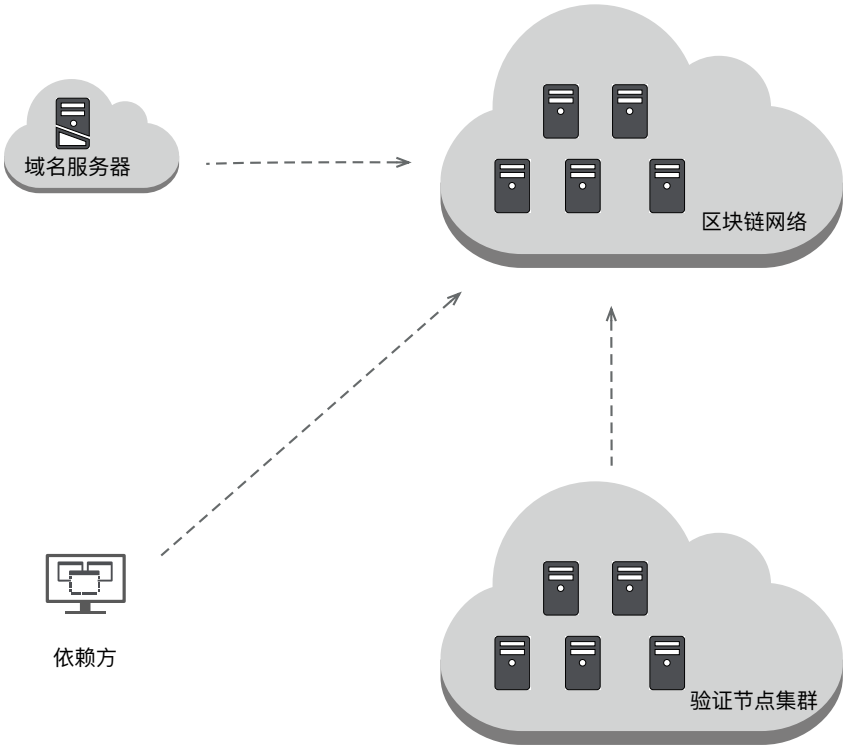

图 4.6 平台搭建

阿里云服务器配置如下：

- CPU: 1 核
- 操作系统: Ubuntu 16.04(64 bit)
- 内存: 2GB
- 磁盘: 40GB
- Golang: 1.8.3
- Ethereum: Titanium(v1.8.7)

本机配置如下：

- CPU: 8 核

- 操作系统: macOs High Sierra(version 10.13.3)
- Chrome: Version 66.0.3359.139(64-bit)

4.3.2　相关参数

本系统实现的认证方案是基于验证次数的方案，在该方案中将涉及到一些参数的设置，如表4.3所示。

参数符号	n	d	t_{adjust}	$Info_{rcvBlock}$	$Info_{curBlcok}$
数值	10	1	1(/块)	Block Hash	Block num
参数含义	身份绑定需要 10 次确认	初始海明距离为 1	每隔 1 块调整一次挑选验证节点准则	接收交易的区块信息为所在区块的哈希值	当前区块的信息为该区块的编号

表 4.3　参数设置及含义

4.3.3　功能测试

本小节对系统实现进行简要测试，依据系统的部署和正常调用流程，检验系统实现的正确性。在本测试中，通过对测试的验证节点 DNS 的修改，将 www.baidu.com 和 cn.bing.com 指向测试的域名服务器，已完成绑定操作；并且，将对 www.baidu.com 设置正确的证书签发者，对 cn.bing.com 设置错误的证书签发者。

以下测试过程中，将给出 www.baidu.com 进行身份认证在域名客户端和验证客户端的相关操作，并在最后给出游览器插件对不同情况的显示结果。

1. 智能合约部署

在智能合约开发和部署过程中，使用 truffle 工具来进行；在完成开发工作后，通过设置 truffle 工程目录下的 truffle.js 文件设置连接区块链网络节点 ip 和端口，然后使用 compile 和 migrate 命令来完成对智能合约的部署。在连接节点上可以看到以下日志：

```
[9:45:59 AM] eth_sendTransaction
[9:46:00 AM]   Transaction: 0x6f...cf
[9:46:00 AM]   Contract created: 0x4e60d62f8dee99d577fa8248843f0c49e4244662
[9:46:00 AM]   Gas usage: 1135286
[9:46:00 AM]   Block Number: 1
[9:46:00 AM]   Block Time: Sun May 06 2018 09:55:59 GMT+0800 (CST)
```

2. 认证接口调用

在域名客户端配置智能合约地址$0x4e60d62f8dee99d577fa8248843f0c49e4244662$，以及需要连接的区块链节点 ip 和端口；运行4.6命令，发起验证请求服务。

```
domain-cli reg 0x67ec76ad62a83a32ece695d5f7e9580c2e625b80 www.baidu.com
```

Listing 4.6 认证请求命令

3. 验证请求调用

在域名客户端配置智能合约地址$0x4e60d62f8dee99d577fa8248843f0c49e4244662$，以及需要连接的区块链节点 ip 和端口；运行 validator-cli 程序，将完成对区块链的监听，获取 example.com 的验证请求，并完成对其的验证工作。

4. 信任列表修改

在域名客户端运行 domain-cli query 命令查询验证请求状态；待绑定完成后，通过运行4.7来修改信任 CA 列表。

```
domain-cli modifyTrustedCAs "Symantec Corporation"
```

Listing 4.7 信任列表修改命令

5. 游览器插件测试

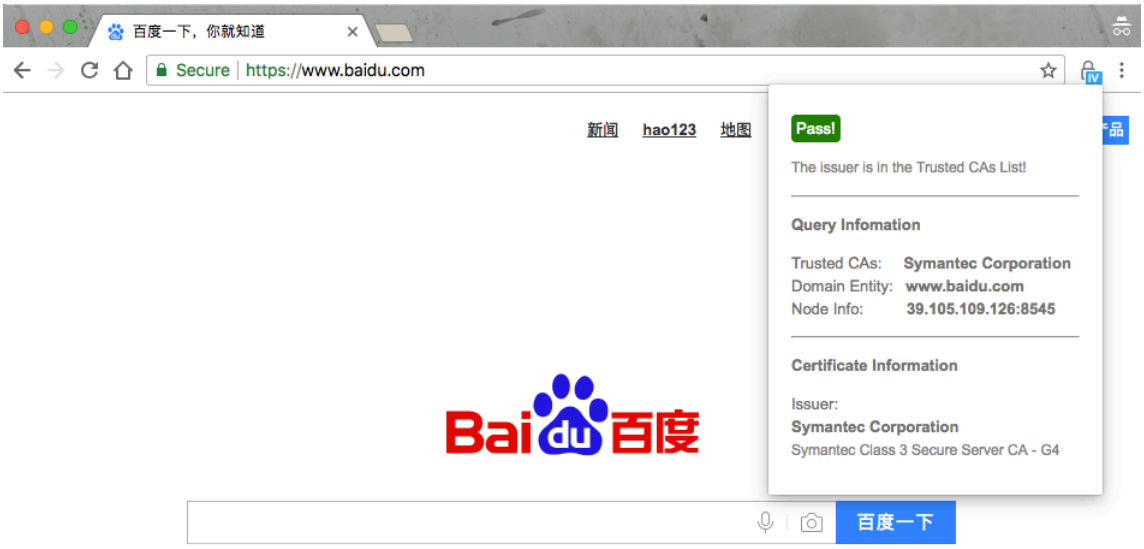

图 4.7 验证通过实例

按照前面的操作，将 www.baidu.com 的信任 CA 设置为了 Symantec Corporation，和其正在使用的证书签发机构是一致的。在游览器端安装 chrome 插件之后，访问 www.baidu.com，插件显示内容如图4.7所示，可以看出本次检验通过。

在插件中，上面区域显示了该域名站点的检验是否通过；中间区域显示了区块链网络中获取的相关信息，包括获取的信任 CA 列表、查询的域名以及介入区块链网络的节点信息；下面区域显示了证书的一些重要信息：证书的组织及签发者。

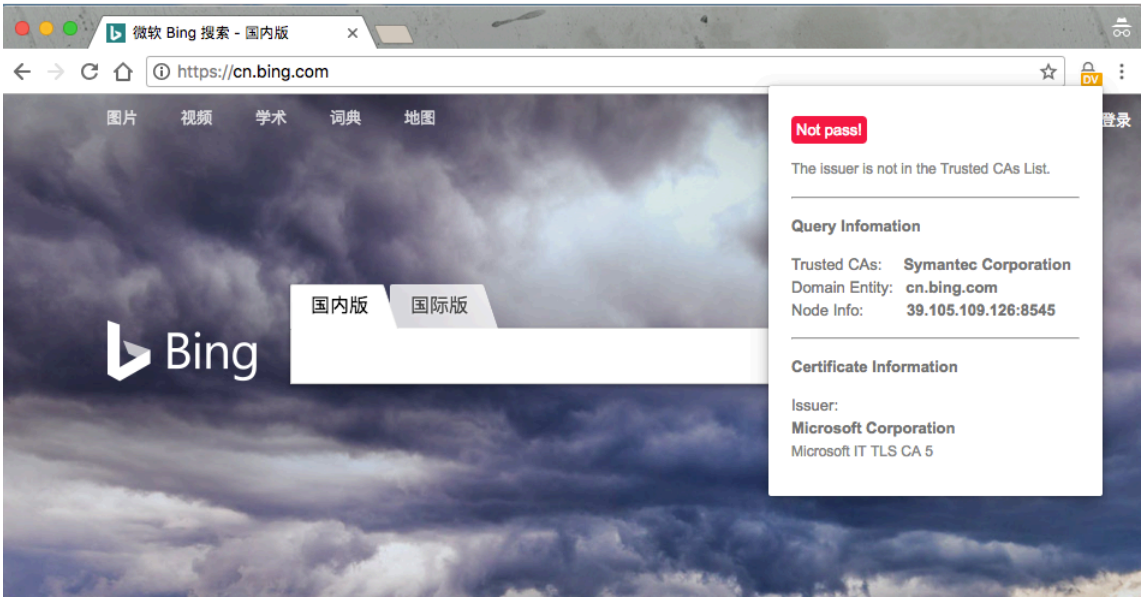

图 4.8　验证未通过实例

在对 cn.bing.com 绑定了错误的信任 CA 之后，访问该站点，插件显示内容如图4.8所示。由于设置的 CA 信任列表和其使用证书的签发者不一致，导致本次检验未通过。

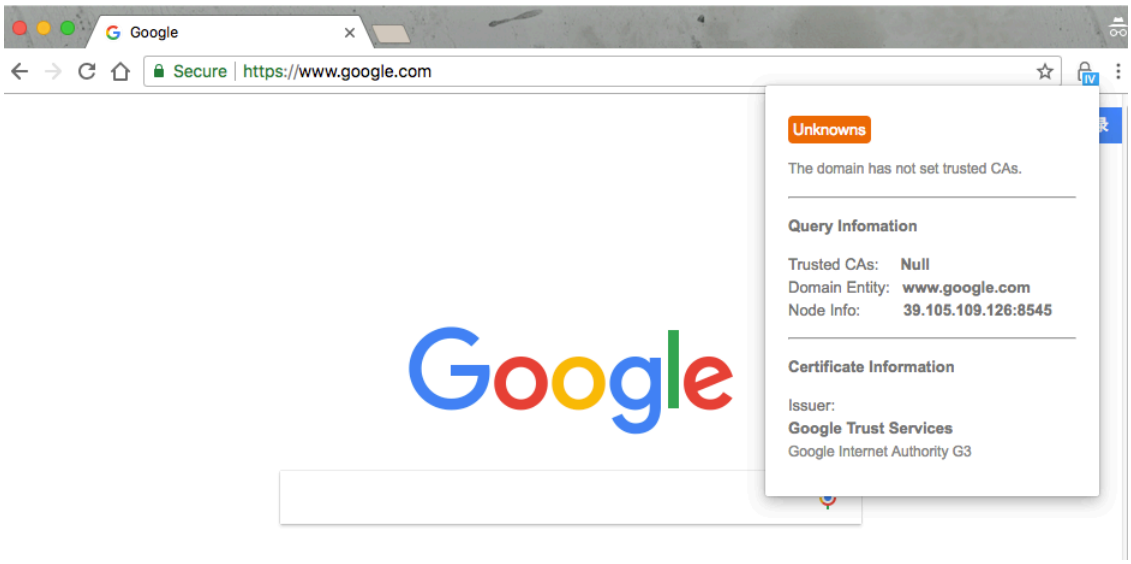

图 4.9　未设置信任列表实例

由于没有对 www.google.com 进行绑定操作，将无法在区块链上获得信任 CA 列表信息，得到结果如图4.9所示，提示未进行信任 CA 列表设置。

通过以上测试可以知道，通过实现的系统可以正常完成域名的身份认证以及信任CA列表的修改；并且通过游览器插件，在访问网页是可以获得相关的验证结果，本系统可以正常工作。

4.3.4 分析与讨论

绑定完成时间分析

按照本文4.3.2小节中的参数设置，在一个身份绑定请求在发起之后100-150块不等的时间内可以完成绑定。以太坊上面的出块时间大约为10s，在本测试环境中，一个绑定的完成时间在半个小时左右。按照每出一个块进行一次验证者选取的难度调整，而在选取过程中比较的是256位的二进制数，所以256块之后一定能完成一次绑定，该时间不会超过1个小时。

绑定完成时间和难度调整的参数设置、本系统中的加入验证节点数目密切相关，如果将本系统投入到实际环境中应用，应该进一步分析它们之间的关系。

交易执行代价分析

在方案设计中包含了所有交易执行的代价，在这里先忽略激励机制中的相关费用，考虑在本文实现系统上面的交易花费。根据底层区块链平台以太坊的特性，其上面执行任意对区块链产生改动的交易都需要消耗所谓的Gas。换句话说，执行的每一个操作都需要花费相应的费用，支付的内容包括执行操作的计算量和存储的内容大小。

对于以太坊上调用智能合约的交易，首先需要将交易发送到以太坊的区块中，其所需的花费被称为 *transaction costs*，是根据数据大大小来计算的；对于交易的执行，需要根据交易的计算量来进行费用评估，被称为 *execution costs*。在以太坊上交易费用的计量单位为 *gas*，根据该[57]可知现在 $1\ gas = 8\ Gwei$。

对于 *example.com* 的绑定、验证、检举和信任列表修改请求执行消耗的代价如表4.4所示。

交易名称	发送数据	交易费用(gas)	执行费用(gas)	总费用(gas)	价格 ($)
绑定请求	Pk_A、$example.com$	189451	165555	355006	1.977
验证操作	Pk_A、$example.com$、$Vcode$	208475	197342	405817	2.26
检举操作	Pk_A、$example.com$、Pk_B	198475	185638	384113	2.139
修改信任列表操作	"CA List"	36906	14674	51580	0.287

表 4.4　交易执行代价

总结与展望

PKI 技术作为能够实现身份鉴别、机密性、完整性、非否认等核心安全服务的基础设施, 在信息系统安全中发挥着重要作用。本文将区块链技术应用于现有的 PKI 系统中，提出了一种针对于 Web PKI 中域名实体的身份认证方案，解决了原有 PKI 体系下信任构建集中化的问题。

通过对现有 PKI 系统的分析，可以得知所有的信任都是以中心化的 CA 作为支点，这样做大大减弱了现有系统设计的复杂度，使得 PKI 被广泛的部署到日常生活和商业应用当中。中心化的 CA 作为 PKI 系统的核心，拥有以上优势的同时也存在相应的弊端和潜在的危险，导致了本系统的权利分配不均并且容易受到 CA 不端操作的影响。本文第三章对 PKI 中存在的其它问题也进行描述，并对现有的相关解决方案进行了归纳和总结。

本文给出方案的主要思想是将 PKI 系统中用户信任的 CA 列表通过区块链来记录，拥有公开且不可篡改的特性。与具有类似思路的已有方案是基于 DNS 的授权实体命名 (DANE) 方案，该方案是将域名的信任 CA 列表记录到 DNS 服务器上，但该方案中的身份认证工作依赖于 DNS 的身份管理服务，并且其安全性严重依赖于 DNS 服务的安全。与之相比，本文给出基于区块链的身份认证协议，其不需要依赖可信第三方来完成，而是通过区块链网络中的节点进行身份的验证和确认；同时，所有的操作和内容都记录在区块链上，由于其不可篡改的特性更加安全有效。

在完成方案设计之后，对方案进行了分析，论证了方案的可行性和安全性；其后对方案的实现方式进行了讨论，可以基于智能合约完成或是通过开发新的链来完成；本选选择较为简单的基于智能合约的方式进行了系统设计和实现，对实现的各个模块进行了功能性测试。

本给出的方案中涉及到很多参数的选择，如挑选验证节点的难度调整时间、验证的时间、验证的次数等，对于不同的实现环境和安全强度将会有不同设置，下一步的工作应该对这些参数的设置给出相关的理论分析和相关建议；同时，本文中给出了基于以太坊上智能合约的实现，如果部署到真实的环境当中，由于以太坊自身效率的问题，其认证效率会大大折扣，并且需要付出较高的代价；最后，本文在实现过程中给出域名客户端和验证节点客户端没有图形化的界面，需要进一步完善客户端的完整性，才能简单便捷的被投入使用。

将以上提到的工作进一步完成之后，本文给出的基于区块链的方案实际上已经实现了对域名身份的认证，完成认证的公私钥对完全可以用在安全通信的建立，取代现

有的 PKI 系统，实现一个基于区块链的公钥基础设施；该基础设施将不需要有可信第三方，完全依靠网络中的用户来完成身份的认证，并且具有公开、透明、防篡改的特性，但是达到这种状态还需要很长一段时间的发展，以及可能需要更好的去中心化的身份认证方案被提出。

参考文献

[1] PRINS J R, CYBERCRIME B U. DigiNotar Certificate Authority breach’ Operation Black Tulip’ [J]. Fox-IT, November, 2011.

[2] RISTI I. Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications. M. 2014.

[3] DUCKLIN P. The TURKTRUST SSL certificate fiasco-what really happened, and what happens next[J]. Naked Security. SOPHOS, 2013, 8.

[4] WENDLANDT D, ANDERSEN D G, PERRIG A. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing.[C]// USENIX Annual Technical Conference. Vol. 8. 2008: 321–334.

[5] ALICHERRY M, KEROMYTIS A D. Doublecheck: Multi-path verification against man-in-the-middle attacks[C]// Computers and communications, 2009. iscc 2009. ieee symposium on. IEEE. 2009: 557–563.

[6] Convergence[EB/OL]. https://convergence.io/.

[7] MODELL M, BARZ A, TOTH G, et al. Certificate patrol[J]. Online at https://addons. mozilla. org/en-US/firefox/addon/certificate-patrol, 2014.

[8] SOGHOIAN C, STAMM S. Certified lies: Detecting and defeating government interception attacks against SSL (short paper)[C]// International Conference on Financial Cryptography and Data Security. Springer. 2011: 250–259.

[9] MARLINSPIKE M, PERRIN T. Internet-Draft: Trust Assertions for Certificate Keys[J]. 2012.

[10] EVANS C, PALMER C, SLEEVI R. Public key pinning extension for HTTP[R]. R. 2015.

[11] BARNES R L. DANE: Taking TLS authentication to the next level using DNSSEC[J]. IETF Journal, 2011, 7(2).

[12] KASTEN J, WUSTROW E, HALDERMAN J A. Cage: Taming certificate authorities by inferring restricted scopes[C]// International Conference on Financial Cryptography and Data Security. Springer. 2013: 329–337.

[13] ECKERSLEY P. Internet-draft: Sovereign key cryptography for internet domains[J]. 2012.

[14] LAURIE B, LANGLEY A, KASPER E. Certificate transparency[R]. R. 2013.

[15] KIM T H.-J, HUANG L.-S, PERRIG A, et al. Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure[C]// Proceedings of the 22nd international conference on World Wide Web. ACM. 2013: 679–690.

[16] RYAN M D. Enhanced Certificate Transparency and End-to-End Encrypted Mail.[C]// NDSS. 2014.

[17] BASIN D, CREMERS C, KIM T H.-J, et al. ARPKI: attack resilient public-key infrastructure[C]// Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM. 2014: 382–393.

[18] CHEVAL V, RYAN M, YU J. DTKI: a new formalized PKI with no trusted parties[J]. ArXiv preprint arXiv:1408.1023, 2014.

[19] NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system[J]. 2008.

[20] BUTERIN V, et al. Ethereum white paper[J]. GitHub repository, 2013.

[21] ZHENG Z, XIE S, DAI H.-N, et al. Blockchain challenges and opportunities: A survey[J]. Work Pap.–2016, 2016.

[22] PETERS G, PANAYI E, CHAPELLE A. Trends in cryptocurrencies and blockchain technologies: a monetary theory and regulation perspective[J]. 2015.

[23] BOCEK T, RODRIGUES B B, STRASSER T, et al. Blockchains everywhere-a use-case of blockchains in the pharma supply-chain[C]// Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on. IEEE. 2017: 772–777.

[24] SCHNEIDER M. Design and Prototypical Implementation of a Blockchain-Based System for the Agriculture Sector[J]. 2017.

[25] HARDJONO T, SMITH N. Cloud-based commissioning of constrained devices using permissioned blockchains[C]// Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security. ACM. 2016: 29–36.

[26] INSTITUTE N R. Survey on Blockchain Technologies and Related Services[J]. 2016.

[27] ALI M, NELSON J C, SHEA R, et al. Blockstack: A Global Naming and Storage System Secured by Blockchains.[C]// USENIX Annual Technical Conference. 2016: 181–194.

[28] LOIBL A, NAAB J. Namecoin[J]. Namecoin. info, 2014.

[29] AKINS B W, CHAPMAN J L, GORDON J M. A whole new world: Income tax considerations of the Bitcoin economy[J]. Pitt. Tax Rev., 2014, 12: 25.

[30] SHARPLES M, DOMINGUE J. The blockchain and kudos: A distributed system for educational record, reputation and reward[C]// European Conference on Technology Enhanced Learning. Springer. 2016: 490–496.

[31] CARBONI D. Feedback based Reputation on top of the Bitcoin Blockchain[J]. ArXiv preprint arXiv:1502.01504, 2015.

[32] BALDI M, CHIARALUCE F, FRONTONI E, et al. Certificate Validation Through Public Ledgers and Blockchains.[C]// ITASEC. 2017: 156–165.

[33] MATSUMOTO S, REISCHUK R M. IKP: Turning a PKI Around with Blockchains.[J]. IACR Cryptology ePrint Archive, 2016, 2016: 1018.

[34] AZARIA A, EKBLAW A, VIEIRA T, et al. Medrec: Using blockchain for medical data access and permission management[C]// Open and Big Data (OBD), International Conference on. IEEE. 2016: 25–30.

[35] WEISE J. Public key infrastructure overview[J]. Sun BluePrints OnLine, August, 2001: 1–27.

[36] HOUSLEY R, FORD W, POLK W, et al. Internet X. 509 public key infrastructure certificate and CRL profile[R]. R. 1998.

[37] ABADI M, BIRRELL A, MIRONOV I, et al. Global Authentication in an Untrustworthy World.[C]// HotOS. 2013.

[38] COOPER D. Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile[J]. 2008.

[39] MYERS M, ANKNEY R, MALPANI A, et al. X. 509 Internet public key infrastructure online certificate status protocol-OCSP[R]. R. 1999.

[40] TOPALOVIC E, SAETA B, HUANG L.-S, et al. Towards short-lived certificates[J]. Web 2.0 Security and Privacy, 2012.

[41] LAURIE B, KASPER E. Revocation transparency[J]. Google Research, September, 2012.

[42] DOUCEUR J R. The sybil attack[C]// International workshop on peer-to-peer systems. Springer. 2002: 251–260.

[43] ANTONOPOULOS A M. Mastering Bitcoin: unlocking digital cryptocurrencies[M]. " O'Reilly Media, Inc.", 2014.

[44] VASIN P. Blackcoin's proof-of-stake protocol v2[J]. URL: https://blackcoin. co/blackcoin-pos-protocol-v2-whitepaper. pdf, 2014.

[45] CASTRO M, LISKOV B, et al. Practical Byzantine fault tolerance[C]// OSDI. Vol. 99. 1999: 173–186.

[46] LARIMER D. Delegated proof-of-stake (dpos)[J]. Bitshare whitepaper, 2014.

[47] SCHWARTZ D, YOUNGS N, BRITTO A, et al. The Ripple protocol consensus algorithm[J]. Ripple Labs Inc White Paper, 2014, 5.

[48] ATENIESE G, BONACINA I, FAONIO A, et al. Proofs of space: When space is of the essence[C]// International Conference on Security and Cryptography for Networks. Springer. 2014: 538–557.

[49] EYAL I, SIRER E G. Majority is not enough: Bitcoin mining is vulnerable[C]// International conference on financial cryptography and data security. Springer. 2014: 436–454.

[50] SZABO N. Smart contracts: building blocks for digital markets[J]. EXTROPY: The Journal of Transhumanist Thought,(16), 1996.

[51] CHURYUMOV A. Byteball: a decentralized system for storage and transfer of value. J. 2016.

[52] CACHIN C. Architecture of the Hyperledger blockchain fabric[C]// Workshop on Distributed Cryptocurrencies and Consensus Ledgers. 2016.

[53] FROMKNECHT C, VELICANU D, YAKOUBOV S. A Decentralized Public Key Infrastructure with Identity Retention.[J]. IACR Cryptology ePrint Archive, 2014, 2014: 803.

[54] RYSTSOV D. Nidaba: a distributed scalable PKI with a stable price for certificate operations[J].

[55] AL-BASSAM M. SCPKI: a smart contract-based PKI and identity system[C]// Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts. ACM. 2017: 35–40.

[56] Certificate info[EB/OL]. `https://github.com/blupig/certificate-info`.

[57] Ethgasstation[EB/OL]. `https://ethgasstation.info/`.

致谢

转眼间三年的硕士时间即将过去，在这里首先要特别感谢关志老师在这三年中对我的指导和关怀。关老师是一位在信息安全和密码领域知识渊博的学者，带领我去学习和了解专业相关的知识，并教我如何去研究并解决问题。感谢关老师在我研究生学习上和毕业论文上的耐心指导，以及平时对我们在生活上的关心和照顾。同时也要感谢陈钟老师在研究生其间对我的指导和帮助，感谢实验室其他老师在这三年中给我的教导。

感谢孟宏伟、唐聪师兄带领我进入区块链领域，感谢 Abba 与我在区块链方面的交流，感谢刘超、王珂、李悦在毕业论文书写期间的讨论和建议；也感谢赵晓濛在三年研究生工作和学习中的帮助，感谢苏超、赖蔚两位师弟在实验室工作中的协助；感谢已经毕业的王康达、刘洪元、王诗吟、董秋香等师兄师姐在研究生学习和生活中的帮助，感谢三位室友李奕、邱兆鹏、林武桃三年的帮助和陪伴，也感谢马凌霄、董自鸣等大学同学的陪伴。

最后感谢我的父母和家人，在生活和学习中始终给予我帮助和支持，是我不断前进的力量源泉。

在此论文完成之际，谨向三年研究生学习生活中给予我关心和帮助的良师益友及亲人们致以最诚挚的谢意！

北京大学学位论文原创性声明和使用授权说明

原创性声明

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

论文作者签名：　　日期：　　年　　月　　日

学位论文使用授权说明

（必须装订在提交学校图书馆的印刷本）

本人完全了解北京大学关于收集、保存、使用学位论文的规定，即：

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

（保密论文在解密后遵守此规定）

论文作者签名：　　导师签名：　　日期：　　年　　月　　日