[CA 认证中心开发技术文档]

（密码学原理与实践-实践部分）

[项目名称：CA 认证中心]

[姓名： 王斌]

[学号： 1190202319]

目录

1.背景与意义

1.1 项目开发意义

CA 认证系统，即 CA 证书颁发系统，是公钥基础设施（PKI）中的核心环节，是公钥加密过程中的第三方权威认证方，负责密钥和证书的产生、发布、管理、存储和撤销等功能，广泛用于电子商务等需要非对称加密的信息传输场景中。CA 为信息传输的双方提供公私钥加密环境，提供身份认证、安全传输、不可否认性和数据完整性等功能。

1.2 国内外现状及技术综述

美国是最早提出 PKI 概念的国家，并于 1996 年成立了美国联邦 PKI 筹委会。与 PKI 相关的绝大部分标准都有美国制定，其 PKI 技术在世界上处于领先地位。2000 年 6 月 30 日，美国总统克林顿正式签署美国《全球及全国商业电子签名法》，给与电子签名、数字证书以法律上的保护，这一决定使电子认证问题迅速成为各国政府关注的热点。加拿大在 1993 年就已经开始了政府 PKI 体系雏形的研究工作，到 2000 年已经在 PKI 体系方面获得重要进展。加拿大与美国代表了发达国家 PKI 发展的主流。

欧洲在 PKI 基础建设方面也成绩显著。已颁布了 93/1999EC 法规，强调技术中立、隐私权保护、国内与国外相互认证以及无歧视等原则。为了解决各国 PKI 之间的协同工作问题，他采取了一系列策略：如积极自主相关研究所、大学和企业研究 PKI 相关技术；自主 PKI 互操作性相关技术研究，并建立 CA 网络及其顶级 CA。并于 2000 年 10 成立了欧洲桥 CA 指导委员会，于 2001 年 3 月 23 日成立欧洲桥 CA。

我国的 PKI 技术从 1998 年开始起步，由于政府和各有关部门近年来对 PKI 产业的发展给予了高度重视，2001 年 PKI 技术被列为”十五“863 计划信息安全主题重大项目，并于同年 10 月成立了国家 863 计划信息安全基础设施研究中心。国家计委也在制定新的计划来支持 PKI 产业的发展，在国家电子政务工程中明确提出了要构建 PKI 体系。目前，我国已全面推动 PKI 技术研究于应用。

1998 年国内第一家以实体形式运营的上海 CA 中心（SHECA）成立。目前，国内的 CA 机构分为区域型、行业型、商业性和企业型四类；截至 2002 年底，前三种 CA 机构已有 60 余家，58%的省市建立了区域 CA，部分部委建立了行业 CA。其中全国型的行业 CA 中心有中国金融认证中心 CFCA、CTCA 中国电信认证中心等。区域型 CA 有一定地区性，也称地区 CA，如上海 CA 中心、广东电子商务认证中心。

目前为止，X.509 是一种非常通用的证书格式。所有的证书都符合 ITU-T X.509 国际标准，因此(理论上)为一种应用创建的证书可以用于任何其他符合 X.509 标准的应用。X.509 证书的结构是用 ASN1(Abstract Syntax Notation One) 进行描述数据结构，并使用 ASN.1 语法进行编码。

在一份证书中，必须证明公钥及其所有者的姓名是一致的。对 X.509 证书来说，认证者总是 CA 或由 CA 指定的人，一份 X.509 证书是一些标准字段的集合，这些字段包含有关用户或设备及其相应公钥的信息。X.509 标准定义了证书中应该包含哪些信息，并描述了这些信息是如何编码的(即数据格式)

一般来说，一个数字证书内容可能包括基本数据（版本、序列号) 、所签名对象信息（ 签名算法类型、签发者信息、有效期、被签发人、签发的公开密钥)、CA 的数字签名，等等。

目前使用最广泛的标准为 ITU 和 ISO 联合制定的 X.509 的 v3 版本规范(RFC5280），其中定义了如下证书信息域：

版本号(Version Number)：规范的版本号，目前为版本 3，值为 0x2；

序列号（Serial Number)：由 CA 维护的为它所发的每个证书分配的一的列号，用来追踪和撤销证书。只要拥有签发者信息和序列号，就可以唯一标识一个证书，最大不能过 20 个字节；

签名算法（Signature Algorithm)：数字签名所采用的算法，如：

sha256-with-RSA-Encryption、ccdsa-with-SHA2S6；

颁发者（Issuer)：发证书单位的标识信息，如“C=CN，ST=Beijing, L=Beijing, O=org.example.com，CN=ca.org。example.com”；

有效期(Validity): 证书的有效期很，包括起止时间。

主体(Subject) : 证书拥有者的标识信息（Distinguished Name)，如：“C=CN，ST=Beijing, L=Beijing, CN=person.org.example.com”；

主体的公钥信息(Subject Public Key Info)：所保护的公钥相关的信息：

公钥算法 (Public Key Algorithm）公钥采用的算法；

主体公钥（Subject Unique Identifier)：公钥的内容。

颁发者唯一号（Issuer Unique Identifier)：代表颁发者的唯一信息，仅 2、3 版本支持，可选；

主体唯一号（Subject Unique Identifier)：代表拥有证书实体的唯一信息，仅 2，3 版本支持，可选：

扩展（Extensions，可选）：可选的一些扩展。中可能包括：

Subject Key Identifier：实体的秘钥标识符，区分实体的多对秘钥；

Basic Constraints：指明是否属于 CA;

Authority Key Identifier：证书颁发者的公钥标识符；

CRL Distribution Points: 撤销文件的颁发地址；

Key Usage：证书的用途或功能信息。

此外，证书的颁发者还需要对证书内容利用自己的私钥添加签名， 以防止别人对证书的内容进行篡改

X.509 规范中一般推荐使用 PEM(Privacy Enhanced Mail)格式来存储证书相关的文件。证书文件的文件名后缀一般为 .crt 或 .cer 。对应私钥文件的文件名后缀一般为 .key。证书请求文件的文件名后缀为 .csr 。有时候也统一用 pem 作为文件名后缀。

PEM 格式采用文本方式进行存储。一般包括首尾标记和内容块，内容块采用 Base64 进行编码。

2.需求分析

2.1 总体需求

完成一个认证系统的相关所需功能，可供商家、用户、银行和普通用户申请证书，每个申请用户均需在系统注册账号并通过实名认证后申请证书。在网站申请页面填写正确信息后，由管理员审核通过后，签发格式为.cer 的 x509 标准格式证书。用户可以输入自定义密码，来获取使用其加密后的私钥（私钥不直接传输存储，防止泄露），证书下载及私钥分发在系统界面中提供操作按钮，申请用户直接下载并使用。同时用户如果发现密钥泄露，可以通过撤销来冻结自己的证书，管理员拥有反撤销的能力恢复证书。此外，后端提供使用证书加解密，签名和验证签名的相关函数，商家或银行可根据相关方法并结合自己需求后正确使用证书。在前端编写更加通用的证书代理，实现代码的松耦合，兼容各种代码，在代理中实现使用证书的加解密，验证证书，私钥签名等操作，使用缓存机制，可以在证书中心离线的情况下正常运行。同时，如果检测到之前使用过的证书被再次使用，会直接加载缓存，避免多次下载，提高运行效率。用户拥有对自己信息查看并修改权力（查看个人信息，包括修改头像、电话、邮件等，查看实名信息，实名信息仅限实名认证成功前修改），管理员拥有管理用户的权力（比如注销、恢复用户，更新用户信息，提升用户权限，重置用户密码等）。

数据库中在涉及到用户信息（比如实名信息，电话、邮箱等敏感信息）和证书私钥提取（证书别名，证书提取私钥的密码）的地方要进行加密，防止信息泄露。

此外，系统会记录用户的操作，生成系统日志，便于管理员复核和审计。

2.2 功能需求

本次实验中，CA 的功能需求主要有如下几点：

1. 接收验证用户数字证书的申请
2. 生成证书、存储证书
3. 向申请者颁发（或拒绝颁发）数字证书
4. 接收用户的数字证书的查询、撤销、私钥获取、下载证书请求
5. 数字证书的归档
6. 密钥归档
7. 提供撤销列表，用于展示撤销的证书
8. 提供证书使用工具（这里实现的是证书代理和 java 相关工具类）
9. 系统日志
10. 数据库加密
11. 用户信息及密码修改
12. 用户实名认证
13. 用于防止恶意提交表单的验证码生成验证机制

14. 后台管理用户（注销、恢复用户，重置用户密码，更新用户信息，管理实名信息）
15. 后台管理证书（撤销和反撤销证书）
16. 后台管理申请（通过和驳回）

2.3 性能需求

指导书中未明确指出性能需求，不过根据使用的技术，预计如下：

由于使用了 Redis 基于内存的快速的数据库用于存储验证码，用户 token 等常用信息，同时在后端层之间精简了对于用户隐藏或者不必要的信息，加快层之间信息传输速度，同时 MyBatis-Plus 会自动帮我们优化 sql 语句，使得语句的效率更高，因此如果平台的内存带宽资源和网络环境理想的情况下，我们预计：

1. 该系统至少能够同时满足 1000 人的访问；
2. 用户登录时间小于 3s；
3. 系统可以 24 小时不间断工作；
4. 可以简单与电商和银行平台对接；
5. 前端页面使用 Vue 开发，平台兼容性好，能够在各种浏览器上运行，一次加载，后续操作很快。

3.概要设计

3.1 技术路线

1. 前端使用 Vue 从零开发需要的所有界面
2. 后端使用 SpringBoot+MyBatis-Plus 开发
3. 证书代理（用于兼容各种开发语言，提高对接效率，提高证书操作速度）使用 SpringBoot 开发，监听系统 9266 端口
4. 数据库使用 MySQL 存储证书系统用户信息、管理员信息、证书基本信息（不含证书密钥）等，使用 Redis 存储登陆注册验证码、用户登陆分配 Token 等信息
5. 密钥归档使用 keystore 文件，该文件自带密码，只有提供文件密码和证书别名以及证书密码的情况下才能从该文件获取证书的私钥，安全性较高

3.2系统功能概要

3.2.1 用户功能

1. 登陆注册 CA 系统
2. 管理自身信息
3. 修改密码
4. 申请证书（只有实名认证后才可以提交申请）
5. 申请管理（修改申请信息，重新发起申请）
6. 证书管理（召回证书，下载证书，获取证书私钥）
7. 实名认证（只有实名认证后才能成为个人版，拥有申请证书的能力）

3.2.2 管理员功能

1. 用户管理（注销，还原，更新用户信息，重置密码）
2. 申请管理（通过申请生成证书，驳回申请）
3. 证书管理（召回证书，撤销召回）
4. 实名管理

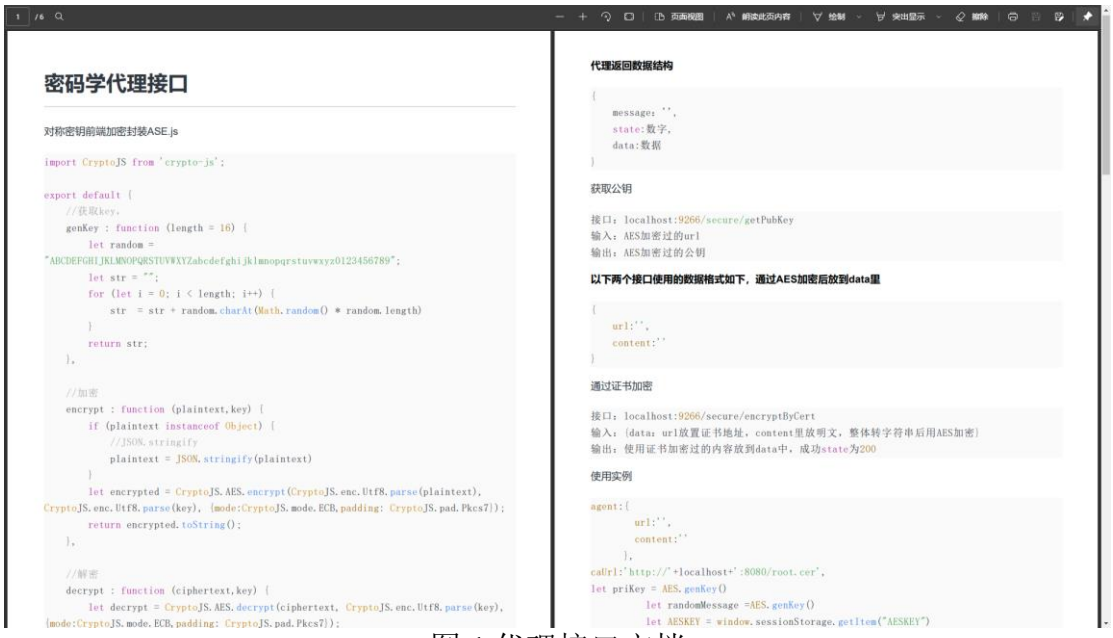
3.2.3 证书代理功能

1. 基础功能：自动验证证书真伪，并对比是否在撤回列表，如果提供加密私钥的密码，自动验证私钥真伪，以下操作都会自动执行上面的基础功能
2. 加密通信功能（代理与需要使用服务的程序之间使用 AES 加密，加密的密钥在程序运行时生成）
3. 提供使用证书（公布 url，会自动缓存避免每次下载加快速度）加解密接口
4. 提供使用自身证书公私钥（证书和加密后的私钥放在代理服务器的固定位置，如果需要私钥，请提供私钥访问密码）加解密接口
5. 提供使用自身证书公私钥（证书和加密后的私钥放在代理服务器的固定位置，如果需要私钥，请提供私钥访问密码）签名消息

4.详细设计

4.1 证书代理器

为了方便适配不同的开发语言和框架，避免每次加密都需要下载证书，以及在 CA 服务器离线状态可用（利用上次缓存的撤销列表以及根证书、个人版根证书和商业版根证书），开发证书代理器，提供使用证书加解密，使用自身证书加解密和签名验签服务，并且在上述操作过程中都会验证证书有效性（签名有效、是否过期和是否在撤销列表中），利用私人证书私钥加密还会自动验证密码是否正确。对接的开发者只需要阅读证书代理器的接口文档，实现 http 请求即可完成对证书的一系列操作。



密码学代理接口

对称密钥前端加密封装ASE.js

```
import CryptoJS from 'crypto-js';

export default {
    //获取key,
    genKey : function (length = 16) {
        let random =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
        let str = "";
        for (let i = 0; i < length; i++) {
            str  = str + random.charAt(Math.random() * random.length)
        }
        return str;
    },

    //加密
    encrypt : function (plaintext,key) {
        if (plaintext instanceof Object) {
            //JSON.stringify
            plaintext = JSON.stringify(plaintext)
        }
        let encrypted = CryptoJS.AES.encrypt(CryptoJS.enc.Utf8.parse(plaintext),
CryptoJS.enc.Utf8.parse(key), {mode:CryptoJS.mode.ECB,padding: CryptoJS.pad.Pkcs7});
        return encrypted.toString();
    },

    //解密
    decrypt : function (ciphertext,key) {
        let decrypt = CryptoJS.AES.decrypt(ciphertext, CryptoJS.enc.Utf8.parse(key),
{mode:CryptoJS.mode.ECB,padding: CryptoJS.pad.Pkcs7});
```

代理返回数据结构

```
{
    message: '',
    state:数字,
    data:数据
}
```

获取公钥

接口：localhost:9266/secure/getPubKey
输入：AES加密过的url
输出：AES加密过的公钥

以下两个接口使用的数据格式如下，通过AES加密后放到data里

```
{
    url:'',
    content:''
}
```

通过证书加密

接口：localhost:9266/secure/encryptByCert
输入：{data：url放置证书地址，content里放明文，整体转字符串后用AES加密}
输出：使用证书加密过的内容放到data中，成功state为200

使用实例

```
agent:{
        url:'',
        content:''
    },
caUrl:'http://'+localhost+':8080/root.cer',
let priKey = AES.genKey()
        let randomMessage =AES.genKey()
        let AESKEY = window.sessionStorage.getItem("AESKEY")
```

图 1.代理接口文档

4.2 加解密设计

在登陆时，由前端生成一个随机 AES 密钥（128 位），并使用该密钥加密一个随机字符串，与登陆信息一起使用证书服务器的根证书公钥加密（使用证书代理器）发送给证书服务器，证书服务器使用私钥解密后提取 AES 密钥，解密随机串作为返回结果，之后使用 AES 加密返回给前端，前端得到后使用 AES 解密后验证其与之前的随机串是否相同，如果相同则登录成功，并且建立安全的通信信道，之后所有的通信内容都是用该 AES 密钥加密。该方法同样可以**防止中间人攻击**，中间人无法获取到通信的 AES 密钥，也就无法解密其截获的其他内容。由于添加验证码，可以有效防止穷举方法破解用户密码。

```
request:{
  username:'',   用户名
  password:'',   密码
  verifyCode:'', 验证码
  verifyKey:'',  验证码id
  privateKey:'', AES私钥
  randomStr:''   随机串
},
```

图 2.前端请求格式

在注册时，前端使用证书服务器的根证书公钥加密注册信息，同时引入验证码，防止恶意注册，在注册成功后会被引导到登录页面。

普通操作，由于引入 Token 和私钥自动过期（30 分钟）的机制，因此用户的请求和服务器发出的数据都是抗 30 分钟内的重放攻击，所以普通的操作没有做额外的**抗重放攻击**，使用 AES 私钥加密通信内容，在客户端和服务器之间往返。

敏感操作，例如：管理员的同意、驳回、召回、撤销召回证书，注销、撤销注销、重置用户密码操作、普通用户的撤销证书，获取证书私钥等操作，使用更严苛的**抗重放机制**，将时间设定在 1 分钟，保证操作的安全性。

上面的所有请求都经过了签名算法签名，保证无法被篡改。

4.3 签名设计

非对称签名算法使用 java 自带的库函数实现，具体代码如下

```java
/**
 * 签名
 * @param data 待签名数据
 * @param privateKey 私钥
 * @return 签名
 */
public static String sign(String data, String privateKey) throws Exception {

    byte[] keyBytes = decryptBASE64(privateKey);
    PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(keyBytes);
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    PrivateKey key = keyFactory.generatePrivate(keySpec);
    Signature signature = Signature.getInstance("MD5withRSA");
    signature.initSign(key);
    signature.update(data.getBytes());
    return new String(Base64.encodeBase64(signature.sign()));
}
```

图 3.非对称签名函数

验签算法如下：

```java
/**
 * 验签
 * @param srcData 原始字符串
 * @param publicKey 公钥
 * @param sign 签名
 * @return 是否验签通过
 */
public static boolean verify(String srcData, PublicKey publicKey, String sign) throws Exception {
    byte[] keyBytes = publicKey.getEncoded();
    X509EncodedKeySpec keySpec = new X509EncodedKeySpec(keyBytes);
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    PublicKey key = keyFactory.generatePublic(keySpec);
    Signature signature = Signature.getInstance("MD5withRSA");
    signature.initVerify(key);
    signature.update(srcData.getBytes());
    return signature.verify(Base64.decodeBase64(sign.getBytes()));
}
```

图 4.非对称验证签名函数

由于在最初使用 ca 证书中心的时候，用户没有使用证书，所以无法使用上面的签名方法，只能使用对称密钥签名的哈希签名范式，前端签名的方法如下

```
sign:AES.encrypt(this.$md5(data),priKey)
```

图 5.前端对称签名方法

后端得到后做将 data 做 md5 变换，并将 sign 解密比对即可验证签名。

```java
protected boolean vrfySign(String msg,String sign,String priKey) throws Exception {
    String s = AES.decrypt(sign,priKey);
    String s1 = DigestUtils.md5DigestAsHex(msg.getBytes());
    return s1.equals(s);
}
```

图 6.验证签名函数

4.3 MySQL 数据库设计-存储安全

主要有 4 个表格，分别是用户信息表、管理员信息表、证书信息表、实名信息表

4.3.1 用户信息表

用户的密码并不在数据库中直接存储，而是在收到用户注册密码后，生成强随机的盐值，将密码与盐值混合后做 md5 变换，得到的结果存到 password，盐值存到 salt，可以有效的防止数据库泄露时用户密码暴露，导致黑客撞库攻击用户的其他账号密码。用户的手机号和邮箱都是采用 AES 加密后存储到数据库中，防止用户个人隐私泄露。

```
CREATE TABLE `t_user`  (
  `uid` int NOT NULL AUTO_INCREMENT COMMENT '用户id',
  `username` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '用户名',
  `password` char(32) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '密码',
  `salt` char(36) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '盐值',
  `phone` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '电话号码',
  `email` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '电子邮箱',
  `gender` int NULL DEFAULT NULL COMMENT '性别:0-女，1-男',
  `avatar` varchar(250) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '头像',
  `is_delete` int NULL DEFAULT NULL COMMENT '是否删除：0-未删除，1-已删除',
  `created_user` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '日志-创建人',
  `created_time` datetime NULL DEFAULT NULL COMMENT '日志-创建时间',
  `modified_user` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '日志-最后修改执行人',
  `modified_time` datetime NULL DEFAULT NULL COMMENT '日志-最后修改时间',
  `utype` int NULL DEFAULT NULL COMMENT '用户类型：1为个人版，2为商业版（授权商家和银行）',
  PRIMARY KEY (`uid`) USING BTREE,
  UNIQUE INDEX `username`(`username`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 40 CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = DYNAMIC;

SET FOREIGN_KEY_CHECKS = 1;
```

图 7.用户信息表创建

uid	username	password	salt	phone	email	gender	avatar	is_delete	created_user	cre
1	tim	5047FDD6E19E4914D7D5	C20C82FB-83DD-4EB6-BBE3-D15738008AE9	VZBj/zbwMpeP0/SntMQU/g==	SmfJV8kOslE9B/nD+8lnBw==	1		0	tim	
2	test1	7A0C3D96994B5683039B	2E9DAEFA-3362-4305-954C-E5C159F1D80A	kuyPJHUxkUZ9XsFQ+ISSyA==	sW5vkNsVZEmsZYEZFcgu6Q==	0		0	test1	202
3	test2	4C677D1112D295555B41	(Null)	(Null)	(Null)	0		0	test2	202
4	test3	19FE4D6652FC8D1FCEE9	070579EC-1F03-443A-819C-10388A00EF10	(Null)	(Null)	0		0	test3	202
5	tim1	123	(Null)	(Null)	(Null)	1	(Null)	0	tim1	
6	test4	C8FCF1DB99D30BD7B4DE	43113495-6B60-4A4D-8A05-876684243 9BB	(Null)	(Null)	1	(Null)	0	test4	202
7	test5	784BE1C6D882AD44AA0C	6A3C058E-6EB8-489D-88E6-3A895FC1155B	(Null)	(Null)	0	(Null)	0	test5	202
8	test6	55C3C1C6719CCFF946DC	92BF83C4-932C-4F2F-A7C9-93FFBFCB7419	(Null)	(Null)	0	(Null)	0	test6	202
9	test7	41183365A8FD1B657109	3A9BCA3F-2BBA-4CB4-A1FF-75A18D16776B	(Null)	(Null)	(Null)	(Null)	0	test7	202
10	test8	40E89C16905D6A657C78	322708F2-15A3-458E-9413-00EE43D5B6F6	(Null)	(Null)	(Null)	(Null)	0	test8	202
12	test9	8D5B5320856D15F24333	639986EA-61F9-46B9-B562-75B7FE6EB1AF	(Null)	(Null)	(Null)	(Null)	0	test9	202
13	bin	F78763AAFCED6944DFFD	05BAFF25-B550-40D4-A8D9-5ED605AEB403	ioe4jDVgKnC3asiZIl8l6A==	PtfGNKRglAvSkBsEUlyHOG7ttK1MSv92XlEN0ILI3Pc=	(Null)	(Null)	0	bin	202
14	bin1	A8ACB4D6C448E08CB3E6	4AE5B4F8-9834-43A8-97E8-B5ACFFCA6231	ioe4jDVgKnC3asiZIl8l6A==	PtfGNKRglAvSkBsEUlyHOG7ttK1MSv92XlEN0ILI3Pc=	(Null)	(Null)	0	bin1	202
15	bin2	4A76F95571AF76A74561	BAFA2F68-C842-4108-A6BD-1CA603407C64	ioe4jDVgKnC3asiZIl8l6A==	PtfGNKRglAvSkBsEUlyHOG7ttK1MSv92XlEN0ILI3Pc=	1	(Null)	0	bin2	202
18	1	17A6CC29F706A05429AE	9F51B4F3-835B-43A7-8BCB-2D6CCF77074E	l90bL1b/Fv0PYHE5zrJVag==	l90bL1b/Fv0PYHE5zrJVag==	1	(Null)	0	1	202
19	12343	935946480F9FE0B376A80	7CD715C1-E7DA-438B-BCDC-B0177F44529F	l90bL1b/Fv0PYHE5zrJVag==	l90bL1b/Fv0PYHE5zrJVag==	1	(Null)	0	12343	202
20	111111	16970D61ACA2BA3A523F	B0765D6D-2EE0-4AD8-A1B9-3DB7687B65FE	ioe4jDVgKnC3asiZIl8l6A==	PtfGNKRglAvSkBsEUlyHOG7ttK1MSv92XlEN0ILI3Pc=	1	(Null)	0	111111	202
22	1111111	DA311C09111D9AA3E1DE	5B9BFC4C-CE8B-4450-BF9A-B1B896FA9FB4	ioe4jDVgKnC3asiZIl8l6A==	PtfGNKRglAvSkBsEUlyHOG7ttK1MSv92XlEN0ILI3Pc=	1	(Null)	0	1111111	202
23	xufannan	EDD59091977168DD60C6	04986190-B32B-4F82-A83E-85691E3C67BC	ioe4jDVgKnC3asiZIl8l6A==	PtfGNKRglAvSkBsEUlyHOG7ttK1MSv92XlEN0ILI3Pc=	1	(Null)	0	xufannan	202
24	qwer1234	F5C1F031103C8E5A0FC37	EAFA46FD-5795-404F-9077-CC7728E0F816	ahfzg8OymRuKlt6afp3JSQ==	ahfzg8OymRuKlt6afp3JSQ==	1	(Null)	0	qwer1234	202
25	qwer1231	13920477F6A515DD8413	8856AE45-7824-481B-A3B7-F94D3BB6EA53	ahfzg8OymRuKlt6afp3JSQ==	ahfzg8OymRuKlt6afp3JSQ==	0	(Null)	0	qwer1231	202
26	111222	BDD07A3C233A6AF08DEF	D552A2CB-D6D5-4F77-A90C-ECCF884F91CB	l90bL1b/Fv0PYHE5zrJVag==	l90bL1b/Fv0PYHE5zrJVag==	1	(Null)	0	111222	202
27	qqqqqq	858E1964AB29E8C16AC5	1217790F-2BED-42F0-BC41-9E1CEDF7942F	7CQ2ZW0+ZbwU3kYoTcb9fA==	7CQ2ZW0+ZbwU3kYoTcb9fA==	0	(Null)	0	qqqqqq	202
28	QWERT	76375B9C276CC7756055	185A78C1-1D9B-4DD0-859A-206489105298	ahfzg8OymRuKlt6afp3JSQ==	ahfzg8OymRuKlt6afp3JSQ==	0	(Null)	0	QWERT	202
29	1234567	BB39707C72F44E9224105	012B5E4C-3300-4E02-9D2F-55C808FD8B6F	JhnoSbh8iGyax7afpHlC/w==	JhnoSbh8iGyax7afpHlC/w==	0	(Null)	0	1234567	202
30	12345678	4700DDF37FA082B0C811	5668F18B-5606-4E04-8F26-BE49902852D6	ahfzg8OymRuKlt6afp3JSQ==	ahfzg8OymRuKlt6afp3JSQ==	1	(Null)	0	12345678	202
31	user1234	6C2FD160CCD1B91FAD0A	B066F48A-78C3-4170-A722-DFCA156FF588	kLAKEPUUhpZhGNGjCbsybQ==	WiH9BDuP6W4PL1P6qbpzAwLQ8F1SpiH9yypfOzE+PZE=	0	http://local	0	user1234	202
32	tim2333	123	(Null)	(Null)	(Null)	(Null)	(Null)	(Null)	(Null)	
34	qqqqqqq	C03A2FB72C813A4C5161	E9AE8BD6-BC72-4231-84C1-29D49928B994	d86tDz/N3fDBKxOM64iF4A==	qHoZl2jJjdyyhl/qpWJ+Vw==	1		0	qqqqqqq	202
35	qqqqqqqq	89DA79736BE584B806B1	4E577A2B-9295-47E9-A596-622BA6BCB6DC	d86tDz/N3fDBKxOM64iF4A==	qHoZl2jJjdyyhl/qpWJ+Vw==	1		0	qqqqqqqq	202
36	aaaaaaaa	1CE7AEECFA622CF12C7E	E7515B87-7825-416B-94FF-8FFCC7F5468F	ZgXatgaGua5YmTqEFp7vjQ==	PYPV1M/KdKv3wlhwBK0iiA==	1	http://local	0	aaaaaaaa	202
37	xufannannan	26D89DB134EC18502CB5	2C092747-A205-453C-8DB9-1672F31F0ACC	jiBTXo9ZNyLfjVBRWKt7OA==	m6s7S5OG6JNBy6+xFxaSWQ1dOD2NXriN7Fn6XIL6PP4=	1		0	xufannannan	202
38	ZhanXianyou	37CAA4EC5AF990F3CCA1	4D9C3BF4-B45C-4DA2-BAF2-8AB6B11F5710	Q69ml1cLM9u28BcR+j8XRw==	rLY1vcZxK+q6iw2ll+OHEW4IyAhVeskXqAGQIn25tb0=	1		0	ZhanXianyou	202
39	11111	4BD4E6DC9220571ADE10	D9685DC7-6961-43B9-8380-81FD2C85AB83	ahfzg8OymRuKlt6afp3JSQ==	ahfzg8OymRuKlt6afp3JSQ==	1		0	11111	202
40	cacenter	8119811F96A9CA0602C4	8A358C3A-DAB2-4B46-A25D-5BEB179853B6	ioe4jDVgKnC3asiZIl8l6A==	PtfGNKRglAvSkBsEUlyHOG7ttK1MSv92XlEN0ILI3Pc=	0	http://local	0	cacenter	202

图 8.用户信息表展示（敏感信息已加密处理）

4.3.2 管理员信息表

密码存储方案同上，即使获得了表格，也很难推测出管理员的密码，保证了存储的安全性，同时手机号和邮箱同样采用了 AES 加密来保障安全

```
CREATE TABLE `t_su`  (
  `uid` int NOT NULL AUTO_INCREMENT COMMENT '用户id',
  `username` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '用户名',
  `password` char(32) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '密码',
  `salt` char(36) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '盐值',
  `phone` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '电话号码',
  `email` varchar(30) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '电子邮箱',
  `gender` int NULL DEFAULT NULL COMMENT '性别:0-女，1-男',
  `avatar` varchar(250) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '头像',
  `is_delete` int NULL DEFAULT NULL COMMENT '是否删除：0-未删除，1-已删除',
  `created_user` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '日志-创建人',
  `created_time` datetime NULL DEFAULT NULL COMMENT '日志-创建时间',
  `modified_user` varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '日志-最后修改执行人',
  `modified_time` datetime NULL DEFAULT NULL COMMENT '日志-最后修改时间',
  PRIMARY KEY (`uid`) USING BTREE,
  UNIQUE INDEX `username`(`username`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 14 CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = DYNAMIC;
```

图 9.管理员信息表创建

uid	username	password	salt	phone	email	gender	avatar	is_delete	created_user	created_time	modified_user	m
13	root	68C80B814EA6C03348DE	8C9DC4D5-1AD9-4251-816E-C3A27524A1C	ioe4jDVgKnC3asiZIl8l6A==	PtfGNKRglAvSkBsEUlyHO	1	http://local	0	root	2021-11-24 14:38:30	root	20

图 10.管理员信息表展示（敏感信息已加密处理）

4.3.3 证书信息表

证书的私钥是分开存储的，分为密钥文件、数据库中的表格，只有同时获得二者还有密钥文件的密码和加密数据库的 AES 密钥才可以还原出一个证书的密码，也保证了存储安全.

```
CREATE TABLE `t_ca`  (
  `cid` int NOT NULL AUTO_INCREMENT COMMENT '证书id',
  `uid` int NULL DEFAULT NULL COMMENT '证书持有者id',
  `password` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '证书在证书库中密码',
  `created_user` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '日志-创建人',
  `created_time` datetime NULL DEFAULT NULL COMMENT '日志-创建时间',
  `modified_user` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '日志-修改人',
  `modified_time` datetime NULL DEFAULT NULL COMMENT '日志-修改时间',
  `c_type` int NULL DEFAULT NULL COMMENT '用户类型：1为个人版，2为商业版（授权商家和银行）',
  `description` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '证书说明描述',
  `start_time` datetime NULL DEFAULT NULL COMMENT '证书生效时间',
  `valid_time` int NULL DEFAULT NULL COMMENT '证书有效期，单位：天',
  `c_state` int NULL DEFAULT NULL COMMENT '证书状态：0-已申请，等待审核，1-过审核，已生成证书 2-驳回 3-召回',
  `username` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '拥有者用户名',
  `alias` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '证书别名（用于从密钥库中取出私钥）',
  `ca_c` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '国家',
  `cn` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '拥有者名称',
  `l` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '市',
  `o` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '组织',
  `ou` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '组织单位',
  `st` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '省',
  `id` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '写在证书中的id',
  PRIMARY KEY (`cid`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 30 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
```

图 11.证书信息表创建

cid	uid	password	created_user	created_time	modified_user	modified_time	type	description	start_time	valid_time	state	username	alias	c	cn	l	o	ou
20	37	xlNh4mdzKvikA1	user1234	2021-11-27 13:08	user1234	2021-12-14 08:03:2	1	K7iku9gI4zhLUg	2021-11-30 00	9999		XUJXmOb/v65	SDpC6O(	中国	王斌	哈尔滨	哈工大	信息
21	36	ymlgi6O7CF4TbI	aaaaaaaa	2021-11-27 15:34	root	2021-11-27 20:16:5	1	DB9vz4xXp+hN	2021-11-30 00	666		yBpLS8UkSHHI	fc8EJ5H3	aaa	aaa	aaa	aaa	aa
22	37	YENJGJW+XRpT	user1234	2021-11-27 15:34	su_root	2021-12-14 08:16:3	2	GQlLuT+Szyqe4	2021-11-30 00	6665		XUJXmOb/v65	Ze+q9n4	ss	ss	ss	ss	ss
23	31	(Null)	user1234	2021-11-27 16:58	user1234	2021-12-09 10:42:3	2	MaD/D7RxblJoC	2021-11-30 00	123		XUJXmOb/v65	(Null)	z	z	z	z	z
24	36	(Null)	aaaaaaaa	2021-11-27 16:58	aaaaaaaa	2021-11-27 18:04:0	1	l1SzsFT7j7mdB0	2021-11-30 00	123		yBpLS8UkSHHI	(Null)	w	w	w	w	w
25	37	xlt/rrUIZ1+Km/R	user1234	2021-11-27 20:21	user1234	2021-12-14 00:21:0	2	3E6FaRvcINq0B	2021-11-27 00	9999		XUJXmOb/v65	hqK4fOB	中国	胡丰康	大连理	大连理	信息
26	37	(Null)	user1234	2021-11-27 22:20	user1234	2021-12-08 15:36:1	2	T4DTGQQhrXim	2021-11-23 00	100		XUJXmOb/v65	(Null)	祖国	binw	合并	哈工大	信息
27	37	(Null)	xufannannan	2021-11-27 22:49	xufannannan	2021-11-27 22:49:5	1	wFYeLU99p2x0R	2021-11-10 00	12		uTPP+5XI/cKKI	(Null)	asd	asd	asd	ads	asd
28	38	5W2UdXpq1Mm	ZhanXianyou	2021-11-28 14:17	su_root	2021-12-14 08:17:4	2	L8+9eeJxpDicbL	2021-11-28 00	9999		z46fuFc468mb	PTRi2ssn(	中国	bank	哈尔滨	哈尔滨	计算
29	39	VxUQXJz3/FpdE	11111	2021-11-28 14:51	su_root	2021-12-14 08:17:4	2	ahfzg8OymRuKl	2021-11-27 00	11		9o8mnt4+0M7	ZmdCg6r	1111	1111	111	111	111
30	37	(Null)	user1234	2021-12-10 22:46	user1234	2021-12-10 22:46:3	1	Qmf9POdoqdvc	2021-12-28 00	1234		XUJXmOb/v65	(Null)	x	x	x	x	x
31	4	+hxPx+ed8GuU(	test1	2021-12-11 10:07	su_root	2021-12-14 08:17:2	2	EKGIlWMmYaYV	2021-12-29 00	523		EESpSr5OPHm	81uWplC	中国	test1	哈尔滨	12305	1562
32	2	(Null)	test1	2021-12-11 10:29	test1	2021-12-11 10:29:3	1	svPC3QN3wr8Je	2021-12-29 00	123		EESpSr5OPHm	(Null)	z	z	z	z	z
33	40	1VMiqZEjvBvo1j	cacenter	2021-12-11 18:34	su_root	2021-12-14 10:44:1	1	+CErWLLO/dJlu	2021-12-12 00	365		GYPep5cymIcn	5a6WQ1(	中国	证书中(	哈尔滨	哈工大	信息
34	40	RO4ZCdo5sLMj	cacenter	2021-12-11 18:37	root	2021-12-11 19:03:3	2	U3jr1MiNoZZ/Vi	2021-12-12 00	365		GYPep5cymIcn	5C7wqZC	中国	证书中(	哈尔滨	哈工大	信息
35	1	qYD+xCNby0rg/	jim	2021-12-12 15:26	root	2021-12-12 15:27:0	1	JhnoSbh8iGyax7	2021-12-27 00	11		zSlkk7EkBn4nl	c444jryEH	123	233	11	11	11
36	37	aRfq9exCdRHtU	user1234	2021-12-14 00:24	root	2021-12-14 00:24:1	1	8bK61i2Lh0YCD	2021-12-28 00	123		XUJXmOb/v65	b+FEmfA	cc	cc	cc	cc	cc
37	37	0i86nMVTrmrfjM	user1234	2021-12-14 08:11	su_root	2021-12-14 08:17:5	1	svPC3QN3wr8Je	2021-12-28 00	123		XUJXmOb/v65	5KM7ja0(	vv	vv	vv	vv	vv

对证书密码，别名，描述信息，所有者用户名采用加密的方式，防止证书密钥和用户信息泄露

图 12.证书信息表展示（私钥相关或用户相关的信息已经加密处理）

4.3.4 实名信息表

存储用实名信息，由于无法接入国家相关网络验证，这里当用户提交时，只要通过前端规则验证，默认认证成功，修改用户状态从游客变成个人版。这里存储的个人信息都经过 AES 加密，防止数据泄露。

```
DROP TABLE IF EXISTS `t_real`;
CREATE TABLE `t_real`  (
  `rid` int NOT NULL AUTO_INCREMENT COMMENT '事务编号',
  `uid` int NOT NULL COMMENT '用户uid',
  `username` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL COMMENT '用户名',
  `name` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL COMMENT '真实姓名',
  `id` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL COMMENT '身份证号',
  `state` int NOT NULL COMMENT '事务状态 0-待提交 1-待审核 2- 审核通过',
  PRIMARY KEY (`rid`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;

SET FOREIGN_KEY_CHECKS = 1;
```

图 13.用户实名信息表

rid	uid	username	name	id	state
1	1	2	NqO1flYF9I1kLmUI4ZzeN	snmn1fIpYDHn0LLRJrxX5A	1
2	41	fffff	WtRiXZBNKq0x2fkIahfZiA	iaCYdu42pzfesKeD+TPFP/	2
3	42	klkkk	sq4ZiLHlAf13i8IOE6ph/A=	wWupILJ2wQHNQwu+Gx	2

图 14.用户实名信息表展示（真实姓名和身份证号已经加密处理）

4.4 用户私钥分发

由于所有的私钥都是在服务器这边生成保存，所以在用户私钥未泄露的情下可以给用户提供多次私钥分发服务，首先用户输入一串密码，后台获取到这个密码后将密码做 md5 变换，可以得到一段定长的值，将这个值作为 AES 密钥加密

用户私钥，将加密后的结果返回给用户保存，用户只有同时提供密码和加密后的私钥才能解密出真正的私钥，实现了安全的私钥保存，如果用户的私钥丢失，用户也可以点击撤回证书，来阻止证书继续生效，被人利用。

4.5 证书生成和存储

证书使用 x.509 标准格式，可以直接安装到操作系统中，证书存储在 CA 文件夹下，密钥库、根证书（签发私人版根证书和商业版根证书）、私人版根证书、商业版根证书存放在文件夹的根目录，私人版证书存放在其下的 personal 目录中，商业版证书存放在其下的 commercial 目录中，文件的命名方式为证书别名加上.cer 后缀。

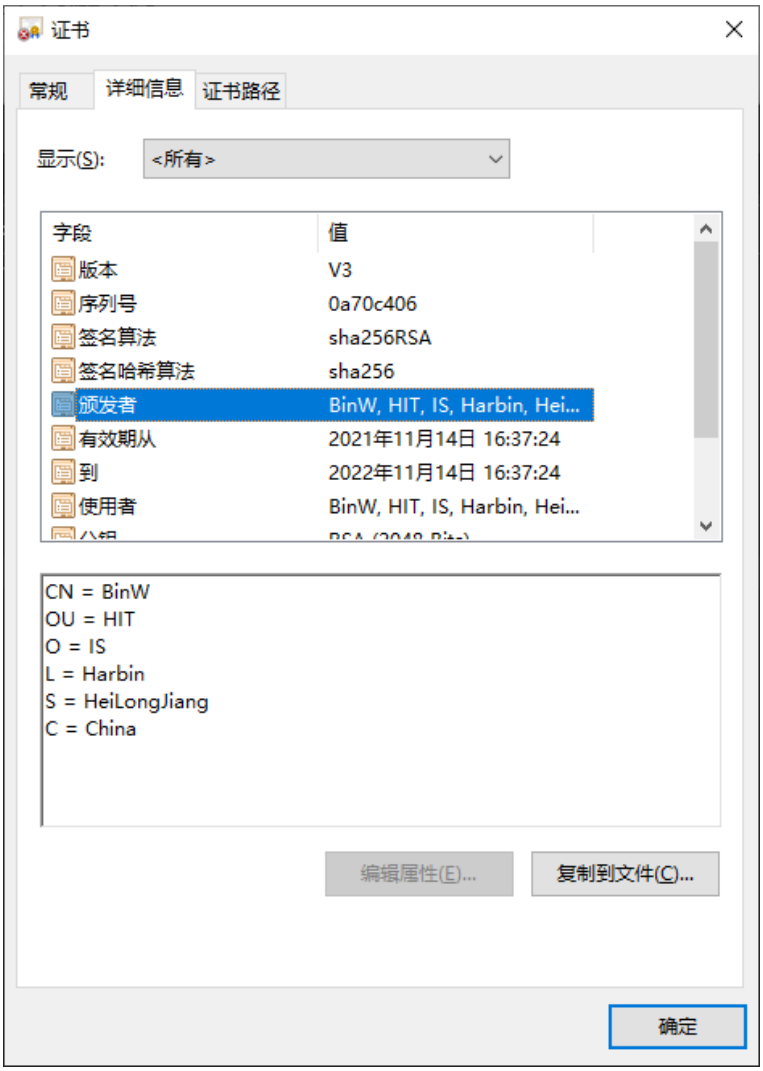


图 15.使用 win10 打开根证书

4.6 证书下载

根证书、个人版根证书、商业版根证书提供永久下载链接，而其他证书需要用户登录后下载。

4.7 证书撤销列表

提供过一个永久查询撤销列表的地址，其中内容为当前处于撤销状态的所有

证书，内容使用 CA 的私钥签名并使用电子信封的方式加密。

4.8 用户信息

用户可以更新自身信息，包括邮箱、电话和头像信息，管理员可以更改用户类型，分为个人版和商业版，用户在刚刚注册的时候为个人版用户，可以联系管理员提供证明后提升为商业版用户。个人版用户只能够申请个人版的证书，而商业版不限制申请类型。用户可以通过提供旧密码修改自身密码，对于遗忘自身密码的用户，提供证明给管理员后，管理员可以强制重置用户密码，将新密码发给用户，用户再做修改。

用户如果想要申请证书，也必须要进行实名认证，没有经过实名认证的用户处于游客状态，无法申请证书。

4.9 申请管理

用户可以发出证书申请，管理员在审核纸质材料和申请内容后，可以选择驳回和同意，驳回后用户可以修改申请后重新发出申请，同意则会生成证书。

4.10 证书管理

用户可以在不需要证书或者证书密钥泄露的情况下选择撤回证书，除此之外，也可以完成下载证书和获取私钥操作，还可以查看证书信息。

管理员不可以获取用户证书的私钥，只能撤回和撤销撤回证书，以及查看证书信息，防止私钥的泄露。

4.11 验证码

对于注册和登录操作，设置验证码，可以有效的防止黑客恶意提交表单和穷举用户密码。

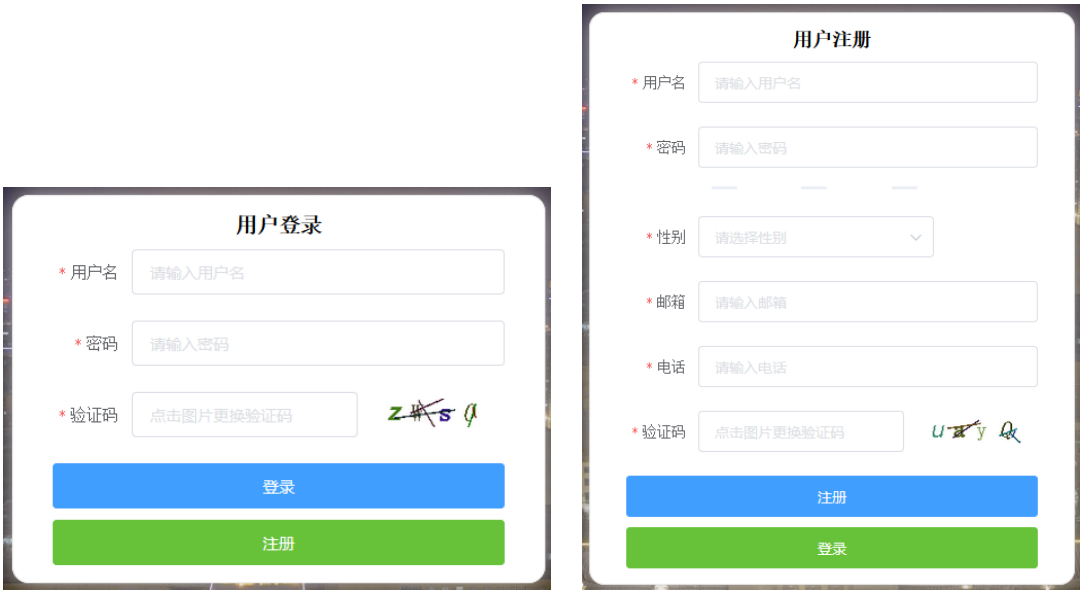


图 16.登录注册页面

4.12 审计日志功能

提供审计日志功能，对于用户（普通用户和管理员用户）的操作进行记录，

在出现事故时，方便进行责任的划分，同时对于日常维护（例如：封禁非法操作试图攻击服务器的用户）也有很大帮助。

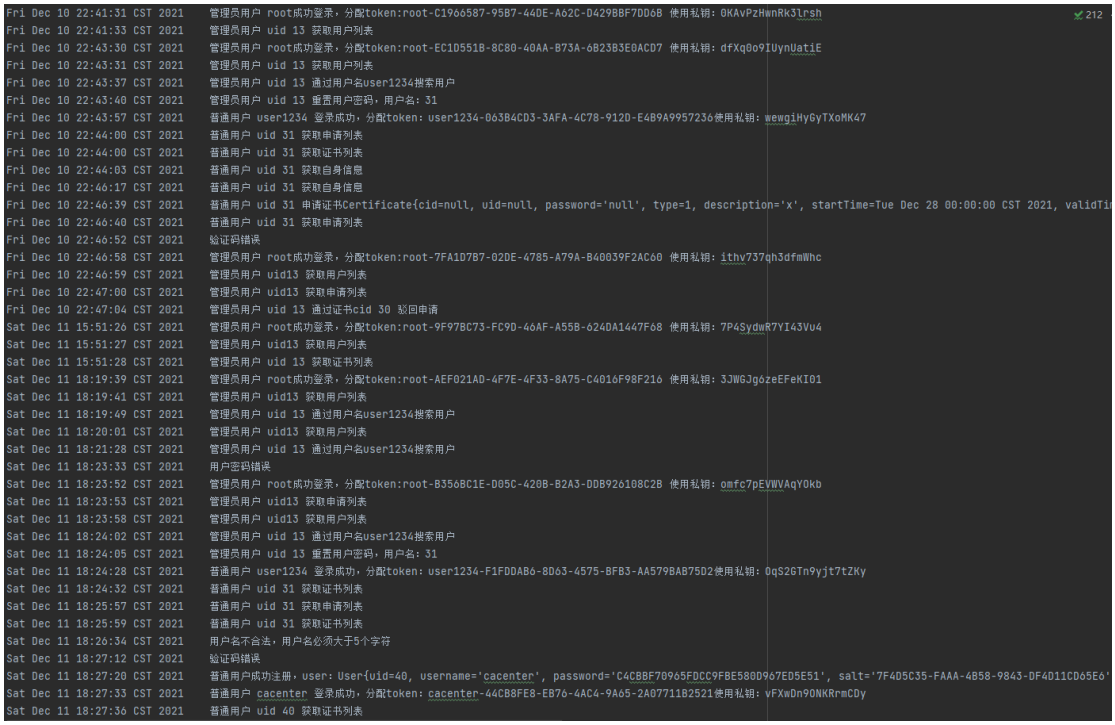


图 17.系统日志

4.13 后端开发证书使用工具

封装了 java 语言的证书使用函数，函数详细使用方式写在注释中，方便后端开发者使用。

```java
public interface ICaService {
    /**
     * 用证书公钥解密
     * @param data 解密数据
     * @param cert 解密证书
     * @return 解密结果
     */
    String decryptByCert(String data,String cert);

    /**
     * 用证书公钥加密
     * @param data 加密数据
     * @param cert 加密证书
     * @return 加密结果
     */
    String encryptByCert(String data,String cert);

    /**
     * 用我的证书公钥解密
     * @param data 解密数据
     * @return 解密结果
     */
    String decryptByMyCertByPublic(String data);

    /**
     * 用我的证书公钥加密
     * @param data 加密数据
     * @return 加密结果
     */
    String encryptByMyCertByPublic(String data);

    /**
     * 用我的证书私钥解密
     * @param data 解密数据
     * @param key 加密私钥的密码
     * @return 解密结果
     * @throws Exception 如果密码错误或者证书不存在会抛出异常
     */
    String decryptByMyCertByPrivate(String data,String key) throws Exception;
```

图 18.部分封装的证书函数

5.实现与测试

CA 网站使用了证书代理，所以需要运行证书代理，复制证书代理生成的 AES 密钥，实现与代理服务器之间的加密通信

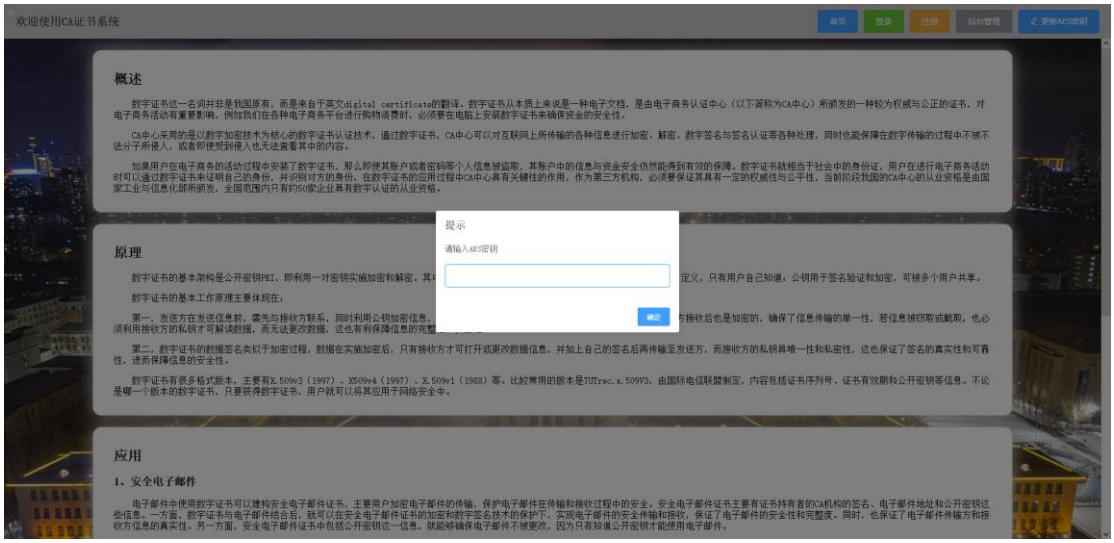


图 19.首次打开首页，提示输入代理启动时随机生成的密钥

5.1 注册

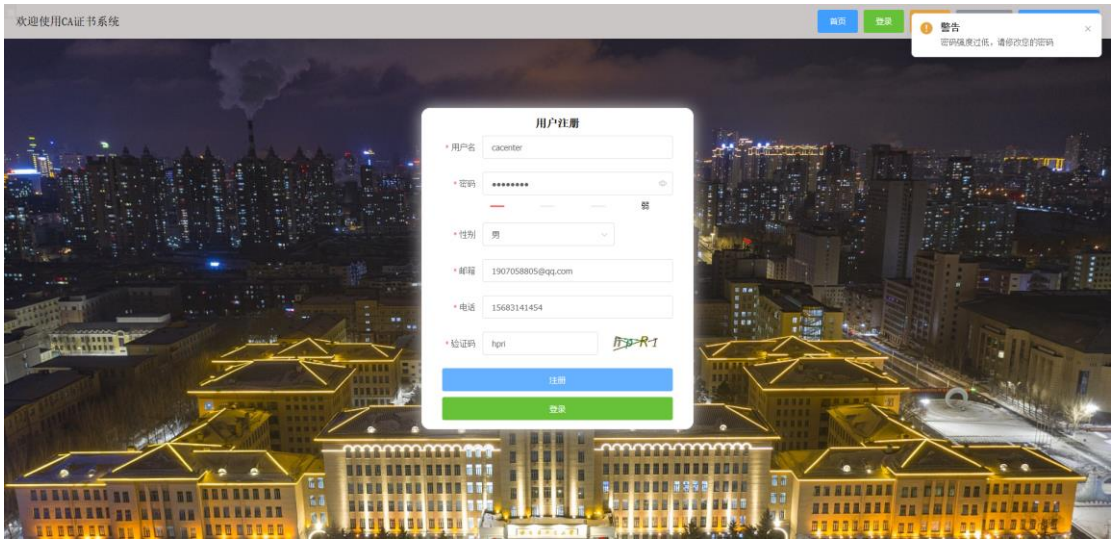


图 20.用户注册页面

在注册时会检查密码强度，弱密码无法注册成功，密码由四类字符组成，大写字母，小写字母，数字和特殊字符，且密码至少要 8 位以上，强度按照如下分级：

弱：只含有一种字符
中：含有两种或者三种字符
强：含有四种字符

5.2 登录

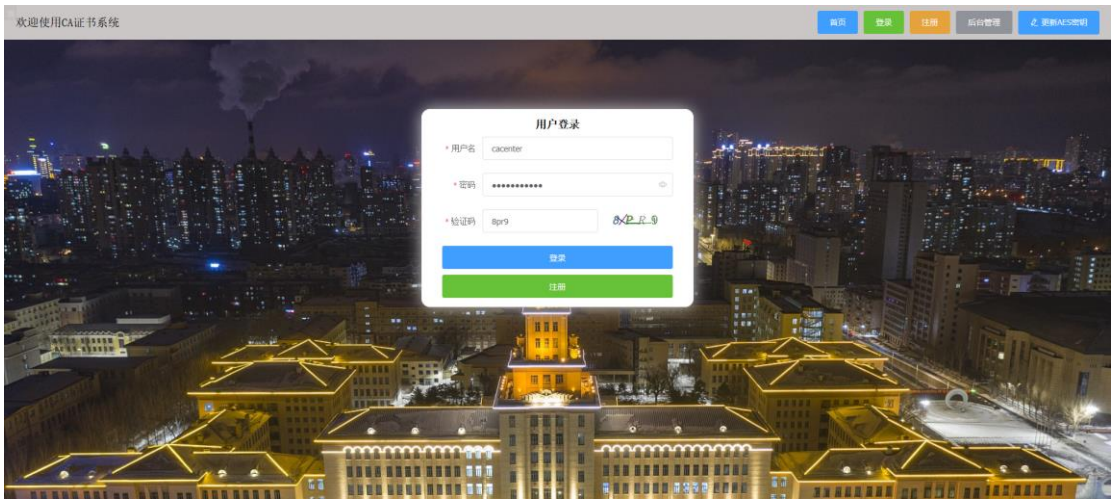


图 21.用户登录页面

用户输入账号密码和验证码后进行登录，如果验证码不正确后台会返回验证码不正确，如果验证码正确，账号密码不正确，后台会提示用户名或密码异常。

5.3用户实名认证

用户在刚刚注册完成后，处于游客状态，只有实名认证后才可以申请证书。

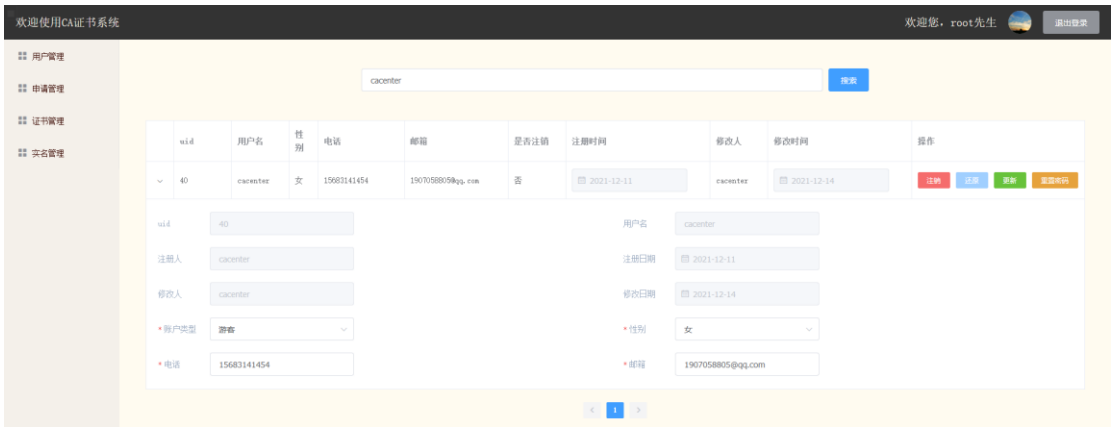


图 22.管理员用户信息查看

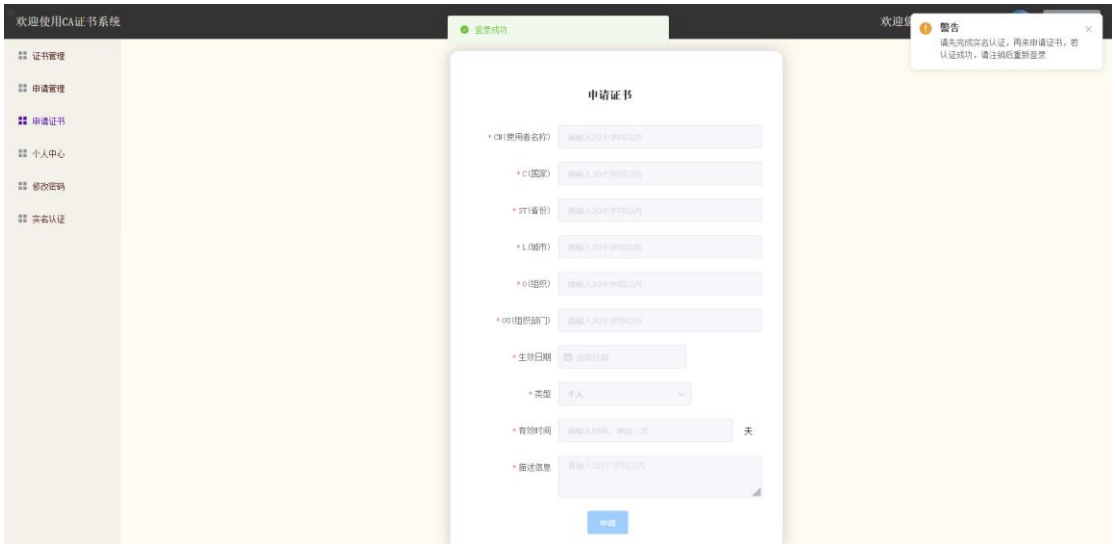


图 23.用户实名认证前无法申请证书

图 24.用户首先实名认证

图 25.用户实名认证成功

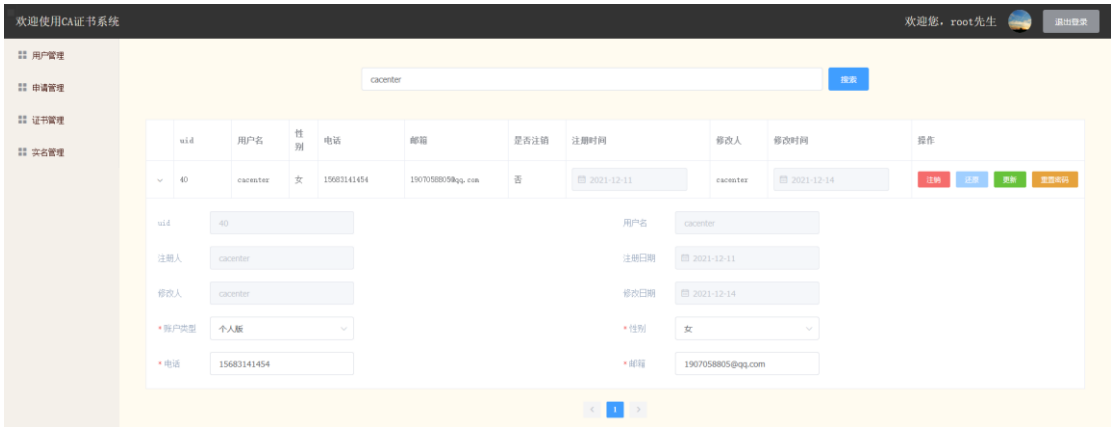


图 26.管理员用户信息查看

5.4 管理员管理用户信息

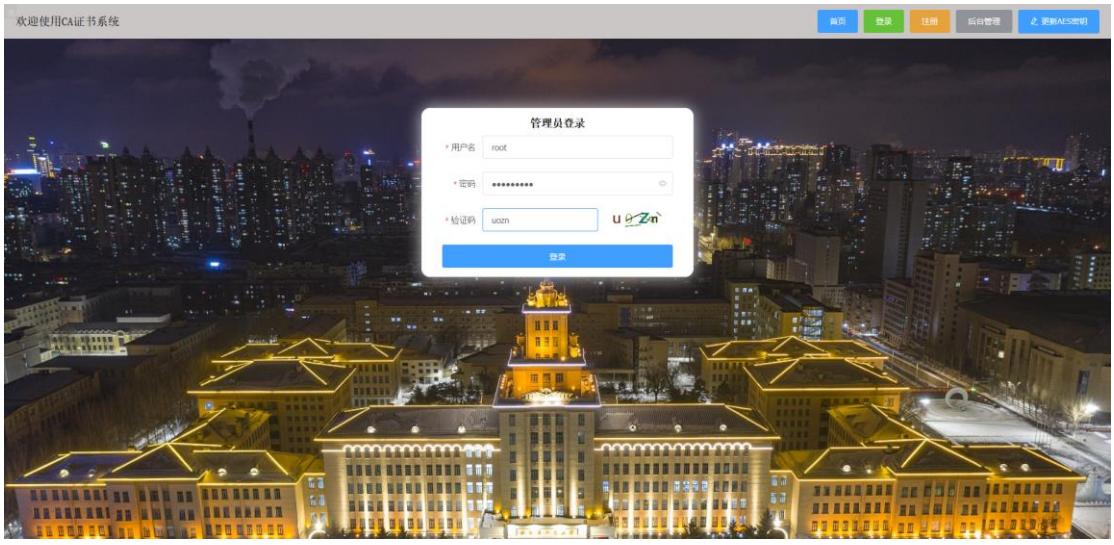


图 27.管理员登录页面

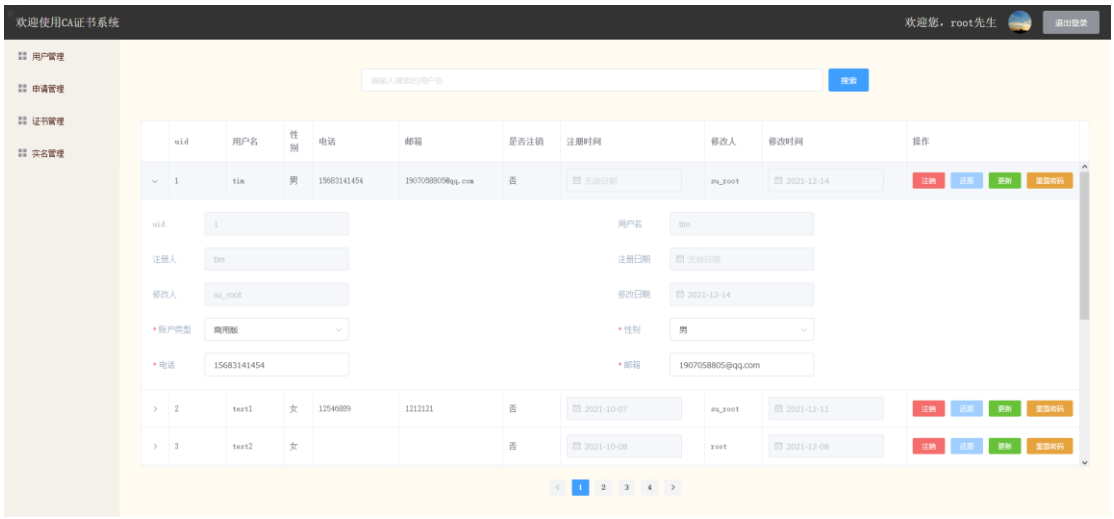


图 28.管理员用户信息管理页面

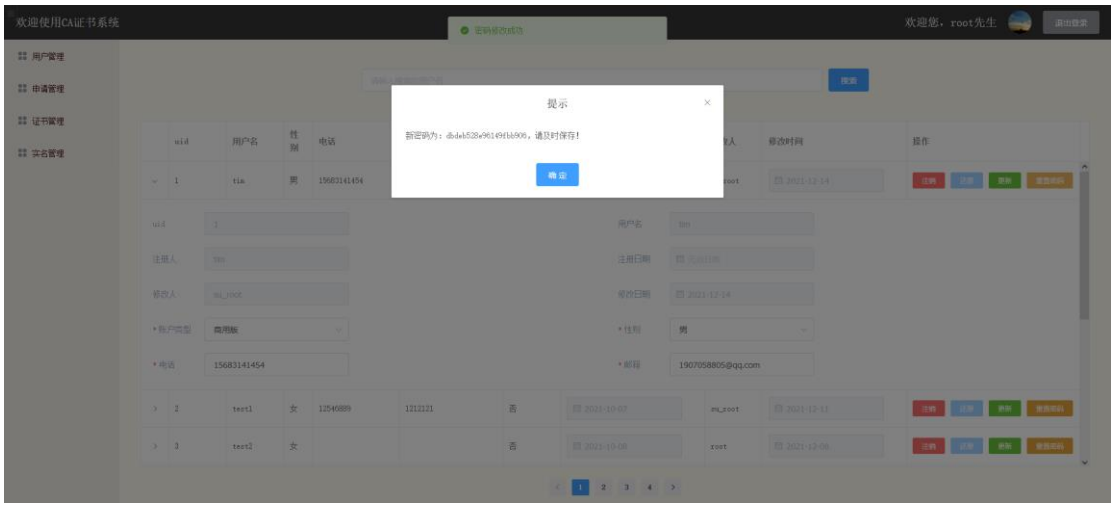


图 29.管理员重置用户密码

管理员登录后可以更新用户信息，修改账户类型，性别，电话，邮箱等，也可以注销或者还原用户，还可以重置用户密码。

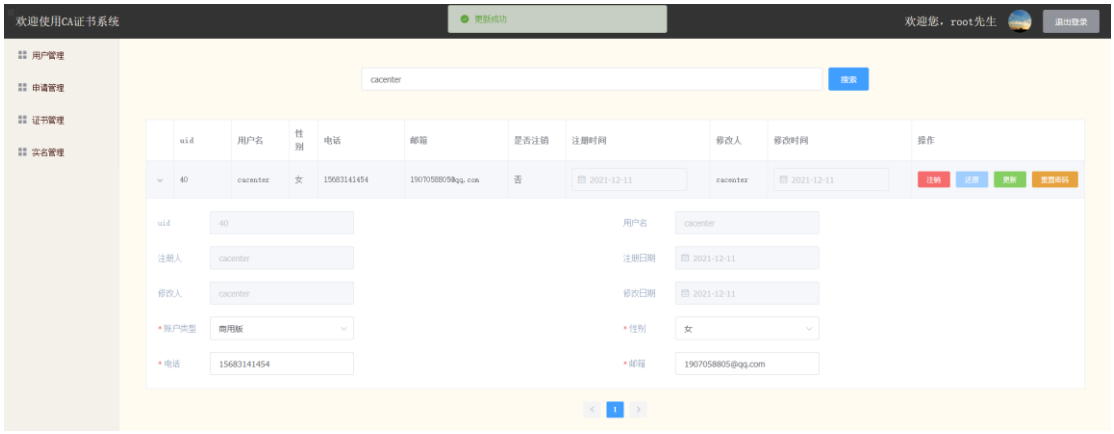


图 30.管理员更新用户信息

这里将刚刚申请的 cacenter 用户更新为商业版用户，方便后面演示。

5.5 申请证书

图 31.申请个人版证书

图 32.申请商业版证书

这里分别申请个人版和商业版证书，方便后续查看区别。

5.6 修改信息

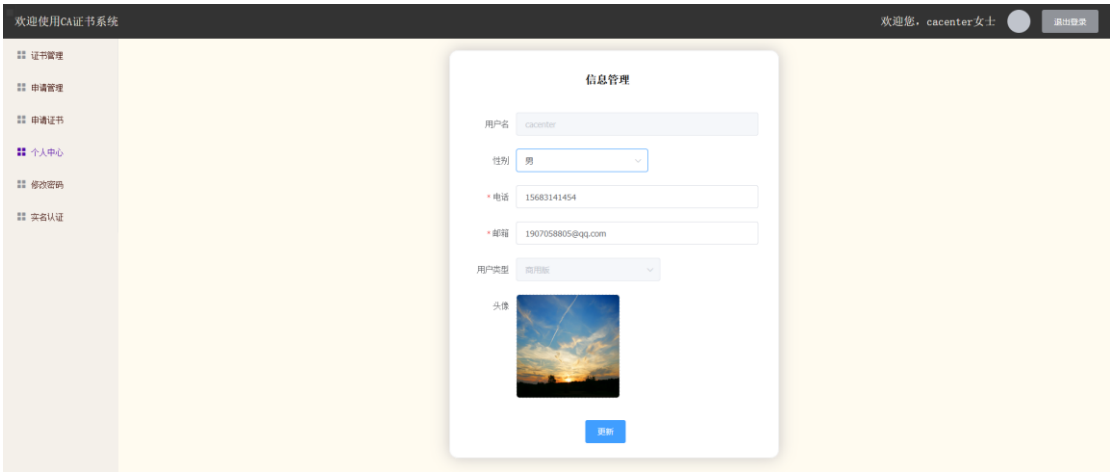


图 33.用户上传头像修改性别

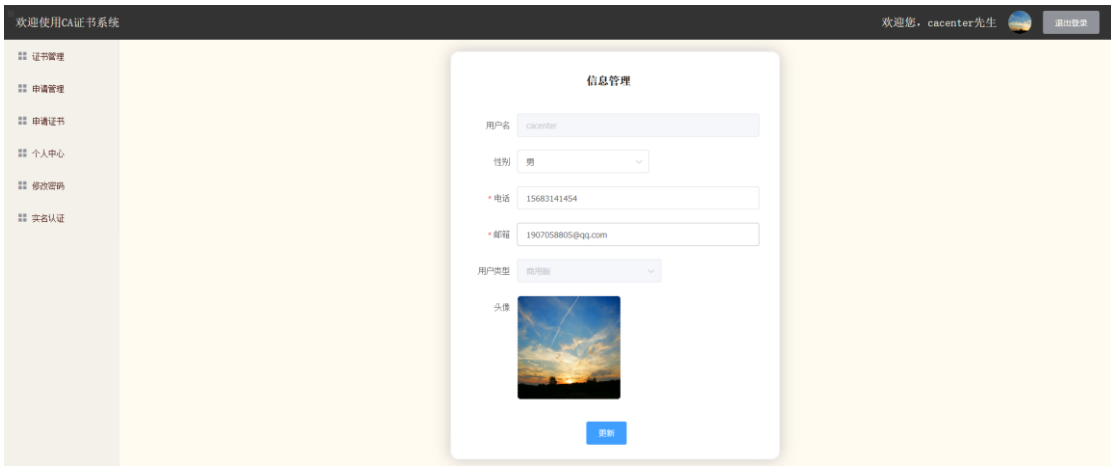


图 34.修改后的界面

用户可以完成修改自身信息，上传头像等操作

5.7 修改密码

图 35.用户修改密码

用户可以通过提供新旧密码，来修改自身密码。

5.8 管理员申请管理

图 36.管理员审核证书申请

这里可以看到上面申请的两个证书，这里可以驳回个人版，通过商业版查看区别。

5.9 用户申请管理

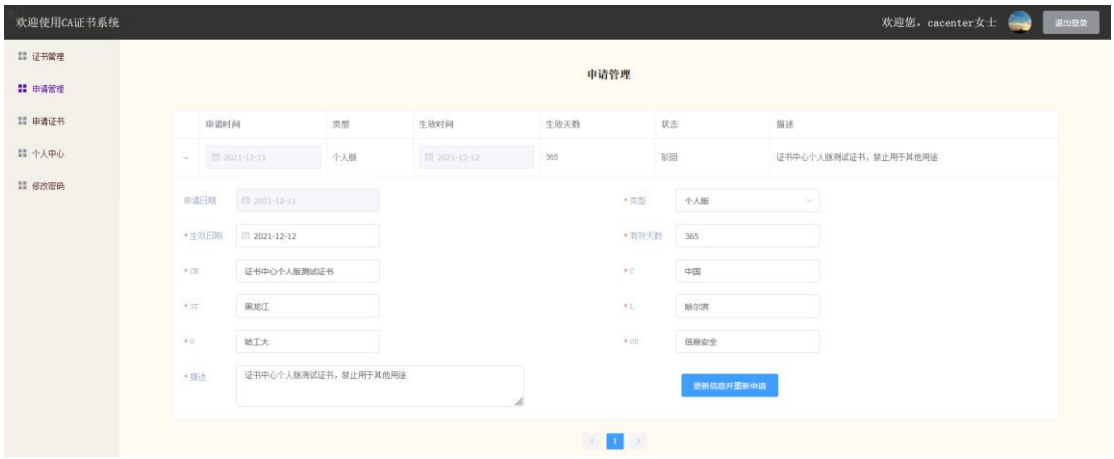


图 37.用户申请管理

在用户的申请管理里面可以看到刚刚驳回的证书，这里重新申请，后台予以通过。

5.10 用户证书管理

图 38.用户证书管理

在证书管理中，可以下载证书，召回证书，和获取证书私钥。

首先测试下载证书，将两个证书下载打开

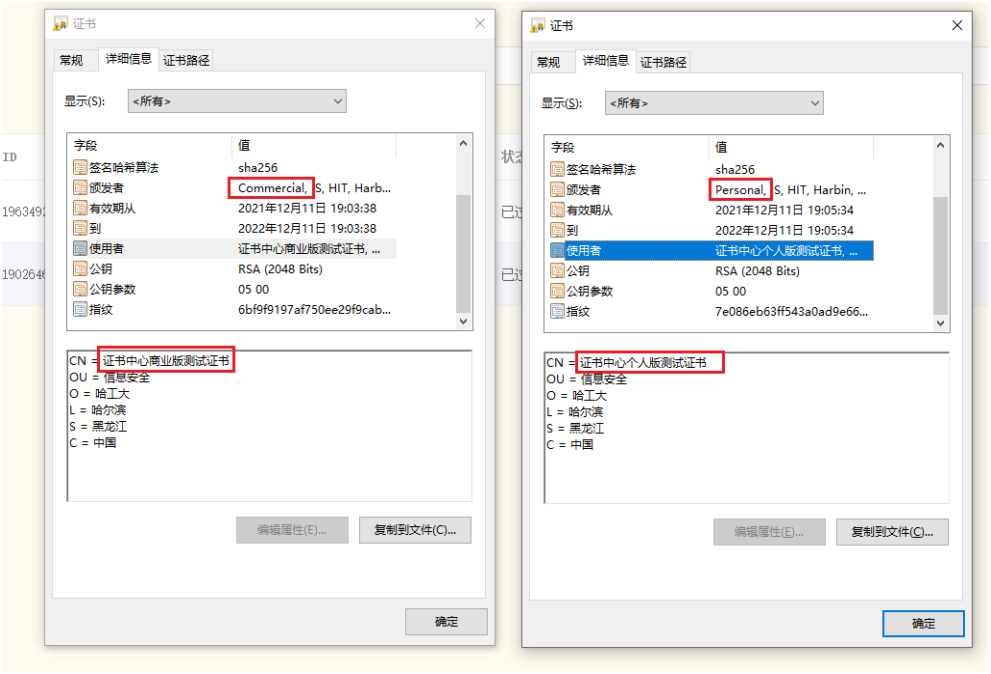


图 39.用户查看下载的证书

尝试撤回个人版证书，发现证书出现在撤销列表中

图 40.用户撤销证书

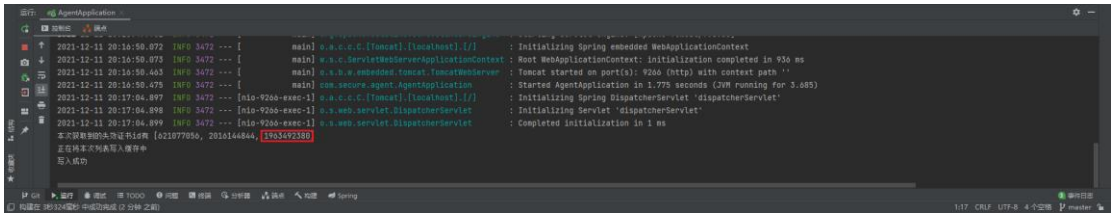


图 41.证书代理获取撤销列表结果

最后获取用户私钥，当用户点击获取私钥后，会让用户输入用于加密私钥的密码，之后才会返回私钥

图 42.用户输入加密私钥的密码

图 43.服务器返回加密后的私钥

5.11 证书代理个人证书的配置

图 44.证书代理配置示意图

MyCa 文件夹配置如下

图 45.私人密钥和证书配置示意图

5.12 管理员撤销撤回证书

管理员撤销撤回刚刚由用户误撤销的证书

图 46.管理员管理证书

可以发现刚刚撤销的证书已经不在列表中

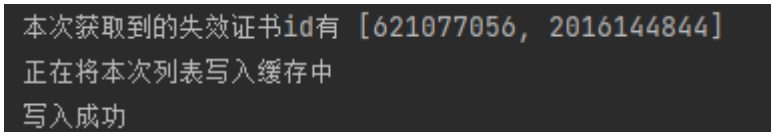


图 47.证书代理获取撤销列表

6.结束语

这次密码学实践，自己从头到尾设计了一个 CA 认证系统，从构思到克服一个个困难，可谓是路途坎坷。在这次实验过程中，最大的困难是证书编程部分，之前从未有过类似的经验。经过长达三周的时间，在学习多篇博客后，终于封装出了一个可以满足本次实验需求的证书类，在完成这个类后，其他的很多问题都可以迎刃而解。

这次实验极大的提升了自己的代码能力，之前从未接触过这么大代码量的项目。经过这次实验，提升了自己前后端编程的熟练度，对于安全编程有了更深的理解。

参考文献

[1]江为强,陈波.PKI/CA 技术的起源、现状和前景综述[J].西南科技大学学报(自然科学版),2003(04):75-78+83.
[2]https://www.cnblogs.com/daizhongxing/p/11593137.html 常见密码正则表达式
[3]https://www.cnblogs.com/demodashi/p/8458113.html Java 使用 RSA 加密解密签名及校验
[4]https://blog.csdn.net/lk2684753/article/details/88642423 CA 基本常识:X.509 标准