

本科生毕业论文（设计）



题 目  **基于Netty的高性能RPC服务器设计与实现**

学 院  **软件学院**

专 业  **软件工程 080902**

学生姓名  **邹 绿 州**

学 号  **2014141463318** 年级 **2014**

指导教师  **左航**

教务处制表

二Ο一七年十二月十五日

**基于Netty的高性能RPC服务器设计与实现**

软件工程

学生 邹绿洲 指导老师 左航

**[摘要]** 当今时代，互联网日益成为我们生产生活的必备工具，电子投票也逐渐开始取代传统的纸质投票。电子投票具有电子化和网络化、低成本化、快速性和便捷性、高度匿名性等优点，能够发挥保证民主的广泛参与性。选举、投票中层出不穷的各种急需解决的问题，说明了对安全的、公正的电子选票系统的需求与日俱增。

本作品在B／S架构下设计并实现了一种安全高效的电子投票系统。结合电子投票的基本原理，采用关联环签名技术实现匿名注册，利用盲签名盲化选票和引入身份序列码保证选票唯一性的方法，设计了一个抗第三方欺诈的安全电子投票系统。该系统不仅实现了投票协议应具备的基本安全要求，并且具有抗选票碰撞、可在任意阶段弃权、不依赖于任何可信第三方和匿名通信信道的功能。

本系统与传统的电子投票系统相比，在各个环节中避免了人工因素对最后投票的真实有效性的干扰。并且通过计算机统计，可以大大提高最后对选票结果统计的速度，从而能够极大的节约人力物力以及时间，使得每场电子投票的结果能够更加的公正公开，普遍为投票人所相信。

本项目开发出的作品参加2017年全国大学生信息安全大赛获得了三等奖。

**[主题词]** 远程过程调用; Netty

**Safe Voting-System Based on Linkable-Ring-Signature**

Software Engineering

Student： Xu Qinkun Adviser: Liang Gang

**[Abstract]** In today's era, the Internet has become a necessary tool for our production and life, and electronic voting has gradually begun to replace traditional paper voting.Electronic voting has the advantages of being electronic and networked, low-cost, fast and convenient, highly anonymized, and can play a wide role in ensuring democracy.The urgent need to address the issues of the election and the middle of the vote illustrates the growing need for a secure, just electronic ballot system.

This paper designs and implements a safe and efficient electronic voting system in B/S mode.

In combination with the basic principle of electronic voting, associated ring signature technology was adopted to realize the anonymous registration, blind signature blind to make use of the vote and introducing identity sequence code to ensure uniqueness, the method of designing a fraud against a third party security electronic voting systems. The system not only realizes the voting agreement should have the basic safety requirements, and have vote collision resistance, can be in any stage abstained, do not rely on any trusted third party and anonymous communication channel function.

Compared with the traditional electronic voting system, the system avoids the interference of the artificial factor in the final voting. And through the computer statistics, can significantly improve the speed of statistics, the result of the ballot so that they can greatly save manpower and time, which makes the results of each electronic voting more fair and open, generally by voters believe.

The work developed by this project won the third prize in the 2017 national college students information security competition.

**[Key Words]** Electronic voting; Linkable-ring signature; Blind signature

目 录

[1 绪论 1](#_Toc502054735)

[1.1 项目背景 1](#_Toc502054736)

[1.2 国内外研究现状 3](#_Toc502054737)

[1.2.1 RPC研究现状 3](#_Toc502054738)

[1.2.2 Netty研究现状 4](#_Toc502054739)

[1.3 项目主要工作 4](#_Toc502054740)

[1.3.1 项目概述 4](#_Toc502054741)

[1.3.2 功能描述 5](#_Toc502054742)

[1.3.3 创新点及特色 5](#_Toc502054743)

[1.4 报告组织与结构 7](#_Toc502054744)

[2 项目涉及的相关知识和技术简介 8](#_Toc502054745)

[2.1数字签名理论 8](#_Toc502054746)

[2.2盲签名理论 11](#_Toc502054747)

[2.3环签名理论 12](#_Toc502054748)

[2.4 WEB技术 13](#_Toc502054749)

[2.4.1 前端技术 13](#_Toc502054750)

[2.4.2 后端技术 13](#_Toc502054751)

[2.5本章小结 16](#_Toc502054752)

[3 项目工作和进展情况 17](#_Toc502054753)

[3.1项目的概况 17](#_Toc502054754)

[3.2项目的开发内容和结果 17](#_Toc502054755)

[3.2.1 系统架构设计 17](#_Toc502054756)

[3.2.2 系统概要设计 19](#_Toc502054757)

[3.2.3 系统详细实现 24](#_Toc502054758)

[3.3项目开发计划及阶段性完成情况 31](#_Toc502054759)

[3.3.1 项目开发计划 31](#_Toc502054760)

[3.3.2 项目完成情况 32](#_Toc502054761)

[3.4本章小结 39](#_Toc502054762)

[4 讨论与体会 40](#_Toc502054763)

[4.1对项目开发过程的体会 40](#_Toc502054764)

[4.2对项目的评价 40](#_Toc502054765)

[4.3本章小结 41](#_Toc502054766)

[5 小结 42](#_Toc502054767)

[参考资料 43](#_Toc502054768)

声 明 [45](#_Toc502054769)

[致 谢 46](#_Toc502054770)

[附录1 任务申请书 47](#_Toc502054771)

[附录2 可行性研究报告 48](#_Toc502054772)

[附录3 项目开发计划 49](#_Toc502054775)

[附录4 数据要求说明书 51](#_Toc502054778)

[附录5 需求规格说明书 54](#_Toc502054782)

[附录6 用户手册概要 57](#_Toc502054787)

[附录7 概要设计说明书 68](#_Toc502054792)

[附录8 数据库设计说明书 74](#_Toc502054795)

[附录9 组装测试计划 79](#_Toc502054799)

[附录10 详细设计说明书 104](#_Toc502054802)

[附录11 模块开发说明 122](#_Toc502054805)

[附录12 单元测试 147](#_Toc502054808)

# 绪论

现在的互联网企业的产品往往都是体量非常巨大的，这些产品都是有若干个子项目一起组成。这些子项目往往由各个团队不同的开发人员进行合作开发，这些子项目又提供各种各样的服务。各个服务提供有不同的功能，处于系统的不同位置，有的偏向于底层，用的偏向于表现层。

根据软件设计架构的理论，以及出于可复用组件的考虑，往往在一个组件中提供的功能就不用再另一个组件中重复提供。也就是说，当A团队开发一个项目时，需要的功能模块或者说服务在B团队开发的项目中已经存在，那么A团队就不需要再进行重复开发，也就是我们平常说的“重复造轮子”。但是如何获得B团队项目中提供的功能呢？很早以前，我们采取的办法是直接拷贝代码到项目中，这未尝不是一种解决方案，但是这种方式会有如下问题：

问题一：直接拷贝代码可能导致项目代码量急剧增大，如果依赖的其他项目过多，将会使得项目臃肿不堪，难以维护。

问题二：由于开发依赖代码的并不是本团队成员，后期代码维护将是一个重大难题，可能因为依赖代码的不稳定导致项目的风险急剧增大。

问题三“如果依赖的服务需求发生变化，或者需要升级功能，对于拷贝的代码来说将会无能为力。

问题四：如果依赖的功能或者服务出现bug，只能由原开发人员进行修复或者需要其进行帮助修复，这往往会带来不必要的沟通成本和人力成本。

上面列出的问题只是可能出现的一部分问题，除了在这个方向上进行的思考，我们还应该从另一个方向上进行思考，即：这样的做法是否符合软件工程的要求。软件工程理论要求我们构建有效的，可维护的软件，而这样做肯定不符合可维护的要求。

而一些RPC框架的出现就可以很好的解决上述出现的问题，服务A和服务B可以通过网络传输来调用对方的函数或者数据，也更好的提高了代码的可重用性，减少了系统模块功能的复杂性。

## 项目背景

当下的互联网公司中绝大多数都已经的广泛使用各种RPC框架进行服务间的通信，是各个项目不可分割的组成部分，比如使用世界互联网科技巨头Google开源的grpc框架以及社交网络巨头Facebook开源的Thrift框架。在国内有以电子商务起家的阿里巴巴集团开源的dubbo框架最具代表性，此外还有搜索科技公司百度开源的brpc,以及新浪开源的轻量级RPC框架Motan等。

不管是国内还是国外，各大互联网公司都在大量的使用RPC框架进行项目的构建，随着技术的发展，以及互联网产品用户量和数据量的急剧提高，怎样更好的设计一套更优秀的RPC框架将会是程序开发人员和架构师们不断追求的目标。而作为开发人员，怎样更好的理解RPC的原理，怎样在实际开发工作中利用好现有的RPC框架，怎样能够让RPC框架能够发挥更好的性能，都是非常重要的问题。

根据腾讯大数据发布的《腾讯2017代码报告》显示，腾讯2017年新增代码超过5亿行，项目总数31908个。在一个大型的互联网公司中，这么多的项目肯定不是一一独立的，其中有专门针对某个业务的部门，比如我们每天使用的微信，但是微信是一个体量非常大的应用，这个应用肯定需要多个项目最支撑，如与信息安全相关的，需要专门做信息安全的团队；与数据处理和数据分析相关的，需要大数据团队；与风险控制相关的，需要风控团队；与基础架构相关的，需要基础架构团队。而其中，可能大数据项目需要依赖基础架构项目提供的基础设施服务，也可能需要风险控制项目提供的服务。

在一个互联网公司中，做中间件的团队通常是为整个集团的各个项目提供服务的，比如消息队列，分布式缓存，分布式搜索，分布式计算以及云服务等等。

这些所有的场景，归根结底都涉及到一个重要的问题，即：怎样在一个项目中调用另一个项目提供的服务？或者说，怎样在一台机器上调用另一台机器提供得函数或者方法呢？我们知道，不同的项目是运行在不同的服务器上的，也就是其拥有的内存空间是不同的，当然不能直接进行调用。这时，我们自然想到了使用网络进行传输，即使用网络传输我们需要调用的方法或者说需要的数据。而这正是RPC所要解决的重要问题。

俗话说，授人以鱼不如授人以渔，为了理解RPC的底层原理，了解其设计思想以及改进现有的RPC框架，我们完全可以自己动手开发一套RPC框架，这将是一个很有用的尝试。为了使工作更加有效和节约，本文选择几个重要的方向进行研究，避免重复造同样的轮子。

【基于Netty的RPC服务器的优势】Netty是一个

【多种序列化协议进行支持】像总统或者主席选举，这样大规模的投票事件，选民中有很大部分选民为海外工作选民，并且很多意外状况的发生会导致部分选民无法到达投票地点进行投票，或者是错过投票截止时间，这样的意外严重影响了投票的公平性。

【】

## 国内外研究现状

### **1.2.1 RPC框架研究现状**

电子投票方案的研究已经以及有长达三十多年的历史，Chaum等人于1981年提出了第一个具有密码学意义的的电子投票方案 [2]，它采用公钥密码方案，利用数字签名使得投票人身份得以隐藏，并通过计算机和网络来完成整个计算过程。1985年，Cohen Benaloh和Michael J. Fischer等人提出了基于同态加密技术的电子投票方案[9]。接着Benaloh,Yung等人在Cohen方案的基础上提出了优化计算效率的方案。

近年来，随着学者不断研究，涌现出众多电子投票方案；然而，有些方案计算量巨大，难以应用于大规模选举活动中，而有些则在安全方面存在着较大的漏洞。Fujioka，Okamoto和Ohta于1992年最先提出了适合大规模投票的方案，方案的核心在于运用了比特承诺技术和盲签名技术[10]。至此，电子投票系统在实用性方面得到了至关重要的突破，并且在社会各界得到了十分广泛的应用。之后，许多大学和研究机构不断地对该方案进行了改进，开发出了一些安全电子投票软件系统。其中较为著名的有MIT开发的EVOX系统和华盛顿大学开发的Sensu系统。

在电子投票系统的研究和发展过程中，研究者们又设计了适合不同场合目的的电子投票系统。为了解决电子投票方案中的问题，Cramer[11] 提出的投票方案，利用了门限同态加密、电子公告牌和零知识证明等技术，具有效率高，匿名性强和广义验证性等优点。随后，为了避免投票过程中发生的选民买卖选票、候选人强迫投票等不法行为，Benaloh[12]提出了电子投票 “无收据性”的概念，即投票者不能够向第三方显示其投票给了哪一位候选人。在此基础上，Martin证明了在选民可以多次投票的情况下，Benaloh的方案不具备这一特性。Lee在Cramer方案基础上进行改进[13]，引入了一个“可信任的第三方”作为验证，以此实现无收据性，但在实际投票过程中，“可信任的第三方”是难以达到的条件[14]。

### **1.2.2 Netty研究现状**

Rivest和Shamir等人在其论文《How to leak a secret》中首次提出了环签名方案[15]。在该论文中，作者描述了这样一个场景：某位内阁成员想要举报首相的不法行为，如果这位内阁成员以实名的方式举报首相那么很有可能遭到首相的报复。但如果内阁成员以匿名的方式举报首相，那么这样的举报很难引起记者的注意，甚至会被认为是恶意中伤。这样就陷入了两难的境地，作者为了解决类似于这样的难题，提出了环签名方案。

环签名方案本质上是一种群签名方案。不同的是，环签名不存在群管理员，无需组内成员之间的任何合作。组内任何一个成员想要生成环签名，只需要使用其他所有成员的公钥和自己的私钥即可生成环签名，不需要其他成员的任何帮助，而且不会受到任何人的干扰。当接收者受到该成员的签名后，只能验证其是否出自该“环”，无法得知签名人的确切身份。

普通的群签名方案中群管理权限过大，导致群签名容易被跟踪找到签名人的具体信息[16]。环签名的出现，解决了这一难题。环签名具有使签名者无条件匿名的特点，验证者唯一能够确定的是该签名出自环中的某一成员，但无从得知该成员的确切身份，从而达到了匿名的要求。如今，环签名在匿名电子投票，情报组织，电子商务，发腐等领域都有着十分广泛的应用[17]。近年来，涌现出一大批学者从事环签名的研究。然而，在国内对环签名的研究仍然处于起步阶段。并且，在各种环签名方案被提出的同时，也存在一些继续解决的问题，正是这些问题阻挡了其投入商业使用的进程。如：环签名的长度总是随着环中全体成员的数量线性增长，环签名的安全性仍有待提高[18]。

关联环签名是在环签名的基础上，增添了关联标签[19]。使得验证者能确定不同签名是否来自于同一签名者[20]。然而，现有的关联环签名方案均是建立在离散对数公钥密码体制上，且绝大多数方案仅具备弱匿名性的特点。因此，如何设计出基于其他公钥密码体制、具有强匿名性的关联环签名具有重要的理论价值和研究意义[21]。

## 项目主要工作

### **1.3.1 项目概述**

本系统实现了一个基于关联环签名的安全电子投票系统。采用统一身份识别码解决了选票碰撞这样的问题。同时，通过在投票过程中引入可公开验证的信息公告栏机制，允许选民在任意投票阶段中途弃权，并避免了第三方机构的不法行为对投票过程的破坏及对选票结果的干扰。此外，将具有较大计算量的关联环签名放到注册阶段，在投票和计票阶段则利用基于身份序列码的普通个体签名进行身份验证，无需借助任何匿名通信信道，有效地提高了系统运行效率，适合于大规模的选举活动。

### **1.3.2 功能描述**

本系统提供了如下功能：发起投票，匿名投票，查看结果，公开验证。

**表1.1：系统具体功能列表**

| ***功能*** | ***详细描述*** |
| --- | --- |
| **发起投票** | 投票中心在网站上发起投票，填写投票主题等相关信息，之后添加投票者邮件地址，网站对每位投票者发起邀请。 |
| **匿名投票** | 在投票之前，投票者接受到邮件邀请，之后利用投票者消息进行匿名注册，投票中心只能确认其为合法投票者之一，但无法确定投票者真实身份。  确认身份后投票者获得唯一标识身份的序列码，进入投票页面进行投票。 |
| **查看结果** | 投票结束后，投票者和投票中心可以前往网站投票结果页面查看本次投票结果。 |
| **公开验证** | 在选举的中间过程，选举的中间结果不会被泄漏，且投票的状态变化公布在网站的公告信息表中，投票人和投票中心可凭借手中持有信息对信息表进行公开验证，对各阶段进行有效监督。 |
| **双机热备** | 使用两台服务器，对运行过程中产生的数据在两台服务器上同时保存，当一台服务器出现故障时，由于负载均衡的作用，可以立即切换到由另一台服务器提供服务。 |

### **1.3.3 创新点及特色**

对比于其他电子投票系统，本项目的创新点在于：

1. **具有灵活性，选民可在任意阶段弃票**

投票者在任意投票环节弃权，并不会影响选举的正常进行。每一个阶段的操作的数据，都基于上一个阶段，因此当投票者没有进行前一个操作，对应的数据不会出现在下一个阶段的信息表中，则该投票者的投票不会被计算，则投票者在任何阶段的弃票都不会影响下一个阶段。

在注册阶段，用户部分信息会公布在于List1上，包含标识用户身份的序列码。只有具有序列码的投票者才能进行下一个阶段--投票阶段，因此在注册阶段弃票不会影响后续投票功能。

同时只有在投票后公布的List2中出现了序列码的投票者的选票才能在下一个阶段被计算，因此在投票阶段弃权不会影响后续计票。

在计票阶段，投票人将消盲后的选票公布于计票信息表List3中，系统根据List3进行计票。超过计票期限后，如果投票者仍然没有将选票公布于List3中，则被视为弃权，不参与选票的计算。因此，投票者在任何阶段弃权都不会影响整个系统的工作，则本产品具有稳固性。

1. **不依赖任何第三方和匿名通道，实现公平可验证性**

对投票者进行无条件匿名注册，同时为投票者发放统一身份识别码用于后续的投票及计票过程，任何其他验证者只能确定投票者的合法性而无法得知其真实身份，因此不需要借助匿名通道。

同时，每个过程都接受监督检验，任何不诚实行为也可以被制止。

从整个系统的角度，涉及到投票者信息的投票中心会将信息公布到对应的公告信息表中。

在投票阶段，投票中心将｛消息，对消息的关联环签名，对消息的普通个体签名，Ni｝等数据加入到注册信息表List1中，同时公布List1，接受选民公开验证；在注册截止日期内，未能及时注册的投票者可以公布自己的签名，要求注册中心为其发放身份序列码和对应签名，并公布其注册信息于List1表中；同时已注册者可以在List1中查询自己的信息；如果查询不到，投票者可以公开自己的签名列表{明文消息，对消息的关联环签名，对消息的普通个体签名，Ni}，并公开验证签名。对消息的关联环签名和私钥对消息进行的普通个体签名的正确性，进而要求投票中心在所有人的监督下为其发放身份序列码和对应签名，并公布投票者相应信息于List1上。能够有效防止注册中心的不诚实行为。

在投票阶段，将序列号、选票、选票的签名数据存入List2。投票结束后投票管理机构公布List2，接受选民的公开验证。在投票截止日期内，未能及时投票的选民可以公开投票数据，公布自己的盲化选票和签名内容，要求投票中心为其发送签名并将其投票结果公布于List2中；同时已投票者查询List2，如果查询不到自己的投票信息，投票者可以公布自己的身份序列码、盲化选票和签名，在大众的监督下，要求投票中心为其签名并发送选票、选票的签名，并将其投票结果公布到List2中。能够有效防止数据管理机构的不诚实行为。

在计票阶段，投票人将消盲后的选票公布于计票信息表List3中，能够有效防止投票中心的不诚实行为。

从而，整个系统的各个模块独立、公开、透明，接受众人的监督，具有公开可验证性。

从投票者角度，任何不诚实的投票行为也会被发现和制止，也具有公平可验证性。

首先，保证了投票人的身份合法具有注册资格，进而获得注册机构为其分配的唯一标识身份的序列码Ｎ。其次，在其后的投票和计票环节中，都使用Ni，且Ni分别公布在List2，List3中，如投票者重复投票，则会在对应表中出现冗余，从而杜绝投票者重复投票行为。

因此，每个过程都将接受大家的监督，即可以防止第三方泄密，同时也具有公平可验证性。

1. **提升了关联环签名验证身份的效率，适用于大规模选举**

环签名的很难应用于大规模投票的主要原因是，在签名生成阶段会使用大规模的模幂运算，计算非常耗时。为此，本项目将计算耗时的签名生成放在注册阶段，实现合法投票者的匿名注册，并通过为合法投票者发放一个唯一的身份识别码Ｎ，并结合自己产生的公私钥对，用于后续投票和计票阶段的匿名身份认证。由于本项目在投票阶段仅使用公私钥对实现普通的个体签名及验证，因而有效地保证了投票阶段的高效性和实用性，适合于大规模选举。

## 报告组织与结构

第一部分：绪论。主要介绍了电子投票的发展和应用，以及本报告的研究背景和研究工作，电子投票系统的设计目的；

第二部分：项目相关理论。对设计电子投票技术所采用的技术进行了介绍。研究分析了电子投票系统的技术原理；

第三部分：电子投票系统设计工作和进展情况。

第四部分：项目的讨论与体会。

第五部分：小结。本章对全文工作进行总结，指出了还需改进的地方。

# 项目涉及的相关知识和技术简介

## 2.1数字签名理论

**1.数字签名的发展**

随着计算机和互联网技术的发展，人们越来越多地利用网络进行交流。这其中包含一些十分重要且机密的信息，由于TCP/IP协议的不安全性，对数据的加密传输显得十分重要[21]。这其中涌现出诸多技术，如对称加密技术，非对称加密技术，防火墙，安全协议以及数字签名等[22]。其中，数字签名的使用十分广泛，在电子商务、电子投票以及政府机构等领域用的最多。数字签名是进行身份认证的重要技术，它保证了发送信息的完整性，发送者对发送的信息无法否认并且可以确定消息发送者的身份[22]。

**2.数字签名的功能**

数字签名，顾名思义，就是对消息进行电子的签名。类似于日常生活中，我们对重要文件进行的签名。一旦我们签署了某项文件，那么这个文件就具有法律效益。一方面，签名可以保证文件的真实有效性。另一方面，一旦文件签署完成，签名人就不能更改文件内容，如果是否发生纠纷，签名人无法否认在此文件上签过名这一事实[23]。同样的，数字签名也有相似的功能：

（1）确认信息是由签名者发送的；

（2）确认文件从被签署到接收者接收文件的过程中，没有被修改过；

（3）签名者不能够否认信息是由自己发出的。

**3.数字签名原理**

数字签名是基于公钥密码算法和消息摘要技术实现的，下面对其一一介绍。

（1）公钥密码算法

1976年以前，所有的加密算法采用同一种模式：甲方选择某一种加密规则，对信息进行加密；乙方使用相同的规则，对信息进行解密。由于加密和解密采用同样的密钥，这种加密技术被称为“对称加密算法”[16]。

这种加密模式存在一个致命的弱点，甲方必须把加密的密钥告诉乙方。利用网络进行密钥传输是十分不安全的，因此保存和传递密钥就成了最令人头疼的问题[24]。

1976年，两位计算机科学家Whitfield Diffie 和 Martin Hellman，提出了一种崭新的构想，在不直接传输密钥的情况下，完成解密过程。这被称为"Diffie-Hellman密钥交换算法"。这个想法启发了其他密码学家。人们意识到，加密和解密可以使用不同的规则，只要这两个密钥之间存在某种对应关系即可，这样就避免了直接传输密钥的不安全性。

这种新的加密模式被称为“非对称加密算法”（也即公钥加密算法）。该算法主要有三个步骤：

步骤一：乙方生成两把密钥（公钥和私钥）。公钥是公开的，任何人都可以获得，私钥则是保密的。

步骤二：甲方获取乙方的公钥，然后用它对信息加密。

步骤三：乙方得到加密后的信息，用自己的私钥解密。

（2）消息摘要

消息摘要可以将消息通过哈希函数转换成一个具有固定长度的且值唯一的字符串[25]。值唯一的意思是不同的消息生成的摘要是不同的，并且能够确保唯一。此外，该过程不可逆，即不能通过摘要反推明文。因此，一旦原文被修改了很少量的信息（甚至是1bit），其生成的消息摘要也会千差万别，利用这一特性，可以验证消息的完整性[26]。

（3）数字签名

假设有通信双方Alice和Bob，Alice要向Bob发送消息M，两者使用数字签名技术确保数据的完整性，其步骤如下：

a. Bob首先获取自己的公私钥对，并将公钥广播，这样Alice即可接收到Bob的公钥

**PubKey**

**PriKey**

**图2.1 Bob生成公私钥对**

b. Alice首先利用哈希函数（如：MD5）生成待发送消息M的消息摘要D

**Hello Bob：**

**I have some good news to tell you …**

**Yours:**

**Alice**

**fef8b9d5c49b2c37fb3e8b20a09ba7c3a752e35e2f08ab83aa629885ee3b88c4**

**SHA256**

**D**

**M**

**图2.2 Alice生成消息摘要**

c. 接着，Alice利用Bob的公钥对消息摘要D进行加密，生成密文E。

**U2FsdGVkX19XcqHZ6+k6fk1Z+DHXzZVssnGZAN3BRNje4yR8/gH0i5G6lVk5FHNN**

**LJA4ly5mM/WbI1NcUTe6sIMB8VOfbV/qNzVs7xJ7RdfhLindfEjxwFJZ9XwECVC5**

**fef8b9d5c49b2c37fb3e8b20a09ba7c3a752e35e2f08ab83aa629885ee3b88c4**

**PubKey**

**E**

**D**

**图2.3 Alice加密摘要**

d. 最后，Alice将密文E与明文信息M合并，一同发送出去。

**U2FsdGVkX19XcqHZ6+k6fk1Z+DHXzZVssnGZAN3BRNje4yR8/gH0i5G6lVk5FHNNLJA4ly5mM/WbI1NcUTe6sIMB8VOfbV/qNzVs7xJ7RdfhLindfEjxwFJZ9XwECVC5**

**E**

**M**

**Hello Bob：**

**I have some good news to tell you …**

**Yours:**

**Alice**

**Send**

**图2.4 发送签名文件**

e. Bob接收到Alice发送的消息后，使用自己的私钥解密密文E，其对应的明文为D1，并使用相同的哈希函数（SHA256）生成明文的消息摘要D2，最后比较D1和D2的值是否相等，如果相等则表示消息未被篡改，否则认为消息被篡改。

以上即为数字签名的全部步骤。

**4.数字签名的性质**

(1) 不可伪造性：数字签名是对消息摘要进行非对称加密后生成的密文。由于摘要函数的唯一性，不同的明文（哪怕只相差1bit）生成的消息摘要差异是非常大的，因此敌手几乎不可能伪造出这样的数字签名。

(2) 不可篡改性(保证消息的完整性)：由于数字签名首先要对消息原文进行消息摘要，消息摘要选取的哈希函数必须具有值唯一性以及抗碰撞性等特性。因此，即使对原文修改了1bit量的数据，修改后的信息生成的消息摘要与原来的消息摘要差异也是巨大的。所以，对原文消息进行的任何修改，都无法通过验证。

(3) 不可否认性：对于普通的签名方案来说，一旦签名人对某一消息进行了签名之后，即可接受公众的检验。验证者只要知道签名人的公钥和签名算法即可利用验证等式进行验证。签名人的公钥是公开的，并且签名算法也是公开的，所以一旦签名完成，签名者无法否认该签名来源于自身。因此，数字签名具有不可否认性。

## 2.2盲签名理论

随着互联网的普及，人们越来越多地将传统的社会活动转移到了互联网上面。比如网上购物，电子邮件以及本文提到的电子投票等。然而，在互联网带来方便的同时，也有许多不安全的因素产生。任何人都可以借助工具截获网上的数据。因此，用户的信息在不进行任何处理就传输的情况下，很容易泄露用户的个人隐私甚至是用户密码。基于以上原因，学者们开始研究各种解决方案，其中盲签名作为一种新型的数字签名方案在1982年被Chaum首次提出，这一方案核心在于对签名消息的保密，因此而得到了广泛的应用[27]。

盲签名本质上也是一种数字签名，与一般数字签名不同的是签名者不知道待签名消息的具体内容，这样即确保了信息的安全性，又保护了用户的隐私。盲签名方案的原理如下所示：

**UnBlinding**

Signature

**Blinding**

Message

**Signing**

Key

Blind Signature

**Signature**

**Verifier**

Message

True/False

**图2.5 盲签名原理图**

如上图所示：用户首先对消息进行盲化，将盲化后的消息发送给签名者；签名者接收到消息后，使用自己的私钥对盲化消息进行签名，再讲签名发送给用户；用户将消息进行消盲后，得出的数据便是对原消息的盲签名。验证环节与一般数字签名类似，这里不再赘述。

盲签名作为一种特殊的数字签名，拥有一些普通数字签名所不具备的性质：

（1）正确性：使用盲签名算法所输出的签名数据，使用验证函数总可以得出正确的结果。

（2）匿名性：这是盲签名最重要的性质。除了消息拥有者本身外，任何人都不知道消息的具体内容。

（3）不可伪造性：在不知道签名人私钥的情况下，任何人都无法伪造出能使验证方程成立的签名。

（4）不可跟踪性：签名消息公开之后，签名人无法得知签名的确切时间。

## 2.3环签名理论

Rivest和Shamir等人在其论文《How to leak a secret》中首次提出了环签名方案。在该论文中，作者描述了这样一个场景：某位内阁成员想要举报首相的不法行为，如果这位内阁成员以实名的方式举报首相那么很有可能遭到首相的报复。但如果内阁成员以匿名的方式举报首相，那么这样的举报很难引起记者的注意，甚至会被认为是恶意中伤。这样就陷入了两难的境地，作者为了解决类似于这样的难题，提出了环签名方案。

环签名方案本质上是一种群签名方案。不同的是，环签名不存在群管理员，无需组内成员之间的任何合作。组内任何一个成员想要生成环签名，只需要使用其他所有成员的公钥和自己的私钥即可生成环签名，不需要其他成员的任何帮助，而且不会受到任何人的干扰。当接收者受到该成员的签名后，只能验证其是否出自该“环”，无法得知签名人的确切身份。

普通的群签名方案中群管理权限过大，导致群签名容易被跟踪找到签名人的具体信息[28]。环签名的出现，解决了这一难题。环签名具有使签名者无条件匿名的特点，验证者唯一能够确定的是该签名出自环中的某一成员，但无从得知该成员的确切身份，从而达到了匿名的要求。如今，环签名在匿名电子投票，情报组织，电子商务，发腐等领域都有着十分广泛的应用。近年来，涌现出一大批学者从事环签名的研究。然而，在国内对环签名的研究仍然处于起步阶段。并且，在各种环签名方案被提出的同时，也存在一些继续解决的问题，正是这些问题阻挡了其投入商业使用的进程。如：环签名的长度总是随着环中全体成员的数量线性增长，环签名的安全性仍有待提高[29]。

关联环签名是在环签名的基础上，增添了关联标签。使得验证者能确定不同签名是否来自于同一签名者。然而，现有的关联环签名方案均是建立在离散对数公钥密码体制上，且绝大多数方案仅具备弱匿名性的特点。因此，如何设计出基于其他公钥密码体制、具有强匿名性的关联环签名具有重要的理论价值和研究意义[30]。

## 2.4Web技术

### **2.4.1 前端技术**

#### 2.4.1.1 jQuery

**1. 简介**

jQuery是一个快速、简洁的JavaScript框架，是继Prototype之后又一个优秀的JavaScript代码库（或JavaScript框架）。jQuery设计的宗旨是“write Less，Do More”，即倡导写更少的代码，做更多的事情。它封装JavaScript常用的功能代码，提供一种简便的JavaScript设计模式，优化HTML文档操作、事件处理、动画设计和Ajax交互。

**2. 语言特点**

**1).快速获取文档元素**

jQuery的选择机制构建于Css的选择器，它提供了快速查询DOM文档中元素的能力，而且大大强化了JavaScript中获取页面元素的方式。

**2).提供漂亮的页面动态效果**

jQuery中内置了一系列的动画效果，可以开发出非常漂亮的网页，许多网站都使用jQuery的内置的效果，比如淡入淡出、元素移除等动态特效。

**3).创建AJAX无刷新网页**

AJAX是异步的JavaScript和ML的简称，可以开发出非常灵敏无刷新的网页，特别是开发服务器端网页时，比如PHP网站，需要往返地与服务器通信，如果不使用AJAX，每次数据更新不得不重新刷新网页，而使用AJAX特效后，可以对页面进行局部刷新，提供动态的效果。

**4).提供对JavaScript语言的增强**

jQuery提供了对基本JavaScript结构的增强，比如元素迭代和数组处理等操作。

**5).增强的事件处理**

jQuery提供了各种页面事件，它可以避免程序员在HTML中添加太事件处理代码，最重要的是，它的事件处理器消除了各种浏览器兼容性问题。

**6).更改网页内容**

jQuery可以修改网页中的内容，比如更改网页的文本、插入或者翻转网页图像，jQuery简化了原本使用JavaScript代码需要处理的方式。

### **2.4.2 后端技术**

#### 2.4.2.1 Struts2

**1.简介**

Struts2是一个基于MVC设计模式的Web应用框架，它本质上相当于一个servlet，在MVC设计模式中，Struts2作为控制器(Controller)来建立模型与视图的数据交互。

**2.工作原理**

Suruts2的工作原理可以用下面这张图来描述，下面我们分步骤介绍一下每一步的核心内容

**图2.6 Struts2体系结构**

一个请求在Struts2框架中的处理大概分为以下几个步骤

1)、客户端向Servlet容器（例如Tomcat）发送一个请求，容器首次接收到这个请求后初始化对应的Servlet。

2)、这个请求经过一系列的过滤器（Filter）的过滤。

3)、FilterDispatcher是控制器的核心，就是mvc中c控制层的核心。下面粗略的分析下我理解的FilterDispatcher工作流程和原理：FilterDispatcher进行初始化并启用核心doFilter

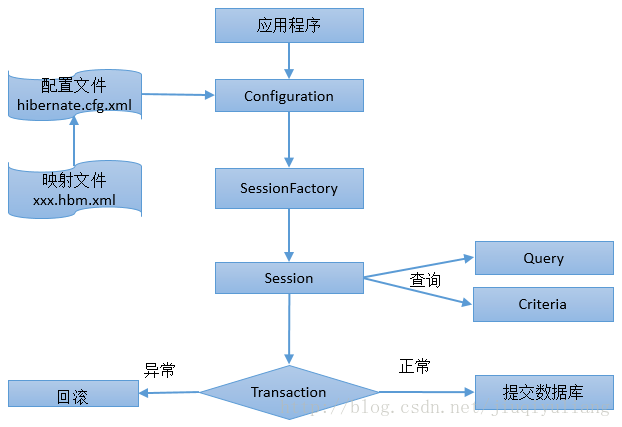
4)、ActionProxy通过ConfigurationManager询问框架的配置文件，找到需要调用的Action类 ,这里，我们一般是从struts.xml配置中读取。

5)、ActionProxy创建ActionInvocation的实例。

#### 2.4.2.3 Hibernate

**1.简介**

Hibernate是一个开放源代码的ORM（Object Relationship Mapping，对象关系映射）框架，它是对传统的JDBC进行的封装，它将POJO与数据库表建立映射关系，是一个全自动的orm框架，Hibernate可以自动生成SQL语句，自动执行。

**2.Hibernate核心**

**图2.7 Hibernate核心架构**

从上图中，我们可以看出Hibernate六大核心接口，两个主要配置文件，以及他们直接的关系。Hibernate的所有内容都在这了。那我们从上到下简单的认识一下，每个接口进行一句话总结。

1）、Configuration接口:作用是配置并实例化Hibernate。

2）、SessionFactory接口:负责Hibernate的初始化工作。

3）、Session接口:负责持久化对象的CRUD操作

4）、Transaction接口:负责事务

5）、Query接口和Criteria接口:负责执行各种数据库查询

**3.Hibernate的优/缺点**

**优点：**

1）、更加对象化以对象化的思维操作数据库，我们只需要操作对象就可以了，开发更加对象化。

2）、移植性因为Hibernate做了持久层的封装，你就不知道数据库，你写的所有的代码都具有可复用性。

3）、Hibernate是一个没有侵入性的框架，没有侵入性的框架我们称为轻量级框架。对比Struts的Action和ActionForm，都需要继承，离不开Struts。Hibernate不需要继承任何类，不需要实现任何接口。这样的对象叫POJO对象。

4）、Hibernate代码测试方便。

5）、提高效率，提高生产力。

**缺点：**

1）、使用数据库特性的语句，将很难调优

2）、对大批量数据更新存在问题

3）、系统中存在大量的攻击查询功能

## 2.5本章小结

本章对数字签名理论、盲签名理论及环签名理论做了一个简单的介绍，在用户进行注册前，系统需要初始化，基于环签名理论初始用户群；在注册阶段，涉及到投票者对消息进行关联环签名、用私钥对消息进行普通个体签名；投票中心使用签名验证算法验证用户的签名，判断用户是否重复申请。注册中心分配给用户一个用于唯一标识身份的序列码，也要对其进行签名。在投票阶段，盲化选票还涉及到了盲签名等理论。此为，对网站的前后端开发技术做了一个简单的介绍。

# 项目工作和进展情况

## 3.1项目的概况

因为安全电子投票系统的受众注重投票各阶段的安全，为保证用户的积极使用，我们必须考虑以下几点问题：

1. 电子投票网站在安全、公正的前提下，要延承电子投票快速便捷的优点；
2. 投票人完成匿名不依赖任何第三方和匿名通信信道，即无条件匿名；
3. 网站能作为投票中心和投票人之间的中枢支持两方互相验证信息；
4. 现存的理论化电子投票方案大多无法有效地应用于实际投票，为解决这一问题，需要避免选票碰撞，允许中途弃票，提高投票效率。

## 3.2项目的开发内容和结果

### **3.2.1 系统架构设计**

#### 3.2.1.1 系统架构总体设计

整个系统可分为三个部分，投票用户端、网站端、投票中心端。投票用户作为关联环体

**图3.1 系统架构示意图**

中的某一成员进行匿名投票表决。网站端作为投票用户和投票中心之间的中枢架起相互信任的桥梁，对整个投票过程中数据的真实有效负责，并在公告牌上明示注册信息表List1，投票信息表List2，计票信息表List3以供相互查对。投票中心端验证每一个投票用户的身份合法性，为合法投票者发放身份序列码Ni，验证盲化选票的合法性，并为盲化选票签名。

1. 投票者：作为环体中的某一成员对候选人进行投票。任何人只能确定投票者的合法性而无法得知其真实身份
2. 投票中心：在网站上创建投票，设置投票人范围。验证每一个投票者身份的合法性，为合法投票者发放身份序列码Ni，并为Ni签名。
3. 网站：在投票者、投票中心与后台服务器建立信息交互桥梁，显示信息。
4. 后台服务器：初始化系统；邀请合法投票人进行注册；验证盲化选票的合法性，并为合法选票进行盲签名。对投票者去盲后的选票进行验证并签名，然后统计并公布最终结果。同时公布注册、投票、计票阶段对应的信息表。
5. 信息表：List1注册信息表，List2投票信息表，List3计票信息表，接受各方的监督与验证。

#### 3.2.1.2 系统模块划分



**图3.2 系统功能模块图**

1. 注册模块：首先由注册中心创建投票主题，设置投票者范围。注册模块根据范围发送邀请。投票者接受邀请后，进行匿名注册，其中使用关联环签名实现。同时注册模块为每一位投票者分配标识唯一身份的序列码，实现抗选票碰撞。结合前述二者，保证了投票者的秘密性。其他人只能知道某序列码是合法投票者中的一员，而无法知道其具体身份。同时，注册模块公布注册信息表，保证未及时注册的投票者及时注册。
2. 投票模块：投票者生成选票后，使用盲签名盲化选票，发送给对应功能模块。后者再为该选票进行签名，保证选票的正确性。同时投票模块发布投票信息表，可防止投票者未及时投票，以及防止选票泄露，并接受公众监督。
3. 计票模块：计票模块根据投票模块的数据为选票进行消盲，并进行计票，且将选票信息公布，接受公众监督。
4. 双机热备：使用配置相同的两台服务器，对运行过程中产生的数据在两台服务器上同时保存，当一台服务器出现故障时，由于负载均衡的作用，可以立即切换到由另一台服务器提供服务。

### **3.2.2 系统概要设计**

#### 3.2.2.1 系统用例图设计

整个系统完成的功能详见如下用例图：



**图3.3 系统用例图**

#### 3.2.2.2 系统流程设计

##### 3.2.2.2.1 系统概要流程设计

1. 投票中心在网站上创建投票，设置投票人范围；
2. 网站通过注册中心的设置，向投票人发起邀请；
3. 投票者通过邀请在投票中心进行匿名注册，投票中心分配给投票者用于标识唯一身份的序列码；
4. 投票者发送盲化选票，投票中心为其盲签名；
5. 网站进行计票，公布信息表List1（包含用户序列码、及其签名、对消息的关联环签名、私钥对消息进行的普通个体签名、公钥），List2（身份序列码、盲化后的选票、选票的盲签名），List3（身份序列码、消盲后的选票），及选票的最终结果，接受验证。

##### C:\Users\xuqinkun\Desktop\系统流程图.jpg3.2.2.2.2系统具体流程设计

**图3.4 系统具体流程图**

* **注册阶段**

通过验证的合法投票者在网站进行匿名注册，具体步骤如下：

1. 首先对系统进行初始化，所有具备选举资格的选民的公钥组成集合L={y1,y2,……,yn}。
2. 投票者Vi(1≤i≤n)对消息m进行关联环签名，生成签名(C,S1,S2,……,Sn,Y)，同时为自己生成另外一对基于RSA签名体制的公私钥对，并用RSA私钥对消息m进行普通的数字签名，最终将选票消息，两个签名以及自己生成的公钥一起发送给投票中心。
3. 投票中心根据关联环签名验证算法，首先验证签名(C,S1,S2,……,Sn,Y)的合法性。如果合法，则检查该签名是否存在于注册信息表中，如果存在，则说明Vi重复注册，拒绝此次注册请求；如果不存在，则接受此次注册请求。
4. 如果Vi是新的合法投票者，投票中心使用收到的公钥来验证普通签名是否合法。若正确，投票中心为Vi选取一个随机的具有唯一标识的身份序列码Ni，并对Ni签名；若不正确，投票中心要求投票者重新发送消息。最后，投票中心将身份序列码Ni和签名结果发送给投票人Vi，同时将相关信息加入到注册信息表List1中。

* **投票阶段**

在规定的投票期限内，投票者按照如下规定进行投票

1. 首先，投票者Vi生成电子选票Ei，然后随机选择一盲化因子对选票进行盲化。
2. Vi对选票进行签名，然后将盲化选票、Ni、选票签名发给投票中心。
3. 投票中心首先检查Ni是否已经存在于投票信息表List2中。如果存在，拒绝此次请求；如果不存在，验证签名的正确性。
4. 若签名通过，投票中心为Vi的盲化选票进行签名，并将盲化选票和签名发送给Vi，同时将相关数据加入到投票信息表List2中；如果验证失败，则丢弃此次选票内容，并要求投票者重新发送。
5. 投票结束后公布List2，接受公开验证。在投票公示时间内，未能在投票期间及时投票的投票者可以公开投票，公布自己的盲化选票和签名，要求投票中心为其发送签名并将其投票结果公布于List2中；同时已投票者查询List2，如果查询不到自己的投票信息，投票者可以公布自己的盲化选票和签名，在所有人的监督下，要求投票中心为其签名并发送盲签名等信息给投票者，并将其结果公布到List2中。如果公示期结束，投票者依旧没有投票，则被视为中途放弃。

* **计票阶段**

投票人Vi将消盲后的选票公布于计票信息表List3中，网站根据List3统计并公布选举的最终结果（只有Ni存在于List2中的投票者才有权参与如下计票过程）：

1. Vi对投票中心的盲签名做消盲，进而得到投票中心对选票的签名。
2. Vi对选票Ei签名，然后将相关信息公布于List3中。任何人均可验证最终计票信息的正确性。若投票中心和投票人对投票的签名都正确，说明选票为Vi的合法投票，并且选票没有被篡改。如果Ni在List3中重复出现，说明投票者Vi重复投票，仅对其选票统计一次。
3. 超过机票期限，投票者依然未将选票公布于List3中，则被视为中途弃权。
4. 网站根据List3统计并公布投票的最终结果。

#### 3.2.2.3 网站数据处理模块设计

服务器数据处理模块使用C++开发，用于处理投票用户和投票中心传来的数据。结构图如下：

**图3.5 数据处理模块结构图**

数据处理模块主要完成分配公私钥对、关联环签名的签名与验证、盲签名的签名与验证、普通RSA签名的签名与验证等功能。在整个系统中发挥了至关重要的作用。

#### 3.2.2.4 网站设计

Web后端使用SSH框架开发，并采用Tomcat作为服务器；主要分为投票管理中心和用户投票中心两部分。系统采用MVC架构，使用Strus2作为控制层，负责数据的处理与转发；Hibernate框架负责对象的持久化工作；Spring负责事务的管理以及对象的创建工作。Web前端使用jsp实现，并使用jQuery、BootStrap作为UI支持，使用Ajax技术提高了用户交互。



**图3.6 后台管理中心架构图**



**图3.7 用户投票中心架构图**

### **3.2.3 系统详细实现**

**3.2.3.1 BlindSignature（盲签名算法部分）**

* **盲签名算法实现**

1. **BlindBigInt类**

**表3.1 类BlindBigInt的实现**

| **编号** | **函数名** | **说明** |
| --- | --- | --- |
| 1 | BlindBigInt::BlindBigInt() | 构造函数1 |
| 2 | BlindBigInt::BlindBigInt(int) | 构造函数2 |
| 3 | BlindBigInt::BlindBigInt(int,int) | 构造函数3 |
| 4 | BlindBigInt::~BlindBigInt() | 析构函数 |
| 5 | BlindBigInt::isZero() | 判断是否为0 |
| 6 | BlindBigInt::toULong() | 将数值转换成是十进制的长整型数 |
| 7 | BlindBigInt::toString() | 将数值转换成字符串 |
| 8 | BlindBigInt::toHexString() | 将数值转换成十六进制格式表示的字符串 |
| 9 | BlindBigInt::operator[]() | 获得数值特殊一位上的值 |
| 10 | BlindBigInt::operator\*() | 将两个BlindBigInt数值相乘 |
| 11 | BlindBigInt::operator\*=() | 将两个BlindBigInt数值相乘并赋值 |
| 12 | BlindBigInt::operator+() | 将两个BlindBigInt数值相加 |
| 13 | BlindBigInt::operator>=() | 比较当前BlindBigInt数值是否大于等于参数BlindBigInt数值 |
| 14 | BlindBigInt::operator>() | 比较当前BlindBigInt数值是否大于参数BlindBigInt数值 |
| 15 | BlindBigInt::operator<=() | 比较当前BlindBigInt数值是否小于等于参数BlindBigInt数值 |
| 16 | BlindBigInt::operator<() | 比较当前BlindBigInt数值是否小于参数BlindBigInt数值 |
| 17 | BlindBigInt::operator/() | 将两个BlindBigInt数值相除 |
| 18 | BlindBigInt::operator==() | 判断两个BlindBigInt数值是否相等 |
| 19 | BlindBigInt::operator%() | 将两个BlindBigInt数值取余 |
| 20 | BlindBigInt::operator<<=() | 对BlindBigInt数值做左移位操作 |
| 21 | BlindBigInt::operator|=() | 将两个BlindBigInt数值做或位操作 |
| 22 | BlindBigInt::operator&=() | 将两个BlindBigInt数值做和位操作 |
| 23 | BlindBigInt::operator-() | 将两个BlindBigInt数值相减 |
| 24 | modPow() | x^ymodm |
| 25 | binToHex() | 将十进制字符串转换为十六进制字符串 |
| 26 | bitset::add() | 做位相加操作 |
| 27 | bitset::subtract() | 做位相减操作 |

1. **BlindRSA类**
2. **实现欧拉函数的变量**

**表3.2 实现欧拉函数的变量**

| 编号 | 变量 | 说明 |
| --- | --- | --- |
| 1 | Int p,q | 十七位的随机数,p!=q |
| 2 | BlindBigIntn | n=p\*q |
| 3 | BlindBigIntphi | Phi=(p-1)(q-1) |
| 4 | BlindBigInte | gcd(e,phi)==1 |
| 5 | BlindBigIntd | [e\*d==1]mod n |

1. **实现欧拉函数的函数**

**表3.3 实现欧拉函数的函数**

| **编号** | **函数名** | **说明** |
| --- | --- | --- |
| 1 | BlindRSA::BlindRSA() | 构造函数1 |
| 2 | BlindRSA::BlindRSA(int) | 构造函数2 |
| 3 | BlindRSA::BlindRSA(int,int) | 构造函数3 |
| 4 | BlindRSA::~BlindRSA() | 析构函数 |
| 5 | BlindRSA::GetPHI() | 返回变量phi |
| 6 | BlindRSA::getP() | 返回变量p |
| 7 | BlindRSA::getQ() | 返回变量q |
| 8 | BlindRSA::getPublicKey() | 返回公钥 |
| 9 | BlindRSA::getPrivateKey() | 返回私钥 |
| 10 | BlindRSA::getModulus() | 返回modulus=p\*q |
| 11 | BlindRSA::setPublicKey(unsignedint) | 设置公钥 |
| 12 | BlindRSA::setPublicKey(BlindBigInt) | 设置公钥 |
| 13 | BlindRSA::encrypt() | 加密 |
| 14 | BlindRSA::decrypt() | 解密 |
| 15 | BlindRSA::modInverse() | 反向取余 |
| 16 | BlindRSA::isPrime() | 素数验证 |
| 17 | BlindRSA::isPrimeMR() | 素数验证 |
| 18 | BlindRSA::isPrimeDiv() | 素数验证 |
| 15 | BlindRSA::gcd(int,int) | Euclidean算法 |
| 16 | BlindRSA::gcd(BlindBigInt,BlindBigInt) | Euclidean算法 |

**3.2.3.2 LinkableRingSignature（关联环签名部分）**

* **关联环签名算法实现**

1. 结合函数（combiningfunction）

结合函数环签名方案中的一个重要组成部分。

定义为结合函数，其中，k是输入，v是随机选取的初始值，是上的任意值，输出z是上的值。

选定k和v的值后，结合函数有以下的特点：

1. 对于每个s,，以及其他给定输入，方程，是从到输出z的一一映射。
2. 对于每个s,，给定一个b比特长读的值z，和除以外的所有的值，可以求解满足的b比特长度值。
3. 如果一个攻击者不能你想求解陷门方程中的某个方程，那么给定k , v和z,它就很难根据求解中的任何一个值。
4. 结合算数为以下形式：

其中定义为。

1. 只要输出z与v相等，该方程的计算过程就会连成一个环形，这也是环签名名称的由来。



**图3.8 环签名示意图**

1. **关联环签名代码实现**

**CommonFunction类**

**表3.4 类CommonFunction实现**

| **编号** | **函数名** | **说明** |
| --- | --- | --- |
| 1 | CommonFunctions::GetGroupParameters() | 获得群参数 |
| 2 | CommonFunctions::IntegerToString() | 将CryptoPP::integer转为字符串 |
| 3 | CommonFunctions::GenerateString() | 将向量中的各值转为字符串 |
| 4 | CommonFunctions::Hash1() | SHA1哈希函数 |
| 5 | CommonFunctions::Hash2() | SHA2哈希函数 |

**LinkableRingSignProver类**

1. **实现的变量**

**表3.5 LinkableRingSignProver变量**

| **编号** | **变量** | **说明** |
| --- | --- | --- |
| 1 | Integer\_private\_key | 私钥 |
| 2 | unsignedint\_self\_identity | 当前环成员 |
| 3 | unsignedintnum\_members | 环成员人数 |
| 4 | Integerg | 群 |
| 5 | Integerq | 群序 |
| 6 | stringm | 签名消息 |
| 7 | vector<Integer>public\_keys | 公钥向量 |
| 8 | IntegerC | 关联环参数 |
| 9 | IntegerY | 关联环参数 |
| 10 | vector<Integer>S | 关联环参数 |

1. **实现的函数**

**表3.6 LinkableRingSignProver函数**

| **编号** | **函数名** | **说明** |
| --- | --- | --- |
| 1 | LinkableRingSignProver::LinkableRingSignProver() | 构造函数 |
| 2 | LinkableRingSignProver::GenerateSignature() | 产生签名 |
| 3 | GeneratePublicPrivateKeys() | 产生公私钥对 |

1. **LinkableRingSignVerifier类**
2. **实现的变量**

**表3.7 LinkableRingSignVerifier变量**

| 编号 | 变量 | 说明 |
| --- | --- | --- |
| 1 | IntegerC | 关联环参数 |
| 2 | IntegerY | 关联环参数 |
| 3 | vector<Integer>S | 关联环参数 |
| 4 | unsignedintnum\_members | 环成员人数 |
| 5 | Integerg | 群 |
| 6 | Integerp | 素数 |
| 7 | Integerq | 群序 |
| 8 | stringm | 签名消息 |
| 9 | vector<Integer>public\_keys | 公钥向量 |

1. **实现的函数**

**表3.8 LinkableRingSignVerifier函数**

| **编号** | **函数名** | **说明** |
| --- | --- | --- |
| 1 | LinkableRingSignVerifier::LinkableRingSignVerifier() | 构造函数 |
| 2 | LinkableRingSignVerifier::VerifySignature() | 验证函数 |

## 

## 3.3项目开发计划及阶段性完成情况

### **3.3.1 项目开发计划**

**表3.9 项目开发计划**

|  |  |  |
| --- | --- | --- |
| **任务名称** | **开始时间** | **计划完成时间** |
| **项目立项** | 2017年4月1日 | 5日 |
| 相关资料收集 | 2017年4月6日 | 5日 |
| 用户访问 | 2017年4月11日 | 2日 |
| 系统调研 | 2017年4月13日 | 2日 |
| 可行性分析 | 2017年4月15日 | 2日 |
| 制定项目计划 | 2017年4月17日 | 2日 |
| **知识积累** | 2017年4月24日 | 7日 |
| 建立开发环境 | 2017年4月27日 | 3日 |
| 相关知识学习 | 2017年5月1日 | 14日 |
| 相关工具学习 | 2017年5月15日 | 5日 |
| **需求分析** | 2017年5月20日 | 4日 |
| 需求建模 | 2017年5月25日 | 2日 |
| 编写需求规格书 | 2017年5月27日 | 2日 |
| **设计** | 2017年5月29日 | 4日 |
| 架构设计 | 2017年6月2日 | 6日 |
| 数据设计 | 2017年6月8日 | 5日 |
| 界面设计 | 2017年6月13日 | 3日 |
| 功能模块设计 | 2017年6月18日 | 6日 |
| 编写设计文档 | 2017年6月24日 | 2日 |
| 编码实现 | 2017年6月24日 | 28日 |
| **测试** | 2017年7月23日 | 3日 |
| 单元测试 | 2017年7月24日 | 1日 |
| 集成测试 | 2017年7月25日 | 1日 |
| 系统测试 | 2017年7月26日 | 1日 |
| **总结** | 2017年11月20日 | 14日 |
| 编写总结报告 | 2017年11月20日 | 14日 |

### **3.3.2 项目完成情况**

系统大部分功能已经实现，详细完成情况如下：

**3.3.2.1 发起投票**

投票中心在网站登录注册后创建投票信息，输入主题、内容等信息创建投票。



**图3.9 创建投票**

之后，投票中心在候选名单中添加候选人信息和选民列表。



**图3.10 编辑投票详情**



**图3.11 添加候选人**

投票中心在网站上编辑邮件内容，网站根据投票中心提供的邮件地址对每位投票者进行邀请。



**图3.12 邮件邀请**

**3.3.2.2 匿名投票**

每位投票者通过邮件接收邀请，网站对用户提供的个人信息进行RSA加密，之后作为关联环的消息。



**图3.13 查看邮件**

点击邮件链接。通过网站生成关联环签名，投票中心可以通过环签名确认投票者的身份合法性，并分配给投票者用于标识唯一身份的序列码，用户完成匿名注册。



**图3.14 生成关联环签名**

用户下载含有投票信息以及签名的文件到本地，利用该文件登录投票中心。



**图3.15 用户登录**



**图3.16 选取候选人**

投票者凭借身份序列码进行匿名投票，投票过程用户将选票内容盲化，交给投票中心进行盲签名，避免双方篡改信息。



**图3.17 盲签名投票**

**3.3.2.3 查看结果**

系统包括三个阶段：注册阶段，投票阶段，计票阶段。三个阶段限定时间完成，投票结束前不显示投票结果。



**图3.18 查看参与的投票**

投票结束后，投票者和投票中心可以前往网站投票结果页面查看本次投票结果。



**图3.19 投票结果**

**3.3.2.4 公开验证**

在选举的中间过程，选举的中间结果不会被泄漏，且投票的状态变化公布在网站的公告信息表中，注册阶段结束公布List1，投票阶段结束公布List2，计票阶段结束公布List3。如下列图示。



**图3.20 注册信息表（List1）**



**图3.21 投票信息表List2**



**图3.22 计票信息表List3**

投票人和投票中心可凭借手中持有信息对信息表进行公开验证，对各阶段进行有效监督。

**3.3.2.5 双机热备**

使用两台相互独立的服务器，每台服务器上安装相同的应用和数据库。在系统运行过程中产生的数据，同时在两台服务器上保存。首先由一台服务器提供服务，当一台服务器发生故障时，可以立即切换到另一台服务器。

## 3.4本章小结

本章首先介绍了项目的基本概况。然后，介绍了系统的概要设计，具体架构及详细设计。最后，对项目的关键部分的实现做了介绍。

# 讨论与体会

## 4.1对项目开发过程的体会

本项目来源于《全国大学生信息安全大赛》参赛作品，项目的选题由小组成员共同完成。本人负责网站的设计与开发（从数据库到前端页面的设计），因此这里着重讲网站开发过程中遇到的问题及体会。

由于我刚刚接触Web开发不久，在项目开发的过程中时时碰壁。首先，项目所使用的服务器是Tomcat，Tomcat在使用前要配置环境变量，对于新手来说，这是经常容易出错的地方。其次，项目中使用到的框架包含许多jar包，在导入jar包的时候往往会遗漏部分jar包，从而导致抛出“ClassNotFoundException”的异常。此外，遇到的一个比较棘手的问题是，由于框架版本更新后配置发生了一些变化，而网上教程用的是老版本的，因此在配置问题上纠缠了很长时间。这也让我深深体会到要读官方API的重要性。

除此之外，是在持久化开发过程中遇到的一些问题。在数据存取的过程中，往往会遇到表与表之间有相互依赖的情况，这时候会涉及到两个或多个对象的持久化；而Hibernate在存取多个对象的过程中必须在一个事务中进行，因此只能开启一个session（回话）进行持久化。对于刚开始接触Hibernate框架的我，存储每一个对象都会开启一个session，因而造成持久化失败。

虽然在项目的开发过程中经历了诸多困难，但是将Bug一个个的消除后，内心是十分快乐的。并且，在项目开发的过程中，编程能力也渐渐地得到了提高。此外，阅读文档及API的能力也得到了锻炼。总之，此次项目开发可谓收获颇丰。

## 4.2对项目的评价

本项目创新点如下：

1. **不依赖任何第三方和匿名通道，实现公平可验证性**

系统可以有效地避免由第三方机构欺诈行为引起的安全隐患。

从整个系统的角度，项目构建方案中存在投票中心第三方机构，但由于在选举的任意阶段都采用独立的信息公告机制，且分别设置了公示时间，供投票者查询自己的信息是否得到正确公布。与此同时，在所有人的监督下能够及时发现并有效制止第三方机构的不诚实行为。因此任何第三方机构单独或者合谋均无法冒充投票者进行投票，也不能篡改或者删除投票者的投票信息，避免了第三方机构的恶意行为或合谋攻击引起的安全隐患，即系统的安全性不依赖于对第三方机构绝对可信的前提假设。

从投票者角度，首先，保证了投票人的身份合法具有注册资格，进而获得注册机构为其分配的唯一标识身份的序列码Ｎ。其次，在其后的投票和计票环节中，都使用N，且Ｎ分别公布在List2，LIst3中，如投票者重复投票，则会再练个表中出现冗余，从而杜绝投票者重复投票行为。因此，投票者的任何不诚实的投票行为会被及时发现和制止。

因此，每个过程都将接受大家的监督，即可以防止第三方泄密，同时也具有公平可验证性。

1. **提升了关联环签名验证身份的效率，适用于大规模选举**

环签名的很难应用于大规模投票的主要原因是，在签名生成阶段会使用大规模的模幂运算，计算非常耗时。为此，本产品将计算耗时的签名生成放在注册阶段，实现合法投票者的匿名注册，并通过为合法投票者发放一个唯一的身份识别码Ｎ，并结合自己产生的公私钥对，用于后续投票和计票阶段的匿名身份认证。由于本产品在投票阶段仅使用公私钥对实现普通的个体签名及验证，因而有效地保证了投票阶段的高效性和实用性，适合于大规模选举。

1. **解决了匿名性，安全性，公平性之间的矛盾**

电子投票系统的安全需求存在内在的矛盾性。

矛盾一：一方面，投票系统需要向投票者提供证据表明前者正确地理解了后者的意图；另一方面，投票系统不能提供给投票者任何票据，使投票者能够向第三方证明自己的投票内容。

矛盾二：一方面，投票系统需要向公众提供证据表明所有的选票都被正确的计数；另一方面，这些证据不能损害投票者的匿名性。

针对矛盾一，本产品在用户生成选票、投票前时，会用友好的交互方式，向用户显示用户的投票内容，让用户进行确定。在投票后，不产生任何票据，则用户不能向第三方证明自己的投票内容。

针对矛盾二，在计票阶段，本年产品的计票模块会公布用户选票信息表List3，包含标识唯一用户的序列码N，消盲后的选票。投票者可以根据List3查看是否有自己序列码，从而判断自己的选票是否被进行了正确计算。同时，其他人无法根据序列码判断每个序列码对应的具体的用户，因此保证了投票者的匿名性。

本系统提出的电子投票系统的安全需求矛盾的解决方案，并且基于此方案构建出了满足匿名性，安全性，公平性的投票系统。

## 4.3本章小结

本章主要是毕业设计从开始到结束的总体回顾。还回顾了项目开发经历，总结了在整个项目开发过程中所遇到的困难和所提出的对问题的解决方案，以及从这次项目开发中所吸取的经验。同时，客观的分析和评价了项目的优势与创新点，介绍了项目解决的关键问题。

# 小结

目前，系统已经实现了大部分功能，并且考虑了大部分细节问题。针对项目初衷，对项目的改进可以从下面几个方面入手：

1.系统初始化

系统初始化过程涉及到关联环签名的生成，这个过程会做大量的幂运算，因此会花费大量时间，从而降低系统性能。可以利用GPU的多核并行计算的特点，将循环拆分成单条指令进行并行计算，从而减少系统的初始化时间。

2.界面

在一些细节上面处理得还不够好，比如发送邮件失败后不能把出错的详细信息反馈给用户。可以增加异常处理页面，将错误信息实时反馈给用户。

# 参考资料

[1] 张国印，王玲玲，马春光．环签名研究进展[J]．通信学报，2007，28(5)：109-116．

[2] L.Chaum.Untraceable electronic mail, return address and digital pseudonyms[J]. Communications of the ACM,1981, 24(2):84-88.

[3] PENG K.A general and efficient countermeasure to relation attacks in mix-based e-voting[J].International Journal of Information Security,2011,10(1):49-60.

[4] FUJIOKA A, OKAMOTO T,OHTA K.A practical secret voting scheme for large scale elections[C]. Advances in Cryptology-AUSCRYPT 1992,LNCS 718.Berlin:Springer.Verlag,1993:244-25 1.

[5] U W, WANG S.A secure and practical electronic voting scheme[J].Computer Communication,1999,22(3):279-286.

[6] 陈晓峰，王育民．基于匿名通讯信道的安全电子投票方案[J]．电子学报，2003，31(3)：390-393．

[7] CHEN Xiaofeng,WU Qianhong,ZHANG Fangguo,eta1.New receipt-free voting scheme using double-trapdoor commitment[J].Information Sciences,201l,181(8):1493—1502.

[8] 熊丹．关联环签名及其在电子投票中的应用研究[D]．成都：西南交通大学，2015．

[9] J.Cohen, M.Fischer. A Robust and Verifiable Cryptographically Secure Election Scheme, Proc. of the 26th IEEE Symposium on the Foundations of Computer Science, 1985, pp.372-382

[10] 王继林，伍前红，陈德人，王育民．匿名技术的研究进展[J]．通信学报，2005,26(2)：112-118

[11] 李轲．环签名理论及其应用研究[D]. 济南:济南大学,2016.

[12] 张国印，王玲玲，马春光．环签名研究进展[J]．通信学报，2007，,28(5)：109-117

[13] 韩玮．电子选举中的安全技术研究[D]．上海：上海交通大学，2008．

[14] D.Chaum. E. Heyst. Group Signature. Advances in Cryptology-Eurocrypt'91 LNCS 547.Berlin: Soringer-Verkag.1992:257-265

[15] Camnisch J, Stadler M. Efficient and generalized group signatures[C].EUROCRYPT 1997, LNCS 1294, Berlin:Springer-Verlag, 1997: 465-479

[16] IVEST R, SHAMIR A, TAUMAN Y. How to leak a secret[C].Advances in Cryptology-Asiacrypt 2001. LNCS 2248. Berlin: Springer-Verlag.2001:552-565.

[17] 史有辉，李伟生．盲签名研究综述[J]．计算机工程与科学，2005，27(7)：83-94．

[18] 吴龚飞．安全的电子投票系统的设计与实现[J]．电脑迷，2014，7(13）：23-23．

[19] 刘彪．环签名算法研究与应用[D]．西安：西安电子科技大学，2012．

[20] 胡程瑜．环签名体制的研究[D]．济南：山东大学，2008．

[21] 田莹，李星，段海新．基于Mix．Net的电子投票系统的安全需求矛盾及解决办法[D]．北京：清华大学信息网络工程研究中心，2006．

[22] 张文芳，熊丹，王小敏．基于关联环签名的抗第三方欺诈安全电子投票方案[J]．西南交通大学学报，2015，50(5)：906-941．

[23] 徐紫枫，曾康，周福才．基于时间释放加密和数字签名的匿名电子投票方案[J]．计算机应用与软件，2016,33(12)．

[24] 石培荣．一种高安全的电子投票系统的设计与实现[D]．成都：电子科技大学，2014．

[25] 王永恒，徐晨，陈经纬，吴文渊．基于HElib的安全电子投票方案[J]．计算机应用研究，2016，33：1-9．

[26] 范曦，魏评．电子投票系统安全问题的分析与研究[J]．数字技术与应用，2014：179-179．

[27] 王继林，伍前红，陈德人，王育民．匿名技术的研究进展[J]．通信学报，2005,26(2)：112-118．

[28] 张文芳，熊丹，王小敏，陈桢，刘旭东．基于RSA公钥密码体制的可选择可转换关联环签名[J]．2017，40(5)：1168.1180．

[29] 徐秋亮．改进门限RSA数字签名体制[J]．计算机学报，2000，23(5)：449-453．

[30] 刘莉，金正平．一个基于RSA的无证书多重签名方案[J]．四川大学学报，2016，48(2):162-168．

# 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，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，也不包含为获得四川大学或其他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谢意。

本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

学位论文作者（签名） \_\_\_\_\_\_\_\_\_\_\_\_ 2017年12月 15 日

论文指导教师（签名） \_\_\_\_\_\_\_\_\_\_\_\_ 2017年12月 15 日

# 致 谢

首先感谢父母的辛勤付出，是他们养育了我并在我生活和学习中给予了无微不至的关爱。其次，要感谢我的导师梁刚老师，是他为我排除万难，解决了我在项目开发以及论文撰写过程中遇到的种种难题。同时，要感谢和我共同参与全国信息安全大赛的队友，正是有了他们的合作，我才能够在大赛中取得良好的成绩。最后，感谢所有给予我帮助的老师和同学。

能圆满的完成毕业设计，离不开四川大学四年的培养熏陶，更离不开软件学院各位老师的殷勤教导。

毕业设计的完成，使我在四川大学的大学生涯画上了一个圆满的句号。虽然，在川大的学习生活即将结束，但是这段经历必将使我终身受益。

# 附录1 任务申请书

当今时代，以信息技术为核心的新一轮科技革命正在孕育兴起，互联网日益成为创新驱动发展的先导力量，深刻改变着人们的生产生活，有力推动着社会发展，电子投票也逐渐开始取代传统的手工投票。电子选票具有电子化和网络化、低成本化、快速性和便捷性、高度匿名性等优点，能够发挥保证民主的广泛参与性。选举、投票中层出不穷的各种亟待解决的问题，说明了对安全的、公正的电子选票系统的需求与日俱增。

电子投票是通过使用计算机与网络通信，利用高速计算的计算机负责整个投票过程，通过互联网能够将身在不同地方的投票人联系起来，在保证合法的投票人身份的匿名性的前提下，每个投票人都能够根据自己的主观意愿投出选票。同时与传统投票相比，电子投票能够避免在各个环节中人工因素对最后投票的真实有效性进行干扰，并且通过计算机统计，可以大大提高最后对选票结果统计的速度，从而能够极大的节约人力物力以及时间，使得每场电子投票的结果能够更加的公正公开，普遍为投票人所相信。

在大学生活中我们也经常会遇到投票选举的活动，这类活动往往通过在线投票系统来完成。我们经常会看到某个同学的票数突然增加了成百上前票，很显然这类投票系统安全系数较低，很容易通过现有的刷票工具进行刷票。因此，我打算开发一款能够防止刷票的投票软件。

# 附录2 可行性研究报告

## 引言

**1.1编写目的**

目的是明确该项目的各个部分是否具有实现的可行性。

预期读者：本文档的预期读者为指导老师、项目的开发人员、业务和需求分析人员、测试人员、用户文档编写者、项目管理人员，也适用于客户。

**1.2背景**

开发软件名称：基于关联环签名的安全电子投票系统

项目任务提出者：韩镓维，许勤昆，凌雪，唐钰葆

项目开发者：许勤昆

用户：所有需要开展投票活动的单位或个人

实现软件单位：许勤昆

项目与其他软件，系统的关系：

本项目采用C++实现数字签名部分（关联环签名，盲签名等）；采用MySQL来实现数据库；网站后端采用SSH框架，前端采用BootStrap以及jQuery等框架。

**1.3定义**

无。

**1.4参考资料**

《软件文档写作教程》电子工业出版社 主编：马平、黄冬梅

## 可行性研究

**2.1 数字签名可行性**

数字签名作为一项成熟的理论，目前各种语言都有针对其实现的插件，例如：C++加密库Crypto++，以及java JDK自带的安全库security等都实现了公私钥的生成，以及对消息的加密解密算法。

**2.2 盲签名算法可行性**

盲签名是一种特殊的数字签名。目前有多种盲签名的实现方案，本项目采用的是基于RSA公钥密码的盲签名方案，它与普通数字签名的差别在于，签名用户首先对消息进行盲化，签名者不知道签名消息的具体内容。因此，该方案只需要将消息进行盲化后再进行普通数字签名即可实现。

**2.3 关联环签名技术可行性**

关联环签名所依赖的各种函数均可采用C++实现。

**2.4 Web技术可行性**

目前web技术众多，本项目前端采用了BootStrap和jQuery，后端采用的是SSH框架，数据库使用MySQL。

# 附录3 项目开发计划

## 1 引言

**1.1编写目的**

目的是计划该项目的完成进度。

预期读者：本文档的预期读者为指导老师、项目的开发人员、业务和需求分析人员、测试人员、用户文档编写者、项目管理人员，也适用于客户。

**1.2背景**

开发软件名称：基于关联环签名的安全电子投票系统

项目任务提出者：韩镓维，许勤昆，凌雪，唐钰葆

项目开发者：许勤昆

用户：所有需要开展投票活动的单位或个人

实现软件单位：许勤昆

项目与其他软件，系统的关系：

本项目采用C++实现数字签名部分（关联环签名，盲签名等）；采用MySQL来实现数据库；网站后端采用SSH框架，前端采用BootStrap以及jQuery等框架。

**1.3定义**

无。

**1.4参考资料**

《软件文档写作教程》电子工业出版社 主编：马平、黄冬梅

## 2 项目开发计划

**表3.1 项目开发计划表**

|  |  |  |
| --- | --- | --- |
| **任务名称** | **开始时间** | **计划完成时间** |
| **项目立项** | 2017年4月1日 | 5日 |
| 相关资料收集 | 2017年4月6日 | 5日 |
| 用户访问 | 2017年4月11日 | 2日 |
| 系统调研 | 2017年4月13日 | 2日 |
| 可行性分析 | 2017年4月15日 | 2日 |
| 制定项目计划 | 2017年4月17日 | 2日 |
| **知识积累** | 2017年4月24日 | 7日 |
| 建立开发环境 | 2017年4月27日 | 3日 |
| 相关知识学习 | 2017年5月1日 | 14日 |
| 相关工具学习 | 2017年5月15日 | 5日 |
| **需求分析** | 2017年5月20日 | 4日 |
| 需求建模 | 2017年5月25日 | 2日 |
| 编写需求规格书 | 2017年5月27日 | 2日 |
| **设计** | 2017年5月29日 | 4日 |
| 架构设计 | 2017年6月2日 | 6日 |
| 数据设计 | 2017年6月8日 | 5日 |
| 界面设计 | 2017年6月13日 | 3日 |
| 功能模块设计 | 2017年6月18日 | 6日 |
| 编写设计文档 | 2017年6月24日 | 2日 |
| 编码实现 | 2017年6月24日 | 28日 |
| **测试** | 2017年7月23日 | 3日 |
| 单元测试 | 2017年7月24日 | 1日 |
| 集成测试 | 2017年7月25日 | 1日 |
| 系统测试 | 2017年7月26日 | 1日 |
| **总结** | 2017年11月20日 | 14日 |
| 编写总结报告 | 2017年11月20日 | 14日 |

# 附录4 数据要求说明书

## 1引言

**1.1编写目的**

目的是明确系统中各项功能和非功能性需求实现时所需要的数据，并可以此数据为根据设计数据库。同时为概要设计和详细设计提供设计依据。

预期读者：本文档的预期读者为指导老师、项目的开发人员、业务和需求分析人员、测试人员、用户文档编写者、项目管理人员，也适用于客户。

**1.2背景**

开发软件名称：基于关联环签名的安全电子投票系统

项目任务提出者：韩镓维，许勤昆，凌雪，唐钰葆

项目开发者：许勤昆

用户：所有需要开展投票活动的单位或个人

实现软件单位：许勤昆

项目与其他软件，系统的关系：

本项目采用C++实现数字签名部分（关联环签名，盲签名等）；采用MySQL来实现数据库；

网站后端采用SSH框架，前端采用BootStrap以及jQuery等框架。

**1.3定义**

无

**1.4参考资料**

《软件文档写作教程》电子工业出版社 主编：马平、黄冬梅。

## 2数据的逻辑描述

数据进行逻辑描述时可把数据分为动态数据、静态数据和内部生成数据。所谓静态数据是指在运行过程中主要作为参考的数据，它们在很长一段时间内不会变化的，一般不随运行而改变。所谓动态数据包括所有在运行中要发生变化的数据以及在运行中要输入、输出的数据。内部生成数据是指根据数据库各个表的内在连接关系可以生成新的所需要的数据，同时可以进行一些加权计算，这些数据可以在SQL语句中体现。进行描述时应把各数据元素逻辑地分成若干组，例如函数、源数据或对于其应用更为恰当的逻辑分组。给出每一数据元的名称包括缩写和代码、定义或物理意义度量单位、值域、格式和类型等有关信息。

**2.1静态数据**

**表4.1 静态数据**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **数据名称** | **数据类型** | **数据长度** | **数据描述** |
| **管理员信息(admin)** | name | varchar | 20 | 用户名 |
| password | varchar | 20 | 密码 |
| authority | int | 1 | 权限等级 |
| email | varchar | 20 | 邮箱 |
| id\_card | varchar | 20 | 身份证号 |
| realname | varchar | 10 | 真实姓名 |
| last\_login\_date | date | - | 最近一次登录日期 |

**2.2动态输入数据**

**表4.2 静态数据**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **数据名称** | **数据类型** | **数据长度** | **数据描述** |
| **选民信息** | name | varchar | 20 | 姓名 |
| email | varchar | 20 | 邮箱 |
| address | varchar | 200 | 地址 |

**2.3动态输出数据**

**表4.3 静态数据**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **数据名称** | **数据类型** | **数据长度** | **数据描述** |
| **投票活动信息** | title | varchar | 100 | 投票标题 |
| content | varchar | 500 | 投票内容 |
| create\_time | date | - | 创建时间 |
| start\_time | varchar | - | 投票开始日期 |
| end\_time | varchar | - | 投票结束日期 |
| type | varchar | 20 | 投票类型 |
| **候选名单信息** | name | varchar | 20 | 候选名称 |
| decription | varchar | 500 | 候选描述 |
| create\_time | date | - | 创建日期 |

**2.4内部生成数据**

列出向用户或开发单位中的维护调试人员提供的内部生成数据。

## 3数据的采集

**3.1要求和范围**

按数据元的逻辑分组来说明数据采集的要求和范围指明数据的采集方法说明数据采集工作的承担者是用户还是开发者。具体的内容包括：

输入数据的来源主要是建立数据库时录入的数据在进行相应操作时会进行数据更新。数据只能由操作员更新。

# 附录5 需求规格说明书

## 1引言

**1.1编写目的**

目的是明确系统中需要实现的各项功能和非功能性需求，并可以此说明为根据来设计系统。同时为概要设计和详细设计提供设计依据。

预期读者：本文档的预期读者为指导老师、项目的开发人员、业务和需求分析人员、测试人员、用户文档编写者、项目管理人员，也适用于客户。

**1.2背景**

开发软件名称：基于关联环签名的安全电子投票系统

项目任务提出者：韩镓维，许勤昆，凌雪，唐钰葆

项目开发者：许勤昆

用户：所有需要开展投票活动的单位或个人

实现软件单位：许勤昆

项目与其他软件，系统的关系：

本项目采用C++实现数字签名部分（关联环签名，盲签名等）；采用MySQL来实现数据库；

网站后端采用SSH框架，前端采用BootStrap以及jQuery等框架。

**1.3定义**

无。

**1.4参考资料**

《软件文档写作教程》电子工业出版社 主编：马平、黄冬梅

## 2任务概述

**2.1目标**

将数字签名技术应用于投票系统，一方面可以保护选民隐私安全，另一方面可以防止选票过程中不法分子篡改选票、刷票等操作。

**2.2用户的特点**

需要开展选举活动的任何单位或个人。

**2.3假定和约束**

无。

## 3需求规定

**3.1对功能的规定**

* **发起投票**

投票中心在网站上发起投票，填写投票主题等相关信息，之后添加投票者邮件地址，网站对每位投票者发起邀请。

* **匿名投票**

在投票之前，投票者接受到邮件邀请，之后利用投票者消息进行匿名注册，投票中心只能确认其为合法投票者之一，但无法确定投票者真实身份。

确认身份后投票者获得唯一标识身份的序列码，进入投票页面进行投票。

* **查看结果**

投票结束后，投票者和投票中心可以前往网站投票结果页面查看本次投票结果。

* **公开验证**

在选举的中间过程，选举的中间结果不会被泄漏，且投票的状态变化公布在网站的公告信息表中，投票人和投票中心可凭借手中持有信息对信息表进行公开验证，对各阶段进行有效监督。

* **双机热备份**

利用MySQL数据库提供的服务，将数据实时备份到另一台服务器上

**3.2对性能的规定**

所谓性能需求，是指对软件产品的功能性需求强加的条件。软件产品的性能需求包括系统的可靠性、可维护性、可扩充性等。这里可以看到性能需求涉及的范围很广，软件产品本身不是孤立存在的，还涉及到诸多外在环境的影响。性能需求必须考虑到软件既要可用，还要易用。对于本系统的性能需求有以下几点：

1. **清晰性：**系统的整体结构和功能区域要清晰，能够清楚表现出整个系统运行的脉络和途径，而不是毫无逻辑的运行。
2. **可扩展性：**在设计系统架构时，要考虑到架构是否易于扩展。
3. **可移植性：**在设计过程中采用通用的程序设计语言和运行支撑环境，尽量不用与系统的底层相关性强的语言，保证系统的可移植性。
4. **扩展性需求：**要求软件易于扩展，将来添加功能时，修改一个模块而不会影响其他模块，尽量防止连锁反应；要求在软件设计和实现是尽量遵守面向对象五大原则（单一职责原则、开放封闭原则、依赖倒置原则、接口隔离原则、Liskov替换原则）

**3.3输入输出要求**

无。

**3.4数据管理能力要求（针对软件系统）**

无。

**3.5故障处理要求**

无。

**3.6其他专门要求**

* **界面需求：**

1. **界面易懂性：**界面中的文字标识，含义准确、清楚、用通用词汇。
2. **风格一致性：**界面分块后，前中后期开发时界面图形和风格一致，遵循简洁一致的原则。

## 4运行环境规定

**4.1设备**

无。

**4.2支持软件**

运行环境：Windows、JDK1.8、Tomcat9

开发平台：MyEclipse

开发语言：Java、C++、HTML、CSS、JavaScript

数据库：MySQL

**4.3接口**

数据库接口。

**4.4控制**

无。

# 附录6 用户手册概要

## 1.引言

**1.1编写目的**

目的是明确系统中需要实现的各项功能和非功能性需求，并可以此说明为根据来设计系统。同时为概要设计和详细设计提供设计依据。

预期读者：本文档的预期读者为指导老师、项目的开发人员、业务和需求分析人员、测试人员、用户文档编写者、项目管理人员，也适用于客户。

**1.2背景**

开发软件名称：基于关联环签名的安全电子投票系统

项目任务提出者：韩镓维，许勤昆，凌雪，唐钰葆

项目开发者：许勤昆

用户：所有需要开展投票活动的单位或个人

实现软件单位：许勤昆

项目与其他软件，系统的关系：

本项目采用C++实现数字签名部分（关联环签名，盲签名等）；采用MySQL来实现数据库；

网站后端采用SSH框架，前端采用BootStrap以及jQuery等框架。

**1.3定义**

无。

**1.4参考资料**

《软件文档写作教程》电子工业出版社 主编：马平、黄冬梅

## 2用途

**2.1功能**

* 发起投票
* 匿名投票
* 查看结果
* 公开验证
* 双机热备份

**2.2性能**

无。

## 3运行环境

**3.1硬设备**

无。

**3.2支持软件**

运行环境：Windows、JDK1.8、Tomcat9

开发平台：MyEclipse

开发语言：Java、C++、HTML、CSS、JavaScript

数据库：MySQL

**3.3数据结构**

无。

## 4使用过程

**4.1安装与初始化**

**1.下载并安装JDK**

**图6.1**

**1）进入Oracle官网下载JDK**

**2）双机程序安装JDK**

**图6.2**

**3）配置环境变量**

打开“控制面板”，进入“系统和安全”，进入“系统”，进入“高级系统设置”，“环境变量”

**图6.3**

**在系统变量中添加环境变量**

JAVA\_HOME:JDK安装路径

**图6.4**

CLASS\_PATH: **.;**%JAVA\_HOME%\lib**;**%JAVA\_HOME%\lib\tools.jar(.表示当前路径)

**图6.5**

编辑已有的变量PATH，在变量值最后添加参数：**;**%JAVA\_HOME%\bin**;**%JAVA\_HOME%\jre\bin**;**。

**图6.6**

最后，在命令行中输入：java –version，如果出现如下信息，则表示配置成功

**图6.7**

**2.下载并安装Tomcat**

**1）进入Tomcat官网http://tomcat.apache.org/，在导航栏中点击相应版本，进入下载页面**

**图6.8**

根据操作系统类型，下载相应的压缩包

**图6.9**

将压缩包解压

**图6.10**

**2）配置环境变量**

新建环境变量

CATALINA\_BASE：D:\Servers\apache-tomcat-9.0.0.M20

CATALINA\_HOME：D:\Servers\apache-tomcat-9.0.0.M20

TOMCAT\_HOME：D:\Servers\apache-tomcat-9.0.0.M20

**图6.11**

**3）测试**

在“命令行窗口”中输入startup，可以看到Tomcat启动信息

**图6.12**

****

**图6.13**

在浏览器中输入**http://localhost:8080**，进入Tomcat管理页面



**图6.14**

**3.部署项目**

将项目压缩文件VoteSystem.war放入Tomcat安装路径webapp目录下面

**图6.15**

**4.运行Tomcat**

在cmd下输入命令startup，启动Tomcat

**5.运行程序LinkableRingSignature.exe**

**图6.16**

**4.2操作说明**

**1）进入网站首页**



**图6.17 网站首页**

**2）本系统针对不同场景设置了不同的投票类型，点击“开启议事投票”即可进入后台登录页面**

**图6.18 后台登录页面**

**3）登录成功后进入投票管理中心**



**图6.19 投票管理中心首页**

**4）创建投票**

点击菜单栏“创建投票”，输入投票活动信息



**图6.20 创建投票**

**5）添加候选人**

**图6.21 添加候选人**

**6）邀请选民参与投票**



**图6.22 添加选民**

# 附录7 概要设计说明书

## 1引言

**1.1编写目的**

目的是对系统进行初步设计，为详细设计提供参考。

预期读者：本文档的预期读者为指导老师、项目的开发人员、项目的维护人员、测试人员、用户文档编写者、项目管理人员，也适用于客户。

**1.2背景**

开发软件名称：基于关联环签名的安全电子投票系统

项目任务提出者：韩镓维，许勤昆，凌雪，唐钰葆

项目开发者：许勤昆

用户：所有需要开展投票活动的单位或个人

实现软件单位：许勤昆

项目与其他软件，系统的关系：

本项目采用C++实现数字签名部分（关联环签名，盲签名等）；采用MySQL来实现数据库；

网站后端采用SSH框架，前端采用BootStrap以及jQuery等框架。

**1.3定义**

无。

**1.4参考资料**

《软件文档写作教程》电子工业出版社 主编：马平、黄冬梅

## 2总体设计

**2.1系统用例图设计**

整个系统完成的功能详见如下用例图：



**图7.1 系统用例图**

**2.2 系统流程设计**

* **系统概要流程设计**

1）投票中心在网站上创建投票，设置投票人范围；

2）网站通过注册中心的设置，向投票人发起邀请；

3）投票者通过邀请在投票中心进行匿名注册，投票中心分配给投票者用于标识唯一身份的序列码；

4）投票者发送盲化选票，投票中心为其盲签名；

****5）网站进行计票，公布信息表List1（包含用户序列码、及其签名、对消息的关联环签名、私钥对消息进行的普通个体签名、公钥），List2（身份序列码、盲化后的选票、选票的盲签名），List3（身份序列码、消盲后的选票），及选票的最终结果，接受验证。

**图7.2 系统具体流程图**

* **系统具体流程设计**

**注册阶段**

通过验证的合法投票者在网站进行匿名注册，具体步骤如下：

1）按照提前生成好的公私钥对系统进行初始化，其中公钥集合L={y1,y2,……,yn}由所有具备选举几个者公钥组成。

2）投票者Vi(1≤i≤n)利用关联环签名生成算法对消息m进行关联环签名，生成签名(C,S1,S2,……,Sn,Y)，同时为自己生成另外一对基于RSA签名体制的公私钥对，并利用其中的私钥对消息m进行普通的个体签名，最终将消息，两个签名以及自己生成的公钥一起发送给投票中心。

3）投票中心按照关联环签名验证算法，首先验证签名(C,S1,S2,……,Sn,Y)的正确性。若正确，则检查该签名中的Y值是否已存在于注册信息表List1中，如果存在，则说明Vi重复注册，拒绝此次请求；如果不存在，说明Vi为新的合法投票者。

4）如果Vi是新的合法投票者，投票中心利用收到的公钥验证普通签名是否正确。若正确，投票中心为Vi随机选取一个具有唯一标识的身份序列码Ni，并对Ni签名；若不正确，投票中心要求投票者重新发送消息。最后，投票中心将身份序列码Ni和签名结果发送给投票人Vi，同时将相关信息保存到注册信息表List1中。

5）注册结束后网站公布List1,接受公开验证。在注册公示时间内，未能在注册期及时注册的投票者可以公开注册，公布自己的签名，要求投票中心为其发放身份序列码和对应签名，并将其信息添加到List1中，同时已注册者查询List1，如果查询不到自己的注册信息，投票者可以公示自己的签名，并公开验证消息m的关联环签名和普通签名的正确性，进而要求投票中心在所有人的监督下为其发放身份序列码和对应签名，并添加投票者的相应信息于List1上。上述方法能够有效防止注册中心的不诚实行为。此外，若公示时间结束后，投票者依然没有注册，则被视为自动弃权。

**投票阶段**

在规定的投票期限内，投票者按照如下协议进行投票（只有Ni存在于List1的投票者才有权参与如下投票过程）：

1）投票者Vi首先生成电子选票Ei，然后随机选择一盲化因子对选票进行盲化。

2）Vi对选票进行签名，然后将盲化选票、Ni、选票签名发给投票中心。

3）投票中心首先检查Ni是否已经存在于投票信息表List2中。如果已经存在，说明投票者Vi重复投票，拒绝此次请求；如果不存在，验证签名的正确性。

4）若签名正确，投票中心为Vi的盲化选票进行签名，并将盲化选票和签名发送给Vi，同时将相关信息保存到投票信息表List2中；若不正确，则要求投票者重新发送。

5）投票结束后公布List2，接受公开验证。在投票公示时间内，未能在投票期间及时投票的投票者可以公开投票，公布自己的盲化选票和签名，要求投票中心为其发送签名并将其投票结果公布于List2中；同时已投票者查询List2，如果查询不到自己的投票信息，投票者可以公布自己的盲化选票和签名，在所有人的监督下，要求投票中心为其签名并发送盲签名等信息给投票者，并将其结果公布到List2中。如果公示期结束，投票者依旧没有投票，则被视为中途放弃。

**计票阶段**

1）投票人Vi将消盲后的选票公布于计票信息表List3中，网站根据List3统计并公布选举的最终结果（只有Ni存在于List2中的投票者才有权参与如下计票过程）：

2）Vi对投票中心的盲签名做消盲，进而得到投票中心对选票的签名。

3）Vi对选票Ei签名，然后将相关信息公布于List3中。任何人均可验证最终计票信息的正确性。若投票中心和投票人对投票的签名都正确，说明选票为Vi的合法投票，并且选票没有被篡改。如果Ni在List3中重复出现，说明投票者Vi重复投票，仅对其选票统计一次。

4）超过机票期限，投票者依然未将选票公布于List3中，则被视为中途弃权。

5）网站根据List3统计并公布投票的最终结果。

**2.3 网站数据处理模块设计**

****服务器数据处理模块使用C++开发，用于处理投票用户和投票中心传来的数据。结构图如下：

**图7.3 数据处理模块结构图**

数据处理模块主要完成分配公私钥对、关联环签名的签名与验证、盲签名的签名与验证、普通RSA签名的签名与验证等功能。在整个系统中发挥了至关重要的作用。

**2.4 网站设计**

Web后端使用SSH框架开发，并采用Tomcat作为服务器；主要分为投票管理中心和用户投票中心两部分。系统采用MVC架构，使用Strus2作为控制层，负责数据的处理与转发；Hibernate框架负责对象的持久化工作；Spring负责事务的管理以及对象的创建工作。Web前端使用jsp实现，并使用jQuery、BootStrap作为UI支持，使用Ajax技术提高了用户交互。



**图7.4 后台管理中心架构图**



**图7.5 用户投票中心架构图**

# 附录8 数据库设计说明书

## 1引言

**1.1编写目的**

目的是对系统的数据库结构和设计进行详细描述。

预期读者：本文档的预期读者为指导老师、项目的开发人员、项目的维护人员、测试人员、用户文档编写者、项目管理人员，也适用于客户。

**1.2背景**

开发软件名称：基于关联环签名的安全电子投票系统

项目任务提出者：韩镓维，许勤昆，凌雪，唐钰葆

项目开发者：许勤昆

用户：所有需要开展投票活动的单位或个人

实现软件单位：许勤昆

项目与其他软件，系统的关系：

本项目采用C++实现数字签名部分（关联环签名，盲签名等）；采用MySQL来实现数据库；

网站后端采用SSH框架，前端采用BootStrap以及jQuery等框架。

**1.3定义**

无。

**1.4参考资料**

《软件文档写作教程》电子工业出版社 主编：马平、黄冬梅

## 2结构设计

**2.1数据库环境说明**

本系统采用的数据库Mysql，使用root用户登录数据库创建各种数据库对象。

**2.2数据表**

1. List1

**表8.1 数据表list1**

| 字段名 | 数据类型 | 长度 | 描述 | 必要性 |
| --- | --- | --- | --- | --- |
| id | varchar | 40 | 用户序列码 | 必要 |
| signature | varchar | 200 | 序列码签名 | 必要 |
| ring\_signature | varchar | 200 | 关联环签名 | 必要 |
| personal\_signature | varchar | 200 | 普通个体签名 | 必要 |
| public\_key | varchar | 200 | 公钥 | 必要 |

1. List2

**表8.2 数据表list2**

| 字段名 | 数据类型 | 长度 | 描述 | 必要性 |
| --- | --- | --- | --- | --- |
| id | varchar | 40 | 主键 | 必要 |
| blind\_vote | varchar | 200 | 序列码签名 | 必要 |
| signed\_vote | varchar | 200 | 关联环签名 | 必要 |

1. List3

**表8.3 数据表list3**

| 字段名 | 数据类型 | 长度 | 描述 | 必要性 |
| --- | --- | --- | --- | --- |
| id | varchar | 40 | 主键 | 必要 |
| signature | varchar | 200 | 序列码签名 | 必要 |
| ring\_signature | varchar | 200 | 关联环签名 | 必要 |
| personal\_signature | varchar | 200 | 普通个体签名 | 必要 |
| public\_key | varchar | 200 | 公钥 | 必要 |

1. 管理员信息表（admin）

**表8.4 管理员信息表**

| 字段名 | 数据类型 | 长度 | 描述 | 必要性 |
| --- | --- | --- | --- | --- |
| id | varchar | 40 | 主键 | 必要 |
| name | varchar | 20 | 帐号 | 必要 |
| password | varchar | 50 | 密码 | 必要 |
| authority | int | 4 | 常住地址 | 必要 |
| email | varchar | 50 | 联系邮箱 | 必要 |

1. 投票信息表（voteinfo，用于保存一次投票事件的详细信息）

**表8.5 投票信息表**

| 字段名 | 数据类型 | 长度 | 描述 | 必要性 |
| --- | --- | --- | --- | --- |
| id | int | 10 | 主键 | 必要 |
| title | varchar | 200 | 标题 | 必要 |
| content | varchar | 500 | 详情 | 必要 |
| create\_time | datetime | (标准日期格式) | 创建时间 | 必要 |

1. 候选名单表（voteitem）

**表8.6 候选名单表**

| 字段名 | 数据类型 | 长度 | 描述 | 必要性 |
| --- | --- | --- | --- | --- |
| id | int | 10 | 主键 | 必要 |
| name | varchar | 50 | 名称 | 必要 |
| description | varchar | 500 | 简介 | 必要 |
| create\_time | datetime | (标准日期格式) | 创建时间 | 必要 |
| infoid | int | 10 | 外键（参考voteinfo.id） | 必要 |

1. 图片信息表（image，保存候选名单的图片路径）

**表8.7 图片信息表**

| 字段名 | 数据类型 | 长度 | 描述 | 必要性 |
| --- | --- | --- | --- | --- |
| id | int | 10 | 主键 | 必要 |
| path | varchar | 100 | 图片路径 | 必要 |
| itemid | int | 10 | 外键（参考voteitem.id） | 必要 |

1. 投票结果表（ballot，保存投票结果）

**表8.8 数据表投票结果表**

| 字段名 | 数据类型 | 长度 | 描述 | 必要性 |
| --- | --- | --- | --- | --- |
| id | int | 10 | 主键 | 必要 |
| infoid | int | 10 | 外键（参考voteinfo.id） | 必要 |
| itemid | int | 10 | 外键（参考voteitem.id） | 必要 |
| userid | varchar | 40 | 外键（参考user.id） | 必要 |

1. 选民邮箱（email，向指定选民的邮箱发起投票请求）

**表8.9 选民邮箱**

| 字段名 | 数据类型 | 长度 | 描述 | 必要性 |
| --- | --- | --- | --- | --- |
| id | int | 10 | 主键 | 必要 |
| account | varchar | 50 | 邮箱帐号 | 必要 |

1. 邮件信息表（receiver，保存已发送邮件的信息）

**表8.10 邮件信息表**

| 字段名 | 数据类型 | 长度 | 描述 | 必要性 |
| --- | --- | --- | --- | --- |
| id | int | 10 | 主键 | 必要 |
| email | varchar | 100 | 邮箱帐号 | 必要 |
| send\_date | datetime | 标准日期格式 | 发送日期 | 必要 |
| infoid | int | 10 | 外键（参考voteinfo.id） | 必要 |

**2.3数据库逻辑设计**

**图8.1 数据库逻辑模型**

## 3运用设计

**3.1安全性设计**

**防止用户直接操作数据库的方法**

数据库只能由管理员通过后台进行访问，没有其他途径操作数据库。

**角色与权限**

本系统除数据库管理员外没有任何其他用户，因此未对用户进行角色及权限的分配，数据库管理员具有所有对象的操作权限。

# 附录9 组装测试计划

## 1引言

**1.1编写目的**

目的是安排对系统将要进行的测试，并预估测试结果，供测试人员参考。

预期读者：本文档的预期读者为指导老师、项目的开发人员、项目的维护人员、测试人员、用户文档编写者、项目管理人员，也适用于客户。

**1.2背景**

开发软件名称：基于关联环签名的安全电子投票系统

项目任务提出者：韩镓维，许勤昆，凌雪，唐钰葆

项目开发者：许勤昆

用户：所有需要开展投票活动的单位或个人

实现软件单位：许勤昆

项目与其他软件，系统的关系：

本项目采用C++实现数字签名部分（关联环签名，盲签名等）；采用MySQL来实现数据库；

网站后端采用SSH框架，前端采用BootStrap以及jQuery等框架。

**1.3定义**

无。

**1.4参考资料**

《软件文档写作教程》电子工业出版社 主编：马平、黄冬梅

需求说明书

概要设计说明书

## 2计划

**2.1系统说明**

**表9.1**

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **所属模块** | **测试步骤** | **预期结果** |
| 1 | 投票管理中心  注册/登录模块 | 进入后台注册页面 | 输入账号密码，注册失败给出提示信息；  注册成功后，进入后台登录页面 |
| 进入后台登录页面 | 输入正确账号密码后进入后台管理首页，否则提示：“用户名或密码错误” |
| 2 | 创建投票 | 进入创建投票活动页面 | 输入投票活动信息后，创建投票活动，在投票活动列表中显示投票活动列表 |
| 3 | 添加候选人 | 进入添加候选人页面 | 填写候选人信息，并上传候选人照片，在投票信息详情页面可以查看候选人信息 |
| 4 | 邀请选民 | 进入选民邀请页面 | 正确填写邮箱信息及上传选民信息表，邮箱发送成功后提示“发送成功”，否则提示“发送失败” |
| 5 | 选民注册 | 点击邮箱链接，进入选民注册页面 | 注册成功后，向用户提供证书，用户下载证书 |
| 6 | 选民登录 | 进入登录页面  用户上传证书 | 验证成功后进入投票中心页面 |
| 7 | 选民投票 | 选择候选人进行投票 | 投票成功后，提示用户投票成功；用户不可重复投票 |
| 8 | 数字签名 | 关联环签名模块 | 注册阶段，为用户生成关联环签名 |
| 盲签名模块 | 为选票生成盲签名 |
| 消盲阶段 | 将盲化选票消盲，还原选票信息 |

**2.2 测试内容**

**2.2.1 公私钥对产生测试**

**测试目的**

保证产生的公私钥对符合使用的条件并且不重复。

**测试用例及结果**

**表9.2**

|  |  |
| --- | --- |
| 测试用例ID | LR-01-01 |
| 测试用例名称 | 公私钥对正常产生 |
| 测试用例说明 | 根据输入参与投票的人数产生相对的公私钥对 |
| 预置条件 | 通过Socket获得参与投票的人数 |
| 输入 | 参与投票的人数 |
| 预期结果 | 生成对应人数的公私钥对，并打印其中一对 |
| 实际结果 | 如图所示，正常生成五十对公私钥，并打印出其中第三十二对。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.3**

|  |  |
| --- | --- |
| 测试用例ID | LR-01-02 |
| 测试用例名称 | 公私钥对无重复情况 |
| 测试用例说明 | 根据输入参与投票的人数产生相对的公私钥对，验证公私钥对中是否存在重复数据。 |
| 预置条件 | 通过Socket获得参与投票的人数 |
| 输入 | 参与投票的人数 |
| 预期结果 | 生成对应人数的公私钥对，并验证是否有数据重复的情况。 |
| 实际结果 | 如图所示，将生成的存放公私钥对的向量先进行去重，然后查看向量的大小，如果向量大小不变，则证明不存在数据重复的情况。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**测试描述**

产生的公私钥对满足公私钥对性质，可正常用于之后的关联环签名。

**2.2.2 关联环签名正确签名验证测试**

**测试目的**

保证关联环签名符合电子投票系统的安全需求。

测试用例及结果

**表9.4**

|  |  |
| --- | --- |
| 测试用例ID | LR-02-01 |
| 测试用例名称 | 关联环签名正常运行 |
| 测试用例说明 | 关联环签名在投票人签名后经过投票中心验证为合法投票人 |
| 预置条件 | 无 |
| 输入 | 投票人需要签名的消息 |
| 预期结果 | 投票中心验证为合法投票人 |
| 实际结果 | 如图所示，关联环签名通过投票中心的验证为合法投票人。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.5**

|  |  |
| --- | --- |
| 测试用例ID | LR-02-02 |
| 测试用例名称 | 关联环签名签名私钥异常 |
| 测试用例说明 | 关联环签名在投票人签名阶段使用了他人私钥 |
| 预置条件 | 无 |
| 输入 | 投票人需要签名的消息 |
| 预期结果 | 投票中心验证为非合法投票人 |
| 实际结果 | 如图所示，关联环签名通过投票中心的验证为非合法投票人。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.6**

|  |  |
| --- | --- |
| 测试用例ID | LR-02-03 |
| 测试用例名称 | 关联环签名验证时公钥异常 |
| 测试用例说明 | 关联环签名在验证阶段疏漏了部分签名人 |
| 预置条件 | 无 |
| 输入 | 投票人需要签名的消息 |
| 预期结果 | 投票中心验证为非合法投票人 |
| 实际结果 | 如图所示，关联环签名通过投票中心的验证为非合法投票人。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.7**

|  |  |
| --- | --- |
| 测试用例ID | LR-02-04 |
| 测试用例名称 | 关联环签名验证时公钥异常 |
| 测试用例说明 | 关联环签名在验证阶段公钥环中有数据遭到修改 |
| 预置条件 | 无 |
| 输入 | 投票人需要签名的消息 |
| 预期结果 | 投票中心验证为非合法投票人 |
| 实际结果 | 如图所示，关联环签名通过投票中心的验证为非合法投票人。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.8**

|  |  |
| --- | --- |
| 测试用例ID | LR-02-05 |
| 测试用例名称 | 关联环签名验证时参数错误 |
| 测试用例说明 | 关联环签名在验证阶段参数C遭到修改 |
| 预置条件 | 无 |
| 输入 | 投票人需要签名的消息 |
| 预期结果 | 投票中心验证为非合法投票人 |
| 实际结果 | 如图所示，关联环签名通过投票中心的验证为非合法投票人。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.9**

|  |  |
| --- | --- |
| 测试用例ID | LR-02-06 |
| 测试用例名称 | 关联环签名验证时参数错误 |
| 测试用例说明 | 关联环签名在验证阶段参数Y遭到修改 |
| 预置条件 | 无 |
| 输入 | 投票人需要签名的消息 |
| 预期结果 | 投票中心验证为非合法投票人 |
| 实际结果 | 如图所示，关联环签名通过投票中心的验证为非合法投票人。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.10**

|  |  |
| --- | --- |
| 测试用例ID | LR-02-07 |
| 测试用例名称 | 关联环签名验证时参数错误 |
| 测试用例说明 | 关联环签名在验证阶段参数向量S有值遭到修改 |
| 预置条件 | 无 |
| 输入 | 投票人需要签名的消息 |
| 预期结果 | 投票中心验证为非合法投票人 |
| 实际结果 | 如图所示，关联环签名通过投票中心的验证为非合法投票人。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.11**

|  |  |
| --- | --- |
| 测试用例ID | LR-02-08 |
| 测试用例名称 | 关联环签名验证时消息异常 |
| 测试用例说明 | 关联环签名在验证阶段传入的消息遭到修改 |
| 预置条件 | 无 |
| 输入 | 投票人需要签名的消息 |
| 预期结果 | 投票中心验证为非合法投票人 |
| 实际结果 | 如图所示，关联环签名通过投票中心的验证为非合法投票人。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**测试描述**

关联环签名中，投票中心可以在不知道投票人身份的情况下确认他的合法身份，如果签名验证中出现异常和篡改，则确认为非合法投票人。

**2.2.3 普通RSA数字签名正确签名验证测试**

（1）测试目的

保证普通RSA数字签名符合电子投票系统中数据传输的安全需求。

（2）测试用例及结果

**表9.12**

|  |  |
| --- | --- |
| 测试用例ID | LR-03-01 |
| 测试用例名称 | 普通RSA数字签名正常运行 |
| 测试用例说明 | 在消息传递的过程中对消息签名，并在接收消息方进行验证，确认没有被篡改。 |
| 预置条件 | 无 |
| 输入 | 投票人需要传递的消息 |
| 预期结果 | 验证成功 |
| 实际结果 | 如图所示，消息确认未被篡改。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.13**

|  |  |
| --- | --- |
| 测试用例ID | LR-03-02 |
| 测试用例名称 | 普通RSA数字签名接收方消息被篡改 |
| 测试用例说明 | 普通RSA数字签名验证阶段的消息与签名时不符 |
| 预置条件 | 无 |
| 输入 | 投票人需要传递的消息 |
| 预期结果 | 验证失败 |
| 实际结果 | 如图所示，验证失败，证明数据被篡改。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.14**

|  |  |
| --- | --- |
| 测试用例ID | LR-03-03 |
| 测试用例名称 | 普通RSA数字签名接收方签名异常 |
| 测试用例说明 | 普通RSA数字签名验证阶段的签名发生错误 |
| 预置条件 | 无 |
| 输入 | 投票人需要传递的消息 |
| 预期结果 | 验证失败 |
| 实际结果 | 如图所示，验证失败。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

1. 测试描述

为保证系统中信息传递的安全，在信息传递过程中均使用普通RSA数字签名，防止信息被篡改。

**2.2.4 盲签名正确签名验证测试**

1. 测试目的

保证选票盲签名符合电子投票系统中的安全需求。

1. 测试用例及结果

**表9.15**

|  |  |
| --- | --- |
| 测试用例ID | LR-04-01 |
| 测试用例名称 | 盲签名过程正常运行 |
| 测试用例说明 | 首先投票人对选票盲化，投票中心对盲化选票签名，之后投票人对选票消盲 |
| 预置条件 | 无 |
| 输入 | 选票信息 |
| 预期结果 | 盲签名成功，选票未被篡改 |
| 实际结果 | 如图所示，盲签名成功，选票前后信息一致，确认未被篡改。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.16**

|  |  |
| --- | --- |
| 测试用例ID | LR-04-02 |
| 测试用例名称 | 盲签名过程盲化选票被修改 |
| 测试用例说明 | 在数据传输时盲化选票被篡改 |
| 预置条件 | 无 |
| 输入 | 选票信息 |
| 预期结果 | 盲签名失败 |
| 实际结果 | 如图所示，盲签名失败，同时消盲后可见选票信息前后不一致。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.17**

|  |  |
| --- | --- |
| 测试用例ID | LR-04-03 |
| 测试用例名称 | 盲签名过程盲化签名被修改 |
| 测试用例说明 | 在数据传输时签名被修改，投票中心无法确认选票的真实性 |
| 预置条件 | 无 |
| 输入 | 选票信息 |
| 预期结果 | 盲签名失败 |
| 实际结果 | 如图所示，盲签名失败，投票中心验证这不是自己签署的选票。 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

1. 测试描述

为保证投票人和投票中心都能验证选票信息未被篡改的同时，也能对选票信息保密，使用盲签名操作选票，并确认该过程的安全性。

**2.2.5 Web后台管理中心测试**

1. 测试目的：

保证后台管理中心正常运行

1. 测试用例及结果

**表9.18**

|  |  |
| --- | --- |
| 测试用例ID | LR-05-01 |
| 测试用例名称 | 后台登录测试 |
| 测试用例说明 | 管理员登录后台 |
| 预置条件 | 无 |
| 输入 | 用户名及密码 |
| 预期结果 | 用户名和密码匹配则进入后台，否则给出错误提示信息。 |
| 实际结果 | 1.登录失败页面    2.登录成功后跳转页面 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.19**

|  |  |
| --- | --- |
| 测试用例ID | LR-05-02 |
| 测试用例名称 | 后台登录测试 |
| 测试用例说明 | 创建投票信息 |
| 预置条件 | 无 |
| 输入 | 填写投票信息 |
| 预期结果 | 在用户填写所有表单并点击提交按钮之后，进入投票管理页面并显示所创建的投票内容 |
| 实际结果 | 1.创建投票信息    2.点击提交后进入投票管理页面 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.20**

|  |  |
| --- | --- |
| 测试用例ID | LR-05-03 |
| 测试用例名称 | 编辑投票信息 |
| 测试用例说明 | 修改已有投票的信息 |
| 预置条件 | 在已有的投票信息基础上进行编辑 |
| 输入 | 修改投票信息 |
| 预期结果 | 在用户填写所有表单并点击提交按钮之后，进入投票管理页面并显示修改的投票内容 |
| 实际结果 | 1.修改投票信息    2.点击提交后进入投票管理页面 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.21**

|  |  |
| --- | --- |
| 测试用例ID | LR-05-04 |
| 测试用例名称 | 查看投票信息 |
| 测试用例说明 | 查看投票信息的详细内容 |
| 预置条件 | 在已有的投票信息基础上进行查看 |
| 输入 | 点击“详情”按钮 |
| 预期结果 | 显示投票信息详细内容 |
| 实际结果 | 1.若无数据，给出提示信息    2.若有数据，显示内容 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.22**

|  |  |
| --- | --- |
| 测试用例ID | LR-05-05 |
| 测试用例名称 | 添加候选项目 |
| 测试用例说明 | 为投票内容添加候选项目 |
| 预置条件 | 在已有的投票信息基础上进行添加 |
| 输入 | 点击“添加数据”按钮 |
| 预期结果 | 将填写的数据写入数据库，如果有图片，将其相对路径填入数据库，并将文件保存在响应的文件夹下 |
| 实际结果 | 1.填写表单    2.提交后进入候选名单页面  表格显示：    图片显示：    3.数据库中插入相应数据 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.23**

|  |  |
| --- | --- |
| 测试用例ID | LR-05-06 |
| 测试用例名称 | 添加选民 |
| 测试用例说明 | 为投票内容添加选民 |
| 预置条件 | 在已有的投票信息基础上进行添加 |
| 输入 | 填写表单  C:\Users\admin\Desktop\QQ图片20170525164105.pngQQ图片20170525164105 |
| 预期结果 | 指定选民收到投票邮件 |
| 实际结果 | 1.邮件发送后，给出提示信息    2.发送成功后，进入投票信息详情页，可以看到已经有一条收件人的信息    3.选民查看邮箱 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.24**

|  |  |
| --- | --- |
| 测试用例ID | LR-05-07 |
| 测试用例名称 | 选民注册 |
| 测试用例说明 | 选民根据邮箱提供的链接，前往注册中心注册 |
| 预置条件 | 投票管理中心发送了投票邀请邮件 |
| 输入 | 点击链接 |
| 预期结果 | 将填写的数据写入数据库，如果有图片，将其相对路径填入数据库，并将文件保存在响应的文件夹下 |
| 实际结果 | 1.填写表单    2. 注册成功后提示：并给出UUID  3.List1中插入相应数据 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.25**

|  |  |
| --- | --- |
| 测试用例ID | LR-05-08 |
| 测试用例名称 | 选民登录 |
| 测试用例说明 | 根据注册得到的序列码进行登录 |
| 预置条件 | 用户需先注册，得到序列码 |
| 输入 | 序列码 |
| 预期结果 | 输入错误的序列码后给出错误提示信息；输入有效地序列码后进入投票中心首页 |
| 实际结果 | 1.输入错误的序列码    2. 输入正确的序列码 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表格9.26**

|  |  |
| --- | --- |
| 测试用例ID | LR-05-09 |
| 测试用例名称 | 开始投票 |
| 测试用例说明 | 用户进行投票 |
| 预置条件 | 用户需登录 |
| 输入 | 点击投票 |
| 预期结果 | 投票后生成对应结果信息 |
| 实际结果 | 1.进入投票页面    2. 点击投票 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

**表9.27**

|  |  |
| --- | --- |
| 测试用例ID | LR-05-10 |
| 测试用例名称 | 结果查看 |
| 测试用例说明 | 查看投票结果 |
| 预置条件 | 用户需登录 |
| 输入 | 无 |
| 预期结果 | 得到用户投票的信息 |
| 实际结果 | 1.用户个人投票信息    2. 结果统计 |
| 结果分析 | 测试通过 ； 测试未通过 🞏 |

# 附录10 详细设计说明书

## 1引言

**1.1编写目的**

目的是对系统每一个模块的实现进行详细的设计。

预期读者：本文档的预期读者为指导老师、项目的开发人员、项目的维护人员、测试人员、用户文档编写者、项目管理人员，也适用于客户。

**1.2背景**

开发软件名称：基于关联环签名的安全电子投票系统

项目任务提出者：韩镓维，许勤昆，凌雪，唐钰葆

项目开发者：许勤昆

用户：所有需要开展投票活动的单位或个人

实现软件单位：许勤昆

项目与其他软件，系统的关系：

本项目采用C++实现数字签名部分（关联环签名，盲签名等）；采用MySQL来实现数据库；

网站后端采用SSH框架，前端采用BootStrap以及jQuery等框架。

**1.3定义**

无。

**1.4参考资料**

《软件文档写作教程》电子工业出版社 主编：马平、黄冬梅

需求说明书

概要设计说明书

## 2详细设计

**2.1网站数据处理模块实**

**2.2.1 BlindSignature（盲签名算法部分）**

* **盲签名算法实现**

一个盲签名体制中存在两个参与实体：一个是消息拥有者，另一个是签名人。

**图10.1 盲签名算法示意图**

盲签名方案一般有如下六个阶段：

(1) 初始化阶段。一个概率多项式时间算法，输入安全参数，输出签名人的公私钥对（pk,sk）,私钥sk由签名人保密，公钥对公开。

(2) 盲化阶段。消息拥有者引入盲化因子T，对消息M进行盲化处理，得到盲化后的消息，并将发送给签名人。

(3) 盲化签名阶段。一个概率多项式时间的交互协议，签名人用私钥sk对盲化后的消息进行盲化签名，得到盲化后的签名，并将发送给消息拥有者。

(4) 盲化验证阶段。消息拥有者用公钥pk和盲化后的消息对盲化后的签名进行盲化验证，记作，输出“1”（表示签名有效），“0”（表示签名无效）。

(5) 去盲阶段。若盲化验证阶段输出“1”，说明签名有效，消息拥有者用盲化因子T，消息M和公钥pk对盲化后的签名进行去盲，得到签名σ。

(6) 签名验证阶段。用消息M和公钥pk对签名σ验证，记作，输出“1”（表示签名有效），“0”（表示签名无效）。

* **盲签名代码实现**

**（1）BlindBigInt类**

**表10.1 类BlindBigInt的实现**

| **编号** | **函数名** | **说明** |
| --- | --- | --- |
| 1 | BlindBigInt::BlindBigInt() | 构造函数1 |
| 2 | BlindBigInt::BlindBigInt(int) | 构造函数2 |
| 3 | BlindBigInt::BlindBigInt(int,int) | 构造函数3 |
| 4 | BlindBigInt::~BlindBigInt() | 析构函数 |
| 5 | BlindBigInt::isZero() | 判断是否为0 |
| 6 | BlindBigInt::toULong() | 将数值转换成是十进制的长整型数 |
| 7 | BlindBigInt::toString() | 将数值转换成字符串 |
| 8 | BlindBigInt::toHexString() | 将数值转换成十六进制格式表示的字符串 |
| 9 | BlindBigInt::operator[]() | 获得数值特殊一位上的值 |
| 10 | BlindBigInt::operator\*() | 将两个BlindBigInt数值相乘 |
| 11 | BlindBigInt::operator\*=() | 将两个BlindBigInt数值相乘并赋值 |
| 12 | BlindBigInt::operator+() | 将两个BlindBigInt数值相加 |
| 13 | BlindBigInt::operator>=() | 比较当前BlindBigInt数值是否大于等于参数BlindBigInt数值 |
| 14 | BlindBigInt::operator>() | 比较当前BlindBigInt数值是否大于参数BlindBigInt数值 |
| 15 | BlindBigInt::operator<=() | 比较当前BlindBigInt数值是否小于等于参数BlindBigInt数值 |
| 16 | BlindBigInt::operator<() | 比较当前BlindBigInt数值是否小于参数BlindBigInt数值 |
| 17 | BlindBigInt::operator/() | 将两个BlindBigInt数值相除 |
| 18 | BlindBigInt::operator==() | 判断两个BlindBigInt数值是否相等 |
| 19 | BlindBigInt::operator%() | 将两个BlindBigInt数值取余 |
| 20 | BlindBigInt::operator<<=() | 对BlindBigInt数值做左移位操作 |
| 21 | BlindBigInt::operator|=() | 将两个BlindBigInt数值做或位操作 |
| 22 | BlindBigInt::operator&=() | 将两个BlindBigInt数值做和位操作 |
| 23 | BlindBigInt::operator-() | 将两个BlindBigInt数值相减 |
| 24 | modPow() | x^ymodm |
| 25 | binToHex() | 将十进制字符串转换为十六进制字符串 |
| 26 | bitset::add() | 做位相加操作 |
| 27 | bitset::subtract() | 做位相减操作 |

**（2）BlindRSA类**

**（a）实现欧拉函数的变量**

**表10.2 实现欧拉函数的变量**

| 编号 | 变量 | 说明 |
| --- | --- | --- |
| 1 | Int p,q | 十七位的随机数,p!=q |
| 2 | BlindBigIntn | n=p\*q |
| 3 | BlindBigIntphi | Phi=(p-1)(q-1) |
| 4 | BlindBigInte | gcd(e,phi)==1 |
| 5 | BlindBigIntd | [e\*d==1]mod n |

**(b)实现欧拉函数的函数**

**表10.3 实现欧拉函数的函数**

| **编号** | **函数名** | **说明** |
| --- | --- | --- |
| 1 | BlindRSA::BlindRSA() | 构造函数1 |
| 2 | BlindRSA::BlindRSA(int) | 构造函数2 |
| 3 | BlindRSA::BlindRSA(int,int) | 构造函数3 |
| 4 | BlindRSA::~BlindRSA() | 析构函数 |
| 5 | BlindRSA::GetPHI() | 返回变量phi |
| 6 | BlindRSA::getP() | 返回变量p |
| 7 | BlindRSA::getQ() | 返回变量q |
| 8 | BlindRSA::getPublicKey() | 返回公钥 |
| 9 | BlindRSA::getPrivateKey() | 返回私钥 |
| 10 | BlindRSA::getModulus() | 返回modulus=p\*q |
| 11 | BlindRSA::setPublicKey(unsignedint) | 设置公钥 |
| 12 | BlindRSA::setPublicKey(BlindBigInt) | 设置公钥 |
| 13 | BlindRSA::encrypt() | 加密 |
| 14 | BlindRSA::decrypt() | 解密 |
| 15 | BlindRSA::modInverse() | 反向取余 |
| 16 | BlindRSA::isPrime() | 素数验证 |
| 17 | BlindRSA::isPrimeMR() | 素数验证 |
| 18 | BlindRSA::isPrimeDiv() | 素数验证 |
| 15 | BlindRSA::gcd(int,int) | Euclidean算法 |
| 16 | BlindRSA::gcd(BlindBigInt,BlindBigInt) | Euclidean算法 |

**2.2.2 LinkableRingSignature（关联环签名部分）**

* **关联环签名算法实现**

**(1) 结合函数（combiningfunction）**

结合函数环签名方案中的一个重要组成部分。

定义为结合函数，其中，k是输入，v是随机选取的初始值，是上的任意值，输出z是上的值。

选定k和v的值后，结合函数有以下的特点：

a) 对于每个s,，以及其他给定输入，方程，是从到输出z的一一映射。

b) 对于每个s,，给定一个b比特长读的值z，和除以外的所有的值，可以求解满足的b比特长度值。

c) 如果一个攻击者不能你想求解陷门方程中的某个方程，那么给定k , v和z,它就很难根据求解中的任何一个值。

d) 结合算数为以下形式：

其中定义为。

e) 只要输出z与v相等，该方程的计算过程就会连成一个环形，这也是环签名名称的由来。



**图10.2 环签名示意图**

**2）参数初始化**

设q是一个素数，是q阶子群,在G中求解离散对数问题是困难的。，是的单向散列函数。对i=1,2，...，n，每一用户拥有的私钥对，且。

**3) 签名生成**

设消息，公钥集合为，签名者私钥为，对应公钥为，其中，利用如下算法产生关联环签名：

a) 计算和，然后选择，计算。

b) 对，选择，计算。

c) 计算。

d) 最后，输出签名。

**（1）签名验证**

验证者受到关于m及L的签名后，按如下步骤验证签名的正确性：

a)计算，然后对于计算，，，。

b) 检验是否成立，如果成立，则接受签名，否则拒绝此签名。

* **关联环签名代码实现**

**(1) CommonFunction类**

**表10.4 类CommonFunction实现**

| **编号** | **函数名** | **说明** |
| --- | --- | --- |
| 1 | CommonFunctions::GetGroupParameters() | 获得群参数 |
| 2 | CommonFunctions::IntegerToString() | 将CryptoPP::integer转为字符串 |
| 3 | CommonFunctions::GenerateString() | 将向量中的各值转为字符串 |
| 4 | CommonFunctions::Hash1() | SHA1哈希函数 |
| 5 | CommonFunctions::Hash2() | SHA2哈希函数 |

**(2) LinkableRingSignProver类**

**a) 实现的变量**

**表10.5 LinkableRingSignProver变量**

| **编号** | **变量** | **说明** |
| --- | --- | --- |
| 1 | Integer\_private\_key | 私钥 |
| 2 | unsignedint\_self\_identity | 当前环成员 |
| 3 | unsignedintnum\_members | 环成员人数 |
| 4 | Integerg | 群 |
| 5 | Integerq | 群序 |
| 6 | stringm | 签名消息 |
| 7 | vector<Integer>public\_keys | 公钥向量 |
| 8 | IntegerC | 关联环参数 |
| 9 | IntegerY | 关联环参数 |
| 10 | vector<Integer>S | 关联环参数 |

**b) 实现的函数**

**表10.6 LinkableRingSignProver函数**

| **编号** | **函数名** | **说明** |
| --- | --- | --- |
| 1 | LinkableRingSignProver::LinkableRingSignProver() | 构造函数 |
| 2 | LinkableRingSignProver::GenerateSignature() | 产生签名 |
| 3 | GeneratePublicPrivateKeys() | 产生公私钥对 |

**(3) LinkableRingSignVerifier类**

1. **实现的变量**

**表10.7 LinkableRingSignVerifier变量**

| 编号 | 变量 | 说明 |
| --- | --- | --- |
| 1 | IntegerC | 关联环参数 |
| 2 | IntegerY | 关联环参数 |
| 3 | vector<Integer>S | 关联环参数 |
| 4 | unsignedintnum\_members | 环成员人数 |
| 5 | Integerg | 群 |
| 6 | Integerp | 素数 |
| 7 | Integerq | 群序 |
| 8 | stringm | 签名消息 |
| 9 | vector<Integer>public\_keys | 公钥向量 |

1. **实现的函数**

**表10.8 LinkableRingSignVerifier函数**

| **编号** | **函数名** | **说明** |
| --- | --- | --- |
| 1 | LinkableRingSignVerifier::LinkableRingSignVerifier() | 构造函数 |
| 2 | LinkableRingSignVerifier::VerifySignature() | 验证函数 |

**2.2.3 NormalRSASignature（普通RSA数字签名部分）**

* **普通RSA数字签名算法实现**

数字签名是签名者的利用自己私钥对消息进行加密生成的签名，验证者利用签名者公钥进行解密验证签名的一种密码技术。

数字签名体制一个数字签名体质主要由下列部分构成：

(1) 一个明文消息空间M：所有待签名的消息组成的集合；

(2) 一个签名空间Ω：所有签名形成的集合；

(3) 一个签名密钥空间K:所有的可能用于生成签名的密钥的集合；

(4) 一个认证密钥空间：所有的可能由于验证签名的密钥结合；

(5) 一个有效的密钥生成算法，其中K和为（3）中定义，N为安全参数空间，当输入安全参数空间N内的一个安全参数时，算法Gen在多项式时间内生成一对公私钥对；

(6) 一个有效的签名算法，对于消息空间M内的一个给定的消息，算法Sign利用算法Gen生成的签名密钥，在多项式时间内生成一个有效签名；

(7) 一个有效的验证算法，对于给定的消息和算法Sign生成的签名，利用算法Gen生成的验证密钥验证签名的正确性。

(8) 对任意的和任意的表示σ是用密钥sk生成的m的签名。

(9) 对于任意的私钥，用pk表示与sk相对应的公钥，则对应于和必有

* **普通RSA数字签名代码实现**

**（1）normalSign类**

**a) 实现的变量**

**表10.9 类normalSign的变量**

| **编号** | **变量** | **说明** |
| --- | --- | --- |
| 1 | stringm | 消息 |
| 2 | AutoSeededRandomPool | 随机种子参数 |
| 3 | IntegerN,P,Q,D,E | RSA签名参数 |
| 4 | stringencoded | 解密字符串 |
| 5 | boolresult | 验证签名结果 |
| 6 | InvertibleRSAFunction | RSA签名钥对 |

**b) 实现的函数**

**表10.10 类normalSign的函数**

| **编号** | **函数名** | **说明** |
| --- | --- | --- |
| 1 | normalSign::normalSign() | 构造函数 |
| 2 | normalSign::getNormalRSASignature() | RSA数字签名 |
| 3 | normalSign::verifyNormalRSASignature() | RSA数字签名验证 |

**2.2 Web后台设计**

后台采用SSH（spring，struts2，Hibernate）框架；spring负责事务管理；struts2为控制层，负责处理前端页面提交的数据，并返回相应的结果；Hibernate负责对象的持久化映射。详细实现如下所示：

1. **Action层**
2. AdminLoginAction
3. 所用变量表

**表10.11 实现AdminLoginAction变量**

| 名称 | 数据类型 | 属性 |
| --- | --- | --- |
| admin | Admin | private |
| result | String | private |

1. 所用函数表

**表10.12 实现AdminLoginAction函数**

| 函数名 | 参数 | 返回类型 |
| --- | --- | --- |
| AdminLoginAction | void | void |
| getResult | void | String |
| getAdmin | void | Admin |
| setAdmin | Admin | void |
| execute | void | String |

响应管理员登录事件：当管理员登录后台时，此action查询该管理员帐号密码是否存在于数据库中；如果存在则验证通过，进入后台管理页面，反之向前端页面发出错误提示信息，要求用户重新输入账号密码。

1. **RegisterAction**
2. 所用变量表

**表10.13 实现RegisterAction变量**

| 名称 | 数据类型 | 属性 |
| --- | --- | --- |
| user | User | private |
| info | VoteInfo | private |

1. 所用函数表

**表10.14 实现RegisterAction函数**

| 函数名 | 参数 | 返回类型 |
| --- | --- | --- |
| RegisterAction | void | void |
| getUser | void | User |
| setUser | User | void |
| getInfo | void | VoteInfo |
| setInfo | VoteInfo | void |
| register | void | String |

1. LoginAction
2. 所用变量表

**表10.15 实现LoginAction变量**

| 名称 | 数据类型 | 属性 |
| --- | --- | --- |
| user | User | private |
| uuid | String | private |

1. 所用函数表

**表10.16 实现LoginAction函数**

| 函数名 | 参数 | 返回类型 |
| --- | --- | --- |
| getUuid | void | String |
| setUuid | String |  |
| getUser | void | User |
| setUser | User | void |
| login | void | String |
| checkUuid | void | boolean |

响应用户登录事件：用户持有唯一的身份序列码（uuid），用户通过此uuid进行登录，此action查询数据表List1中是否存在该uuid，如果存在则登录成功。

1. **AddVoteitemAction**
2. 所用变量表

**表10.17 实现AddVoteitemAction变量**

| 名称 | 数据类型 | 属性 |
| --- | --- | --- |
| info | VoteInfo | private |
| item | VoteItem | private |
| upload | File | private |
| uploadFileName | String | private |
| uploadContentType | String | private |
| savePath | String | private |

1. 所用函数表

**表10.18 实现AddVoteitemAction函数**

| 函数名 | 参数 | 返回类型 |
| --- | --- | --- |
| getInfo | void | VoteInfo |
| setInfo | VoteInfo | void |
| getItem | void | Item |
| setItem | Item | void |
| getUpload | void | File |
| setUpload | File | boolean |
| getUPloadFileName | void | String |
| setUPloadFileName | String | void |
| getUploadContentType | void | String |
| setUploadContentType | String | void |
| getSavePath | void | String |
| setSavePath | String | void |
| addVoteItem | void | String |

添加候选名单：主要负责文件上传以及保存管理员添加的候选名单。通过struts2的标签<s:file/>，将文件保存在属性upload中，uploadFileName保存图片名称，uploadContentType保存文件类型，savePath保存文件存放的路径。

方法addVoteItem()负责文件的保存与数据的持久化。

1. ServiceAction
2. 所用变量表

**表10.19 实现ServiceAction函数**

| 名称 | 数据类型 | 属性 |
| --- | --- | --- |
| info | VoteInfo | private |
| item | VoteItem | private |
| sender | Email | private |
| text | Text | private |

1. 所用函数表

**表10.20 实现ServiceAction函数**

| 函数名 | 参数 | 返回类型 |
| --- | --- | --- |
| getText | void | Text |
| setText | Text | void |
| getSender | void | Email |
| setSender | Email | void |
| getItem | void | VoteItem |
| setItem | VoteItem | void |
| getInfo | void | VoteInfo |
| setInfo | VoteInfo | void |
| getVoteItem | void | String |
| getEmails | void | String |
| setEmails | String | void |
| createVoteInfo | void | String |
| getVoteInfo | void | String |
| deleteVoteInfo | void | String |
| editVoteInfo | void | String |
| getSendEmail | void | String |

1. 主要负责的后台业务

**表10.21 ServiceAction主要负责的后台业务**

| 属性/方法 | 作用/工作 |
| --- | --- |
| info:VoteInfo | 存储投票信息 |
| item:VoteItem | 存储候选项目 |
| sender:Email | 存储发件人邮箱 |
| text:Text | 存储邮件内容 |
| createVoteInfo() | 创建投票 |
| getVoteInfo() | 获取投票信息列表 |
| deleteVoteInfo() | 删除投票信息 |
| editVoteInfo() | 编辑投票信息 |
| getEmail() | 获取数据库存储的邮箱列表 |
| getSendEmail() | 获取一发送的邮件列表 |
| sendEmails() | 发送邮箱 |

1. VoteAction

**表10.22 实现VoteAction**

|  |  |
| --- | --- |
| 属性/方法 | 作用/功能 |
| itemid:int | 候选名单 |
| vote() | 获取用户投票信息，并将处理后的数据存入数据库 |

1. **DAO（数据访问对象）层**

**表10.23 数据访问对象层的实现**

| 类名 | 功能 |
| --- | --- |
| AdminDao.java | 注册功能操作 |
| Email.java | 邮件功能操作 |
| ListDao.java | 列表功能操作 |
| ReceiverDao.java | 接受功能操作 |
| VoteDao.java | 选举功能操作 |

该层主要负责数据的CRUD操作，通过Hibernate的SessionFactory生成一个与数据库的session会话，由该session开启一个事务，在该事务内实现相应的增删改查操作。由于各个类的功能大同小异，这里不再多做赘述。

1. **util层（工具类）**

**表10.24 相关工具的实现**

| 类名 | 功能 |
| --- | --- |
| Base64Util.java | 将数据进行Base64编码以及解码工作，为邮件发送提供数据处理服务 |
| DateUtil.java | 获取系统当前时间并进行格式化 |
| EmailUtil.java | 邮件发送工具类，通过jmail提供的smtp服务发送邮件 |
| FileUpload.java | 文件上传工具类，将上传的文件保存到本地。 |
| GenerateUUID.java | 为新注册的用户生成一个uuid |
| HibernateUtil.java | SessionFactory以及Session的创建工具类 |
| TCPUtil.java | TCP客户端工具类，对每一次与C++代码的交互建立一次TCP连接，并发送和接收相应的数据。 |

# 附录11 模块开发说明

## 1引言

**1.1编写目的**

目的是对之前设计的系统各个模块进行实现。

预期读者：本文档的预期读者为指导老师、项目的开发人员、项目的维护人员、测试人员、用户文档编写者、项目管理人员，也适用于客户。

**1.2背景**

开发软件名称：基于关联环签名的安全电子投票系统

项目任务提出者：韩镓维，许勤昆，凌雪，唐钰葆

项目开发者：许勤昆

用户：所有需要开展投票活动的单位或个人

实现软件单位：许勤昆

项目与其他软件，系统的关系：

本项目采用C++实现数字签名部分（关联环签名，盲签名等）；采用MySQL来实现数据库；

网站后端采用SSH框架，前端采用BootStrap以及jQuery等框架。

**1.3定义**

无。

**1.4参考资料**

《软件文档写作教程》电子工业出版社 主编：马平、黄冬梅

需求说明书

概要设计说明书

## 2模块开发情况表

**2.1 数据处理模块主要代码**

**2.1.1 BlindRSA**

**代码清单11.1 BlindRSA.cpp**

|  |
| --- |
| namespace RSAUtil  {  #define A\_MAX 25  #define RAND\_LIMIT 0xFFFF    BlindRSA::BlindRSA(int p1, int q1){  BlindRSA::e = 0;  BlindRSA::d = 0;  BlindRSA::p = p1;  BlindRSA::q = q1;    srand(time(0));  BlindRSA::n = BlindBigInt(BlindRSA::p)\*BlindBigInt(BlindRSA::q);  BlindRSA::phi = BlindBigInt(((BlindRSA::p) - 1))\*BlindBigInt(((BlindRSA::q) - 1));  }  BlindRSA::BlindRSA(int p1){  bool isP;  BlindRSA::e = 0;  BlindRSA::d = 0;  BlindRSA::p = p1;    srand(time(0));    //Find q that is prime and not equal to p. Check that p!=q first  do{  BlindRSA::q = int(((double)std::rand() / RAND\_MAX)\*RAND\_LIMIT);  //set the low bit and high bit.  BlindRSA::q = BlindRSA::q | 0x10001;  isP = isPrime(BlindRSA::q);    } while ((BlindRSA::p == BlindRSA::q) || !isP);      BlindRSA::n = BlindBigInt(BlindRSA::p)\*BlindBigInt(BlindRSA::q);  BlindRSA::phi = BlindBigInt(((BlindRSA::p) - 1))\*BlindBigInt(((BlindRSA::q) - 1));    }  BlindRSA::BlindRSA()  {  bool isP;  BlindRSA::e = 0;  BlindRSA::d = 0;    //find p & q, s.t. p!=q && p and q are both prime.  srand(time(0));    do{  BlindRSA::p = int(((double)std::rand() / RAND\_MAX)\*RAND\_LIMIT);    // Set low bit (for oddness) and high bit (to make sure it is large enough).  BlindRSA::p = BlindRSA::p | 0x10001;  isP = isPrime(BlindRSA::p);  }while(!isP);    //Find q that is prime and not equal to p. Check that p!=q first since  //that is the easier check.  do{  BlindRSA::q = int(((double)std::rand() / RAND\_MAX)\*RAND\_LIMIT);  //set the low bit and high bit.  BlindRSA::q = BlindRSA::q | 0x10001;  isP = isPrime(BlindRSA::q);    } while ((BlindRSA::p == BlindRSA::q) || !isP);      BlindRSA::n = BlindBigInt(BlindRSA::p)\*BlindBigInt(BlindRSA::q);  BlindRSA::phi = BlindBigInt(((BlindRSA::p) - 1))\*BlindBigInt(((BlindRSA::q) - 1));  }  BlindRSA::~BlindRSA()  {  }  void BlindRSA::setPublicKey(unsigned int pubKey){  BlindRSA::e = pubKey;  }  // The 2 functions below added by Raghunathan Srinivasan  void BlindRSA::setN(BlindBigInt B)  {  BlindRSA::n = B;  }  // overloaded function created by Raghu  void BlindRSA::setPublicKey(BlindBigInt B)  {  BlindRSA::e = B;  }  // end of code addition  int BlindRSA::getP() const{  return BlindRSA::p;  }  int BlindRSA::getQ() const{  return BlindRSA::q;  }  BlindBigInt BlindRSA::getPublicKey(){  //If e has not been set, calculate e, o/w just return it.  if (BlindRSA::e.isZero()){  calcE();  }  return BlindRSA::e;  }  BlindBigInt BlindRSA::getPrivateKey(){  //If d has not been set, calculate d, o/w just return it.  if (BlindRSA::d.isZero()){  calcD();  }  return BlindRSA::d;  }  BlindBigInt BlindRSA::getPHI() const{  return BlindRSA::phi;  }  BlindBigInt BlindRSA::getModulus() const{  return BlindRSA::n;  }  //calculates m^e mod n  BlindBigInt BlindRSA::encrypt(BlindBigInt msg){  BlindBigInt cipher;    if (BlindRSA::e.isZero()){  calcE();  }  cipher = RSAUtil::modPow(msg, BlindRSA::e, BlindRSA::n);  return cipher;  }  //calculates c^d mod n  BlindBigInt BlindRSA::decrypt(BlindBigInt cipher){  BlindBigInt message;  if (BlindRSA::d.isZero()){  calcD();  }  //Do decryption  message = RSAUtil::modPow(cipher, BlindRSA::d, BlindRSA::n);    return message;  }  /\*\* test code by raghu \*/  /\* end of test code\*/  void BlindRSA::calcE(){    //Find e such that 1 < e < PHI, and e is relatively prime to PHI  BlindBigInt r;  unsigned int high, low;  bool done = false;  BlindBigInt tempPhi;  tempPhi = BlindRSA::phi;    while(!done){  //need to generate a 32-34 bit random number.  //generate 32 bit random num.  //add 33rd bit. either 0,1,or 2.  low = int(((double)std::rand()/RAND\_MAX)\*0xFFFFFFFF);  high = int(((double)std::rand()/RAND\_MAX)\*0x02);  r = BlindBigInt(high, low);    //Make sure r is in the middle 2/3 of PHI.  if ((r>(BlindRSA::phi / 6)) && r<((BlindRSA::phi / 6) \* 5)){  r |= 0x01;  done = (gcd(BlindRSA::phi, r) == 1);  }  }//end while loop.  BlindRSA::e = r;  }  void BlindRSA::calcD(){  //Find d such that de = 1 (mod PHI). d exists if e and PHI are relatively prime.  BlindBigInt response;    if (BlindRSA::e.isZero()){  calcE();  }    response = modInverse(BlindRSA::e, BlindRSA::phi);  BlindRSA::d = response;  }  //Composite testing.  bool isPrime(int p){    bool isP;  //Check if it is divisible by a small prime.  isP = isPrimeDiv(p);  //If it isn't, check for primality using Miller-Rabin algorithm.  if(isP){  isP = isPrimeMR(p);  }  return isP;    }  bool isPrimeMR(int p){    unsigned int a, b, m, j, tempPow2, pow2;  bool maybePrime = true;  BlindBigInt z;  //Easy check.  if(p == 2){  return true;  }  //Check for even numbers.  if(!(p & 0x1)){  return false;  }  //Calculate m, such that p = 1 + (pow2\*m), where pow2 is the largest power  //of 2 that divides p-1.  pow2 = 1;  tempPow2 = 2;  b = 0;  while((p-1)%tempPow2 == 0){  b++;  pow2 = tempPow2;  //multiply by 2.  tempPow2 = tempPow2 << 1;  }  m = (p-1)/pow2;  int iter = 0;  while(iter < 5 && maybePrime){  //a must be less than p.  do{  a = int(((double)std::rand()/RAND\_MAX)\* A\_MAX);  }while(a >= p);  if(a==0 || a==1){  a = 2;  }  j = 0;  z = RSAUtil::modPow(BlindBigInt(a), BlindBigInt(m), BlindBigInt(p));  if(z==1 || z==(p-1)){  //p passes. it may be prime.  maybePrime = true;  }else{  j++;  while(j<b && !(z==(p-1)) && !(z==1)){  z = modPow(z, (unsigned int)2, (unsigned int)p);  j++;  }  if(z == 1){  //p is not prime.  maybePrime = false;  }  else if(j == b && !(z == (p-1))){  //p is not prime.  maybePrime = false;  }  if(z == (p-1)){  //p may be prime.  maybePrime = true;  }  }  iter ++;  }//end while loop.    return maybePrime;  }  bool isPrimeDiv(int p){  // Test all primes < 256.  //use a wheel to generate 1st 2000 primes.  bool response = true;  int primes[] = {2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97,101,103,107,109,113,127,131,137,139,149,151,157,163,167,173,179,181,191,193,197,199,211,223,227,229,233,239,241,251};  int plength = sizeof(primes)/sizeof(int);    for(int i = 0; i < plength; i++){  if(p%primes[i] == 0){  response = false;  break;  }  }  return response;  }  int gcd(int i, int j){  if(j == 0){  return i;  }  else{  return gcd(j, i%j);  }  }  BlindBigInt gcd(BlindBigInt i, BlindBigInt j){  if(j==0){  return i;  }  else{  return gcd(j, i%j);  }    }  //extended Euclidean algorithm. Find b s.t. ab = 1 mod m  BlindBigInt modInverse(BlindBigInt a, BlindBigInt m){  bool neg = false;  BlindBigInt b;  BlindBigInt u1, u2, u3, v1, v2, v3, t1, t2, t3, q;  u1 = 1; u2 = 0; u3 = m;    v1 = 0; v2 = 1; v3 = a;    while(!((u3%v3).isZero())){  q = (u3/v3);  t1 = u1 - (q\*v1);  t2 = u2 - (q\*v2);  t3 = u3 - (q\*v3);  u1 = v1;  u2 = v2;  u3 = v3;  v1 = t1;  v2 = t2;  v3 = t3;    }    //v2 is neg  if (v2[BIGINT\_SIZE-1]==1){  //take 2's comp  v2.flip();  v2 = v2+1;  neg = true;  }    b = v2%m;  if(neg){  b = m-b;  }    return b;  }  } |

**2.1.2 LinkableSignProver**

**代码清单11.2 LinkableRingSignProver.cpp**

|  |
| --- |
| LinkableRingSignProver::LinkableRingSignProver(unsigned int n,  unsigned int identity, Integer g\_in, Integer p\_in,  Integer q\_in, vector<Integer> public\_keys\_in, Integer private\_key\_in):  \_private\_key(private\_key\_in),  \_self\_identity(identity),  member\_size(n),  g(g\_in), p(p\_in), q(q\_in)  {  for(vector<Integer>::iterator itr = public\_keys\_in.begin();  itr != public\_keys\_in.end(); itr++)  {  public\_keys.push\_back(\*itr);  }  }  /\*Function to generate signature.  returns updated arguments (i.e. signature : (c1, s1...sn, y))  \*/  void LinkableRingSignProver::GenerateSignature(string message, Integer &c1,  vector<Integer> &S, Integer &Y)  {  //set message  this->message = message;  //compute h = H2(L)，y = h exp privatekey  string h\_str = GenerateString(public\_keys);  Integer h(Hash2(h\_str, p, q, g).c\_str());    Integer y\_tilde = a\_exp\_b\_mod\_c(h, \_private\_key, p);  RandomPool rng;  Integer u(rng, 0, q - 1);  Integer \*ci = new Integer[member\_size];  Integer \*si = new Integer[member\_size];  string temp = GenerateString(public\_keys) + IntegerToString(y\_tilde) +  message + IntegerToString(a\_exp\_b\_mod\_c(g, u, p)) +  IntegerToString(a\_exp\_b\_mod\_c(h, u, p));  Integer b(Hash1(temp).c\_str());  ci[(\_self\_identity + 1) % member\_size] = b;    //initialise parameter for iteration  unsigned int i = (\_self\_identity + 2) % member\_size;    for(; i != \_self\_identity + 1; i = (i + 1) % member\_size)  {  int j = (i + member\_size - 1) % member\_size;  si[j] = (Integer(rng, 0, q - 1));      temp = Hash1(GenerateString(public\_keys) +  IntegerToString(y\_tilde) + message +  IntegerToString(a\_times\_b\_mod\_c(a\_exp\_b\_mod\_c(g, si[j], p),  a\_exp\_b\_mod\_c(public\_keys[j], ci[j], p), p)) +  IntegerToString(a\_times\_b\_mod\_c(a\_exp\_b\_mod\_c(h, si[j], p),  a\_exp\_b\_mod\_c(y\_tilde, ci[j], p), p)));  ci[i] = Integer(temp.c\_str());  }  //update si\_pi value (i.e. corresponding to self identity)  si[\_self\_identity] = (u % q - a\_times\_b\_mod\_c(\_private\_key,  ci[\_self\_identity], q)) % q;    //update signature parameters (input args)  S.clear();  for(unsigned int i = 0; i < member\_size; i++)  S.push\_back(si[i]);  c1 = ci[0];  Y = y\_tilde;  //free memory  delete[] ci; /\* RESPONSIBLE FOR MEMORY LEAKS!!\*/  delete[] si;  } |

**2.1.3 LinkableRingSignatureVerifier**

**代码清单11.3 LinkableRingSignatureVerifier.cpp**

|  |
| --- |
| LinkableRingSignVerifier::LinkableRingSignVerifier(unsigned int num\_members\_in,  vector<Integer> pub\_keys,  Integer q\_in, Integer p\_in,  Integer g\_in, string m\_in):  member\_size(num\_members\_in),  q(q\_in), p(p\_in), g(g\_in),  message(m\_in)  {  //clear vector  public\_keys.clear();  for(vector<Integer>::iterator itr = pub\_keys.begin();  itr != pub\_keys.end(); itr++)  {  public\_keys.push\_back(\*itr);  }    assert(public\_keys.size() != 0);  }  /\*Function to verify signature  @params: c1, s1..sn, y\_tilde -- refer algo pdf for details.  \*/  bool LinkableRingSignVerifier::verify(Integer &C, vector<Integer> &S,  Integer &Y)  {  //cout << C << endl;  string temp\_str = Hash2(GenerateString(public\_keys), p, q, g);  Integer h(temp\_str.c\_str());    // zi and zi'  Integer zi, zi\_dash;  Integer ci = C;  //cout << endl;  for(unsigned int i = 0; i < member\_size; i++)  {  zi = a\_times\_b\_mod\_c(a\_exp\_b\_mod\_c(g, S[i], p),  a\_exp\_b\_mod\_c(public\_keys[i], ci, p), p);  zi\_dash = a\_times\_b\_mod\_c(a\_exp\_b\_mod\_c(h, S[i], p),  a\_exp\_b\_mod\_c(Y, ci, p), p);  ci = Integer(Hash1(GenerateString(public\_keys) +  IntegerToString(Y) + message +  IntegerToString(zi) +  IntegerToString(zi\_dash)).c\_str());  // cout << ci << endl;  }    #ifdef DEBUG  cout << "C : " << C <<endl;  cout << "H': " << ci <<endl;  #endif  return (C == ci);  } |

**2.2 后台处理模块主要代码**

**2.2.1投票管理模块**

* **AdminAction.java**

**(1) register**

当用户进入后台投票中心注册账户时，调用此方法，该函数首先检查该账户名是否被使用，如果没有被使用则保存该用户。

**代码清单11.4 register**

|  |
| --- |
| **public** String register() **throws** Exception {  AdminDao ad = **new** AdminDao();  **if**(admin==**null** || admin.getName() == **null**) {  **this**.addActionError("请填写用户名！");  **return** ***INPUT***;  }  **if**(ad.findByName(admin.getName()) != **null**) {  **this**.addActionError("该用户已经存在！");  **return** ***INPUT***;  } **else** {  admin.setAuthority(5);  ad.save(admin);  **this**.addFieldError("register\_result", "注册成功！请登录");  }    **return** ***SUCCESS***;  } |

**(2) login**

当用户进入后台管理中心登录页面时，点击登录后调用此方法；该方法验证此用户名及密码是否匹配

**代码清单11.5 login**

|  |
| --- |
| **public** String login() **throws** Exception {  AdminDao ad = **new** AdminDao();  Map<String, Object> application = ActionContext.*getContext*().  getApplication();  **if**(admin==**null** || admin.getName() == **null**) {  **return** ***INPUT***;  }  **if**(ad.validate(admin)) {  List<Admin> list = ad.select(admin);    **if**(list == **null** || list.size() == 0) {  **this**.addActionError("用户名或密码错误！");  **return** ***INPUT***;  }    admin = list.get(0);//加载该管理员个人信息  // 设置登录次数  **if**(admin.getLoginTimes() == **null**)  admin.setLoginTimes(1);  **else**  admin.setLoginTimes(admin.getLoginTimes() + 1);  // 设置登录时间  admin.setLastLogin(DateUtil.*getDateTime*());  ad.update(admin);  application.put("admin", admin);  **if**(admin.getAuthority() == **null** || admin.getEmail() == **null** ||  admin.getId\_card() == **null** || admin.getRealname() == **null**) {  application.put("isComplete", **false**);  }  **return** ***SUCCESS***;  } **else**{  **this**.addActionError("用户名或密码错误！");  System.***out***.println("failed");  **this**.addFieldError("error", "用户名或密码错误");  **return** ***INPUT***;  }  } |

**(3) logout**

当用户注销登录时，调用此方法

**代码清单11.6 logout**

|  |
| --- |
| **public** String logout() **throws** Exception {  Map<String, Object> application = ActionContext.*getContext*().getApplication();  Admin admin = (Admin) application.get("admin");  **if**(admin == **null** || admin.getId() == **null**) {  **return** ***ERROR***;  }  AdminDao ad = **new** AdminDao();  admin = ad.findAdminById(admin.getId());  ad.update(admin);  application.put("admin", **null**);  **return** ***SUCCESS***;  } |

**(4) editAdminInfo**

编辑管理员信息，当用户修改个人信息时，调用此方法，更新被修改了的字段。

**代码清单11.7 editAdminInfo**

|  |
| --- |
| **public** String editAdminInfo() **throws** Exception {  AdminDao ad = **new** AdminDao();  Map<String, Object> application = ActionContext.*getContext*().getApplication();  System.***out***.println("adminId = " + admin.getId());  Admin oldAdmin = ad.findAdminById(admin.getId());  **if**(oldAdmin != **null**) {  **if**(admin.getAuthority() == **null**)  admin.setAuthority(oldAdmin.getAuthority());  **if**(admin.getPassword() == **null**)  admin.setPassword(oldAdmin.getPassword());  **if**(admin.getName() == **null**)  admin.setName(oldAdmin.getName());  **if**(admin.getEmail() == **null**)  admin.setEmail(oldAdmin.getEmail());  **if**(admin.getId\_card() == **null**)  admin.setId\_card(oldAdmin.getId\_card());  **if**(admin.getRealname() == **null**)  admin.setRealname(oldAdmin.getRealname());  **if** (admin.getLoginTimes() == **null**) {  admin.setLoginTimes(oldAdmin.getLoginTimes());  }  **if**(ad.update(admin)) {  System.***out***.println("更新成功");  application.put("isComplete", **true**);  application.put("admin", admin);  **return** ***SUCCESS***;  } **else** {  System.***out***.println("更新失败");  **return** ***ERROR***;  }  } **else**{  **this**.addActionError("无此用户信息");  System.***out***.println("修改失败");  **return** ***ERROR***;  }    } |

* **EmailServiceAction**

发起投票邀请，编辑邮件并发送给选民

**代码清单11.8 sendEmails**

|  |
| --- |
| **public** String sendEmails() **throws** Exception {  HttpServletRequest httpRequest = ServletActionContext.*getRequest*();  String [] emails = httpRequest.getParameterValues("email\_list");  Map request = (Map) ActionContext.*getContext*().get("request");  **if**(emails==**null** || emails.length == 0) {  request.put("resolve\_email\_info", "获取邮箱列表失败");  **return** ***ERROR***;  }    **int** infoid = info.getId();  VoteDao vd = **new** VoteDao();  VoteInfo vi = vd.findVoteInfoById(infoid).get(0);  //获取收件人邮箱  List<String> recivers = Arrays.*asList*(emails); // 避免重复存储  saveReciver(recivers);  String url = getURL(infoid);    // 编辑邮件内容  String content = "\n<h3>投票信息</h3>\n";  content += "<h4>标题：</h4>" + vi.getTitle() + "\n";  content += "<h4>内容：</h4>" + vi.getContent() + "\n";  String link = "<h4>前往注册中心<a href=\"" + url + "\"></a></h4>";  text.setContent( content + text.getContent() + link + url);  Map request\_context = (Map) ActionContext.*getContext*().get("request");    //发送邮件  **if**(EmailUtil.*sendMany*(sender, recivers, text)) {  request\_context.put("info", info);  **return** ***SUCCESS***;  }  **else** {  request.put("resolve\_email\_info", "用户名或密码错误，请重试");  **return** ***ERROR***;  }  } |

* **FileUpDownloadAction**

上传选民信息表（excel文档），解析excel表格，将用户邮箱以json格式发送到前端页面

**代码清单11.9 getUserList**

|  |
| --- |
| **public** **void** getUserList() **throws** Exception {  File file = **new** File(savePath);  **if** (!file.exists()) {  System.***out***.println("文件目录不存在,创建目录："+savePath);  **if** (file.mkdir()) {  System.***out***.println("创建目录"+savePath+"成功");  }  }  // 存储路径  String path = ServletActionContext.*getServletContext*().  getRealPath(savePath) + "\\" + uploadFileName;  // 存储文件到本地  FileUtil.*save*(upload, path);  List<User> userList = ExcelResolveUtil.*resolveExcel*(path);  List<String> emailList = **new** ArrayList<String>();  String emailStr = "";  **for** (**int** i = 0; i < userList.size(); i++) {  User user = userList.get(i);  emailList.add(user.getEmail());  emailStr += user.getEmail();  **if**(i < userList.size() - 1)  emailStr += ";";  }  HttpServletResponse response = ServletActionContext.*getResponse*();  PrintWriter writer = response.getWriter();  writer.println(emailStr);  writer.flush();  } |

* **ServiceAction**

主要涉及与投票活动相关的方法

**(1) createVoteInfo**

创建投票活动

**代码清单11.10 createVoteInfo**

|  |
| --- |
| **public** String createVoteInfo() **throws** Exception {  String type = "默认类型";  **if**(infoType != **null**) {  **if**(infoType.equals("1")) {  type = "多选一";  } **else** **if**(infoType.equals("2")) {  type = "多选多";  } **else** **if**(infoType.equals("3")) {  type = "决议投票";  } **else** {  type = "未知类型";  }    }  info.setType(type);  VoteDao vd = **new** VoteDao();  info.setCreateTime(DateUtil.*getDateTime*());  **if** (vd.save(info)) {  System.***out***.println("创建投票\""+info.getTitle() + "\"成功!");  **return** ***SUCCESS***;  } **else** {  System.***out***.println("创建投票\""+info.getTitle() + "\"失败!");  **this**.addFieldError("error", "创建投票失败");  **return** ***INPUT***;  }  } |

**(2) getVoteInfos**

获取投票活动列表

**代码清单11.11 getVoteInfos**

|  |
| --- |
| **public** String getVoteInfos() **throws** Exception {  System.***out***.println("\*\*\*getVoteInfos\*\*\*");  VoteDao vd = **new** VoteDao();  List<VoteInfo> infos = vd.findVoteInfos();  Map request = (Map) ActionContext.*getContext*().get("request");  request.put("infos", infos);  **return** ***SUCCESS***;  } |

**(3) deleteVoteInfo**

**删除投票活动**

**代码清单11.12 deleteVoteInfo**

|  |
| --- |
| **public** String deleteVoteInfo() **throws** Exception {  VoteDao vd = **new** VoteDao();  **if**(vd.delete(info.getId(),"VoteInfo")) {  System.***out***.println("删除编号：" + info.getId() + "成功");  **return** ***SUCCESS***;  } **else** {  System.***out***.println("删除编号：" + info.getId() + "失败");    **return** ***ERROR***;  }  } |

**(4) updateVoteInfo**

更新投票信息

**代码清单11.13 updateVoteInfo**

|  |
| --- |
| **public** String updateVoteInfo() **throws** Exception {  VoteDao vd = **new** VoteDao();  info.setCreateTime(DateUtil.*getDateTime*());  **if**(!vd.update(info))  **return** ***ERROR***;  **return** ***SUCCESS***;  } |

**(5) addVotePeople**

添加选民

**代码清单11.14 addVoteProplePre**

|  |
| --- |
| **public** String addVotePeoplePre() **throws** Exception {  System.***out***.println("\*\*\*addVotePeoplePre()\*\*\*");  System.***out***.println("info:" + info);  **if**(info == **null** || info.getId() == **null**) {  **return** ***ERROR***;  }  VoteDao vd = **new** VoteDao();  List<VoteInfo> infos = vd.findVoteInfoById(info.getId());  Map request = (Map) ActionContext.*getContext*().get("request");  request.put("info", infos.get(0));  **return** ***SUCCESS***;  } |

**2.2.2 用户投票模块**

* **UserAction**

**(1) login**

选民登录，选民需上传注册阶段

**代码清单11.15 login**

|  |
| --- |
| **public** String login() **throws** Exception {  HttpServletRequest request = ServletActionContext.*getRequest*();  HttpSession session = request.getSession();  System.***out***.println("savePath:" + savePath);  System.***out***.println("uploadFileName:" + uploadFileName);  **if**(uploadFileName == **null**) {  session.setAttribute("login\_error", "请上传密钥文件！");  }  Map<String,String> parameters = **new** HashMap<String, String>();  // 读取文件内容  **if**(upload != **null**) {  InputStream is = **new** FileInputStream(upload);  BufferedReader br = **new** BufferedReader(**new** InputStreamReader(is));  String line = "";  **while** ((line = br.readLine()) != **null**) {  String[] pair = line.split(":");  **if**(pair.length == 2)  parameters.put(pair[0], pair[1]);  }  br.close();  } **else** {  System.***out***.println("File not found");  }  **if**(parameters == **null** || parameters.size() == 0) {  System.***out***.println("文件格式错误");  **return** ***INPUT***;  }  **if**(parameters.containsKey("uuid"))  uuid = parameters.get("uuid");  **if**(parameters.containsKey("infoId"))  infoId = Integer.*parseInt*(parameters.get("infoId"));  Integer keyId = **null**;  **if**(parameters.containsKey("keyId"))  keyId = Integer.*parseInt*(parameters.get("keyId"));  **if**(uuid == **null** || keyId == **null** || infoId == **null**) {  System.***out***.println("验证失败");  **return** ***ERROR***;  }  **if**(checkUuid(uuid)) {  System.***out***.println("该用户合法");  session.setAttribute("uuid", uuid);  session.setAttribute("keyId", keyId);  session.setAttribute("message", **null**);  VoteDao voteDao = **new** VoteDao();  List<VoteInfo> infos = voteDao.findVoteInfoById(infoId);  VoteInfo info = infos.get(0);  System.***out***.println();  **if**(info.getType().equals("多选一")) {  **return** "single";  } **else** **if**(info.getType().equals("多选多")) {  **return** "multiple";  } **else** **if**(info.getType().equals("决议投票")) {  **return** "dicision";  } **else** {  session.setAttribute("login\_error", "该密钥文件无效");  **return** "error";  }  }  **else** {  System.***out***.println("UUID不合法");  session.setAttribute("message", "该序列码不存在");  **return** ***INPUT***;  }  } |

**(2) getPersonVote**

获取个人的投票信息

**代码清单11.16 getPersonVote**

|  |
| --- |
| **public** String getPersonVote() **throws** Exception {  HttpServletRequest request = ServletActionContext.*getRequest*();  HttpSession session = request.getSession();  String uuid = (String) session.getAttribute("uuid");  ListDao ld = **new** ListDao();  List<List2> list2 = ld.find(uuid, "List2");  List<List3> list3 = ld.find(uuid, "List3");  **if**(list2 == **null** || list2.size() == 0)  session.setAttribute("sizeOfList2", 0);  **if**(list3 == **null** || list3.size() == 0)  session.setAttribute("sizeOfList3", 0);  session.setAttribute("list2", list2);  session.setAttribute("list3", list3);  **return** ***SUCCESS***;  } |

* **VoteAction**

**(1) getVoteItem**

获取投票候选项详细信息

**代码清单11.17 getVoteItem**

|  |
| --- |
| **public** String getVoteItem() **throws** Exception {  HttpServletRequest request = ServletActionContext.*getRequest*();  HttpSession session = request.getSession();  VoteDao vd = **new** VoteDao();  List<VoteItem> items = vd.findItem(itemid);  **if**(items == **null** || items.size() == 0)  **return** ***ERROR***;  VoteItem item = items.get(0);  session.setAttribute("item", item);  **return** ***SUCCESS***;  } |

**(2) getVoteItems**

获取投票或选项清单

**代码清单11.18 getVoteItems**

|  |
| --- |
| **public** String getVoteItems() **throws** Exception {  HttpServletRequest request = ServletActionContext.*getRequest*();  HttpSession session = request.getSession();  String uuid = (String) session.getAttribute("uuid");  ListDao ld = **new** ListDao();  List<List1> list = ld.find(uuid,"List1");  **if**(list != **null** && list.size() > 0) {  List1 l1 = list.get(0);  Integer infoId = l1.getInfoId();  VoteDao vd = **new** VoteDao();  List<VoteInfo> infos = vd.findVoteInfoById(infoId);  System.***out***.println("infoId = " + infoId);  **if**(infos != **null** && infos.size() > 0) {  VoteInfo voteInfo = infos.get(0);  session.setAttribute("items", voteInfo.getItems());  session.setAttribute("info", voteInfo);  **return** ***SUCCESS***;  }  **return** ***ERROR***;  }  **return** ***ERROR***;  } |

(3) singleVote

多选一投票

**代码清单11.19 singleVote**

|  |
| --- |
| **public** String singleVote() **throws** Exception {  HttpServletRequest request = ServletActionContext.*getRequest*();  HttpSession session = request.getSession();  String uuid = (String) session.getAttribute("uuid");  **if**(uuid == **null**) {  session.setAttribute("message", "您还没有登录！请登录后再参与投票");  **return** ***INPUT***;  }    Integer infoId = (Integer) session.getAttribute("infoId");  **if**(infoId == **null** ) {  ListDao listDao = **new** ListDao();  List<List1> list = listDao.find(uuid, "List1");  **if**(list!=**null**) {  infoId = list.get(0).getInfoId();  }  }  /\*\*  \* 验证用户是否已经投票  \*/  ListDao ld = **new** ListDao();  List<List2> list = ld.find(uuid, "List2",infoId);  **if**(list != **null** && list.size() > 0) {  session.setAttribute("message", "您已经参与投票，请勿重复投票！");  **return** ***INPUT***;  }    /\*\*  \* 1.对选票进行关联环签名  \* 2.验证选票的合法性  \*/  String message = "BLIND\_SIGNATURE\_SIGN:" + itemid;  Map<String, String> map = TCPUtil.*sendMessage*(message);  String blind\_vote = map.get("blind\_vote").replace(' ', '\0');  String vote\_signature = map.get("vote\_signature").replace(' ', '\0');    /\*\*  \* 初始化List2  \*/  List2 l2 = **new** List2();  l2.setUuid(uuid);  l2.setBlind\_vote(blind\_vote);  l2.setSigned\_vote(vote\_signature);  l2.setCreateTime(DateUtil.*getDateTime*());  l2.setInfoId(infoId);  /\*\*  \* 保存List2  \*/  **try** {  Session session2 = HibernateUtil.*getSession*();  session2.beginTransaction();  session2.save(l2);  session2.getTransaction().commit();  } **catch** (Exception e1) {  System.***out***.println(e1.getMessage());  session.setAttribute("result", "error");  session.setAttribute("message", "投票失败！");  **return** ***ERROR***;  }  /\*\*  \* 查找选票信息  \*/  session.setAttribute("result", "success");  session.setAttribute("message","投票成功！投票结果将会在一周后公布！");  **return** ***SUCCESS***;  } |

**(2) multipleVote**

多选投票

**代码清单11.20 multipleVote**

|  |
| --- |
| **public** String multipleVote() **throws** Exception {  HttpServletRequest request = ServletActionContext.*getRequest*();  HttpSession session = request.getSession();  Integer infoId = **null**;  **if**(info != **null** && info.getId() != **null**) {  System.***out***.println("infoId:" + info.getId());  infoId = info.getId();  } **else** {  infoId = (Integer) session.getAttribute("infoId");  }    String uuid = (String) session.getAttribute("uuid");  **if**(uuid == **null**) {  alert("您还没有登录！请登录后再参与投票。");  **return** ***INPUT***;  }    /\*\*  \* 验证用户是否已经投票  \*/  ListDao ld = **new** ListDao();  List<List2> list = ld.find(uuid, "List2",infoId);    **if**(list != **null** && list.size() > 0) {  alert("您已经参与投票！请勿重复投票");  **return** ***INPUT***;  }    String[] items = request.getParameterValues("itemId");  **if**(items == **null** || items.length == 0){  alert("请选择至少一个候选项！");  **return** ***INPUT***;  }    **for** (String itemId : items) {  /\*\*  \* 1.对选票进行关联环签名  \* 2.验证选票的合法性  \*/  String message = "BLIND\_SIGNATURE\_SIGN:" + itemId;  Map<String, String> map = TCPUtil.*sendMessage*(message);  String blind\_vote = map.get("blind\_vote").replace(' ', '\0');  String vote\_signature = map.get("vote\_signature").replace(' ', '\0');  /\*\*  \* 初始化List2  \*/  List2 l2 = **new** List2();  l2.setUuid(uuid);  l2.setBlind\_vote(blind\_vote);  l2.setSigned\_vote(vote\_signature);  l2.setCreateTime(DateUtil.*getDateTime*());  l2.setInfoId(infoId);  /\*\*  \* 保存List2  \*/  **try** {  Session session2 = HibernateUtil.*getSession*();  session2.beginTransaction();  session2.save(l2);  session2.getTransaction().commit();  } **catch** (Exception e1) {  System.***out***.println("保存选票失败："+e1.getMessage());  session.setAttribute("result", "error");  session.setAttribute("message", "投票失败！");  **return** ***ERROR***;  }  /\*\*  \* 查找选票信息  \*/  session.setAttribute("result", "success");    }  **return** ***SUCCESS***;  } |

# 附录12 单元测试

**表12.1 单元测试表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **编号** | **所属模块** | **测试步骤** | **预期结果** | **是否通过测试** | **测试人员** |
| 1 | 投票管理中心  注册/登录模块 | 进入后台注册页面 | 输入账号密码，注册失败给出提示信息；  注册成功后，进入后台登录页面 | 是 | 许勤昆 |
| 进入后台登录页面 | 输入正确账号密码后进入后台管理首页，否则提示：“用户名或密码错误” | 是 | 许勤昆 |
| 2 | 创建投票 | 进入创建投票活动页面 | 输入投票活动信息后，创建投票活动，在投票活动列表中显示投票活动列表 | 是 | 许勤昆 |
| 3 | 添加候选人 | 进入添加候选人页面 | 填写候选人信息，并上传候选人照片，在投票信息详情页面可以查看候选人信息 | 是 | 许勤昆 |
| 4 | 邀请选民 | 进入选民邀请页面 | 正确填写邮箱信息及上传选民信息表，邮箱发送成功后提示“发送成功”，否则提示“发送失败” | 是 | 许勤昆 |
| 5 | 选民注册 | 点击邮箱链接，进入选民注册页面 | 注册成功后，向用户提供证书，用户下载证书 | 是 | 许勤昆 |
| 6 | 选民登录 | 进入登录页面  用户上传证书 | 验证成功后进入投票中心页面 | 是 | 许勤昆 |
| 7 | 选民投票 | 选择候选人进行投票 | 投票成功后，提示用户投票成功；用户不可重复投票 | 是 | 许勤昆 |
| 8 | 数字签名 | 关联环签名模块 | 注册阶段，为用户生成关联环签名 | 是 | 许勤昆 |
| 盲签名模块 | 为选票生成盲签名 | 是 | 许勤昆 |
| 消盲阶段 | 将盲化选票消盲，还原选票信息 | 是 | 许勤昆 |