



北京大学

## 硕士研究生学位论文

题目： 基于区块链的 PKI 实体鉴权  
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

任何收存和保管本论文各种版本的单位和个人，未经本论文作者同意，不得将本论文转借他人，亦不得随意复制、抄录、拍照或以任何方式传播。否则一旦引起有碍作者著作权之问题，将可能承担法律责任。



## 摘要

公钥基础设施（PKI）作为保证数据的机密性、完成性、不可抵赖性以及身份鉴别的重要方式，已经被广泛的应用在电子邮件、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 相关的技术和标准逐渐完善，但由于复杂的应用环境和中性化的系统架构，任然存在着各种问题。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些列外在的改进措施，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，一直是近年来 PKI 相关研究尝试解决的问题。

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。针对 PKI 系统中信任架构的问题，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，其安全性不依赖于其它应用实体；另外，所有的操作都通过区块链来完成，提供了公开透明的审计特性。本文对给出方案可能存在的攻击方式进行了分析，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；依据设计给出相应系统的实现，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 Named Entities**

## **Design and Implementation**

Anran Li (Software and Theory of Computer)

Directed by Prof. Zhi Guan

### **ABSTRACT**

With the rapid development of network technology, computer applications have gradually penetrated into all aspects of daily life. The use of networks to complete the exchange and processing of various information has become a part of life. In daily network applications such as e-mail, e-commerce, online banking and resource publishing, it is necessary to communicate with unidentified entities in an open network environment. How to ensure confidentiality, integrity, authentication and non-repudiation during data transmission is always getting highlight attentions. The emergence of public key cryptography, especially the development of Public Key Infrastructure (PKI) technology, is an important way to solve the above problems. After nearly two decades of development, PKI-related technologies and standards have gradually improved. Various types of CA certification centers have been established one after another, and PKI have gained popularity in the applications mentioned above.

Although the PKI system has been widely applied to various scenarios, there are still various problems due to the complex environment and its architecture. These problems are mainly reflected in the trust construction of the system and the credibility of the issuance certificate. The credibility of the issuance certificate can be solved by a series of external improvement measures and standardizing the operations of certificate management. However, the trust building problem of the system needs to adjust the components and architecture of the existing PKI system and balance the trust and rights allocation among various entities. This has always been a problem that PKI related research attempts to solve.

This article will introduce and analyze the existing PKI system architecture, and find out the possible problems in the system according to the lifecycle of the certificate, and also analyze 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. Aiming at the problem of trust architecture in PKI system,

this paper proposes a domain name authentication scheme based on blockchain technology. The domain owner can complete the control of its own trust CAs in a distributed network enviromanet without trusted third party. As an auxiliary tool of the existing PKI system, the solution can help PKI users to improve the control over the issuance of certificates, reduce the absolute power 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

<b>第一章 绪论</b>	<b>1</b>
1.1 背景知识和研究意义 . . . . .	1
1.2 国内外研究现状 . . . . .	2
1.2.1 PKI 的研究现状 . . . . .	2
1.2.2 区块链技术在 PKI 中的应用 . . . . .	4
1.3 本文研究工作和章节安排 . . . . .	5
<b>第二章 PKI 与区块链技术</b>	<b>7</b>
2.1 PKI . . . . .	7
2.1.1 PKI 系统 . . . . .	7
2.1.2 证书 . . . . .	8
2.1.3 系统中存在的问题 . . . . .	10
2.1.4 已有的改善措施 . . . . .	12
2.2 区块链技术 . . . . .	14
2.2.1 比特币 . . . . .	14
2.2.2 共识机制 . . . . .	16
2.2.3 智能合约 . . . . .	17
2.2.4 区块链的应用 . . . . .	18
<b>第三章 基于区块链的命名实体鉴权方案设计</b>	<b>19</b>
3.1 方案概况 . . . . .	19
3.1.1 实体角色 . . . . .	19
3.1.2 工作流程 . . . . .	20
3.2 身份绑定方案设计 . . . . .	21
3.2.1 基于验证时间的身份绑定方案 . . . . .	21
3.2.2 基于验证次数的身份绑定方案 . . . . .	23
3.2.3 检举的工作机制 . . . . .	26
3.2.4 验证交易的附带信息 . . . . .	26
3.2.5 奖励惩罚机制 . . . . .	27
3.3 安全性分析 . . . . .	27

<b>第四章 基于区块链的命名实体鉴权系统设计与实现</b>	<b>31</b>
4.1 系统设计 . . . . .	31
4.1.1 系统角色 . . . . .	31
4.1.2 系统架构 . . . . .	32
4.1.3 实现方式 . . . . .	33
4.2 模块设计 . . . . .	34
4.2.1 智能合约 . . . . .	35
4.2.2 域名客户端 . . . . .	36
4.2.3 验证节点客户端 . . . . .	37
4.2.4 浏览器验证插件 . . . . .	38
4.3 系统测试 . . . . .	39
4.3.1 环境搭建 . . . . .	39
4.3.2 相关参数 . . . . .	40
4.3.3 功能测试 . . . . .	41
<b>总结与展望</b>	<b>45</b>
<b>参考文献</b>	<b>47</b>
<b>致谢</b>	<b>51</b>
<b>北京大学学位论文原创性声明和使用授权说明</b>	<b>53</b>

# 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。

公钥基础设施 PKI 作为利用非对称加密算法原理和技术实现并提供安全服务的技术和规范，已经成为互联网中保证数据传输保密性、完整性和不可否认性的重要组成部分。PKI 体系中的证书，已经被广泛地应用在各种身份认证的场景下，特别是在互联网中建立 SSL 通信的过程中，用于保证客户端对服务器身份的认证。

证书在 PKI 系统中作为公钥和标识绑定的载体，起着流通信用的作用。围绕证书而言，在该系统存在的角色主要包括以下三类：授权机构 (CA)，用户以及依赖方。在 PKI 系统中，用户发起证书签发请求，递交给授权机构；授权机构负责证书的签发吊销等管理工作；依赖方则是证书的受用对象，检查证书的合法性并从中获得对方的公钥。

证书是否有效是通过验证该证书是否由授权机构所签发来决定的。依赖方会事先将所有授权机构的相关证书下载到本地，在通信过程中获取对方证书和证书链来验证证书的有效性；而证书附有公钥和标识的绑定关系，之后会用该公钥与对方进行通信，并认为拥有对应私钥的人就是绑定标识者。也就是说，授权机构是整个系统中的信任支点，系统中的依赖方需要对其签发的证书给予绝对信任。如果授权机构有作恶行为，比如签发虚假的证书，网络中的实体并不能辨认该证书的真伪，导致系统中的依赖方与虚假的身份进行通信。

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。比如在 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<sup>[1]</sup>，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。

由于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。从最初的类似比特币这样的密码货币争相发布,到现在与区块链相关的传统应用出现在大家的视野中,区块链技术逐渐的被应用在其它领域,其去中心化、防篡改的特性成为了这些应用的核心所在。

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 实体之间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经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<sup>[2]</sup>。

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<sup>[3]</sup>; 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, 一些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<sup>[4]</sup> 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<sup>[5]</sup> 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<sup>[5]</sup>; 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<sup>[6]</sup> 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<sup>[7]</sup> 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<sup>[8]</sup> 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<sup>[9]</sup> 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 <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<sup>[10]</sup> 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。Sovereign Keys(SK)<sup>[11]</sup> 是第一个使用公共日志服务器的 PKI 方案。在该方案中, 允许域名所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)<sup>[12]</sup> 是由 Google 提出的另外一种方案, 旨在帮助域名所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)<sup>[13]</sup> 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)<sup>[14]</sup> 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 中单类型日志服务器的信任, 使得其具有更高的安全性。

与 SK 方案相似, Accountable Key Infrastructure(AKI)<sup>[15]</sup> 也允许域名所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)<sup>[16]</sup> 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 区块链技术在 PKI 中的应用

区块链的概念最初在虚拟数字货币 Bitcoin 中提出的<sup>[17]</sup>, 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<sup>[18]</sup> 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。

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 [19] 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<sup>[20]</sup> 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<sup>[21]</sup> 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<sup>[22]</sup> 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<sup>[23]</sup> 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。

### 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 and 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。

本论文的章节结构如下:

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 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。

第四章: 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 workflows,然后对方案进行详细阐述,并对方案的安全性进行分析。

第五章: 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,并对系统进行实现和相关测试。

第六章: 对本文提出的方案进行总结,并对未来的工作提出展望。





## 第二章 PKI 与区块链技术

### 2.1 PKI

本章将对 PKI 系统进行详细介绍，首先从系统架构对 PKI 进行解析，阐述系统中包含的组成部分和各自的功能；其后将对证书进行介绍，包括证书的结构以及证书的生命周期；最后对本系统存在的问题以及相应的解决方案进行简要叙述。

#### 2.1.1 PKI 系统

公钥基础设施 PKI (Public Key Infrastructure) 作为一种遵循标准的密码管理平台，其可以为网络应用提供密钥和证书管理服务，使用户可以在多种应用环境下方便的使用加密和数字签名技术，从而保证数据的机密性、完整性、有效性和抗抵赖性。

本小节将从框架和其包含的相关组件对 PKI 进行介绍。

#### PKI 基本框架

PKI 框架中包括安全和操作策略，安全服务以及支持公钥密钥和证书管理的交互式协议。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<sup>[24]</sup>，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支持并完成数据、凭证在各种不安全环境下的安全交换。

#### PKI 的系统构成

在 PKI 系统中可能会包含以下组成部分：

- 注册机构 (RA)  
RA 的作用是执行身份验证和处理新的数字证书请求、更新数字证书请求和吊销数字证书请求，并将这些请求递交给授权机构。
- 授权机构 (CA)  
CA 将创建和发布数字证书以及证书吊销列表 (CRLs)，其颁发的数字证书将对用户标识（如主体名称）和绑定的公钥进行签名。
- 验证机构 (VA)  
VA 是一个 PKI 的管理实体，可以用来检查数字证书的有效性。当证书的签发者和证书的状态管理服务由不同的实体提供时，将使用到 VA。

- 用户

PKI 面对的用户是证书持有者或者密钥持有者，通过遵从认证运作规范（CPS）和证书策略（CP）获得证书，完成证书和密钥对的绑定。PKI 系统中的用户可以是个体、组织或者非个人实体，有责任保存好自己的私钥不被泄露。

- 依赖方

依赖方在 PKI 系统中接收、验证和接受数字证书。

- 证书策略 (CP)

证书策略是一组安全规则要求，适用于一类应用系统的共同安全需求。

- 认证运作规范 (CPS)

CPS 描述了 CA 提供数字证书服务的规则和处理方式，其中可能会包括提供服务描述，证书生命周期的管理细则、业务信息、法律义务和金融责任等。

图2.1将给出以上所提到组件之间的关系，图中的剪头将表示数字证书和证书状态信息的传递。

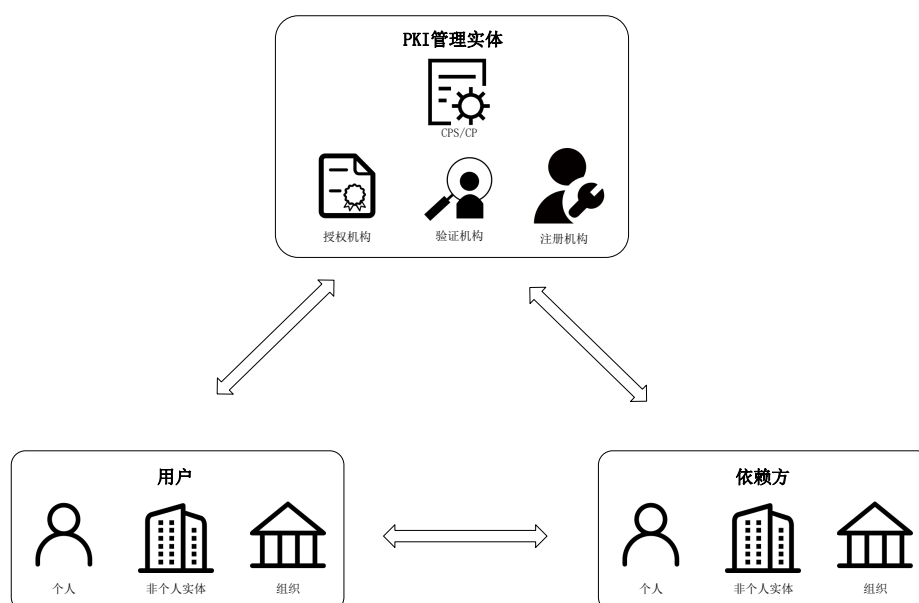


图 2.1 PKI 组成部分

## 2.1.2 证书

### 证书的结构

数字证书作为 PKI 密钥管理服务中的公钥载体，通过授权机构对其的生成、发布、撤销来提供密钥管理服务。数字证书是一个由授权机构 CA 签名的数字文本，其中包含了证书申请者的身份信息比如实体的名称、电子邮件等，以及公钥信息。数字证书

就像身份证一样，是包含第三方（即授权机构）签名的身份证明，依赖方可以通过对证书上面签名来判断身份是否真实有效性。

在 PKI 的发展过程中，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 (International Telecommunication 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。

X.509 v3 版本的数字证书结构如图2.2所示。

证书版本号	证书序列号	签名算法标识符	证书颁发者X.509名称	证书有效期	证书持有者X.509名称	证书持有者公钥	证书颁发者唯一标识符	证书持有者唯一标识符	证书扩展部分

图 2.2 X.509 数字证书结构

## 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2.3所示。

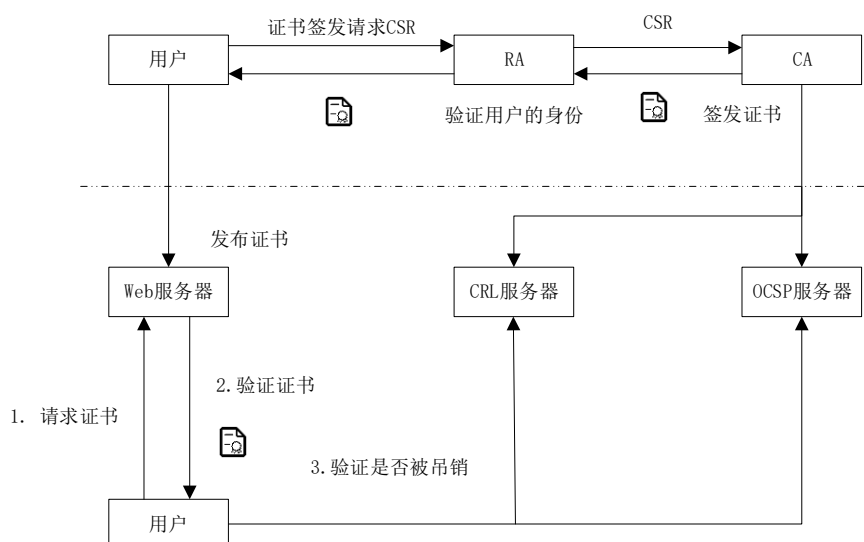


图 2.3 证书的生命周期

### 2.1.3 系统中存在的问题

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。

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<sup>[2]</sup>：

- 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，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任然可以被使用。

- **域名验证过于简单**

DV 证书的签发是基于域名的 WHOIS 协议查询域名所有者信息来完成的，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。

## 2.1.4 已有的改善措施

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2.4所示。

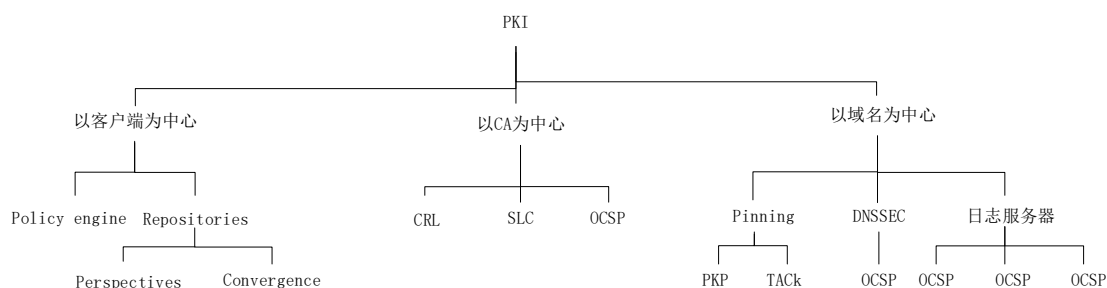


图 2.4 PKI 方案分类

### 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<sup>[25]</sup> 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，证书的一致性等等。

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收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<sup>[4]</sup> 和 Convergence<sup>[6]</sup> 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<sup>[26]</sup>, 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)<sup>[27]</sup> 允许客户端通过 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)<sup>[28]</sup>, 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所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: pinning、DNSSEC、日志服务器。

### 1. Pinning-based (公钥固定)

如 Public Key Pinning(PKP)<sup>[9]</sup> 和 Trust Assertions for Certificate Keys(TACK)<sup>[28]</sup> 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)<sup>[29]</sup>, 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, 为域名所有者提供公共的、可审计的 CA 行为日志, 监督 CA 的行为。例如 Sovereign Keys (SK)<sup>[30]</sup> 要求域名所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)<sup>[12]</sup> 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）<sup>[31]</sup> 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）<sup>[13]</sup> 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）<sup>[15]</sup>，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。

## 2.2 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。

### 2.2.1 比特币

比特币作为一个 P2P 电子货币系统，是由中本聪在 08 年中的一篇论文中提出<sup>[17]</sup> 的。本小节将给出比特币协议的简要概括。

#### 比特币网络

比特币是构建在一个 P2P 网络上的数字货币系统，网络中的节点负责分发并记录交易，客户端通过与该系统的交互完成转账等操作。区块链作为比特币的核心，其被该网络中的所有节点以只追加的方式维护，并且通过工作共识机制保证每个节点上存储的内容是一致的。区块链是由一系列通过哈希值相连的区块组成的，每个区块中包含了比特币网络中被广播的交易。

比特币中的节点通过计算来确定下一区块的生产者，在确定生产者的过程中需要每个节点同时去解决一个工作量证明的问题，即所谓的挖矿，对于特定的区块  $B$  需要计算：

$$SHA256(SHA256(B)) = (0^l || 0, 1^{256-l}) \quad (2.1)$$

其中  $l$  由前面产生的区块链来决定，来保证整个网络中的区块产生速度平均在每 10 分钟一块。当一个参与者找到一个有效的方案时，将可以去生成一个新的块，并将这个块广播给网络中的所有节点。如果该块被其它节点验证是有效的（包括块内包含交易的有效性和工作量证明的有效性等），该块将会被接受并添加到区块链上，重复以上过程来增加区块链的长度，进而完成网络中交易的记录。



比特币提供了两方面的激励来促使节点去挖取新的块，第一种是当成功的挖取到一个新的块时，新块的创建者可以获得奖励，当前的奖励为 12.5 个比特币；第二种奖励来之交易的交易费，纳入块的交易所支付的交易费可以归块的创建者拥有。

### 比特币交易

比特币交易包含了多个输入和输出。每个输出通过一个两元组来表示  $(a, V)$ ，其中  $a$  是发送比特币的数量， $V$  是一个关于谁有权花掉这笔钱的声明。在比特币中，这个声明被称作 *scriptPubKey*，它是由栈式非图灵完备语言书写的比特币脚本。交易的输入指向一个前面交易的输出，同时包含第二个脚本 *scriptSig*，可以用于 *scriptPubKey* 的代码和数据。Coinbase 交易不需要纳入一个先前的交易输出作为输入。

为了向 Bob 发送  $d$  个比特币，Alice 需要 Bob 的 ECDSA 公钥  $pk_b$ ，并将其哈希和发送的数量  $d$  以及一些其它的指令放置在 *scriptPubKey* 中作为一个交易的输出，当然，这笔交易需要至少需要  $d$  数量的比特币作为输入，如 2.1 所示。如果输入的金额大于输出的金额，而且交易提交者没有设置额外的转账地址的话，剩余的金额将会被作为交易费支付给将其纳入块的节点。一旦这笔交易被广播到网络中并收入到一个有效的块中，相应的比特币就属于 Bob 所有了。如果 Bob 想花掉这笔钱，他需要将这笔交易的输出作为新建交易的输入，并附上包含  $pk_b$  对应私钥  $pr_b$  签名的 *scriptSig* 即可。

```

1 Input:
   Previous tx: 0205937d9faa1a313b7826fdab ...
3   Index: 0
   scriptSig: odc23450cdf8cdc710e5e92af7647 ...
5
6 Output:
7   Value: 500000
   scriptPubkey: OP_DUP OP_HASH160 a45f275794fd2337ebf7ddd018c11a21fb6c283
   OP_EQUALVERIFY OP_CHECKSIG

```

Listing 2.1 比特币交易实例

### 匿名性

匿名并不是比特币设计时所考虑的目标。比特币只能通过公钥和 hash 来提供假名，用户在使用过程中可以生成任意多的账户。事实上，很多比特币客户端在交易过程中生成新的身份来保证用户的隐私。

虽然比特币并没有将匿名性作为其设计的目标，比特币用户希望付出一定的代价来获取匿名性，比如有掉钱的风险或者支付更多的交易费用。比如通过交易混合<sup>[32]</sup>的技术来减弱交易地址之间的关联性，使得难追踪资金的流向；或者通过环签名或者零知识证明<sup>[33]</sup>的方案提供匿名性。

比特币是一种去中心化的数字货币，该系统中不存在中心化的银行或货币发行者，其上的货币通过奖励的方式产生，并且奖励的数目会随着时间的增加而减少。

### 2.2.2 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<sup>[34]</sup>：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<sup>[35]</sup>。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<sup>[17]</sup>。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生下一个块，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。

在比特币中使用的哈希算法是 SHA-256，其它的哈希算法如 Blake-256 和 scrypt 也在被使用，同时还有一些共识机制中混入多种哈希算法。

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)<sup>[36]</sup> 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；与 POW 相比，POS 拥有自己

优势和劣势，并且其应用在起来也十分的复杂。

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<sup>[37]</sup>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。

### 2.2.3 智能合约

智能合约的概念于 1994 年被 Nick Szabo 提出<sup>[38]</sup>，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。

不同区块链平台合约语言的自身性质将决定智能合约能够实现的操作，如比特币上使用的基于堆栈的脚本语言，指令集小并且不支持循环等复杂运算，可以实现的功能就很有限。从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，能够实现各种应用逻辑，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广泛的应用场景。

## 2.2.4 区块链的应用

区块链技术最初只是被用于比特币中完成一个数字货币的构建，但区块链技术具有去中心化和防篡改的特性而被逐渐延伸到其它应用场景。比较典型的应用包括金融、物联网、社会公共服务、名誉系统和安全和隐私 5 个方面<sup>[39]</sup>，如图2.5所示。

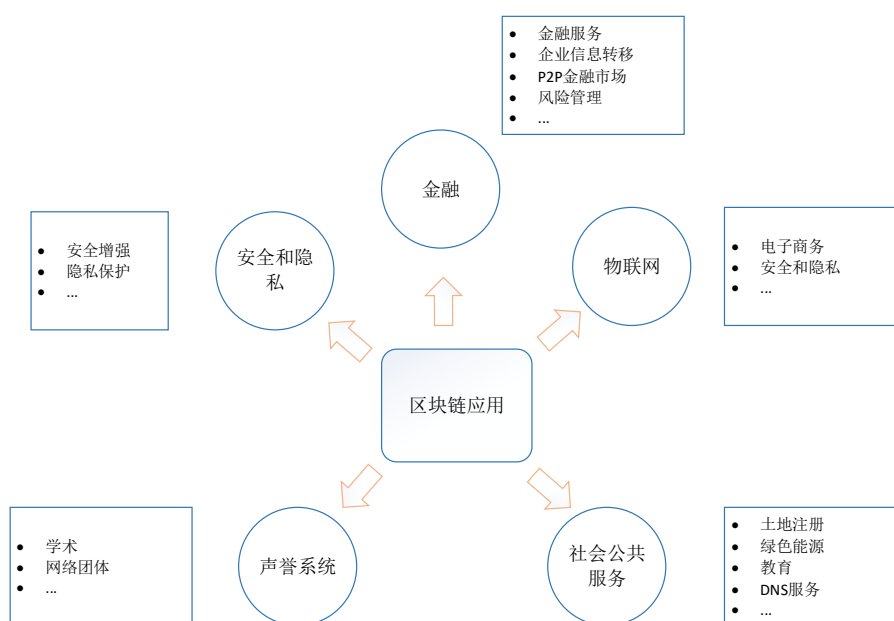


图 2.5 区块链典型的应用领域

金融方面的应用包括金融服务、p2p 金融市场、风险管理等，其中比特币、以太坊就是传统金融服务的另外一种表现形式，区块链给金融相关的应用带来了更好的特性；物联网方面的应用旨在使用区块链技术完成物联网实体之间数据的交换并提高自身的安全和隐私；与社会公共服务相关的区块链应用包括土地注册、绿色能源方面、教育和 DNS 服务等方面；名誉系统旨在利用区块链的不可篡改的特性，记录如学术论文或者网络团体的行为，提供有效的声誉证明。

## 第三章 基于区块链的命名实体鉴权方案设计

本章节将阐述基于区块链的命名实体鉴权方案，首先从对本方案进行宏观描述，包括其中涉及的角色和大体流程；然后对鉴权方案进行阐述，对方案中各角色的操作和使用的方法进行详细描述；最后对设计的方案进行安全性分析。

### 3.1 方案概况

#### 3.1.1 实体角色

针对证书申请和使用流程，在 PKI 系统中涉及的主要角色包含以下三类：

- 授权机构（CA）：授权机构是证书的签发者，在 PKI 系统充当证书签发和管理的角色。
- 证书申请者：证书申请者是 PKI 系统中用户，作为证书的持有者，在初始过程中需要向 CA 提出证书申请，在完成身份验证之后即可获得相应的证书，并在使用过程中提供证书以证明自己的身份，在本场景中，证书的申请者指域名或者站点。
- 证书验证者：证书验证者在 PKI 系统中是依赖方，是证书的受用者，在通信过程中需要认证对方身份的时候，获得证书并完成有效性验证。

不同的角色在 PKI 证书的流转过程中对证书的控制权不尽相同，授权机构作为证书的签发方，对其是否签发证书具有绝对的主动权，甚至可以在未经证书申请者同意的情况下签发恶意证书；证书申请者则只能发起证书申请，不能对证书是否被签发起决定权；证书验证者则只能验证证书是否合法，对于恶意 CA 私自签发的虚假证书，并没有辨别真伪的能力。

可以看出，在传统的 PKI 系统中，证书申请者和证书授权者之间的地位是不对等的，这是由于证书授权方对证书的拥有绝对管理权，所以在 PKI 系统中，需要所有的角色对中心化的 CA 绝对性信任。为了使得证书申请者和证书授权者之间的权利更加均衡，需要提高证书申请方对证书签发的控制权，使用本文中涉及的命名实体鉴权方案，保证未被允许的证书签发机构无法对证书进行私自签发。

在本方案中，以上提到的三类角色具有以下特性：

- 授权机构（CA）：在本方案中，其角色和功能保持不变，负责对发起证书申请的实体进行证书颁发。
- 证书申请者：作为 PKI 系统中的用户，在本方案中其将利用区块链完成身份的认证，并将自己信任的 CA 列表发布在区块链上。

- 证书验证者：作为 PKI 系统中的依赖方，在本方案中验证者可以通过区块链查询通信实体的信任 CA 列表，保证接受到的证书是通信实体信任 CA 签发的。
- 验证节点：在本方案中，验证节点作为证书申请者的身份验证者（域名），通过这些节点完成对区块链上证书申请者的身份认证。

### 3.1.2 工作流程

通过上一章的内容知道，在本方案中涉及的实体主要包含一下三类：授权机构 CA、证书申请者域名、证书验证者浏览器以及区块链上的验证节点。域名作为证书的申请者 and 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，整个方案的工作流程如图3.1所示。

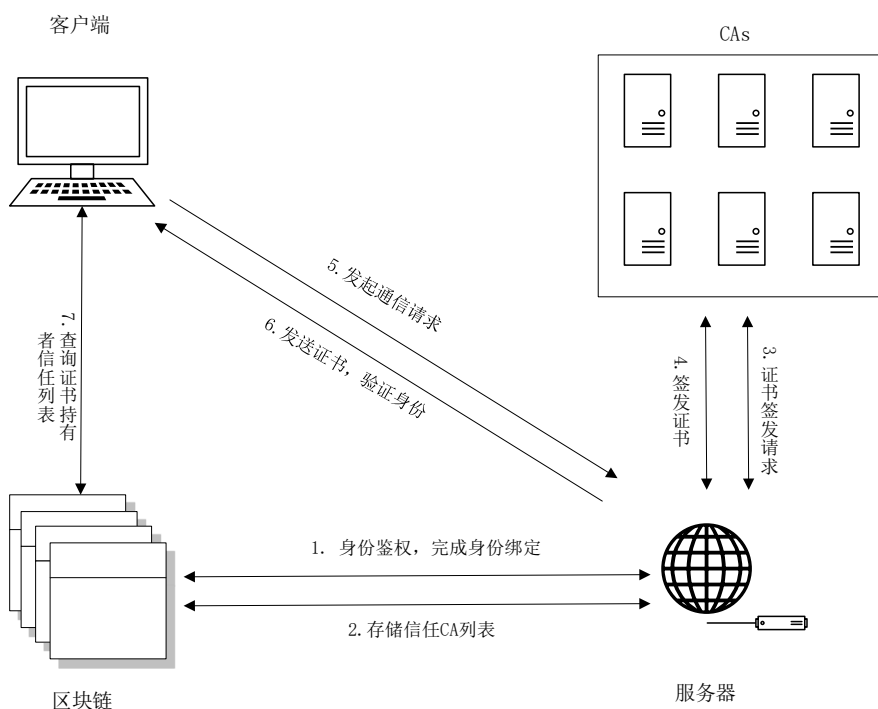


图 3.1 工作流程

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，让整个区块链上的网络节点对请求进行确认，已完成身份的鉴权；当鉴权过程完成之后，域名就可以在区块链上进行自身信任 CA 的控制，将自己信任的 CA 列表记录到区块链上的网络中，供其它实体查阅。

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 身份绑定方案设计

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。本文将提供基于验证时间和基于验证次数的身份绑定方案。

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 基于验证时间的身份绑定方案

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 *example.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:

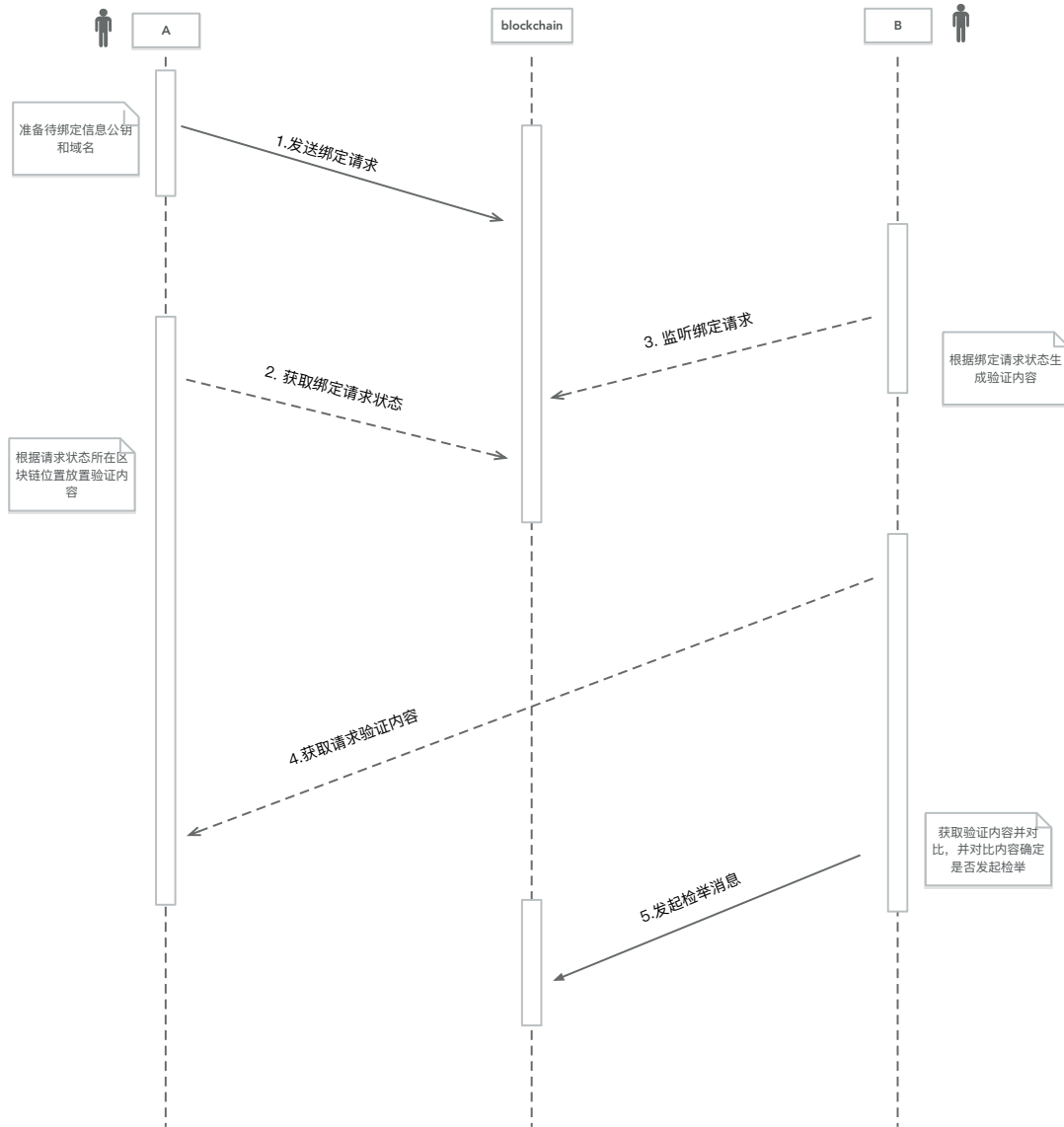


图 3.2 基于时间的方案流程图

1. A 发布绑定交易  $Tx_{req}(Pk_A, example.com)$  到区块链上;
2. 根据需绑定的信息  $(Pk_A, example.com)$ , 计算  $(Path, Chal) = F(Pk_A, example.com)$  放置在自己控制的域名下, 即访问  $example.com/Path$  即可获得到  $Vcode = Sign_{Sk_A}(Chal)$  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  $Tx_{rep}(Pk_A, example.com)$  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:

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

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

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 基于验证次数的身份绑定方案

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  $example.com$ 。
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

基于验证次数的身份绑定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： $v_1, v_2, \dots, v_k$ ，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  $Tx_{vy}(Pk_B, Pk_A, Vcode)$  完成验证。
5. 在经过 K 个验证过后，即完成绑定。

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  $Tx_{vy}$  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。

## 验证者的选取

在本方案中，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  $hash_{cmp}$ ：

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

得到基准值之后，验证节点将通过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：

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

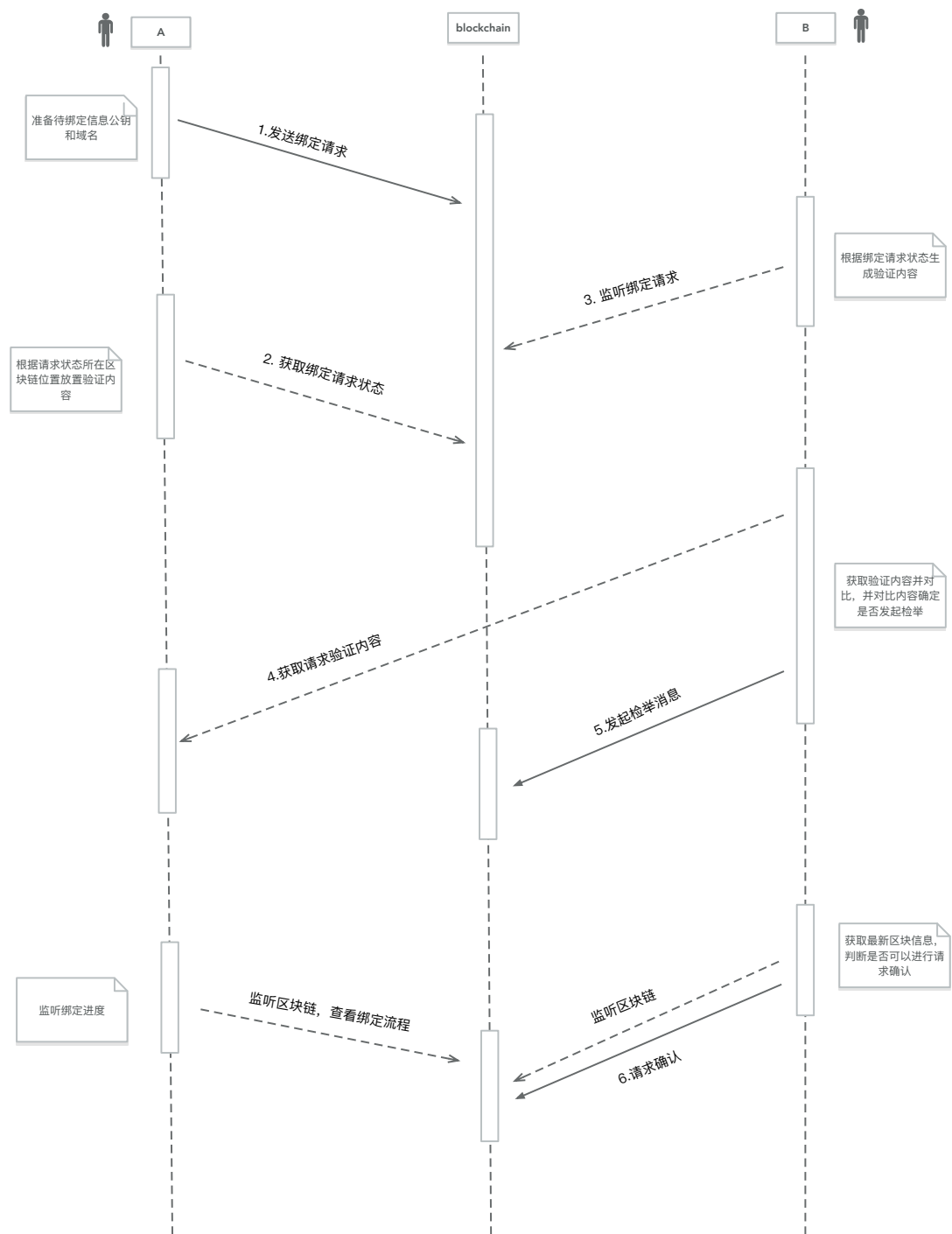


图 3.3 基于验证次数的方案流程图

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； $\Delta_t$  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。

### 3.2.3 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，需要设置检举的机制来完成这一目标。

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，其中包括自身的公钥  $Pk_C$  和绑定者公钥  $Pk_A$  和域名 *example.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，和3.2.2中选择验证节点的方式类似，式3.3中的绑定者公钥  $Pk_A$  将被发起检举者的公钥  $Pk_C$  代替，得到一个基准值，然后挑选出  $n$  个节点对该检举交易进行投票，决定该次检举的有效性。

### 3.2.4 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：

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

其中  $Verify$  是容量证明的验证函数。

### 3.2.5 奖励惩罚机制

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,将可以获得相应的奖励。所以涉及到的奖励和惩罚机制包括以下几点:

- 在发起绑定交易时,绑定者需要付出一定的代价
- 在完成验证之后,验证者将会获得相应的奖励(无论是基于验证次数方案中的验证节点或是对检举交易验证的节点)
- 在发起检举交易时,需要对该交易的发起付出一定担保,在检举交易确认后 will 退回担保
- 在检举交易被认可后,检举发起者和相应的验证者将会获得奖励;在检举交易被否决时,相应的验证者可以获得检举交易的担保费用

第一条规则的设计让身份绑定者需要付出代价,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;第三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进行;第四条规则和第二规则类似,属于激励规则,初始更多的节点加入到本系统中。同时,由于验证者在完成验证或者发现验证不通过的时候,将会获得奖励,而这些奖励不是凭空产生的,而是验证发起者所付出的相应费用。

## 3.3 安全性分析

### 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；否则，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。

### 验证时间的选择

在区块链上有确认交易的概念，以保证交易被正确的收录到了区块链中，并在很大程度上达到不可篡改的安全级别<sup>[17]</sup>。假设区块链上的交易确认块数为  $\Delta$ ，出块的平均时间为  $T_{avg}$ ，在一个  $\Delta * T_{avg}$  的时间之后，交易将被确认有效，在此期间，所有的节点都可以对该条交易的验证内容进行确认，一旦发现不符合的现象，即可发起检举交易，完成检举。





## 第四章 基于区块链的命名实体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；其后给出系统实现的模块模块设计以及系统相关的测试。

### 4.1 系统设计

本章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整体架构，从上层透视系统中的信息交互；最后给出实现本系统的方式建议和本文所采用的方法及其原因。

#### 4.1.1 系统角色

在方案设计的章节中，我们通过对现有 PKI 系统中相关实体的描述，结合方案设计的目标和相关的操作，可以得知，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标识的信任 CA 控制，在区块链上对可信 CA 列表进行管理操作。综上，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

依赖方通过浏览器在 PKI 系统完成的通信工作，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，为依赖方提供证书的进一步检查，达到其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，只需要本系统中的区块链网络中获取通信实体的可信 CA 列表即可，然后与收到的证书签发方进行对比，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，是否有足够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，需要实时的监控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控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，并尝试这些交易打包到最新的区块链中去。每当完成一个区块的创建，将会获得区块链挖矿所分配的奖励。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事，也可以从中获得相应的奖励。普通的网页浏览用户，通过浏览器进行网页浏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。

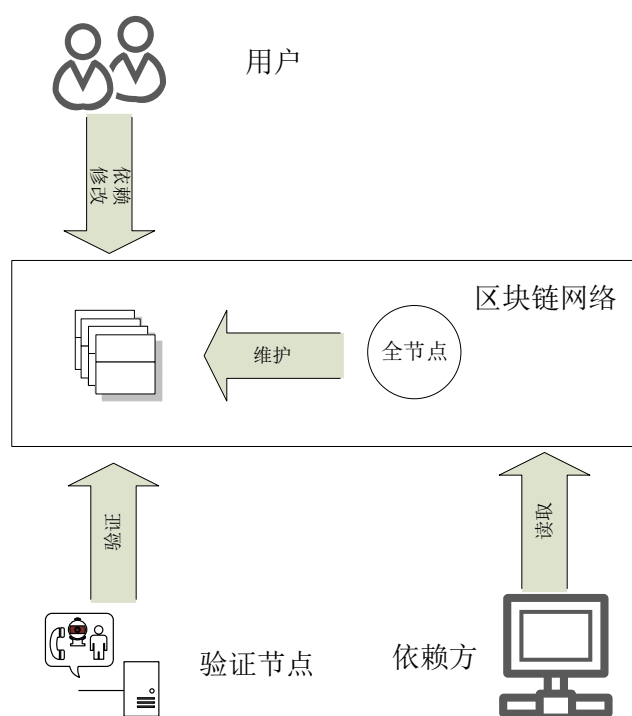


图 4.1 系统架构

### 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。在我看来，实现以上系统有两种方式：一种是利用已有的区块链平台，通过其上的智能合约来完成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的方式而言，实现起来可能更加容易，不需

要对区块链的底层去进行修改，只需要在上次的智能合约中完成相关功能和模块即可，所以难度更小；同时，基于智能合约的方式只需要将方案转换为智能合约，不需要对区块链的底层进行修改，书写的代码也要少一些，因而工程量更小。由于基于现有的区块链平台去开发智能合约，该平台一般情况下已经积累了一定的用户，很多节点已经在上面进行区块链的维护；而新开发一条链需要一段时间才能有住够多的用户加入到其中，能够保证该系统在创建初始就可以安全有效的进行下去，就系统建立的速度而言，基于智能合约方式更加高效。

但是，基于智能合约的方式也存在着相应的缺点，主要表现在性能、健壮性和全节点付出的代价。由于现有支持智能合约的公链性能都很低，并且其上还充斥这其它智能合约的交易，所以在其上开发智能合约会直接导致本系统的请求处理并不会那么及时，性能会比较低下；而创建一条新链，这可以通过一系列优化来提高其性能。另外一个问题就是整个系统的独立性并不好，很大程度上要取决于依赖的区块链平台，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，如果一个全节点想维护本系统中的账本，那么其必须要将该区块链上的所有交易记录都维护下来，包含了一些其它不是很相关的交易，大大提高了其加入本系统的代价。

本文将选择基于智能合约的方式实现，基于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

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。

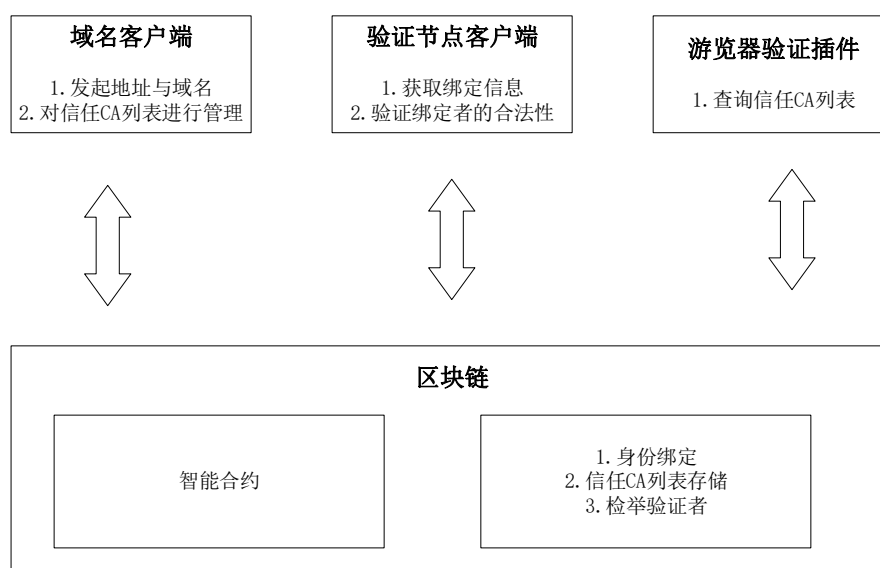


图 4.2 系统模块设计

### 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

根据智能合约模块的功能知道，本模块需要接收域名的认证请求、验证节点的验证信息和检举信息以及域名对信任 CA 列表的修改、其它实体对信任 CA 列表的查询，本模块需要提供的接口如表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

### 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

区块链上的身份是通过公私钥对来完成控制的，所以在该模块中，需要提供密钥管理功能来对区块链上的身份进行管理。其中包括密钥的生成和保存，其中公钥将作为本系统中的地址；其次是需要必要的时候使用私钥完成签名操作，包括对交易的签名和验证信息的签名。

- 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。

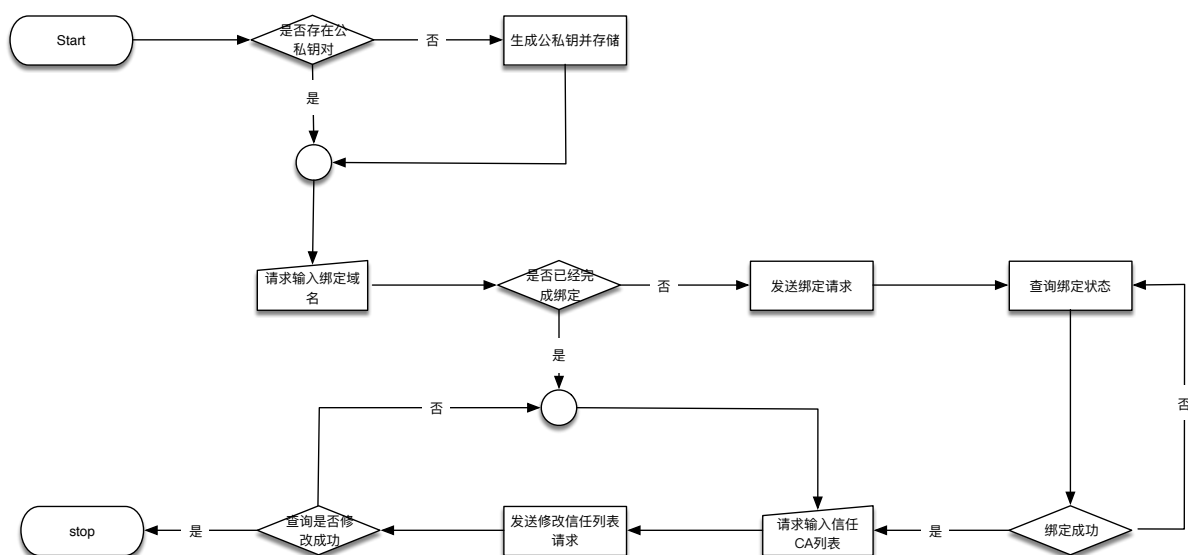


图 4.3 域名客户端模块流程

### 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。

## 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。

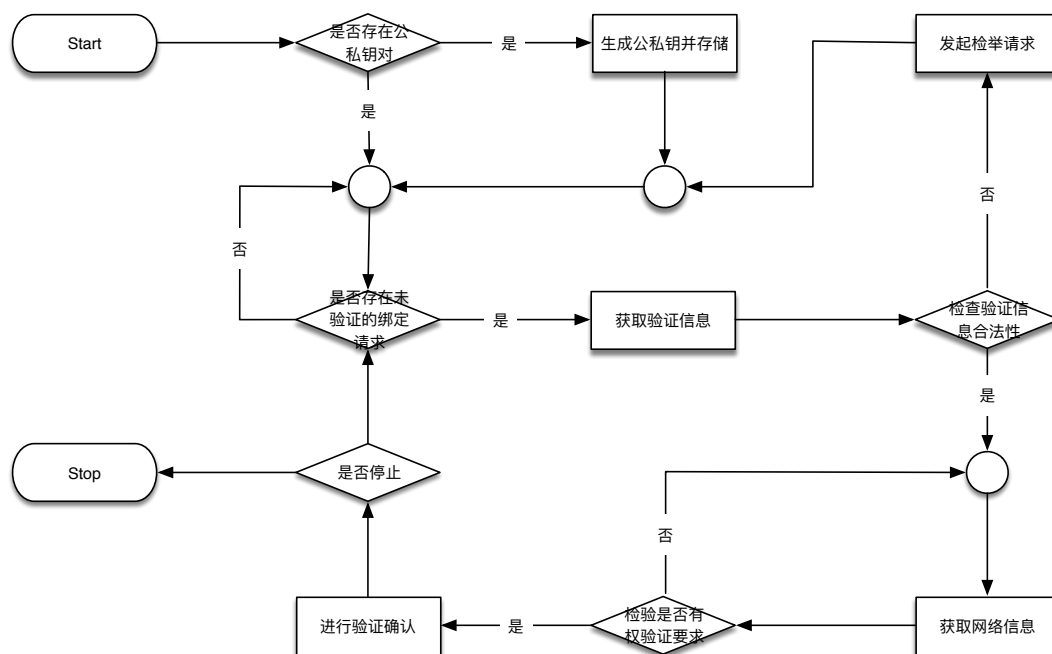


图 4.4 验证客户端模块流程

## 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

浏览器验证插件需要在浏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。

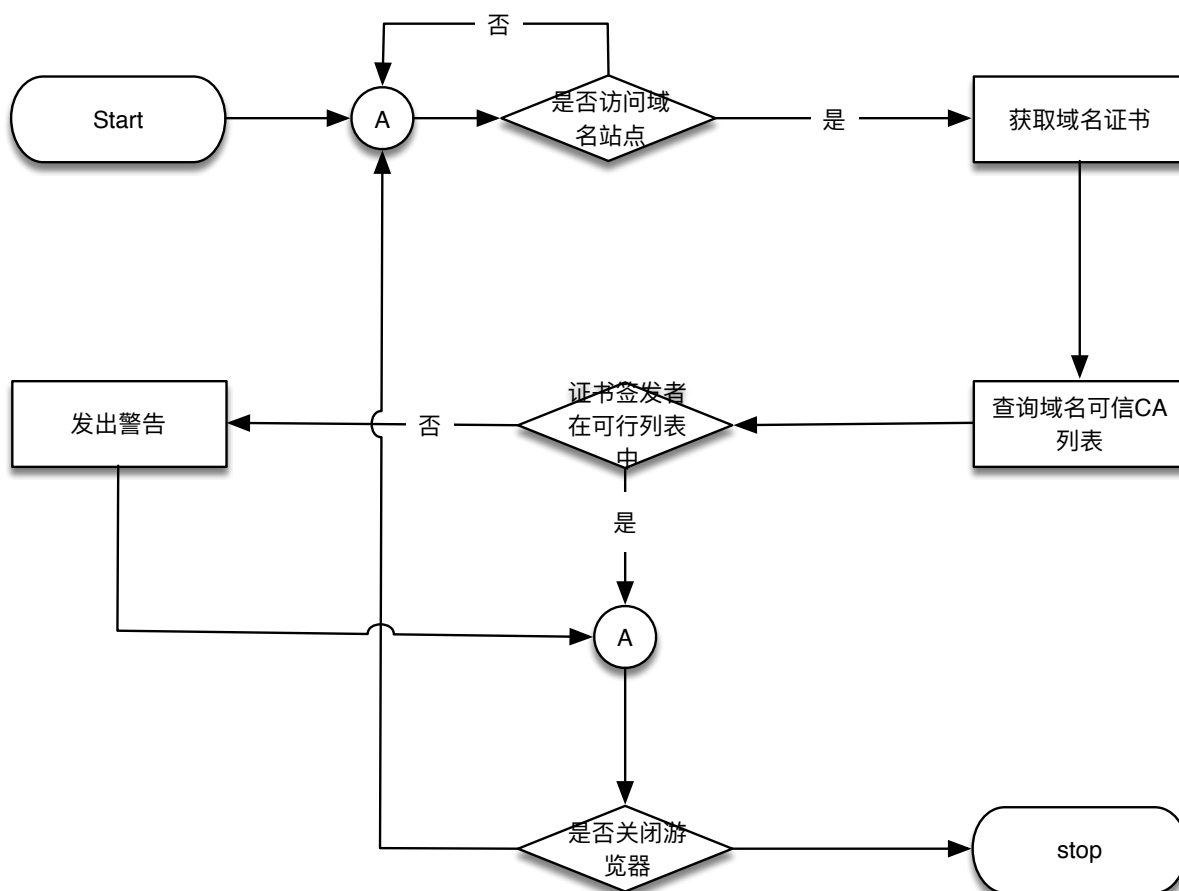


图 4.5 浏览器验证插件模块流程

## 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。

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

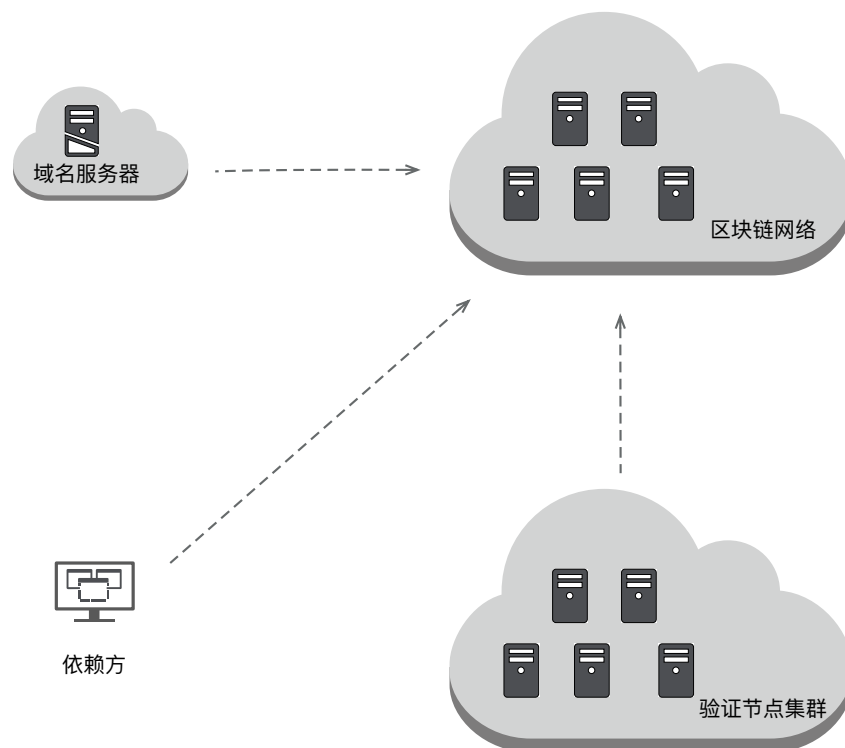


图 4.6 平台搭建

- 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。具体流程如下：

#### 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：

```

1 [9:45:59 AM] eth_sendTransaction
[9:46:00 AM] Transaction: 0
    x6fde67cd2a4419468b0764df675bad9ded7d64358dae8373410a73bb128490cf
3 [9:46:00 AM] Contract created: 0x4e60d62f8dee99d577fa8248843f0c49e4244662
[9:46:00 AM] Gas usage: 1135286
5 [9:46:00 AM] Block Number: 1
[9:46:00 AM] Block Time: Sun May 06 2018 09:55:59 GMT+0800 (CST)

```

Listing 4.1 智能合约部署日志

#### 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；运行一下命令，发起验证请求服务：

```

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

```

得到以下输出：

```

1 2018-05-09 09:48:31,351 :16 - domain-cli - sending the register request...
3 2018-05-09 09:48:31,421 :18 - domain-cli - {'status': 'added', 'txn_receipt':
    AttributeDict({'transactionHash': HexBytes('0
    xf5e7d6b5a51e7279ed8917d88d0a36aa1174856094c5ac7b4c39d82054d05816'), '
    transactionIndex': 0, 'blockHash': HexBytes('0
    x971344c24a7801b1645c47f885c27e258effbae5f1c20c45a2505237aa13dce2'), '
    blockNumber': 2, 'gasUsed': 189579, 'cumulativeGasUsed': 189579, '
    contractAddress': None, 'logs': [], 'status': 1, 'logsBloom': HexBytes('0x..0')
    })}
2018-05-09 09:48:31,421 :22 - domain-cli - The request have been accepted!
5 2018-05-09 09:48:31,421 :24 - domain-cli - waiting for register completed...
2018-05-09 09:48:31,553 :32 - domain-cli - register completed!
7 2018-05-09 09:48:31,554 :42 - domain-cli - please put 3575248931332620452 atwww.
    baidu.com/1851786109301171516.

```

```

2018-05-09 09:48:31,590 :48 - domain-cli - waiting validator to complete
  authorization (0/10)
9 ...
2018-05-09 09:50:32,660 :48 - domain-cli - waiting validator to complete
  authorization (9/10)
11 2018-05-09 09:50:37,695 :48 - domain-cli - waiting validator to complete
    authorization (10/10)
2018-05-09 09:50:37,695 :50 - domain-cli - authorization completed!

```

### 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；通过对区块链的监控，当域名 `exmaple.com` 发起验证时，将运行 `validator-cli` 程序将自动发起验证，得到以下日志：

```

2 2018-05-09 09:48:50,488 :11 - validator-cli - check new authorization request...
2018-05-09 09:48:50,573 :21 - validator-cli - get vcode 3575248931332620452 from
  www.baidu.com/1851786109301171516.
4 2018-05-09 09:48:50,573 :23 - validator-cli - the vcode is right, check passed!
2018-05-09 09:48:50,573 :26 - validator-cli - waiting for new block to confirm
  this request...
6 ...
2018-05-09 09:48:50,574 :26 - validator-cli - waiting for new block to confirm
  this request...
8 2018-05-09 09:48:50,574 :28 - validator-cli - ready to confirm :)
2018-05-09 09:48:50,574 :29 - validator-cli - sending confirmation request...
10 2018-05-09 09:48:50,630 :31 - validator-cli - {'status': 'added', 'txn_receipt':
  AttributeDict({'transactionHash': HexBytes('0
  xab0bcf480e2f9906aaa9c70675158236d6e912931759b768eb0d025a5e0c20d7'), '
  transactionIndex': 0, 'blockHash': HexBytes('0
  x1a01ec45298e0167e158ef4bea077a2e0448ffdb144377f286ed4b8ff4f3c77a'), '
  blockNumber': 3, 'gasUsed': 107323, 'cumulativeGasUsed': 107323, '
  contractAddress': None, 'logs': [], 'status': 1, 'logsBloom': HexBytes('0x0..0'
  )}))}
2018-05-09 09:48:50,630 :37 - validator-cli - confirmation request sent
  successfully.

```

### 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，通过一下来修改信任 CA 列表：

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

得到如下输出：

```
1 2018-05-09 09:50:37,750 :56 - domain-cli - Sending modify Trusted CAs request...
2018-05-09 09:50:37,750 :59 - domain-cli - {'status': 'added', 'txn_receipt':
AttributeDict({'transactionHash': HexBytes('0
x7b39338a2d30624e339eb4b3261efa5fa09378936d2cad07c17fd5406ae40729'), '
transactionIndex': 0, 'blockHash': HexBytes('0
x31a3306ce2fa4e09a9eafc5eaa3a0cf910f69b87d6c93271a2ce0df0f26b955f'), '
blockNumber': 14, 'gasUsed': 46593, 'cumulativeGasUsed': 46593, '
contractAddress': None, 'logs': [], 'status': 1, 'logsBloom': HexBytes('0x0...0
'))})
3 2018-05-09 09:50:37,750 :63 - domain-cli - trusted CAs be modified to Symantec
Corporation
```

## 5. 信任列表查询

在前面的操作中，我们将 [www.baidu.com](http://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。在浏览器端安装 chrome 插件之后，访问 [www.baidu.com](http://www.baidu.com)，插件显示内容如图4.7所示。

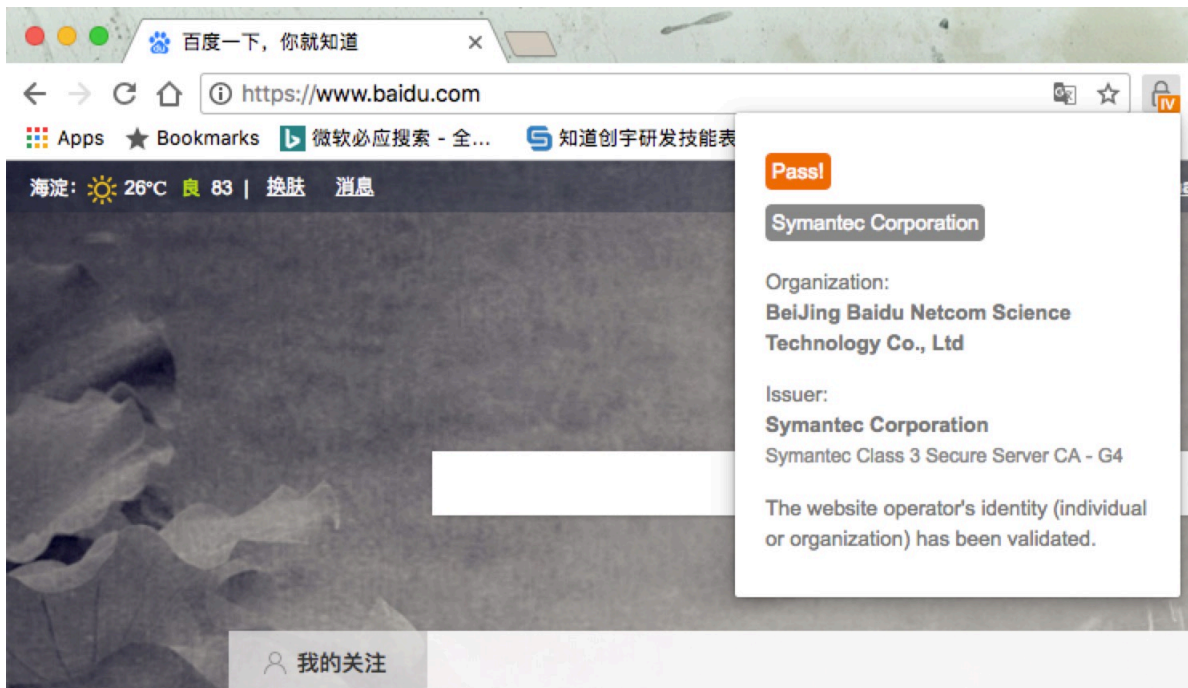


图 4.7 验证通过实例

在上图中，第一行表示是否通过验证，本例中通过验证；第二行表示设置的信任 CA，后面内容为证书相关的信息。

由于没有对 `www.google.com` 进行绑定操作，将无法查询得到信任 CA 列表，所以其无法通过验证，得到结果如图4.8所示。

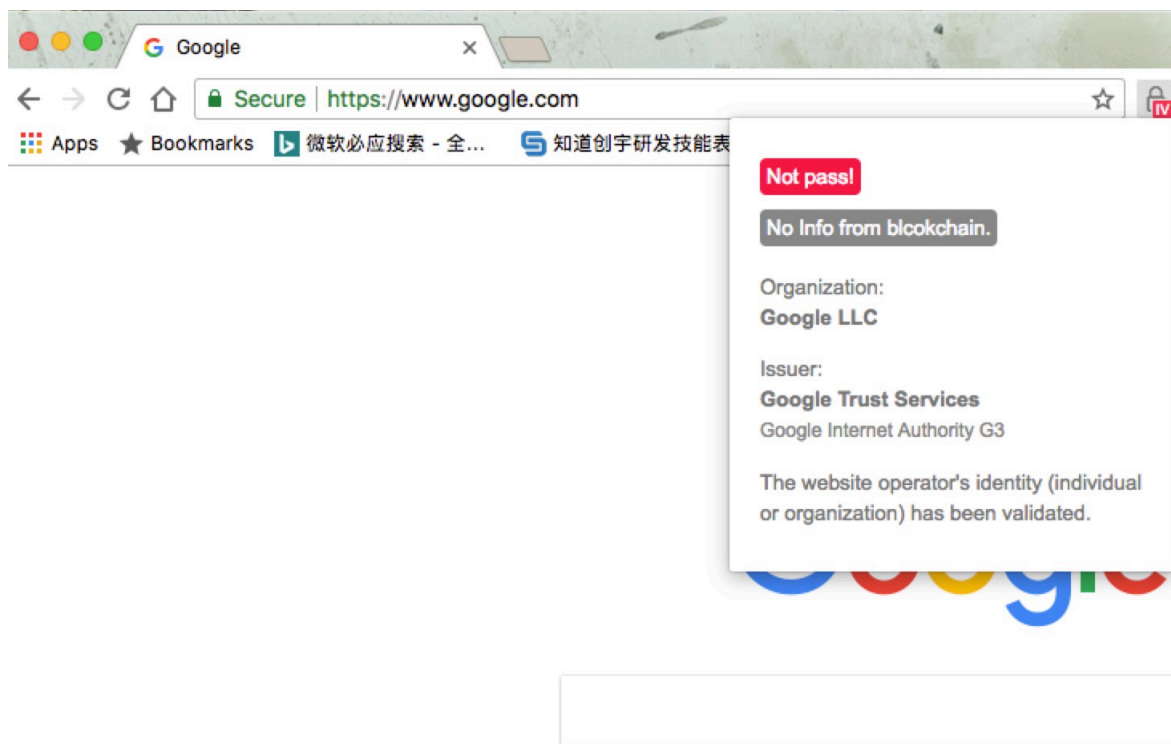


图 4.8 验证未通过实例

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已经信任 CA 列表的绑定；并且通过浏览器插件，在访问网页是可以获得相关的验证结果。

## 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与现有的 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 and 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] J Ronald Prins and Business Unit Cybercrime. “DigiNotar Certificate Authority breach’ Operation Black Tulip’ ” [J]. *Fox-IT*, November, **2011**.
- [2] I Risti. *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications* [M]. Londres, Inglaterra: Feisty Duck, **2014**.
- [3] Paul Ducklin. “The TURKTRUST SSL certificate fiasco-what really happened, and what happens next” [J]. *Naked Security. SOPHOS*, **2013**, 8.
- [4] Dan Wendlandt, David G Andersen and Adrian Perrig. “Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing.” [C]. In: *USENIX Annual Technical Conference*, **2008**: 321–334.
- [5] Mansoor Alicherry and Angelos D Keromytis. “Doublecheck: Multi-path verification against man-in-the-middle attacks” [C]. In: *Computers and communications, 2009. iscc 2009. ieee symposium on*, **2009**: 557–563.
- [6] *Convergence*. <https://convergence.io/>.
- [7] M Modell, A Barz, G Toth *et al*. “Certificate patrol” [J]. Online at <https://addons.mozilla.org/en-US/firefox/addon/certificate-patrol>, **2014**.
- [8] Christopher Soghoian and Sid Stamm. “Certified lies: Detecting and defeating government interception attacks against SSL (short paper)” [J]. In: *International Conference on Financial Cryptography and Data Security*, **2011**: 250–259.
- [9] Chris Evans, Chris Palmer and Ryan Sleevi. *Public key pinning extension for HTTP* [R], **2015**.
- [10] Richard L Barnes. “DANE: Taking TLS authentication to the next level using DNSSEC” [J]. *IETF Journal*, **2011**, 7(2).
- [11] P Eckersley. “Internet-draft: Sovereign key cryptography for internet domains” [J]. **2012**.
- [12] Ben Laurie, Adam Langley and Emilia Kasper. *Certificate transparency* [R], **2013**.
- [13] Mark Dermot Ryan. “Enhanced Certificate Transparency and End-to-End Encrypted Mail.” [C]. In: *NDSS*, **2014**.
- [14] Vincent Cheval, Mark Ryan and Jiangshan Yu. “DTKI: a new formalized PKI with no trusted parties” [J]. *arXiv preprint arXiv:1408.1023*, **2014**.
- [15] Tiffany Hyun-Jin Kim, Lin-Shung Huang, Adrian Perrig *et al*. “Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure” [C]. In: *Proceedings of the 22nd international conference on World Wide Web*, **2013**: 679–690.
- [16] David Basin, Cas Cremers, Tiffany Hyun-Jin Kim *et al*. “ARPKI: attack resilient public-key infrastructure” [C]. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, **2014**: 382–393.
- [17] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system” [J]. **2008**.

- [18] Vitalik Buterin *et al.* “Ethereum white paper” [J]. *GitHub repository*, **2013**.
- [19] Marco Baldi, Franco Chiaraluce, Emanuele Frontoni *et al.* “Certificate Validation Through Public Ledgers and Blockchains.” [C]. In: *ITASEC*, **2017**: 156–165.
- [20] Stephanos Matsumoto and Raphael M Reischuk. “IKP: Turning a PKI Around with Blockchains.” [J]. *IACR Cryptology ePrint Archive*, **2016**, 2016: 1018.
- [21] Conner Fromknecht, Dragos Velicanu and Sophia Yakoubov. “A Decentralized Public Key Infrastructure with Identity Retention.” [J]. *IACR Cryptology ePrint Archive*, **2014**, 2014: 803.
- [22] Denis Rystsov. “Nidaba: a distributed scalable PKI with a stable price for certificate operations” [J].
- [23] Mustafa Al-Bassam. “SCPki: a smart contract-based PKI and identity system” [C]. In: *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, **2017**: 35–40.
- [24] Joel Weise. “Public key infrastructure overview” [J]. *Sun BluePrints OnLine*, August, **2001**: 1–27.
- [25] Martn Abadi, Andrew Birrell, Ilya Mironov *et al.* “Global Authentication in an Untrustworthy World.” [C]. In: *HotOS*, **2013**.
- [26] Dave Cooper. “Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile” [J]. **2008**.
- [27] Michael Myers, Rich Ankney, Ambarish Malpani *et al.* X. 509 Internet public key infrastructure online certificate status protocol-OCSP [R], **1999**.
- [28] Emin Topalovic, Brennan Saeta, Lin-Shung Huang *et al.* “Towards short-lived certificates” [J]. *Web 2.0 Security and Privacy*, **2012**.
- [29] Jakob Schlyter and Paul Hoffman. “The DNS-based authentication of named entities (DANE) transport layer security (TLS) protocol: TLSA” [J]. **2012**.
- [30] P Eckersley. *Sovereign key cryptography for Internet domains*. Website, Nov. 2011 [J].
- [31] Ben Laurie and Emilia Kasper. “Revocation transparency” [J]. *Google Research*, September, **2012**.
- [32] Joseph Bonneau, Arvind Narayanan, Andrew Miller *et al.* “Mixcoin: Anonymity for Bitcoin with accountable mixes” [C]. In: *International Conference on Financial Cryptography and Data Security*, **2014**: 486–504.
- [33] Ahmed Kosba, Andrew Miller, Elaine Shi *et al.* “Hawk: The blockchain model of cryptography and privacy-preserving smart contracts” [C]. In: *Security and Privacy (SP), 2016 IEEE Symposium on*, **2016**: 839–858.
- [34] John R Douceur. “The sybil attack” [C]. In: *International workshop on peer-to-peer systems*, **2002**: 251–260.
- [35] Andreas M Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies* [M]. " O'Reilly Media, Inc.", **2014**.
- [36] Pavel Vasin. “Blackcoin’ s proof-of-stake protocol v2” [J]. URL: <https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf>, **2014**.
- [37] Giuseppe Ateniese, Ilario Bonacina, Antonio Faonio *et al.* “Proofs of space: When space is of the essence” [C]. In: *International Conference on Security and Cryptography for Networks*, **2014**: 538–557.

- [38] Nick Szabo. “*Smart contracts: building blocks for digital markets*” [J]. *EXTROPY: The Journal of Transhumanist Thought*, (16), **1996**.
- [39] Zibin Zheng, Shaoan Xie, Hong-Ning Dai *et al.* “*Blockchain challenges and opportunities: A survey*” [J]. *Work Pap.*–2016, **2016**.



## 致谢



## 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