



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

X 年 X 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

pkuthss 文档模版最常见问题:

`\cite`、`\parencite` 和 `\supercite` 三个命令分别产生未格式化的、带方括号的和上标且带方括号的引用标记: 1, [2]^[1,2]。

若要避免章末空白页, 请在调用 `pkuthss` 文档类时加入 `openany` 选项。

如果编译时不出参考文献, 请参考 `texdoc pkuthss` “问题及其解决”一章“其它可能存在的问题”一节中关于 `biber` 的说明。

关键词: 其一, 其二

Test Document

Anran Li (Software and Theory of Computer)

Directed by Prof. ZhiGuan

ABSTRACT

Test of the English abstract.

KEYWORDS: First, Second

目录

序言	1
第一章 章节	3
第二章 章节	5
第三章 章节	7
第四章 基于区块链的命名实体鉴权方案设计与实现	9
4.1 系统设计	9
4.1.1 系统角色	9
4.1.2 系统架构	10
4.2 身份绑定方案	11
4.2.1 基于验证时间的身份绑定方案	11
4.2.2 基于验证次数的身份绑定方案	12
4.2.3 奖励惩罚机制	13
4.2.4 安全性分析	14
4.3 模块实现	15
4.3.1 智能合约	15
4.3.2 域名客户端	15
4.3.3 验证节点客户端	15
4.3.4 浏览器验证插件	15
第五章 章节	17
结论	19
参考文献	21
附录 A 附件	23
致谢	25
北京大学学位论文原创性声明和使用授权说明	27

序言

pkuthss 文档模版最常见问题:

`\cite`、`\parencite` 和 `\supercite` 三个命令分别产生未格式化的、带方括号的和上标且带方括号的引用标记: 1, [2]^[1,2]。

若要避免章末空白页, 请在调用 *pkuthss* 文档类时加入 `openany` 选项。

如果编译时不出参考文献, 请参考 `texdoc pkuthss` “问题及其解决”一章“其它可能存在的问题”一节中关于 `biber` 的说明。

第一章 章节

pkuthss 文档模版最常见问题:

`\cite`、`\parencite` 和 `\supercite` 三个命令分别产生未格式化的、带方括号的和上标且带方括号的引用标记: 1, [2]^[1,2]。

若要避免章末空白页, 请在调用 *pkuthss* 文档类时加入 `openany` 选项。

如果编译时不出参考文献, 请参考 `texdoc pkuthss` “问题及其解决”一章“其它可能存在的问题”一节中关于 `biber` 的说明。

第二章 章节

pkuthss 文档模版最常见问题:

`\cite`、`\parencite` 和 `\supercite` 三个命令分别产生未格式化的、带方括号的和上标且带方括号的引用标记: 1, [2]^[1,2]。

若要避免章末空白页, 请在调用 *pkuthss* 文档类时加入 `openany` 选项。

如果编译时不出参考文献, 请参考 `texdoc pkuthss` “问题及其解决”一章“其它可能存在的问题”一节中关于 `biber` 的说明。

第三章 章节

pkuthss 文档模版最常见问题:

`\cite`、`\parencite` 和 `\supercite` 三个命令分别产生未格式化的、带方括号的和上标且带方括号的引用标记: 1, [2]^[1,2]。

若要避免章末空白页, 请在调用 *pkuthss* 文档类时加入 `openany` 选项。

如果编译时不出参考文献, 请参考 `texdoc pkuthss` “问题及其解决”一章“其它可能存在的问题”一节中关于 `biber` 的说明。

第四章 基于区块链的命名实体鉴权方案设计与实现

本章节将阐述基于区块链的命名实体鉴权方案，首先从系统设计的整体架构出发，描述本系统中包含的系统角色和架构；然后对鉴权方案进行详细阐述，并理论论证鉴权方案的有效性和可靠性；最后对模块实现的进行阐述。

4.1 系统设计

本部分将从系统涉及的角色和模块对本系统的整体架构进行描述。

4.1.1 系统角色

针对证书申请和使用流程，在 PKI 系统中涉及的角色包含以下三类：

- 证书授权者（CA）：在 PKI 系统充当证书签发和管理的角色。
- 证书申请者：证书申请者在 PKI 系统中作为证书的持有者，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，证书授权者作为证书的签发方，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到决定权；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，保证未被允许签发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：

- 证书授权者（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：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：在本方案中，验证者可以通过区块链查询身份确认实体的信任 CA 列表，保证证书的签发是合法的。
- 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，通过这些节点完成对区块链上证书申请者的身份认证。

4.1.2 系统架构

本系统依托于现有的 PKI 体系，并基于区块链作为存储平台，在其上进行证书申请者信任 CA 列表的存储，系统中的各个角色都要通过本系统中的区块链进行数据交流。更为重要的是，域名站点需要在此基础上进行身份认证，作为信任建立的根基。系统的总体设计如图4.1所示：

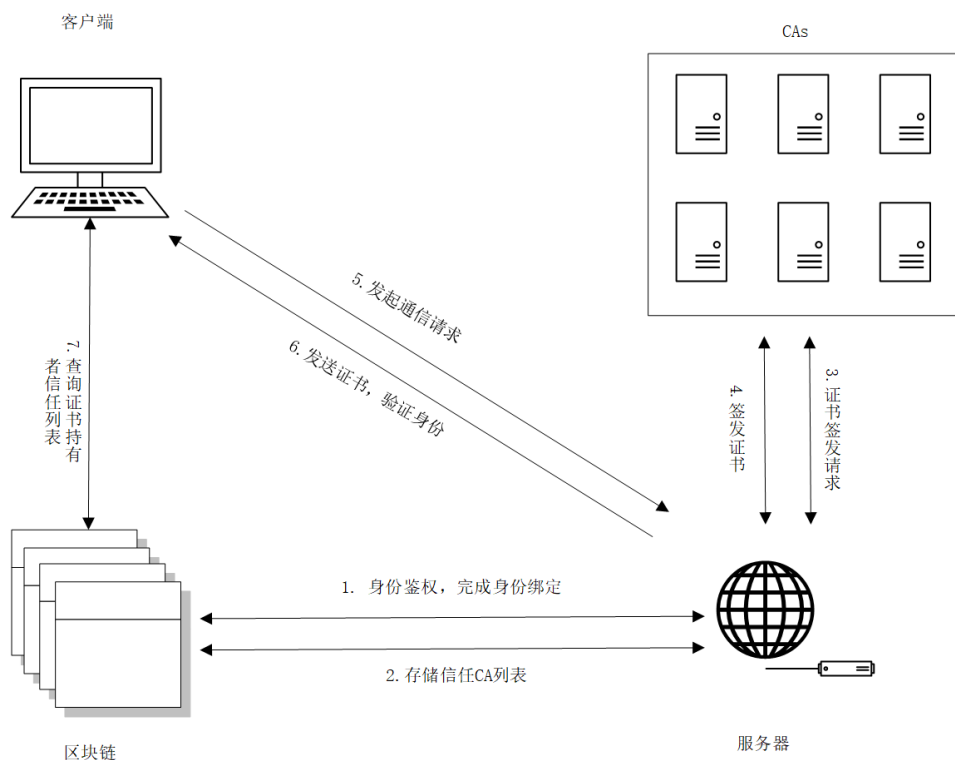


图 4.1 this is for test

在本系统中，域名站点作为证书申请者，向区块链发起身份认证请求，当交易被记录到区块链上之后，域名站点根据交易提供验证信息，供验证节点对其身份进行验证；完成验证之后，域名即可将自己信任的 CA 列表发布到区块链上，并向自己信任的 CA 进行证书申请；完成申请后，即可将证书用在与用户通信的过程中，进行身份证明，此时证书验证者将通过区块链进行信任列表查询，判断该证书是否是合理签发的。

4.2 身份绑定方案

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特性的底层技术，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的，之前需要完成区块链地址（或者公钥）和域名本身的绑定，完成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。

4.2.1 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 Pr_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 Pr_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} ：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。

流程

绑定的具体流程包含以下四步：

1. A 发布绑定交易 $Tx_{req}(Pk_A, example.com)$ 到区块链上；
2. 根据需要的绑定信息 $(Pk_A, example.com)$ ，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 $Vcode = Sign_{Pr_A}(Chal)$ 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 $Chal$:

$$Path_{128} || Chal_{128} = sha256(Pk_A || example.com) \quad (4.1)$$

$$Vcode = Sign_{Pr_A}(Chal) \quad (4.2)$$

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。

4.2.2 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省份绑定。

假设

假设绑定者 A 拥有公钥 Pk_A , 私钥 Pr_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 Pr_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} : 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,具体流程如下:

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 $(Path, Chal) = F(Info_{block})$ 放置在自己控制的域名下, 及访问 $example.com/Path$ 即可获取到 $Vcode = Sign_{Pr_A}(Chal)$ 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 $Tx_{vfy}(Pk_B, Pk_A, C)$ 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, 然后通过公式函数获得(4.1)和(4.2)计算得到 $Path$ 和 $Vcode$ 的值放置到自己的服务器上; 验证者通过(4.1)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_{vfy} 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, 需要合理的选取验证者, 避免节点为了保证自己的利益, 将其他节点的验证交易滞后提交到区块链中的情况; 另外一方面, 如果验证节点不是经过筛选, 而是每个节点都可以对绑定身份进行验证的话, 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; 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, 从中获得更多的获益。

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, 保证验证可以安全有效的进行。当绑定信息被发布到区块链上后, 根据绑定信息 Pk_A 和 $example.com$, 将其装换为基准公钥 Pk_{cmp} :

$$Pk_{std} = sha - 512(Pk_A || example.com) \quad (4.3)$$

在得到基准公钥后, 验证节点计算公钥 Pk_v 与基准公钥 Pk_{cmp} 之间的海明距离, 如果其距离小于某一个值 d , 就具有验证该次绑定的权力, 可以向区块链提交验证结果。为使得能够有住够多的验证者, 让身份绑定过程可以更加高效的进行, 随着时间的推移, 对海明距离的约束 d 将会逐渐变大。

[需要展开一下]

4.2.3 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三点：

- 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（仅在基于验证次数的方案中使用）
- 在发现非法绑定时，验证者将可以发起交易获得相应奖励

如上所述，后两条为奖励机制，其目的是吸引更多的人加入到本系统之中，使得身份绑定可以顺利的完成，并具有较高的真实性；而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。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，而是绑定者所付出的相应费用。

4.2.4 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(4.1)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。

验证交易真实有效

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。

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【这里应该需要添加引用】。假设区块链上的交易确认块数为 Δ ，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。

验证次数的选择

4.3 模块实现

本文以以太坊为例作为区块链底层架构，利用其智能合约完成以上方案的逻辑实现，PKI 各个实体通过与智能合约之间交互，存储并获取相关数据，完成对信任 CA 列表的读写，并最终完成对证书有效性的进一步确认。

根据上面的设计，以以太坊作为区块链基础，本系统分为四个模块进行实现：智能合约、域名客户端、验证节点客户端、浏览器验证插件。

[插入一个图?]

4.3.1 智能合约

智能合约作为可以在区块链上实现其他逻辑操作的一种工具，在本方案的实现中，将给予以太坊作为底层区块链基础，在其上完成符合以上方案的智能合约，完成区块链上的数据存储和逻辑操作。

4.3.2 域名客户端

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，主要包括以下操作：

- 发起身份绑定请求
- 修改信任 CA 列表

4.3.3 验证节点客户端

验证节点是本系统中身份鉴权的重要组成部分，是身份有效性的保证，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。

4.3.4 浏览器验证插件

浏览器 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。

第五章 章节

pkuthss 文档模版最常见问题:

`\cite`、`\parencite` 和 `\supercite` 三个命令分别产生未格式化的、带方括号的和上标且带方括号的引用标记: 1, [2]^[1,2]。

若要避免章末空白页, 请在调用 *pkuthss* 文档类时加入 `openany` 选项。

如果编译时不出参考文献, 请参考 `texdoc pkuthss` “问题及其解决”一章“其它可能存在的问题”一节中关于 `biber` 的说明。

结论

pkuthss 文档模版最常见问题:

`\cite`、`\parencite` 和 `\supercite` 三个命令分别产生未格式化的、带方括号的和上标且带方括号的引用标记: 1, [2]^[1,2]。

若要避免章末空白页, 请在调用 *pkuthss* 文档类时加入 `openany` 选项。

如果编译时不出参考文献, 请参考 `texdoc pkuthss` “问题及其解决”一章“其它可能存在的问题”一节中关于 `biber` 的说明。

参考文献

- [1] Author. “*Title*” [J]. *Journal*, 2014-04-01.
- [2] 作者。“标题” [J]。期刊，2014-04-01。

附录 A 附件

pkuthss 文档模版最常见问题:

`\cite`、`\parencite` 和 `\supercite` 三个命令分别产生未格式化的、带方括号的和上标且带方括号的引用标记: 1, [2]^[1,2]。

若要避免章末空白页, 请在调用 `pkuthss` 文档类时加入 `openany` 选项。

如果编译时不出参考文献, 请参考 `texdoc pkuthss` “问题及其解决”一章“其它可能存在的问题”一节中关于 `biber` 的说明。

致谢

pkuthss 文档模版最常见问题:

`\cite`、`\parencite` 和 `\supercite` 三个命令分别产生未格式化的、带方括号的和上标且带方括号的引用标记: 1, [2]^[1,2]。

若要避免章末空白页, 请在调用 *pkuthss* 文档类时加入 `openany` 选项。

如果编译时不出参考文献, 请参考 *texdoc pkuthss* “问题及其解决”一章“其它可能存在的问题”一节中关于 `biber` 的说明。

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