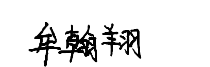


|  |  |
| --- | --- |
| 题 目： | 一种具有隐私保护的多方授 |
|  | 权系统的设计与实现 |

|  |  |
| --- | --- |
| 学 号： | 57119321 |
| 姓 名： | 牟翰翔 |
| 学 院： | 网络空间安全学院 |
| 专 业： | 网络空间安全 |
| 指导教师： | 韩金广 |
| 起止日期： | 2023.01.01-2023.06.05 |

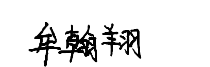
东南大学毕业（设计）论文独创性声明

本人声明所呈交的毕业（设计）论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

论文作者签名： 日期： 2023 年 5 月 1 日

东南大学毕业（设计）论文使用授权声明

东南大学有权保留本人所送交毕业（设计）论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权东南大学教务处办理。

论文作者签名： 导师签名：

日期： 2023 年 5 月 1 日 日期： 年 月 日

# 摘 要

现如今的许多网络应用场景需要通过多个参与方共同验证和授权来完成特定的任务，如金融服务领域中的交易验证和审计、医疗保健领域中的电子病历共享、物联网领域中的智能设备管理等，然而多方授权系统存在潜在的身份泄露问题，威胁着人们的隐私安全。因此，研究如何实现具有隐私保护的多方授权是具有实际意义的研究课题。

本文重点考虑如何设计与实现一种具有隐私保护的多方授权系统，通过合理选取隐私保护协议和多方授权机制，确保隐私保护的安全性和实现可靠的多方授权，并创新性地将两者结合设计新的协议，实现设计目标。在梳理和总结相关领域研究成果的基础上，对目标系统的设计与实现展开如下研究工作：一、方案设计及实现验证。为了满足要求可靠多方授权和安全匿名认证的应用场景，采取了一种基于双线性群配对的多重签名方案实现多方授权机制，并通过零知识成员集合证明技术实现用户对授权机构的匿名身份认证。在实现阶段，基于C++ PBC库编程，对服务请求、身份验证、多重签名等模块进行实现并测试其可用性。二、系统性能与效率分析。通过改变实验组间授权机构数量，测量不同情况下各模块运行时间和中间参数。三、本文分析了系统的可靠性和安全性。该系统可以实现多方授权，且能够提供安全的匿名认证，防止身份泄露。

关键词：多方授权，隐私保护，多重签名，匿名认证

# ABSTRACT

Nowadays many network application scenarios require verification and authorization by multiple parties to complete specific tasks, such as transaction verification and auditing in the financial services field, electronic medical record sharing in the healthcare field, and smart devices in the Internet of Things field. However, identity leakage problem exists in the multi-party authorization systems, threatening people's privacy security. Therefore, it is interesting to design a privacy-preserving multi-party authorization system.

This paper focuses on how to design and implement a privacy-preserving multi-party authorization system. By reasonably selecting a privacy-preserving protocol and a multi-party authorization mechanism, the security of privacy protection and reliable multi-party authorization are ensured, and a new protocol of combination of the two is innovatively designed to achieve the design goals. Based on exsiting research work, the following research work is carried out on the design and realization of the target system: First, scheme design and realization for verification. In order to meet the application scenarios that require reliable multi-party authorization and secure anonymous authentication, a multi-signature scheme based on bilinear group pairing is adopted to realize the multi-party authorization mechanism, and the user's anonymous identity authentication to the authority is realized through the zero-knowledge set membership proof technology. In the implementation stage, based on C++ PBC library programming, modules such as service request, identity verification, and multi-signature are implemented and their usability is tested. Second, system performance and efficiency analysis. By changing the number of authorized institutions between experimental groups, the running time and intermediate parameters of each module are measured under different conditions. Third, this paper analyzes the reliability and security of the system. The system realizes multi-party authorization, and provides secure anonymous authentication to prevent identity leakage.

KEY WORDS: Multi-party authorization, Privacy protection, Multi-signature, Anonymous authentication

# 目 录

[摘 要 I](#_Toc137227325)

[ABSTRACT II](#_Toc137227326)

[目 录 III](#_Toc137227327)

[第一章 绪论 1](#_Toc137227328)

[1.1课题背景和意义 1](#_Toc137227329)

[1.2研究现状 1](#_Toc137227330)

[1.3本文研究内容 3](#_Toc137227331)

[1.4章节安排 4](#_Toc137227332)

[第二章 基础知识与安全模型 5](#_Toc137227333)

[2.1基础知识 5](#_Toc137227334)

[2.1.1 双线性群 5](#_Toc137227335)

[2.1.2 多重签名 5](#_Toc137227336)

[2.1.3 零知识证明和协议 6](#_Toc137227337)

[2.1.4 集合成员证明 7](#_Toc137227338)

[2.1.5 计算困难问题 7](#_Toc137227339)

[2.2多重签名 8](#_Toc137227340)

[2.2.1 一种紧凑的多重签名 8](#_Toc137227341)

[2.2.2 一种基于签名的零知识证明的集合成员证明 10](#_Toc137227342)

[第三章 方案设计 12](#_Toc137227343)

[3.1整体构思 12](#_Toc137227344)

[3.2方案描述 14](#_Toc137227345)

[3.2.1 系统初始化阶段 14](#_Toc137227346)

[3.2.2 用户注册阶段 15](#_Toc137227347)

[3.2.3 服务请求阶段 16](#_Toc137227348)

[3.2.4 验证阶段 17](#_Toc137227349)

[3.3安全性分析 18](#_Toc137227350)

[3.3.1 多方授权 18](#_Toc137227351)

[3.3.2 匿名认证 18](#_Toc137227352)

[第四章 系统实现与实验分析 19](#_Toc137227353)

[4.1系统的实现 19](#_Toc137227354)

[4.1.1 实现方法 19](#_Toc137227355)

[4.1.2 基于角色的访问控制 19](#_Toc137227356)

[4.1.3 C++编程的优势 20](#_Toc137227357)

[4.1.4 PBC库的介绍 21](#_Toc137227358)

[4.1.5 哈希函数 22](#_Toc137227359)

[4.1.6 对数据的测量 23](#_Toc137227360)

[4.2实验测量与数据分析 24](#_Toc137227361)

[第五章 总结与展望 32](#_Toc137227362)

[5.1工作总结 32](#_Toc137227363)

[5.2工作展望 32](#_Toc137227364)

[参考文献 33](#_Toc137227365)

[附录A 仿真系统源代码 35](#_Toc137227366)

[致 谢 54](#_Toc137227367)

# 第一章 绪论

## 1.1课题背景和意义

随着信息技术的发展，人们需要在网络上能够通过多个参与方共同验证和授权来完成特定的任务，例如金融服务领域中的交易验证和审计、医疗保健领域中的电子病历共享、物联网领域中的智能设备管理等，即实现多方授权[1]。如今多方授权系统的应用已经非常成熟，但是其中牵扯到的身份泄露问题威胁着人们的隐私安全。例如，在金融领域中，金融机构需要对多个客户的共同资金的支出进行验证和审计，这需要多个客户的共同授权以确保交易的合法性。然而，在多方授权的交易验证过程中可能会暴露请求授权的交易者的身份信息，造成隐私泄露。

一旦交易者的身份信息遭到泄露，可能会被商家、机构等滥用，甚至被攻击者利用进行身份伪造，直接威胁到了隐私安全，给个人、组织和社会带来许多不利影响。不仅如此，在许多其他的应用场景中，多方授权系统都有可能发生类似的身份泄露问题。

因此，研究具有隐私保护的多方授权系统变得愈加重要。这种系统可以在多个参与方之间实现保护隐私安全的多方授权，有助于保护个人隐私和商业机密，推动数字社会的发展。

## 1.2研究现状

多方授权是一种权限控制机制，它允许多个参与者在分布式系统中协同完成授权操作，同时保护敏感信息的安全。在现代网络环境中，多方授权得到了广泛应用，例如在物联网中涉及到数量庞大的设备和传感器，多方授权技术可以用于物联网场景中的设备认证和访问控制，确保设备之间的数据传输安全[23]。

文献[1]介绍了多方授权的基本原理和现有的多方授权方案。为了实现多方授权，需要建立一个访问控制系统，该系统可以定义访问控制策略和访问控制规则，确保只有经过授权的参与者才能访问资源。多方授权中的访问控制系统可以使用各种访问控制技术，包括基于角色的访问控制、基于属性的访问控制、基于策略的访问控制和基于身份的访问控制。根据访问控制方法的不同，我们调查了多方授权机制的研究现状。

基于角色的访问控制在多方参与的环境中可以通过分配不同角色给不同的参与方，来实现对资源的授权管理。如文献[2]提出了一个基于角色的访问控制模型和多级安全性方案，用于内联网信息系统的保护，该方案使用了基于角色的访问控制模型中的角色继承机制。参与者可以继承其他参与者拥有的角色，并通过合作获得访问加密数据的权限，以解决多方参与者的合作问题。文献[3]介绍了分布式配置管理过程中多方授权和冲突解决的方法。该论文提出了一种基于角色的多方授权机制，它通过将授权和角色之间的映射与配置文件关联起来，实现了对不同角色的授权管理；冲突解决算法则使用一种称为“最短路径合并”的方法，通过将多个版本的配置文件合并成一份具有最少冲突的版本，解决了多个参与者同时修改配置文件时可能出现的冲突。

基于属性的访问控制在多方参与的环境中通过使用属性来定义访问策略和授权规则，实现对资源的授权管理。文献[4]就提出了一种基于属性的加密技术，用于实现对加密数据的细粒度访问控制。该技术将访问控制策略和加密密钥关联起来，对于多方授权问题，该方案使用了门限密钥分发技术，将解密密钥分成多个部分，分发给多个参与者。只有当满足访问控制策略所要求的属性集合的参与者集合满足门限条件时，才能通过合作将密钥重构出来并对加密数据进行解密，确保了多方授权安全可信。

基于策略的访问控制是一种用于管理多方参与者之间访问控制的方案。其中，访问策略定义了参与者可以访问的资源和操作，从而实现了精细化的授权管理。在文献[1]中，作者介绍了基于策略的访问控制的基本概念和实现方法，并讨论了在多方授权场景中实现该方案所面临的挑战和问题，虽然目前相关领域有待研究，但未来多方授权场景下基于策略的访问控制仍可能受到关注。

基于身份的访问控制将用户的身份信息作为访问权限控制的依据，而不是传统的基于角色或基于属性的授权模型。在多方授权场景中，基于身份的访问控制可以实现对多个参与者的细粒度访问控制和权限管理。例如文献[5]解决了在多方授权场景下的身份撤销问题，该方案使用了一种称为“基于证书的撤销”的技术，即当某个用户失效时，其证书将被撤销并加入到一个撤销列表中。此时，其他用户可以使用该撤销列表检查证书是否有效，并且可以使用该列表中的信息来更新其访问控制策略，以便拒绝失效用户的访问请求，保证了数据的安全性和隐私性。

近年来，利用多重签名技术实现多方授权系统得到了越来越多的关注。多重签名是一种基于公钥密码学的数字签名方案，允许多个签名者对同一份文件进行签名，以确保签名的有效性和不可抵抗性。文献[6]详细介绍了多重签名机制并分析了其特点和应用场景。多重签名主要应用于区块链中，以实现多方授权机制，使得区块链的交易更加安全和可靠。利用多重签名实现的多方授权系统属于基于角色的访问控制方法。在这种访问控制方法中，每个角色都被分配了一组权限。例如，在比特币交易中，为了防止交易中的资产被非法转移，每个参与者都有自己的角色，比如交易者、授权机构、注册机构等，只有经过多个授权机构的签名，生成多重签名，交易者才能完成交易。文献[7]就提出了一种紧凑的多重签名方案，适用于类似的小型区块链交易。

虽然多方授权系统在各种场景中的应用已经非常成熟，但它仍存在着隐私安全问题。例如在上文提到的利用多重签名实现的区块链交易中，交易者若想要支出一笔由多个参与方共同管理的资金，需要向授权机构验证自己的身份，以获取签名，然而在这个过程中交易者的身份信息就发生了泄露，并可能会遭到利用。文献[8]提出了一种可以保证隐私安全的单方授权的电子门票方案，其中门票的持有者可以匿名地访问演出场所，并且场所管理者可以有效地进行门票的验证，同时还可以实现可撤销的凭证。在该方案中，门票持有者通过基于属性的凭证技术获取一个匿名的门票凭证，场所管理者可以通过门票持有者提供的门票凭证来验证其门票的有效性，而无需了解门票持有者的身份信息，这样就保护了隐私。

目前，对具有类似匿名认证功能的多方授权机制的研究并不成熟，但是已经有许多相关技术值得关注。本文我们主要关注一种基于零知识证明的成员集合证明技术[9]。在密码学中，成员集合证明用于证明某个元素是否属于一个集合，它可以基于零知识证明实现匿名性，即证明者无需透露所持有的数据信息，只需要证明自己拥有这个集合中的某个元素即可，这种技术可以被应用于各种场景，例如去中心化身份验证、区块链身份管理等。在我们的研究中，我们将重点关注如何将类似技术引入多方授权系统中，以保护身份验证者的隐私安全。

## 1.3本文研究内容

为了设计与实现具有隐私保护的多方授权系统,保证多方授权的可靠性和隐私保护，本文基于对过往研究的总结与思考，尝试选取合适的多方授权机制和隐私保护协议，并将两者结合设计新的协议，通过实验仿真分析其性能、效率与安全性。我们分三点介绍本文的研究内容。

一、设计了一个具有隐私保护的多方授权方案。为了实现多方授权，我们需要从已有的多方授权方案中进行选取。这样的多方授权方案应当满足完整性和不可伪造性，在实际应用中不会被攻击者轻易攻破。它还需要尽量简单，不占用过多资源，具有快速的计算时间，以满足广泛的应用场景。多方授权中的访问控制系统可以使用各种访问控制技术，经过对不同访问控制技术的调查研究，拟采用基于角色的访问控制，通过分配不同角色给不同的参与方，来实现对资源的授权管理，贴合预期的应用场景。通过更为详细的研究，发现利用多重签名技术实现的多方授权系统能有效满足课题需求，多重签名是一种基于公钥密码学的数字签名方案，允许多个签名者对同一份文件进行签名，以确保签名的有效性和不可抵抗性。在本文中拟采用[7]中提出的一种紧凑的多重签名方案，它基于[11]中介绍的BLS签名方案和基于双线性配对的聚合机制。要在这样的多方授权系统中实现隐私保护，需要找到合适的保护机制。我们选取的多方授权方案并未指明用户向授权机构进行认证的方式，为了实现隐私保护，这样的认证应当是匿名的，具有不可伪造性，并且占用较小的空间和足够快速。我们的研究拟采用[9]提出的一种零知识证明的成员集合证明方案，这个方案基于[12]中提出的短签名方案，为实现轻量快速的匿名认证提供支持。

二、系统实现与分析。为了分析该系统的性能和效率以及可用性，需要根据系统协议，实现仿真程序，进行实验测试。所采取的编程语言应当便于系统实现，贴合现实的应用场景，并且尽可能具有更高的效率。系统协议分为四个阶段，一、系统初始化阶段。二、用户注册阶段。三、服务请求阶段。四、验证阶段。对此，我们拟采用C++编程，用类作为数据结构封装该系统中的不同参与方，并根据执行阶段模块化系统功能。PBC（Pairing-Based Cryptography）库是一个免费的 C 语言库，提供基于双线性配对的数学运算功能。它为我们的方案实现中所需的双线性群的生成和运算提供了支持。并且，C++作为底层语言的特性有助于使实现的系统快速高效。

三、方案安全性分析。为了分析我们的方案是否达到设计目标，需要从理论上分析该方案的安全性。安全性分析应当考虑授权与认证两个方面，证明多方授权与匿名认证的不可伪造性。本文我们通过分析证明了我们的方案的安全性。

## 1.4章节安排

在第二章，我们介绍了我们的系统设计中所用到的重要知识，主要是数学与密码学领域的基础知识，比如双线性群，还介绍了我们所用到的方案的安全模型，及其基于的计算困难问题。在第三章，我们自顶向下地介绍了我们的系统的方案设计，描述了方案中的技术细节。在第四章，我们介绍了系统的实现原理，并对实验测试的结果进行了统计和分析，评价了系统的性能和效率。在第五章，我们总结了我们的工作，并且提出了我们的不足。

**第二章 基础知识与安全模型**

2.1基础知识

**2.1.1 双线性群**

定义1（群）: 设是一个非空集合，是上的一个代数运算，即对所有的该集合中的任意两个元素，有，如果满足以下三个条件：(1)结合律，对所有的有 (2)中存在元素，使得对于每一个中的元素都有。(3)对中的每个元素，存在另一个元素使得 ，则称关于运算构成一个群，记为。其中称为单位元，一个群的单位元是唯一的。称为元素的逆元，对各个元素来说，也是唯一的。

定义2（阶）: (1)群的阶：群的元素个数。(2)群中元素的阶：为群中的一个元素，规定单位元，使的最小正整数n叫做元素的阶，如果这样的不存在，则的阶为无限或称为0。

定义3（双线性映射）:[7]中介绍了双线性群的定义。双线性映射定义了三个[素数](https://so.csdn.net/so/search?q=%E7%B4%A0%E6%95%B0&spm=1001.2101.3001.7020)阶乘法循环群，并且定义在这三个群上的映射关系，并且满足以下性质：(1)对于任意中的元素以及属于的整数，。(2)中存在满足。(3)可计算性：存在有效的算法使得所有的中的元素均可计算。

设是一个双线性群生成器，它将安全参数作为输入，并输出乘法群的描述，其中和是素数阶群，是一个有效的、非退化性的双线性映射，和分别是群和的生成元。

**2.1.2 多重签名**

多重签名用多个密钥进行签名来授权服务请求。

我们遵循[9,13]中的定义，将多重签名方案定义为算法和。一个受信机构生成系统参数。每个签名者都生成一对密钥，签名者可以通过各自调用交互式算法来集体签名一个消息，其中是签名者的公钥集合，是签名者的私钥。在协议结束时，每个签名者都输出一个签名。算法在输入一组公钥时，输出一个唯一的聚合公钥。验证者可以通过运行来检查聚合公钥下消息的签名的有效性，输出0或1，分别表示该签名无效或有效。

一个多重签名方案应该满足完整性，也就是说，对于任何，如果我们有，其中，和任何消息，如果所有签名者都输入，那么每个签名者都会输出一个签名，使得。第二，多重签名方案应该满足不可伪造性。多重签名方案的不可伪造性由一个三阶段的博弈定义：

设置。挑战者生成参数和一对密钥作为挑战。它在公钥为的条件下运行敌手。

签名询问。被允许对任何消息进行签名查询，查询对象为任何一组的签名者公钥，这意味着它可以访问预言机，预言机将模拟可信的签名者在签名协议中与其他拥有的签名者互动以签名消息的过程。注意可能会同时运行任意次数的上述查询。

输出。最终，敌手输出了一个伪造多重签名，一个消息，以及一组公钥。如果，没有对进行签名查询，并且，则敌手获胜。

定理 1（广义分叉定理）：设是一个随机算法，是一个在时间内运行的随机算法，最多进行个随机预言机查询，成功的概率为。如果，则 的运行时间最多为，成功的概率至少为，其中的概率取决于的选择和的结果。关于广义分叉算法的定义在[7]中给出，这里不再赘述。

定义 4：如果在时间内运行，执行了签名查询和随机预言机询问，并以至少的概率赢得上述博弈，我们就说是多重签名方案 的一个伪造者。如果不存在伪造者，那么就是不可伪造的。

**2.1.3 零知识证明和协议**

零知识证明由S.Goldwasser、S.Micali及C.Rackoff[25]提出，它指的是证明者能够在不向验证者提供任何有用的信息的情况下，使验证者相信某个论断是正确的。

我们遵循[2,11]中的定义。一对相互作用的算法是关系的知识证明，知识误差，条件是对于所有以概率1接受与的对话；并且存在一个预期的多项式时间算法，称为知识提取器，这样，如果一个伪造的证明者有概率说服接受，那么在给定的可回溯黑盒访问权时，输出一个概率为的的见证。

如果存在一个概率性多项式时间图灵机算法（称为模拟器），那么一个证明系统是诚实验证者零知识的，这样对于任何，与交互后的输出与的输出在计算上不可区分。

协议是一个证明系统 ，其会话形式为，其中和由计算， 是由随机选择的挑战。如果对于某个可有效计算的表述，如果，验证者就接受，给定了的两个接受会话 和，可以有效地计算见证。 而且，存在一个多项式时间模拟器，它在输入和随机字符串上输出的接受会话，这与和之间的真实会话完全没有区别。

我们使用Camenisch和Stadler[16]引入的符号来进行离散对数知识的各种零知识证明和关于离散对数的表述有效性的证明。例如，

它表示一个对整数 和的知识的零知识证明，使得 和 成立，其中 , 和是群 和 的元素。

**2.1.4 集合成员证明**

给定集合，要证明一个元素，称为集合成员证明。

我们遵循[9]中对证明过程的定义，设(Gen, Com, Open)是一个字符串承诺方案的生成、承诺和开放算法。对于一个实例，对于承诺方案和集合的集合成员证明是对以下表述的知识证明：

该证明系统是针对所有承诺方案而定义的。因此，特别是，如果Com是一个完全隐藏的方案，那么语言由所有的承诺组成（假设非空）。因此对于可靠性来说，协议是知识证明的这一点很重要。

**2.1.5 计算困难问题**

定义 5（离散对数问题）：对于一个阶为素数的群，我们定义敌手的为

其中概率是通过 的随机选择和 的随机选取计算。如果能在最多的时间内运行并且有，那么它就能破解离散对数问题。如果不存在这样的敌手，则离散对数问题是困难的。

定义6（计算性共同迪菲-赫尔曼问题）：对于一个阶为素数的群，定义敌手的 为

其中概率是通过 的随机选择和 的随机选取计算。如果能在最多的时间内运行并且有，那么它就能破解计算性共同迪菲-赫尔曼问题。如果不存在这样的敌手，计算性共同迪菲-赫尔曼问题就是困难的。

定义7（计算性共同迪菲-赫尔曼问题）：对于一个阶为素数的群，设是一个预言机，它在输入时返回。定义敌手的为

其中概率是通过 的随机选择和 的随机选取计算。如果能在最多的时间内运行并且有，它就能破解计算性共同迪菲-赫尔曼问题。如果不存在这样的敌手，那么计算性共同迪菲-赫尔曼问题就是困难的。

定义8（强迪菲-赫尔曼假设）：如果对所有概率性多项式时间图灵机的敌手，，其中，且，输出一对，其中的几率在中可以忽略不计，那么我们就说对于一个双线性配对生成器的q-强迪菲-赫尔曼假设成立。

2.2多重签名

**2.2.1 一种紧凑的多重签名**

我们方案中的多方授权机制使用了一种紧凑的多重签名[7]，这种签名基于BLS签名。

首先简要回顾一下BLS签名方案和它的聚合机制。该方案需要：(1) 一个可有效计算的非退化配对 ，其中 为素数阶群，我们设和分别是和的生成元。(2) 一个哈希函数。

BLS签名方案的工作原理如下：(1)密钥生成：选择一个随机的并输出，其中。(2) : 输出。(3)验证函数 : 如果 则接受, 否则拒绝。

这个签名方案支持一个简单的签名聚合过程。给定三元组，其中，可以通过以下计算将签名聚合成一个短聚合签名：

(1)

为了验证 的有效性，计算如下等式:

(2)

注意验证过程需要所有的 ，其中 。在我们的方案中，我们考虑所有被签名的消息都是一样的 ，此时验证条件(2) 就简化为了一个更简单的形式 ，只要求两个配对满足:

(3)

而且, 只需要给验证者一个短的聚合公钥 。

考虑以下流氓公钥攻击：攻击者注册了一个流氓公钥 ，其中是某个不知情的用户Bob的公钥，由攻击者选择。然后，攻击者可以通过提出聚合签名来声称它和Bob都签署了某个消息。这个签名是两个签名的聚合，一个来自，一个来自，因为：

因此, 满足 (3)。实际上，攻击者对消息承诺了Bob用户，而Bob却没有签署 。

在[7]中，作者提出了一种不同的防御方法来抵御这种攻击。接下来，我们会简单介绍[7]中的授权方案和如何证明其安全性。方案流程如下。

首先进行参数生成，生成并输出一个双线性群。

进行密钥的初始化，授权机构选择，计算，并输出，每一个授权机构拥有自己独立的密钥。

进行公钥聚合，输出聚合公钥。

然后进行签名。签名是一个单轮协议。计算，其中。将发送给一个指定的聚合者，在我们的方案中它为用户，它收集所有单个签名并将最终签名计算为。在我们的方案中，在收集每一个签名之前，用户需要在对应授权机构处完成匿名认证。

最后通过如下式子，验证多重签名有效性，如果等式成立证明有效：

[7]中对上述方案进行了安全证明，以抵御流氓公钥攻击，这里我们进行简单的介绍。

作者将他们提出的基于BLS的多签名方案称为，并提出如下定理：在随机预言机模型中，在计算性的共同迪菲-赫尔曼问题下是一个不可伪造的多重签名方案（如定义4所述）。更确切地说，如果，并且共同迪菲-赫尔曼问题是困难的，则称在随机预言机模型中是不可伪造的，其中是单个多重签名中涉及的最大签名者数量，和分别表示在和中计算指数所需的时间，和分别表示在和中计算个多重指数所需的时间。

我对这个定理的证明过程进行了简单总结。作者首先假设了一个针对 多重签名方案的 伪造者，它利用算法进行流氓公钥攻击。经过作者的推导，计算出的整体成功概率为 ，通过定义4我们可以知道，这证明了这个多重签名方案是不可伪造的。然后，作者通过构建算法来证明此定理，使用上述算法运行来自定理1的广义分叉算法，最后证明了如果，在时间上最多运行 并且成功概率，这证明了多重签名方案在随机预言机模型中不可伪造。

**2.2.2 一种基于签名的零知识证明的集合成员证明**

我们引入一种零知识证明的集合成员证明协议[9]，以实现匿名认证。该协议的灵感来自 Camenisch、Neven和shelat [17] 提出的遗忘传输协议。其基本思想是首先验证者向证明者发送集合中每个元素的签名。 因此，证明者收到关于承诺的特定元素的签名。 然后，证明者随机化这个签名，并对其拥有对应的签名进行知识证明。协议的具体内容如下。

我们设为证明者，为验证者。首先一个双线性映射由双线性群生成器PG生成，为的阶数，是的两个生成元，要证明自己拥有集合中的一个元素，为一个承诺，，其中为挑选的随机数，。公共参数为。

首先挑选私钥 ，计算公钥 ，把和发送给，其中遍历中的所有元素。即对元素的签名，也可以认为是证书。

然后挑选随机数，并发送。其中是对的签名随机化的结果。接下来和进行知识证明。

选择 并发送 和 。

发送一个随机数挑战。

发送 和 。随即验证和 是否成立，若成立，则证明成功。

在这个方案中，对元素的签名来自[12]中提出的Boneh-Boyen签名。这里我们简要介绍一下这个短签名方案。其签名者的私钥为，对应的公钥为。消息m上的签名是 。通过计算等式 来完成验证。该方案类似于Dodis和 Yampolskiy的可验证随机函数[18]。

该协议的安全性主要考虑抵御弱选择消息攻击。弱选择消息攻击下的安全性通过以下博弈定义。敌手首先输出条消息。挑战者生成新的密钥对并将公钥连同上的签名一起提供给敌手。 如果敌手成功地在消息 上输出有效签名 ，则敌手获胜。 如果没有概率性多项式时间图灵机敌手赢得这场比赛的概率不可忽略，则该方案在选择消息攻击下不可伪造。实际上，Boneh-Boyen签名具有以下性质：

假设q-强迪菲-赫尔曼假设在中成立。那么Boneh-Boyen 签名方案是q-安全的，可以抵御弱选择消息攻击下的存在性伪造。

以下定理是正确的：如果与双线性群生成器PG相关的 -强迪菲-赫尔曼假设成立，则上述的集合证明协议是集合的集合成员的零知识证明。

此定理的证明主要利用了知识证明的提取特性和随机函数的不可伪造性，根据提取特性，作者推导出对 Boneh-Boyen 签名方案发起弱选择消息攻击的成功概率可忽略不计。

# 第三章 方案设计

## 3.1整体构思

在金融领域中，金融机构需要对多个客户的共同资金的支出进行验证和审计，这需要多个客户的共同授权以确保交易的合法性。然而，在多方授权的交易验证过程中可能会暴露请求授权的交易者的身份信息，造成隐私泄露。其中一个典型的例子是比特币交易，在比特币交易中，为了防止交易中的资产被非法转移，每个参与者都有自己的角色，只有经过多个授权机构的签名，交易者才能完成交易，在签名之前，需要向授权机构验证自己的身份，然而在传统的多方授权系统中，验证身份会使交易者的身份信息发生泄露，并可能会遭到利用。

我们研究的方案的应用场景与比特币交易类似，我们采取基于角色的访问控制方式，将参与者划分为四种角色：用户、服务提供者、注册机构和授权机构。用户是向服务提供者请求某项资源的实体。服务提供者实际管理着单个或多个资源，受信于授权机构。而授权机构实际拥有资源的所有权，拥有可以向服务提供者进行验证的密钥，在我们讨论的场景下，授权机构有多个。注册机构受信于授权机构和服务提供者，合法的用户拥有获取资源的权限，而注册机构用于为合法用户发放证书。图3-1展示了这个系统的应用场景。

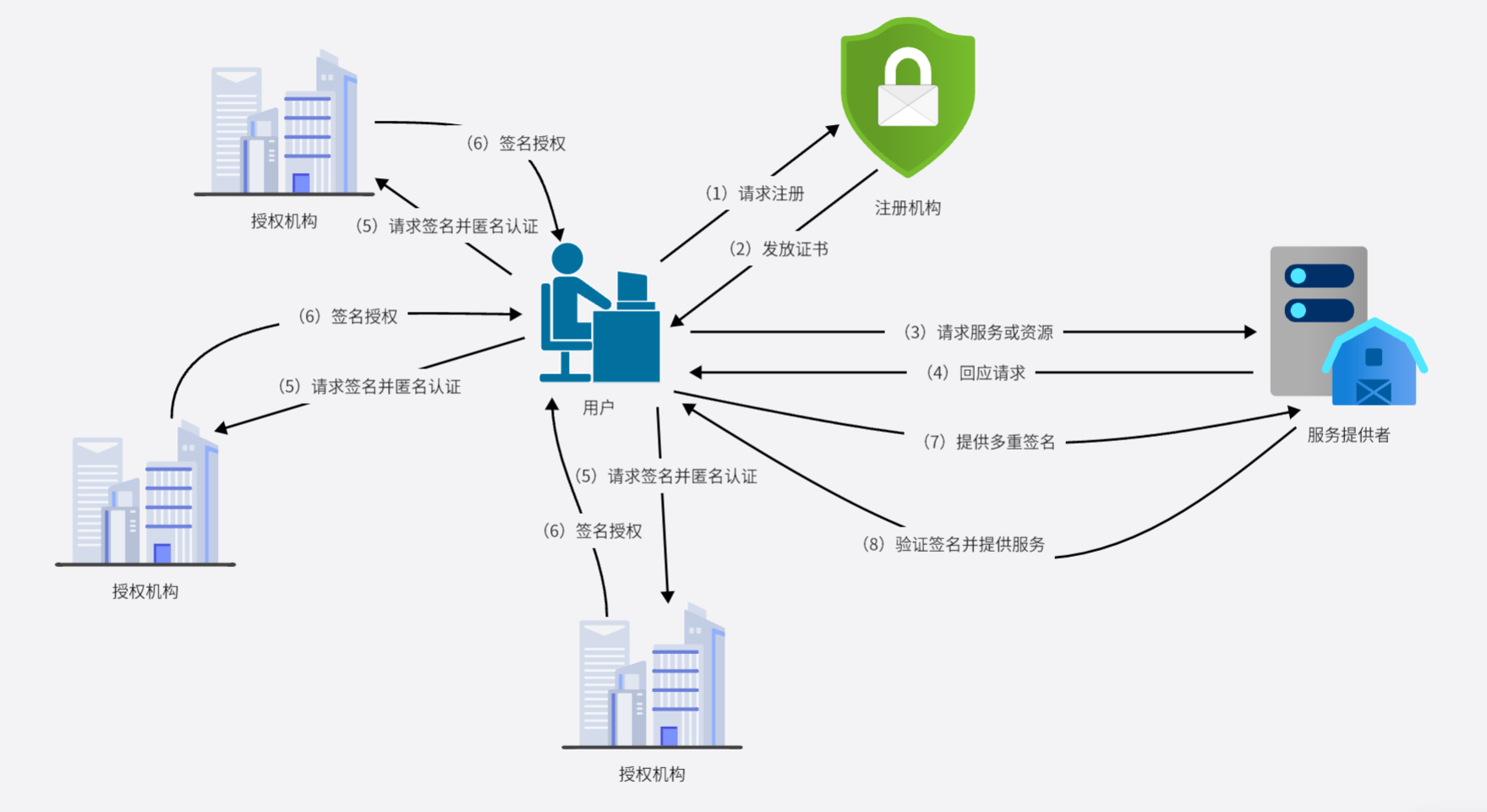


图3-1系统应用场景图

一个合法用户首先要在注册机构处获取证书，然后如果它需要访问某项资源，需要向服务员请求者进行请求，并向授权机构验证身份，以获取授权。根据身份认证和多方授权机制的不同，这个过程有多种实现方式。

对多方授权机制的研究已经非常成熟，我们的研究关注如何设计一种具有隐私保护的多方授权系统，具体而言，对于隐私的保护特指在身份验证阶段，如何防止用户身份泄露。即一个合法的用户，不需要在验证身份的过程中泄露自己在合法用户中的具体信息，比如，在合法用户中的编号、用户名等。

多方授权机制并不是我们研究的重点，我们的需求是找到一种安全的，即满足完整性和不可伪造性，并且轻量、计算时间短，能够适应流行的应用场景（比如比特币交易）的现有方案。我们使用了[7]中提出的一种紧凑的多重签名方案，它被设计用于小型区块链交易，但同样可以用在其他场景中。本文的2.2.1节已经阐述了该方案的安全模型。这个多重签名方案的关键在于用户收集多个授权机构的签名，并将聚合为的多重签名发给服务提供者，根据多重签名验证用户合法性，以获取服务。

如前所述，实现隐私保护的过程实际上是用户进行匿名认证，这样，问题就得到了简化。我们只需要找到一种合适的匿名认证机制。这样的匿名认证又叫做零知识证明的集合成员证明。它应当具有不可伪造性，并且占用较小的空间和足够快速，并且具有匿名性，即用户可以验证自己的合法身份，但不泄露自己的身份信息。

我们采取了[9]提供的基于签名的零知识证明的集合成员证明方案。在这个方案中，注册机构发放的证书基于Boneh-Boyen签名[12]。实现匿名认证的关键在于用户对证书做了随机化处理，而认证过程利用了双线性群的特性。我们同样在2.2.1节阐述了该方案的安全模型。

在我们的方案中，要结合上述的两种协议，设计一个新的系统协议并不是一个难点。因为认证过程和授权过程是分开进行的，这意味着两种协议不会相互干扰。下一页的图3-2展示了我们所构思的系统的执行流程，与前面所述的内容相一致。

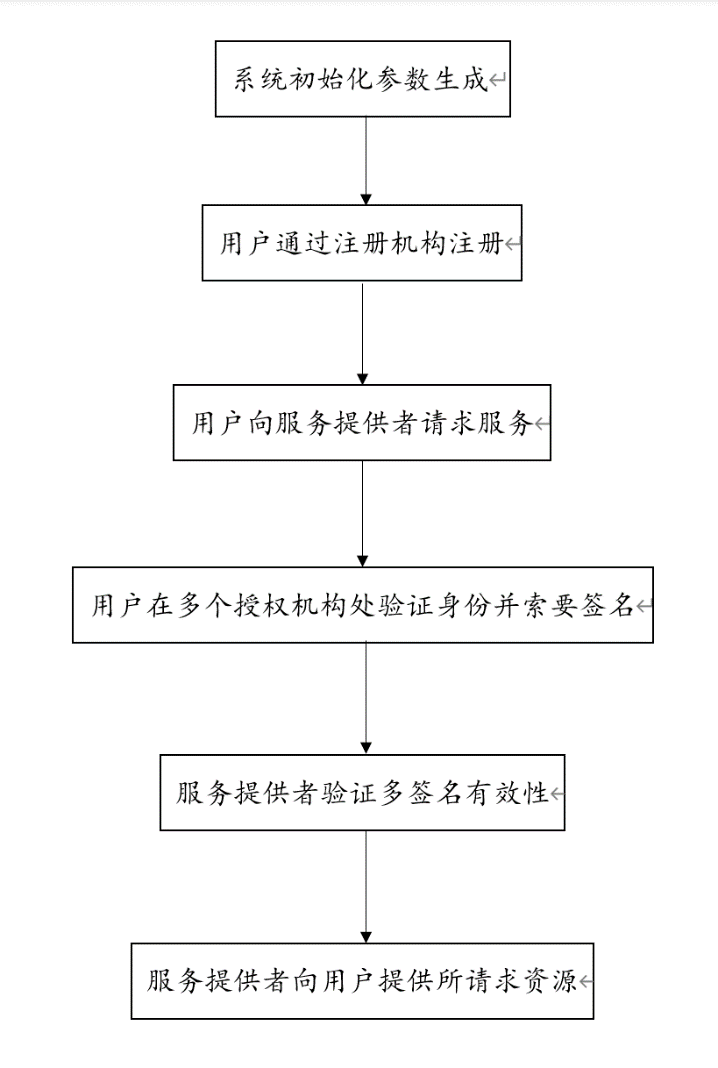


图3-2 系统执行流程图

在接下来的章节中，我们会对我们的方案设计，或者说系统协议作出具体描述，我们会呈现我们的协议并讨论技术细节。

## 3.2方案描述

我们用系统协议的方式来呈现我们的方案，将协议的流程简化为四个阶段。我们将分阶段地阐述我们的协议，首先详细描述每个阶段的协议内容，然后作出具体解释，其中被放在框图中的是各个阶段的协议内容。

**3.2.1 系统初始化阶段**

输入一个安全参数，运行双线性群生成器生成双线性群：

选择生成元。

生成哈希函数：

系统的公共参数为：

图3-3 系统初始化阶段

系统初始化阶段是系统的全局操作，计算生成系统的公共参数，但我们没有指定这些参数具体由哪个机构生成和分发。

双线性生成器由系统实现者定义，根据实现方式不同，有不同形式。在第4章，可以看到我们的实现方法。请注意，这里的双线性群实际上是对称的，即。

这里的指代二进制数。我们没有指定哈希函数的类型，只限定了它们的映射关系。

**3.2.2 用户注册阶段**

注册机构选择私钥 计算公钥 。授权机构选择私钥，计算公钥 。

用户进行注册请求后，注册机构为用户派发编号和证书。

图3-4 用户注册阶段

这里实际上是匿名认证协议的初始化阶段。在2.2.2节我们引入了一种[9]中的集合成员证明方案，在这里情况有所不同。在2.2.2节的安全模型中，我们提出了证明者和验证者两方，既要负责为集合成员颁发签名，即证书，又要对的身份进行验证。但在我们的方案中情况不同。

在我们的方案中，我们有一个专门的注册机构来挑选公私钥，并为合法用户进行注册，即把用户映射到集合中的新元素，我们称为编号，它是此集合成员证明所要证明的元素，它还要为合法用户进行签名，即颁发证书。而验证者由授权机构担任，在2.2.2节中，验证者首先要发送给 和。而在这里这一步已经由注册机构在这个阶段完成，并且只发送了，这是可行的，因为本来就要在中选出。因此，在授权机构对进行身份验证时，跳过了的第一步发送。授权机构只需要就可以实现验证，而是公开的，所以验证过程可行。后面的阶段展示了验证过程。

在这一阶段，授权机构还进行了密钥的初始化，虽然这与认证无关，但我们还是放在了这一阶段进行描述，以为后面的签名做好准备。

**3.2.3 服务请求阶段**

用户进行服务请求后，服务提供者发送给用户,并指明所需要的授权机构，其中是所要请求的资源。

用户向授权机构发送授权请求，通过以下步骤验证身份：

用户选择 并发送 。

用户选择 并发送 和 。

授权机构发送一个随机数挑战。

用户发送 和 。

授权机构验证以下两个式子是否成立，若成立，证明身份合法：

(4)

(5)

用户将发送给授权机构，第个授权机构返回：

图3-5 服务请求阶段

这里，用户可以请求一种数字资源，或是另外形式的服务，这不重要，只要服务提供者把这种服务用二进制串形式的表示即可。例如，用户可以请求一种互联网上的交易服务，本身虽然并不指代服务本身，但可以是获取服务所需的密钥。服务提供者利用隐藏了，但又与相关，这保证了每次的不同，防止有关攻击的实现。

服务提供者必须指明所需的授权机构，使得用户能与它们建立通信，授权机构可以是一个或多个。显然，只有多个授权机构的存在我们的方案才有意义。

匿名认证的过程遵循2.2.2节中的协议。我们可以看到，用户使用随机数对自己的证书进行指数运算，这样的随机化处理隐藏了自己的证书，这样授权机构就无法知道自己的具体身份，但他仍然证明了自己是合法用户中的一员，即拥有中的元素。

其中，是一个承诺。在密码学中，承诺是一个涉及两方的二阶段交互协议，双方分别为承诺方和接收方。它有不可更改性和确定性。隐藏也就是承诺值是不会泄漏有关原消息的任何信息，而绑定就是接收方可以确信收到的消息是该承诺对应的消息，不可能再找到一个不同的消息从而产生同一个承诺。承诺方发送的消息密文，一旦发出就意味着不会再更改，而接收方收到这个消息可以进行结果的验证。这里我们可以看到，(4)中的验证使用到了承诺。

挑战/应答方式即验证者发送一个挑战字串，证明者收到这个字串后，做出相应的应答。在这里，作为随机数挑战。用户进行回应并通过(4)、(5)来验证这个挑战。

(4)和(5)在合法用户的认证下成立，这主要利用了双线性配对的性质，我们在之前已经有过介绍。

认证成功后，用户向每一个授权机构请求签名。授权机构通过进行签名，过程与2.2.1节中的方案一致。我们可以看到，每一个都与当前的授权机构和所有需要的授权机构的集合有关，并且使用了一致的哈希函数，这保证了每次的签名是高度随机、独立的。

**3.2.4 验证阶段**

用户向服务提供者提供多重签名服务提供者验证以下式子是否成立：

(6)

其中

若成立，服务提供者向用户提供资源。

图3-6 验证阶段

这里多重签名的计算非常简单，由BLS多重签名衍生而来。在BLS签名中，多重签名是单个签名的和，而在这里是乘积。而叫做聚合公钥，顾名思义它是对公钥的聚合，但并不是简单的公钥乘积，而是有作为幂，这是因为中同样有作为幂。的出现是因为由计算。

(6)作为验证表达式的成立同样运用了双线性配对的性质。

在这里，我们的方案有一个性能优势。因为我们的多重签名方案和匿名认证方案使用的是同一个双线性配对，这个配对是对称的。不需要分别生成两个双线性配对，因为共用一个不会影响安全性。实际上，这节约了成本。

以上是我们对我们的方案的描述。我们的方案可以适用于广泛的多方授权应用场景，这里我们举一个简单的例子，比特币交易。

多重签名技术让比特币可以实现非常丰富的功能，例如支付通道和闪电网络。多重签名交易也被叫做“M-of-N”交易。M 指的是交易生效所需要的签名数量，N 指的是和本次交易相关的各方的总数量。在比特币交易中，若多方控制一笔资金，就需要用到多方授权来完成交易。一个典型的例子是多重签名钱包。

多重签名地址是 个公钥 的哈希值，即。比特币交易的特殊之处为，与比特币多重签名地址相关的交易数据需要写入区块链中。所以我们应当尽量缩小这样的数据的大小。在我们的方案中，我们使用的多重签名足够紧凑，能有效缩小比特币区块链的大小。

更重要的是，我们为这样的多重签名引入了匿名认证机制。以比特币交易为例，在某些场景下，资金的汇入方也许不希望资金的掌控者了解自己的具体身份，以保护自身隐私，这时，我们的方案就高度适用。

## 3.3安全性分析

**3.3.1 多方授权**

我们的方案使用了[7]中的多重签名方案，在2.2.1节中我们介绍了这个方案的安全模型。

这种签名方案基于BLS签名，对于它的安全性我们主要考虑流氓公钥攻击，即攻击者伪造公钥并与合法授权机构进行签名聚合。在2.2.1节我们已经对它的安全性证明进行了介绍。在这里我们进行总结，这个方案的安全性基于的计算困难问题为离散对数问题和计算性的共同迪菲-赫尔曼问题，这个多重签名方案在理论上具有不可伪造性。因此，我们的多方授权机制是安全可靠的。

**3.3.2 匿名认证**

我们利用[9]中的基于签名的集合成员证明方案来实现隐私保护，它实际上是一个匿名认证方案，我们在2.2.2节介绍了其安全模型。

这种匿名认证方案基于零知识证明，即证明者可以向验证者证明自己的合法身份，但不泄露关于自己身份的任何信息。它的灵感来自于[17]中的遗忘传输协议。对于安全性，这个方案主要考虑弱选择消息攻击，即攻击者尝试伪造有效身份。2.2.2节中已经介绍了对其安全性的证明，这里我们同样进行总结。首先，这个方案中的签名来自[12]中的Boneh-Boyen签名，它可以抵御弱选择消息攻击下的存在性伪造，即攻击成功的概率可以忽略不计。其安全性基于的计算困难问题为离散对数问题、q-强迪菲-赫尔曼假设和-强迪菲-赫尔曼假设。这证明我们的认证方案具有不可伪造性。

**第四章 系统实现与实验分析**

4.1系统的实现

**4.1.1 实现方法**

如上文所述，我们的系统协议适用于多种应用场景，在实际应用中，可以把这种系统视作分布式的。我们对系统进行实现的目的是对我们的方案进行实验仿真测试，以得出数据，再通过数据分析系统的可行性、性能和效率。我们并不是要实现为现实应用场景服务，因为现实的应用场景是基于网络的，其中参与方通过网络通信，而我们的实验模拟在本地进行。

因此，我们对系统的实现有以下要求。首先，我们的实现需要完整地实现系统的功能，能够模拟正确的协议流程。合法用户的认证和请求应当顺利进行，而非法用户的请求应当被拒绝。我们的实现应当允许操作者改变实验的条件，比如授权机构的数量、资源的内容等，这样我们才能通过改变实验条件测量不同情形下的实验数据，满足实验要求。我们的实现是基于编程的，必须简单明了，这就要求我们的代码具有高度可读性，有明晰的注释和清楚的模块划分，而且要有健壮性，不会因为错误的输入出现漏洞。另外，我们的实现需要便于测量。在这一点上，首先，我们的实现应当高度模拟真实网络中的情况，控制误差范围，体现协议高效性。其次，应当便于测量时间。在我们的模拟中，时间是一个重要参数，我们要测量局部事件发生的时间，这就要求测量的误差要小，而且对系统性能的影响要小。

最终，我们确定了我们的实现方法。我们采取C++编程，基于C++面向对象编程的属性，利用类完成系统角色的封装，并且将系统功能模块化。另外，我们的编程还基于PBC库(Pairing-based Cryptography Library)[26]。

利用这样的实现方式具有其优势。不仅能满足我们的实验要求，C++控制台程序还能可视化地输出程序的运行结果，封装出的操作接口也非常简单，没有理解相关原理的人也能进行操作。这样，使得程序的运行过程和输出结果更加清晰易懂，方便后续的数据分析。

**4.1.2** **基于角色的访问控制**

之前提到过，我们的方案采用基于角色的访问控制(RBAC)。在基于角色的访问控制模型里，有3个基础组成部分，分别是：用户、角色和权限。通过定义角色的权限，并对用户授予某个角色来控制用户的权限，实现用户和权限的逻辑分离，方便了权限的管理。它有三个安全原则：最小特权原则、责任分离原则和数据抽象原则。最小特权原则即RBAC可以将角色配置成其完成任务所需的最小权限集合。责任分离原则即可以通过调用相互独立互斥的角色来共同完成敏感的任务，例如要求一个计账员和财务管理员共同参与统一过账操作。数据抽象原则可以通过权限的抽象来体现，例如财务操作用借款、存款等抽象权限，而不是使用典型的读、写、执行权限。

在我们的方案中，有用户、服务提供者、授权机构和注册机构四种角色，并对应了相应的权限。图4-1展示了它们的关系。

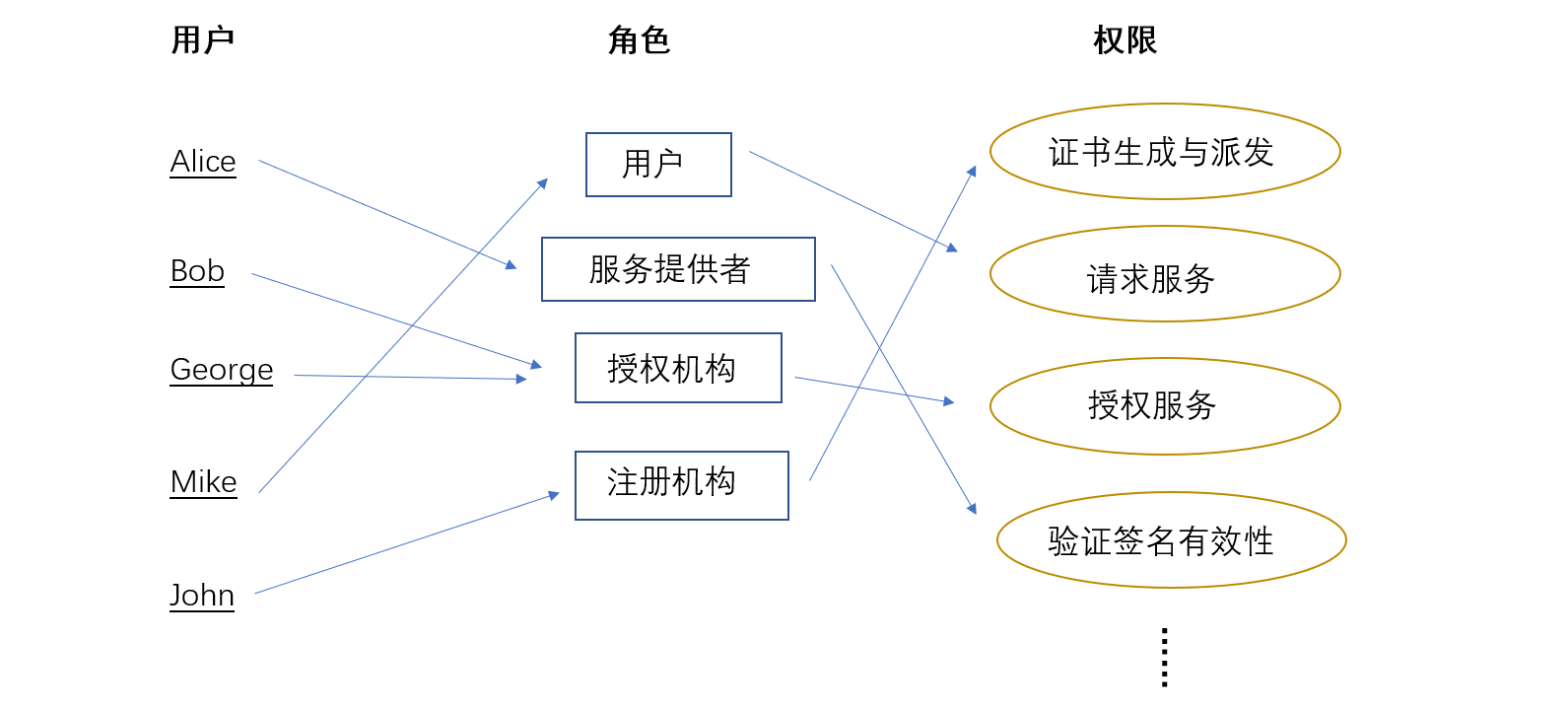


图4-1 我们基于角色的访问控制模型

可以看到，不同用户担任着各类角色，而每一个角色都有自己的权限。基于角色的访问控制简化了用户和权限的关系，使得我们方案的表达非常清晰。并且在这样的模型下我们的系统易扩展、易维护，适用于多种场景。

**4.1.3 C++编程的优势**

从设计思想上来讲，C++面向对象的思想比C面向过程的思想能处理更复杂的系统问题。从语法上讲，C++有封装，继承和多态这三大特性。封装隐藏了实现细节，使得代码模块化。继承通过子类继承父类的方法和属性，实现了代码重用。多态则通过子类重写父类的虚函数，实现了接口重用。C++比C等语言能更好地处理我们的问题，例如，我们用类来封装角色，而C没有类。

如果关注其他高级语言，如Java、Python，C++同样有其优势。我们提到我们使用了PBC库，这个库有其Java版本，但我们选择了C++。

C++相比Java有以下优势。首先，C++的运行速度更快，效率更高，编译为本地机器代码，因为它相对于Java更加底层。这样我们的实现效率就得到了提高，能够更好地控制误差。其次，它提供底层内存控制和系统访问能力，例如动态内存管理，这样我们的实现有更好的内存管理和空间利用效率。

我们用类来封装角色，这时候要考虑角色自身的数据结构。在我们的方案中，角色主要拥有密钥、计算参数，比如注册机构的公钥和私钥，用户收集的签名等，还有一些与密码学没有直接关系的数据，比如资源，还有授权机构的整型编号，在我们的实现中，我们把资源定义为字符串类型。它们都易于实现。而且用公有类型和私有类型可以区分数据的访问权限，例如，公钥属于公有类型而私钥属于私有，这样公钥可以被其他角色直接访问而私钥不可以。

至于角色拥有的权限，即它可以执行的一系列操作，则封装在类的函数中，比如，授权机构用对应函数来对用户提供的资源的哈希值进行签名，返回签名的结果。

**4.1.4 PBC库的介绍**

PBC是实现双线性对运算的函数库，由Stanford大学Ben Lynn博士用C开发并开源的密码库[26]。PBC密码库为双线性对实现提供了接口，是基于双线性对密码体制研究的一个非常有用的辅助工具。

我们的实现大量使用了这个库。可以从<http://crypto.stanford.edu/pbc/>获取这个库的详细介绍和使用手册。我们主要描述我们是怎么使用这个库的。

首先，我们在初始化阶段要生成一个双线性群。在PBC库中，我们用pairing\_init\_pbc\_param(struct pairing\_s \*pairing, pbc\_param\_t par)函数实现这个过程，其中pairing是配对本身，par是初始化配对所用到的参数。在这之前我们还需要用pbc\_param\_init\_a\_gen(pbc\_param\_t par, int r, int q)进行参数par的初始化，它生成一个A类配对参数，存入par中，其中群的阶为r比特长，阶的基域为q比特长，因此群中的元素都为q比特。所谓的A 类配对是在域上的曲线 上构建的，这里不做更多描述。只需要注意，这样获得的配对是对称的，也是我们所需的。我们所取的r和q为使用手册中默认的160和512。

另外，在我们的协议中所有的元素，在PBC库中，都用elment\_t类型表示，这样这些元素才能执行配对操作。每个元素在声明时需要初始化，比如element\_init\_G1(g, pairing)将g初始化为群上的元素，然后可以用element\_random(g)把g指定为这个群上的一个随机的生成元。PBC库还提供了关于这些元素的运算操作，比如element\_mul(a, b, c)把a指定为b和c的乘积。而配对则是通过pairing\_apply(a, b, c, pairing)实现，它把a指定为b和c在配对pairing上配对运算的结果。

关于其他的运算函数，请参考PBC库的使用手册，这里不再赘述。

**4.1.5 哈希函数**

哈希函数，又叫散列函数，指将哈希表中元素的关键键值映射为元素存储位置的函数，把任意长度的输入通过散列[算法](https://m.elecfans.com/v/tag/2562/)变换成固定长度的输出，该输出就是散列值。

回顾我们的协议，我们用到了两个哈希函数：

其中和都是从一个二进制串开始映射的，即。映射到上的一个元素，映射到上的一个数。

首先观察在协议中的用法。被用于对资源做映射操作，即。而用于计算多重签名地址。

但在这里出现了一个明显的问题。实际上，市面上设计的哈希函数的映射都是数与数之间的映射，更准确地说，是二进制数间的映射。但在这里，和都不是二进制数。被定义为了字符串，而则是PBC库中的元素类型。我们将分开讨论这个问题。

首先，PBC库的手册中也提到了这个问题，PBC库并没有给定一个哈希函数实现二进制串到元素之间的映射，因为使用需求是多样的。它的解决方案是，给出了一个函数element\_from\_hash(element\_t e, void \*data, int len)。这个函数将一个数据data映射到了一个元素e上，这个元素需要提前初始化，以指定它属于哪个群或者域。这样，使用者可以使用市面上的任何哈希函数，只要最后把数据通过这个函数映射到元素上即可。

即，把普通二进制串数据的分两步进行哈希。我们以为例，我们的解决方案是。先将字符串通过选定的哈希函数映射到unsigned char数组类型的数据ha上，再通过element\_from\_hash(hm,ha,32)把ha映射到了element\_t类型元素hm上，这里32代表ha的位数。根据的定义，在这里hm是上的元素。映射过程可表示为：

同样的，也可以利用类似的方法解决，但这里出现了新的问题。首先，的含义是，逗号分开的比特串相互级联，即简单的比特拼接，然而，在这里并不是比特串，而是element\_t类型的元素。所以，首先我们要把映射为比特串，好在PBC库同样提供了函数来实现这一点。然后，再将它们按格式级联。最后，执行跟类似的操作，将比特串映射为散列值，再将散列值比特串映射到上。即:

现在我们解决了哈希映射的不匹配问题，但是还没有引入具体的哈希函数。实际上，市面上有许多哈希函数可供选择，也有许多库可以直接调用。在我们的方案中，我们直接使用了SHA256和SHA512作为和，下面介绍这两个哈希函数。

SHA256和SHA512都属于SHA-2 族算法，即安全散列算法2(Secure Hash Algorithm 2)，由美国国家安全局研发，由美国国家标准与技术研究院​​​​​​​（NIST）在2001年发布。对于任意长度的消息，SHA256都会产生一个256比特长的哈希值，称作消息摘要。SHA512则为512比特。SHA256和SHA512的哈希过程有扩散和混淆两个步骤，具体的执行流程这里不再介绍。

这里我们使用这两个哈希函数，主要有两个原因。第一，它们具有较高的安全性，得到了广泛使用，且调用起来较为简单。第二，256比特和512比特与512位的元素位数较为接近，比较适合我们的应用。

我们使用了OpenSSL库提供的SHA256和SHA512。OpenSSL本身是服务于安全通信的，我们只使用了它封装的哈希函数，因为它比较容易配置。

**4.1.6 对数据的测量**

我们实验的目的是测量数据并进行分析，所以实现数据的测量非常关键。我们需要的数据主要分为两个种类，中间参数和所耗费的时间。

测量中间参数是指输出较为重要的参数，如注册机构的公私钥、多重签名的值等。它们的获取十分简单，因为PBC库中提供了一个类似于C++标准输出的函数，以实现把群或域上元素以可读形式打印在标准输出流上。例如，某次计算得到的多重签名的值输出为：[110531671284452176863482206379814381613742560811598461756346926517387149132035915645922377953561328789992457946990106395755904744422335983582900322266682,87386097267026534268329276205673565904857097944540440370830603660097461286532530614355918370861711927545461915148028330027126506433939442742557373731655]。

相比于中间参数，我们更关注时间的测量。这里的时间指各种功能执行所耗费的时间。它直接关系到了我们后续对性能和效率的分析，所以我们对时间测量的要求是要足够精确，尽量保留更多的位数。

我们寻找到了一种足够精确的时间测量方法。QueryPerformanceCounter()这个函数返回高精确度性能计数器的值,它可以以微妙为单位计时。不过它仅仅能返回程序执行以来计数器的值，而不直接地反映时间。这时，我们还需要QueryPerformanceFrequency()函数，它返回计时器的频率。利用计数器的差值除以频率，我们可以得到两次调用间隔的精确时间。这个时间精确到了小数点后四位，例如，某次测得协议第一阶段执行的总时长为52.4167ms。

4.2实验测量与数据分析

我们实验的主要方法是，通过改变授权机构的数量，改变实验条件，并测量程序执行过程中系统各个功能模块所耗费的时间。

我们测得了系统每个阶段执行所耗费的时间，和各种具体步骤所耗费的时间。我们的实验有六个实验组，它们的授权机构数量分别是5、10、15、20、25、30，每个实验组都测得了10组数据并把它们进行了平均化作为最终取用的数据。我们发现有一些阶段和步骤的执行时间受到授权机构数量改变的影响较大，而有一些则分布较为平均。

我们将用图表的方式来呈现我们测量的数据，主要是表格和直方图，并据此对系统的性能和效率进行分析。我们在下面提到的所有测得时间的单位都为毫秒(ms)。

首先，我们将从系统执行四个阶段所用时间来进行分析，进而转向更为局部的执行步骤的时间。下面4.1列出了在不同实验组下系统各个阶段执行的耗费时间。

表4.1 系统各阶段执行耗费时间

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 授权机构数量 | 第一阶段时间 | 第二阶段时间 | 第三阶段时间 | 第四阶段时间 | 总时间 |
| 5 | 110.2234 | 29.7406 | 135.465 | 50.7874 | 319.118 |
| 10 | 104.775 | 41.1871 | 291.732 | 38.9552 | 496.558 |
| 15 | 117.2312 | 56.0114 | 457.052 | 46.7827 | 659.737 |
| 20 | 110.4928 | 68.9493 | 620.035 | 54.9913 | 836.494 |
| 25 | 120.2342 | 101.85 | 784.666 | 70.1868 | 1082.13 |
| 30 | 112.473 | 113.297 | 890.808 | 61.7512 | 1179.17 |

可以看到，在某些阶段实验组间分布较为集中，差异较小，而在某些阶段差异较大。我们从每一个阶段各自进行分析。

图4-2展示了各实验组在第一阶段的执行时间的对比。经过计算，它们的方差为25.17。从图中也可以看出它们的分布较为集中。在系统执行的第一阶段，我们主要完成了双线性群的生成和生成元的初始化，这些步骤理论上来说与授权机构的数量没有关系，因此，分布才会如此平均。我们可以求得各个实验组在第一阶段的执行平均时间为112.5716ms。

图4-2第一阶段的执行时间

接下来是第二阶段的执行时间，如图4-3。

图4-3第二阶段的执行时间

我们可以看到这一阶段的执行时间与第一阶段有很大差异，在于分布的不平均。它们的方差为920.57，明显大于第一阶段的方差。随着授权机构数量的上升，第二阶段的执行时间逐步上升。这是由于第二阶段的参数生成时间有较大组间差异，我们将在后面具体解释。

图4-4第三阶段的执行时间

图4-5第四阶段的执行时间

图4-4和图4-5分别展示了第三阶段和第四阶段的情况。可以看到，第三阶段的执行时间分布与第二阶段类似，方差为70149，这同样是因为授权机构的数量的影响。而在第四阶段，情况发生了变化，数据的方差为101.96。理论上讲，第四阶段的执行不受授权机构数量的影响，这意味着结果的分布应当较为集中。实际上，从图中可以看出，数据的分布的确较为平均。同样的，我们将在后面解释产生以上结果的具体原因。

图4-6总执行时间

在图4-6展现的总执行时间中，我们可以看到总执行时间随授权机构的数量上升而上升。

接下来，我们从各个阶段中具体的执行步骤进行更详细的分析。

首先，我们得知第一阶段各实验组执行时间分布非常平均，这在理论上是因为第一阶段的执行步骤与授权机构的数量无关，比如说双线性群的生成。如表4.2所示。

表4.2 第一阶段各步骤的执行时间

|  |  |  |
| --- | --- | --- |
| 授权机构数量 | 双线性群生成时间 | 生成元初始化时间 |
| 5 | 94.1827 | 11.1049 |
| 10 | 90.8759 | 8.3505 |
| 15 | 100.2423 | 9.4706 |
| 20 | 98.3243 | 7.3823 |
| 25 | 104.1234 | 7.742 |
| 30 | 101.011 | 9.6889 |

可以明显看到两个步骤的执行时间组间分布较为均匀。而且，第一阶段的大部分时间在进行双线性群的生成。

在第二阶段，情况显然有所改变，如下一页的表4.3所示。

表4.3 第二阶段各步骤的执行时间

|  |  |  |  |
| --- | --- | --- | --- |
| 授权机构数量 | 注册机构公私钥生成时间 | 授权机构公私钥生成总时间 | 用户证书和编号生成时间 |
| 5 | 2.693 | 22.7776 | 2.3791 |
| 10 | 2.1384 | 33.8523 | 2.3341 |
| 15 | 2.0907 | 48.0827 | 2.8362 |
| 20 | 1.8895 | 63.1554 | 2.3944 |
| 25 | 3.1265 | 93.7612 | 2.7216 |
| 30 | 1.9874 | 105.147 | 3.1329 |

我们可以看到，注册机构公私钥生成时间和用户证书和编号生成时间的分布仍然非常平均，但授权机构公私钥生成总时间随着授权机构数量上升而上升，这是因为每个授权机构都要进行公私钥的生成。我们将这个总时间除以授权机构的数量，得到平均每个授权机构生成公私钥的时间，并得到图4-7。我们发现，它们都趋于一个均值。

图4-7每个授权机构生成公私钥的平均时间

下一页的表4.4展示了第三阶段各步骤的执行时间。与第二阶段类似，只有授权机构验证与签名总时间之间有较大差异，且呈上升趋势。

表4.4 第三阶段各步骤的执行时间

|  |  |  |  |
| --- | --- | --- | --- |
| 授权机构数量 | 消息哈希值生成时间 | 授权机构验证与签名总时间 | 多签名生成时间 |
| 5 | 5.3114 | 128.458 | 0.1975 |
| 10 | 6.8898 | 283.137 | 0.1723 |
| 15 | 6.8558 | 447.343 | 0.2755 |
| 20 | 5.5552 | 612.074 | 0.2421 |
| 25 | 6.4327 | 774.641 | 0.2023 |
| 30 | 6.8134 | 882.248 | 0.2653 |

我们同样计算了每个授权机构验证与签名的平均时间。如图4-8。它们同样趋于一个均值。

图4-8每个授权机构验证与签名的平均时间

而第四阶段的主要工作只有验证多签名有效性，所以图4-5就代表了这一步骤的执行时间。关于第四阶段的分析，我们之前已经提到过，这一阶段不受授权机构数量差异的影响。

我们还要提到一个指标，各个阶段在总执行时间中的占比，这样可以直观地看出哪个阶段在系统执行中具有较多的执行成本，哪个较少。这里，我们采取这样的方式，先计算出各实验组在每一个阶段执行时间的平均值，再用饼状图表示它们的比例。如下一页图4-9。

图4-9各阶段执行时间平均值占比

我们可以看出，第三阶段占比接近70%，占有绝大多数时间，而第四阶段只有7%。而由之前的结论我们知道第三阶段的绝大多数时间在执行授权机构的验证与签名。这证明授权机构的验证与签名时间对我们系统的执行时间产生的影响最大，随着授权机构数量的增多，这一步骤的时间上升，而且由之前的分析我们知道，平均的验证与签名时间趋于一致，这说明这样的上升是线性上升。

我们的方案的研究重点是身份认证和多重签名，因此我们还关注了在身份验证和签名阶段中具体步骤的执行时间，特别是身份验证的四个阶段，这里我们回顾一下。在验证第一阶段，用户发送给授权机构，其中为用户证书的随机化结果，以此实现匿名认证。在验证第二阶段，授权机构发送给用户随机数挑战。在验证第三阶段，用户发送给授权机构和 。在验证第四阶段，授权机构验证两等式是否成立。

我们想要统计的是每次单个授权机构验证与签名过程的执行时间，这与总的授权机构数量无关，因此，我们可以任意给定授权机构的数量。这里，我们选取了实验组1，即5个授权机构进行测量。得出数据如下一页的表4.5所示。

我们把授权机构进行了编号。可以从表中看出，所有执行步骤组间的方差都较小，这证明每个授权机构进行验证与签名的执行时间分布集中，不存在太大的波动。我们计算了它们的均值，发现验证第一阶段的执行时间远高于其他阶段。在这个阶段，进行了四个参数的计算，其中有群元素的幂运算、配对运算和乘积运算，第四阶段的平均用时也较高，这一阶段也有类似运算的参与，这证明这些运算更耗时。

表4.5 单个授权机构验证与签名时间

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 授权机构编号 | 参数初始化时间 | 验证第一阶段时间 | 验证第二阶段时间 | 验证第三阶段时间 | 验证第四阶段时间 | 签名时间 |
| 1 | 0.0173 | 26.8727 | 0.9747 | 0.0023 | 11.4218 | 2.9706 |
| 2 | 0.0122 | 22.8955 | 0.9955 | 0.0021 | 9.6615 | 3.7115 |
| 3 | 0.0126 | 25.3616 | 0.8072 | 0.0021 | 8.906 | 3.4671 |
| 4 | 0.0125 | 20.9344 | 0.782 | 0.0018 | 7.62 | 3.1782 |
| 5 | 0.013 | 25.0015 | 0.8234 | 0.002 | 8.1928 | 3.4507 |
| 平均值 | 0.0135 | 24.2131 | 0.8765 | 0.0020 | 9.1604 | 3.3556 |
| 方差 | 3.6E-06 | 4.3 | 0.0080 | 2.6E-08 | 1.7477 | 0.0655 |

以上，我们对我们测量的数据进行了不同方面的分析。首先我们从系统执行的阶段较为宏观地分析了执行时间，然后分析了其中的具体步骤，还分析了各个阶段在总执行时间中的占比。之后，我们进行了更为局部的研究，即研究身份验证与签名中的具体阶段的执行时间。

把所有数据一起进行考虑，我们可以得出如下结论。首先，我们的系统具有良好的性能和效率，在有30个授权机构的情况下，我们的系统的总执行时间只有约1.2秒。我们的系统的执行成本大部分取决于第三阶段，即服务请求阶段的执行。随着授权机构的数量上升，系统的执行时间上升。在第三阶段中，每个授权机构进行身份验证和签名的时间几乎一致。其次，系统执行的稳定性较好，这体现在与实验条件改变无关的测量数据拥有较小的方差，这证明系统的执行较为稳定，在相同条件下不会出现较大的时间差异。

# 第五章 总结与展望

## 5.1工作总结

本文我们设计与实现了一种具有隐私保护的多方授权系统，它的创新之处在于结合了隐私保护和多方授权的相关协议，实现了可靠的多方授权和安全的隐私保护，能够应用于多种现实应用场景，如比特币交易。我们基于过往研究，利用一种多重签名方案实现多方授权，并在授权的认证环节，利用一种基于零知识证明的匿名认证方案实现了隐私保护，理论上这个系统具有足够高的安全性。我们用程序实现了这个系统，并进行了实验测试与分析。在改变授权机构的数量的条件下，用多个实验组测量数据。测得的数据反映出系统的执行时间受授权机构数量的影响较大。总体上来看，我们的系统具有良好的性能和效率。

## 5.2工作展望

我们研究的具有隐私保护的多方授权系统，在以往研究较少。随着近年来互联网越来越多地参与到人们的生活，人们对互联网隐私的重视程度越来越深，我们相信以后相关的研究会更多、更广泛。在此我们从三个方面作出展望。

一、在现实中进行应用。虽然我们的系统经过了实验仿真，但还没有应用于现实场景，例如金融交易、医疗记录共享、供应链管理等对数据安全和权限控制有着严格要求的领域。我们的系统可能能够被应用到这些领域，为相关问题提供有价值的解决方案，但在其实际应用中所面临的需求与挑战，还有待进一步的研究。

二、进一步优化性能与效率。尽管我们的系统在实验中表现良好，但在现实应用中，尤其是在大规模场景下，仍然有改进的空间。通过改进算法、优化数据结构或采用并行计算等技术手段，系统的效率也许能够进一步提升。此外，研究如何减少通信开销、降低计算成本以及优化资源利用等问题也值得关注。

三、扩展系统的功能。本文我们对于如何设计与实现具有隐私保护的多方授权系统，提出了一种可行方案。但是我们的系统还缺少一些实用的功能，例如对合法用户进行身份追踪。这些功能也许能够使我们的系统满足更加广泛的应用需求，它们有待后人的研究。

# 参考文献

1. AC SQUICCIARINI, A CINZIA, SM RAJTMARJER, N ZANNONE, et al. Multi-Party Access Control: Requirements, State of the Art and Open Challenges[C]. The 23nd ACM on Symposium on Access Control Models and Technologies, 2018. 49-49.
2. RS SANDHU, EJ COYNE, HL FEINSTEIN, CE YOUMAN. Role-based access control models[J]. Computer, 1996. 29(2): 38-47.
3. H KINKELIN, H NIEDERMAYER, M MULLER, G CARLE. Multi-party authorization and conflict mediation for decentralized configuration management processes[C]. IFIP/IEEE Symposium on Integrated Network and Service Management, 2019. 5-8.
4. V GOYAL, O PANDEY, A SAHAI, et al. Attribute-based encryption for fine-grained access control of encrypted data[C]. The 13th ACM conference on Computer and communications security, 2006. 89-98.
5. A BOLDYREVA, V KUMAR, V GOYAL, et al. Identity-Based Encryption with Efficient Revocation[C]. The 15th ACM conference on Computer and communications security, 2008. 417-426.
6. B LASHKARI, P MUSILEK. A comprehensive review of blockchain consensus mechanisms[J]. IEEE Access, 2021. 9: 43620-43652.
7. D BONEH, M DRIJVERS, G NEVEN. Compact multi-signatures for smaller blockchains[C]. The 24th International Conference on the Theory and Application of Cryptology and Information Security, 2018.
8. J HAN, L CHEN, S SCHNEIDER, et al. Privacy-preserving electronic ticket scheme with attribute-based credentials[J]. IEEE Transactions on Dependable and Secure Computing, 2019. 18(4): 1836-1849.
9. J CAMENISCH, R CHAABOUNI, A SHELAT. Efficient protocols for set membership and range proofs[C]. The 14th International Conference on the Theory and Application of Cryptology and Information Security, 2008. 234-252.
10. Y XIAO, P ZHANG, Y LIU. Secure and efficient multi-signature schemes for fabric: An enterprise blockchain platform[J]. IEEE Transactions on Information Forensics and Security, 2020. 16: 1782-1794.
11. D BONEH, B LYNN, H SHACHAM. Short signatures from the Weil pairing[C]. The 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, 2001.
12. D BONEH, X BOYEN. Short signatures without random oracles[C]. International Conference on the Theory and Applications of Cryptographic Techniques, 2004.
13. M BELLARE, G NEVEN. Multi-signatures in the plain public-key model and a general forking lemma[C]. The 13th ACM conference on Computer and communications security, 2006. 390-399.
14. BN Jagdale, JW Bakal. A novel authentication and authorization scheme in P2P networking using location-based privacy[J]. Evolutionary Intelligence, 2022. 15(2): 1251-1264.
15. M Wetzels, I Ayoola, S Bogers, et al. Consume: A privacy-preserving authorisation and authentication service for connecting with health and wellbeing APIs[J]. Pervasive and Mobile Computing, 2018. 43: 20-26.
16. J CAMENISCH, M STADLER. Efficient group signature schemes for large groups[C]. The 17th Annual International Cryptology Conference Santa Barbara, 1997.
17. J CAMENISCH, G NEVEN, A SHELAT. Simulatable adaptive oblivious transfer[C]. The 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2007.
18. Y DODIS, A YAMPOLSKIY. A verifiable random function with short proofs and keys[C]. The 8th International Workshop on Theory and Practice in Public Key Cryptography, 2005.
19. AA BATTAH, MM MADINE, H ALZAABI, et al. Blockchain-based multi-party authorization for accessing IPFS encrypted data[J]. IEEE Access, 2020. 8: 196813-196825.
20. KL TAN, CH CHI, KY LAM. Secure and privacy-preserving sharing of personal health records with multi-party pre-authorization verification[J]. Wireless Networks, 2022. 1-23.
21. M MAMUN, A MIYAJI, R LUV, et al. A lightweight multi-party authentication in insecure reader-server channel in RFID-based IoT[J]. Peer-to-Peer Networking and Applications, 2021. 14: 708-721.
22. J SUN, G XU, T ZHANG, et al. Verifiable, Fair and Privacy-Preserving Broadcast Authorization for Flexible Data Sharing in Clouds[J]. IEEE Transactions on Information Forensics and Security, 2022. 18: 683-698.
23. S SHIN, T KWON. A privacy-preserving authentication, authorization, and key agreement scheme for wireless sensor networks in 5G-integrated Internet of Things[J]. IEEE access, 2020. 8: 67555-67571.
24. S LIU, L CHEN, G WU, et al. Blockchain-Backed Searchable Proxy Signcryption for Cloud Personal Health Records[J]. IEEE Transactions on Services Computing, 2023.
25. S GOLDWASSER, S MICALI, C RACKOFF. The knowledge complexity of interactive proof-systems[M]. Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali, 2019. 203-225.
26. B LYNN. PBC library manual 0.5.11[J]. 2006.

**附录A 仿真系统源代码**

以下为主函数所在源文件的代码：

#include <iostream>

#include <pbc.h>

#include <string>

#include <functional>

#include <vector>

#include "RA.h"

#include "AA.h"

#include "USER.h"

#include "SP.h"

#include <openssl/sha.h>

#include "hash.h"

#include <windows.h>

#pragma comment(lib,"libssl.lib")

#pragma comment(lib,"libcrypto.lib")

using namespace std;

bool identification\_verifier(user& USER,authorization\_authority& AA, registration\_authority &RA,pairing\_t pairing,element\_t g1, element\_t g2)

{

//参数初始化

LARGE\_INTEGER t1, t2, tc;

double time;

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t1);

element\_t v, r, V, C, f, t, b, u, Y, c, Zs, Zv, Zr,YY,uu,temp1,temp2,temp3,temp4, temp5, temp6,temp7,temp8,temp9,temp10;

element\_init\_Zr(v, pairing);

element\_init\_Zr(r, pairing);

element\_init\_G1(V, pairing);

element\_init\_G1(C, pairing);

element\_init\_Zr(f, pairing);

element\_init\_Zr(t, pairing);

element\_init\_Zr(b, pairing);

element\_init\_GT(u, pairing);

element\_init\_G1(Y, pairing);

element\_init\_Zr(c, pairing);

element\_init\_Zr(Zs, pairing);

element\_init\_Zr(Zv, pairing);

element\_init\_Zr(Zr, pairing);

element\_init\_GT(uu, pairing);

element\_init\_G1(YY, pairing);

element\_init\_GT(temp1, pairing);

element\_init\_GT(temp2, pairing);

element\_init\_Zr(temp3, pairing);

element\_init\_Zr(temp4, pairing);

element\_init\_Zr(temp5, pairing);

element\_init\_Zr(temp6, pairing);

element\_init\_G1(temp7, pairing);

element\_init\_GT(temp8, pairing);

element\_init\_Zr(temp9, pairing);

element\_init\_GT(temp10, pairing);

QueryPerformanceCounter(&t2);

time = (double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart;

cout << "参数初始化:" << time \* 1000 << "ms" << endl;

//参数设定

QueryPerformanceCounter(&t1);

element\_random(v);

element\_random(r);

element\_pow\_zn(V, USER.get\_certificate(), v);

element\_pow2\_zn(C, g1, USER.get\_number(), g2, r);

element\_random(f);

element\_random(t);

element\_random(b);

pairing\_apply(temp1, V, g1, pairing);

pairing\_apply(temp2, g1, g1, pairing);

element\_neg(temp3, f);

element\_pow2\_zn(u, temp1, temp3, temp2, t);

element\_pow2\_zn(Y, g1, f, g2, b);

QueryPerformanceCounter(&t2);

time = (double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart;

cout << "验证第一阶段：用户发送给授权机构V、C、u、Y:" << time \* 1000 << "ms" <<endl <<"其中V为用户证书的随机化结果（证书的随机数次方），以此实现匿名认证，V的值=";

element\_printf(" % B\n", V);

/\*

验证第一阶段：用户发送给授权机构V、C、u、Y

\*/

QueryPerformanceCounter(&t1);

element\_random(c);

QueryPerformanceCounter(&t2);

time = (double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart;

cout << "验证第二阶段：授权机构发送给用户随机数挑战c:" << time \* 1000 << "ms" << endl;

/\*

验证第二阶段：授权机构发送给用户随机数挑战c

\*/

QueryPerformanceCounter(&t1);

element\_mul(temp4, USER.get\_number(), c);

element\_mul(temp5, v, c);

element\_mul(temp6, r, c);

element\_sub(Zs, f, temp4);

element\_sub(Zv, t, temp5);

element\_sub(Zr, b, temp6);

QueryPerformanceCounter(&t2);

time = (double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart;

cout << "验证第三阶段：用户发送给授权机构 Zs Zv Zr:" << time \* 1000 << "ms" << endl;

/\*

验证第三阶段：用户发送给授权机构 Zs Zv Zr

\*/

//接下来是验证第四阶段：授权机构验证两等式是否成立

QueryPerformanceCounter(&t1);

element\_pow3\_zn(temp7, C, c, g2, Zr, g1, Zs);

pairing\_apply(temp8, V, RA.y, pairing);

element\_neg(temp9, Zs);

element\_pow3\_zn(temp10, temp8, c, temp1, temp9, temp2, Zv);

QueryPerformanceCounter(&t2);

time = (double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart;

cout << "验证第四阶段：授权机构验证两等式是否成立:" << time \* 1000 << "ms" << endl;

if (!element\_cmp(Y, temp7) && !element\_cmp(u, temp10))

return true;

else

return false;

}

bool multisig\_verifier(service\_provider& SP,user& USER, vector<authorization\_authority>& AA\_vec, vector<element\_t>& a\_vec,element\_t& h1,pairing\_t& pairing)

{

LARGE\_INTEGER t1, t2, tc;

double time;

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t1);

element\_t apk;

element\_init\_G2(apk, pairing);

element\_set1(apk);

for (int i = 0; i < AA\_vec.size(); i++)

{

element\_t temp;

element\_init\_G2(temp, pairing);

element\_pow\_zn(temp, AA\_vec[i].pk, a\_vec[i]);

element\_mul(apk, apk, temp);

}

element\_t temp1, temp2, temp3, temp4, one;

element\_init\_G2(temp1, pairing);

element\_invert(temp1, h1);

element\_init\_GT(temp2, pairing);

element\_init\_GT(temp3, pairing);

element\_init\_GT(temp4, pairing);

element\_init\_GT(one, pairing);

pairing\_apply(temp2, USER.get\_multisig(), temp1, pairing);

pairing\_apply(temp3, USER.get\_hash\_of\_message(), apk, pairing);

element\_mul(temp4, temp2, temp3);

element\_set1(one);

if (!element\_cmp(temp4, one))

{

QueryPerformanceCounter(&t2);

time = (double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart;

cout << "验证耗费时间:" << time \* 1000 << "ms" << endl;

return true;

}

else

{

QueryPerformanceCounter(&t2);

time = (double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart;

cout << "验证耗费时间:" << time \* 1000 << "ms" << endl;

return false;

}

}

void signer(int i,vector<authorization\_authority>&AA\_vec ,user& USER,vector<element\_t>&a\_vec , vector<element\_t>& sig\_vec,pairing\_t& pairing)

{

LARGE\_INTEGER t1, t2, tc;

double time;

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t1);

element\_t ai, si;

element\_init\_Zr(ai, pairing);

element\_init\_G2(si, pairing);

int pk\_bytes\_length = element\_length\_in\_bytes(AA\_vec[i].pk);

int total\_length = (AA\_vec.size() + 1) \* pk\_bytes\_length;

unsigned char\* data = reinterpret\_cast<unsigned char\*>(malloc(total\_length));

unsigned char\* dest = data;

unsigned char\* temp = reinterpret\_cast<unsigned char\*>(malloc(pk\_bytes\_length));

element\_to\_bytes(temp, AA\_vec[i].pk);

memcpy(data, temp, pk\_bytes\_length);

data += pk\_bytes\_length;

std::free(temp);

for (int i = 0; i < AA\_vec.size(); i++)

{

unsigned char\* temp2 = reinterpret\_cast<unsigned char\*>(malloc(pk\_bytes\_length));

element\_to\_bytes(temp2, AA\_vec[i].pk);

memcpy(data, temp2, pk\_bytes\_length);

data += pk\_bytes\_length;

std::free(temp2);

}

unsigned char hash\_of\_dest[64];

std::memset(hash\_of\_dest, 0, 64);

HASH1(dest, hash\_of\_dest, total\_length);

element\_from\_hash(ai, hash\_of\_dest, 64);

element\_set(a\_vec[i], ai);

AA\_vec[i].sign(si, USER.get\_hash\_of\_message(), ai, pairing);

element\_set(sig\_vec[i], si);

std::free(dest);

QueryPerformanceCounter(&t2);

time = (double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart;

cout << "授权机构" << i + 1 << "成功签名,签名si=" ;

element\_printf(" % B\n", si);

cout <<endl<< "签名过程耗费" << time \* 1000 << "ms" << endl;

}

int main()

{

//一.系统初始化

//初始化pairing

LARGE\_INTEGER t1, t2, tc,t3,t4,t5,t6;

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t3);

string command="y";

QueryPerformanceCounter(&t5);

cout << "一.系统初始化阶段:" << endl<< "将运行双线性群生成器生成双线性群，输入y继续,输入n退出:";

do {

cin >> command;

} while (command != "y" && command != "n");

if (command == "n")

return 0;

QueryPerformanceCounter(&t1);

pbc\_param\_t par;

int rbits = 160, qbits = 512;

pairing\_t pairing;

pbc\_param\_init\_a\_gen(par, rbits, qbits);

pairing\_init\_pbc\_param(pairing, par);

QueryPerformanceCounter(&t2);

double time = (double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart;

cout << "双线性群生成成功！所用时间为" << time\*1000 <<"ms" << endl;

//初始化生成元

QueryPerformanceCounter(&t1);

element\_t g1, g2, h1, h2;

element\_init\_G1(g1, pairing);

element\_init\_G1(g2, pairing);

element\_init\_G2(h1, pairing);

element\_init\_G2(h2, pairing);

element\_random(g1);

element\_random(g2);

element\_random(h1);

element\_random(h2);

QueryPerformanceCounter(&t2);

time = (double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart;

cout << "生成元初始化成功！所用时间为" << time\*1000 << "ms" << endl;

cout <<"G1的生成元如下"<<endl;

cout << "g1=";

element\_printf(" % B\n", g1);

cout <<endl<< "g2=";

element\_printf(" % B\n", g2);

cout <<endl<< "G2的生成元如下" << endl;

cout << "h1=";

element\_printf(" % B\n", h1);

cout << endl << "h2=";

element\_printf(" % B\n", h2);

//初始化哈希函数

cout << endl << "本实验程序默认使用sha256作为哈希函数H0,sha512作为H1"<<endl<<"输入y继续,输入n退出:";

do {

cin >> command;

} while (command != "y" && command != "n");

if (command == "n")

return 0;

cout << "请设定本次实验中用户所请求的消息m，可以是任意字符串但不能为空:" ;

string m="这是用户请求服务提供者提供的消息";

do {

cin >> m;

} while (m == "");

cout << "请设定本次实验中授权机构的数量(推荐20以内的正整数):";

int how\_many\_AA =5;

do {

cin >> how\_many\_AA;

if (how\_many\_AA <= 0)

cout << "请输入一个正整数:";

} while (how\_many\_AA <= 0);

QueryPerformanceCounter(&t6);

time = (double)(t6.QuadPart - t5.QuadPart) / (double)tc.QuadPart;

cout << "第一阶段总时长"<<time\*1000<<"ms"<<endl;

//二.局部参数初始化和用户注册

QueryPerformanceCounter(&t5);

cout << "二.局部参数初始化和用户注册阶段:" << endl<< "首先初始化注册机构、授权机构、用户及其参数，输入y继续,输入n退出:";

do {

//cin >> command;

} while (command != "y" && command != "n");

if (command == "n")

return 0;

//初始化注册机构、授权机构、用户及其参数

registration\_authority RA(pairing, g1);

cout << "输入y继续，输入n退出:";

do {

//cin >> command;

} while (command != "y" && command != "n");

if (command == "n")

return 0;

LARGE\_INTEGER t7, t8;

user USER(pairing);

vector<authorization\_authority>AA\_vec;

QueryPerformanceCounter(&t7);

for (int i = 0; i < how\_many\_AA; i++)

AA\_vec.push\_back(authorization\_authority(pairing, h1,i+1));

QueryPerformanceCounter(&t8);

time = (double)(t8.QuadPart - t7.QuadPart) / (double)tc.QuadPart;

cout << "授权机构生成公私钥总时间：" << time \* 1000 << "ms" << endl;

cout << "输入y继续，输入n退出:";

do {

cin >> command;

} while (command != "y" && command != "n");

if (command == "n")

return 0;

//注册机构为用户派发证书和编号

RA.issue\_certificate(USER, pairing, g1);

QueryPerformanceCounter(&t6);

time = (double)(t6.QuadPart - t5.QuadPart) / (double)tc.QuadPart;

cout << "第二阶段总时长" << time \* 1000 << "ms"<<endl;

//三.服务请求

//服务请求者将消息的哈希值传递给用户

QueryPerformanceCounter(&t5);

cout << "三.服务请求阶段:" << endl << "首先服务请求者将消息的哈希值传递给用户，输入y继续,输入n退出:";

do {

cin >> command;

} while (command != "y" && command != "n");

if (command == "n")

return 0;

service\_provider SP(m);

USER.set\_hash\_of\_message(SP.hash\_of\_message(pairing));

//初始化签名数组，a\_vec里是生成签名的重要参数，后面验证还会用到所以直接保存到数组里了

vector<element\_t>sig\_vec(AA\_vec.size());

vector<element\_t>a\_vec(AA\_vec.size());

for (int i = 0; i < AA\_vec.size(); i++)

{

element\_init\_G2(sig\_vec[i], pairing);

}

for (int i = 0; i < AA\_vec.size(); i++)

{

element\_init\_Zr(a\_vec[i], pairing);

}

cout <<AA\_vec.size()<< "个授权机构将验证用户身份并提供签名，输入y继续,输入n退出:";

do {

cin >> command;

} while (command != "y" && command != "n");

if (command == "n")

return 0;

//每个授权机构给用户签名，这里直接是指定的所有10个授权机构都需要签名

LARGE\_INTEGER t9, t10;

QueryPerformanceCounter(&t9);

for (int i = 0; i < AA\_vec.size(); i++)

{

cout << "-----------------------------------------------------------------" << endl ;

cout <<"授权机构" << i + 1 << "验证用户身份..."<<endl;

//验证用户身份

if (identification\_verifier(USER, AA\_vec[i], RA, pairing, g1, g2))

cout << "授权机构" << i + 1 << "成功验证用户身份" << endl;

else

{

cout<< "授权机构" << i + 1 << "验证用户身份失败！" << endl;

return 0;

}

//验证完毕后签名

signer(i, AA\_vec, USER, a\_vec, sig\_vec, pairing);

}

cout << "-----------------------------------------------------------------" << endl << endl;

QueryPerformanceCounter(&t10);

time = (double)(t10.QuadPart - t9.QuadPart) / (double)tc.QuadPart;

cout << "授权机构验证与签名总时长：" << time \* 1000 << "ms" << endl;

//计算出多签名

cout << "用户已获取全部签名，将聚合生成多签名，输入y继续,输入n退出:";

do {

cin >> command;

} while (command != "y" && command != "n");

if (command == "n")

return 0;

QueryPerformanceCounter(&t1);

element\_t multisig;

element\_init\_G2(multisig,pairing);

element\_set1(multisig);

for (int i = 0; i < sig\_vec.size(); i++) {

element\_mul(multisig, multisig, sig\_vec[i]);

}

USER.set\_multisig(multisig);

QueryPerformanceCounter(&t2);

time = (double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart;

cout << "多签名计算耗时" << time \* 1000 << "ms，值为:";

element\_printf(" % B\n", multisig);

QueryPerformanceCounter(&t6);

time = (double)(t6.QuadPart - t5.QuadPart) / (double)tc.QuadPart;

cout << "第三阶段总时长" << time \* 1000 << "ms"<<endl;

//四.验证阶段

QueryPerformanceCounter(&t5);

cout << endl<<"四.验证阶段:" << endl ;

//服务提供者验证多签名有效性

cout << "服务提供者将验证多签名有效性，输入y继续,输入n退出:" ;

do {

cin >> command;

} while (command != "y" && command != "n");

if (command == "n")

return 0;

if (multisig\_verifier(SP, USER, AA\_vec, a\_vec, h1, pairing))

{

cout << "服务提供者成功验证多签名，将提供给用户以下消息：" << endl;

SP.print\_message();

}

else

{

cout << "无效的多签名！" << endl;

}

//done

QueryPerformanceCounter(&t6);

time = (double)(t6.QuadPart - t5.QuadPart) / (double)tc.QuadPart;

cout << "第四阶段总时长" << time \* 1000 << "ms"<<endl;

QueryPerformanceCounter(&t4);

time = (double)(t4.QuadPart - t3.QuadPart) / (double)tc.QuadPart;

cout << "总时长" << time \* 1000 << "ms";

cout << endl<<"输入y退出:";

do {

cin >> command;

} while (command != "y" );

}

以下为所有头文件的代码，每个头文件都以” #pragma once”开头：

#pragma once

#ifndef \_\_AA\_H\_\_

#define \_\_AA\_H\_\_

#include <iostream>

#include <pbc.h>

#include <string>

#include <functional>

class authorization\_authority {

private:

element\_t sk;

int number;

public:

element\_t pk;

authorization\_authority(pairing\_t pairing, element\_t h1,int number);

void print();

void sign(element\_t& sig, element\_t hash\_of\_message, element\_t ai, pairing\_t pairing);

};

#endif

#pragma once

#ifndef \_\_HASH\_H\_\_

#define \_\_HASH\_H\_\_

int HASH0(unsigned char\* readstr, unsigned char\* md, int length);

int HASH1(unsigned char\* readstr, unsigned char\* md, int length);

void printfHASH0(unsigned char\* md);

void printfHASH1(unsigned char\* md);

#endif

#pragma once

#ifndef \_\_RA\_H\_\_

#define \_\_RA\_H\_\_

#include <iostream>

#include <pbc.h>

#include <string>

#include <functional>

#include "USER.h"

class user;

class registration\_authority {

private:

element\_t x;//私钥

public:

element\_t y;//公钥

registration\_authority(pairing\_t pairing, element\_t g1);//初始化

void issue\_certificate(user &User, pairing\_t pairing, element\_t g1);//给用户颁发证书

};

#endif // !\_\_RA\_H\_\_

#pragma once

#ifndef \_\_SP\_H\_\_

#define \_\_SP\_H\_\_

#include <iostream>

#include <pbc.h>

#include <string>

#include "hash.h"

using namespace std;

class service\_provider {

private:

string message;

element\_t hashMessage;

public:

service\_provider(const string& m = "这是用户需要的消息");

element\_t& hash\_of\_message(pairing\_t pairing);

void print\_message() { cout << message << endl; }

};

#endif

#pragma once

#ifndef \_\_USER\_H\_\_

#define \_\_USER\_H\_\_

#include <iostream>

#include <pbc.h>

#include <string>

#include <functional>

#include "RA.h"

using namespace std;

class user {

private:

element\_t number;//编号

element\_t certificate;//证书

element\_t hash\_of\_message;//消息的hash值

element\_t multisig;//多签名

public:

user(pairing\_t pairing);//初始化

void set\_number(element\_t n) { element\_set(number, n); }

void set\_certificate(element\_t c) { element\_set(certificate, c); }

void set\_hash\_of\_message(element\_t h) { element\_set(hash\_of\_message, h); } //cout << element\_printf("%B\n",hash\_of\_message); }

void set\_multisig(element\_t n) { element\_set(multisig, n); }

element\_t& get\_certificate() { return certificate; }

element\_t& get\_number() { return number; }

element\_t& get\_hash\_of\_message() { return hash\_of\_message; }

element\_t& get\_multisig() { return multisig; }

void print() {

element\_printf("%B\n", hash\_of\_message);

}

};

#endif

# 致 谢

感谢东南大学和我所在的网络空间安全学院对我的栽培。感谢我的毕业设计导师韩金广老师，在我的研究过程中给予我宝贵的指导与建议。