**基于区块链的身份认证系统**

本项目设计的基于区块链的身份认证系统包括三个主要功能：去中心化用户身份管理，身份认证，用户信息隐私保护。

系统模型主要分为五个部分：系统角色、客户端、区块链、IPFS 和接口层。其中系统涉及到的角色主要有两种，用户和验证方。具体描述如下：

① 普通用户：即系统的身份信息数据拥有者，同时也是访问策略的制定者。

② 验证方：即用户链上身份信息的验证者，保证链上身份信息真实可靠。验证方也可以是服务商，发起对用户身份的认证。

本系统采用以太坊平台实现智能合约的运行，基于 Web 平台实现一个身份管理以及认证系统。接口层主要为客户端、区块链以及 IPFS 提供交互接口，主要有与身份凭证发布相关的注册和验证接口、身份信息加密接口以及获取明文信息的解密接口。本研究中的系统架构总体设计如图 1所示。其中，用户身份凭证发布即用户可信身份信息上链过程，对应图 1 中步骤 1-5，主要分为身份注册（步骤 1-3）和身份验证（步骤 4-5）两个部分，用户通过身份注册实现在区块链中的身份信息上链，而身份验证是为了保证链上身份可信任，通过身份注册和身份验证实现用户身份凭证的发布。步骤 6-9 是用户利用可信身份凭证在服务提供商进行身份认证的过程。

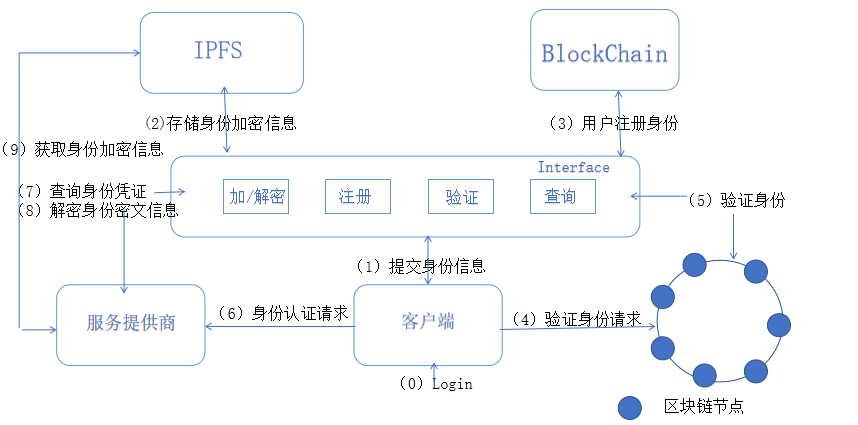


图1.系统架构设计

**1.去中心化用户身份管理**

去中心化的用户身份管理主要通过智能合约实现。部署在区块链（以太坊）上的智能合约是本项目的核心部分，通过合约交互实现主要的业务逻辑。在外部与智能合约交互上可以分为发送交易和消息调用，其中发送交易会修改区块链数据，修改时会产生一笔交易记录，主要用于本项目中用户身份信息数据上链和修改身份凭证状态两个部分；消息调用主要是用于查询区块链上的数据，一次查询不会产生交易记录，主要用于项目中用户身份信息数据的获取上。

智能合约设计如图5所示

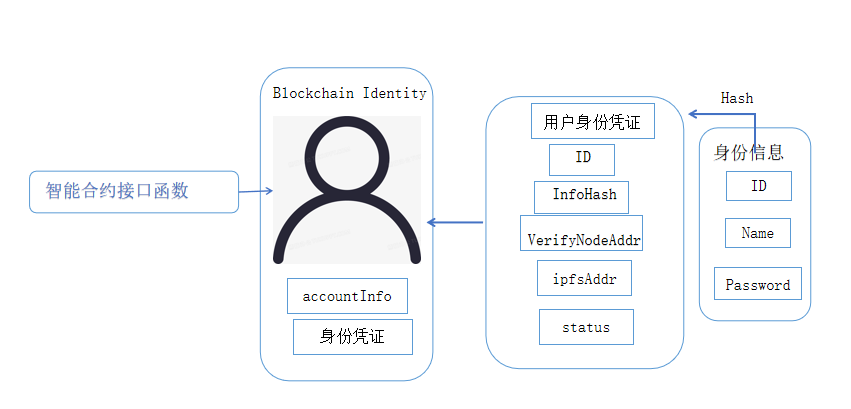


图5 智能合约设计

身份注册是实现用户链上身份存储和管理的前提，本设计中主要涉及到的角色分别是用户（user）、验证节点（verifyNode）和系统（system，区块链身份认证系统）。用户注册身份设计完整流程如下所示：

(1) system:setup→ pk,sk

初始化阶段。系统执行用于生成RSA需要的公私钥对。

(2) user→system:submit(userInfo,verifyNodeAddr)

用户提交身份注册信息。用户向系统提交注册身份信息userInfo 以及验证节点的以太坊地址verifyNodeAddr，通过指定验证节点验证身份信息来确保上链信息真实性。

(3) system:encript(userInfo)→encriptInfo

加密用户身份信息。此步骤由系统执行加密算法，输入用户身份信息userInfo，输出身份加密结果encriptInfo。

(4) IPFS:ipfsAdd(encriptInfo)→ipfsAddr

IPFS 存储用户加密身份信息。此步骤由IPFS 执行，输入步骤(3)中得到的加密结果，输出IPFS 存储时计算得到的哈希值。利用IPFS 存储用户身份信息密文encriptInfo，返回计算得到哈希值ipfsAddr。

(5) system:registerIdentity(userInfo,verifyNodeAddr,ipfsAddr)。

身份信息上链由系统执行，输入用户提交的身份信息userInfo、验证节点地址verifyNodeAddr，以及IPFS 存储时得到的哈希值ipfsAddr（若未存储则为空），输出用户身份凭证唯一标识ID。通过调用智能合约中的身份注册函数实现身份信息上链。用户身份信息在区块链中一次身份注册，生成一个待验证的身份凭证，其状态为pendding。

**2.身份认证**

用户通过一次注册身份后，实现了线下身份上链过程。为了保证链上信息的真实性，被指定验证节点对已注册身份凭证进行验证。验证身份过程设计主要如下：

(1) user verifyNode:→ req(ID,userInfo,sign(sk,userInfo))

用户请求验证链上身份真实性。用户线下向指定的验证节点发起身份验证的请求，该验证节点与注册身份时选择的验证节点为同一个。其中提交请求的参数分别是用户链上身份凭证唯一标识ID、用户注册身份信息userInfo 以及用户私钥sk 对身份信息的签名值sign(sk, userInfo)。

（2）verifyNode:verify(userInfo),verfySignature(sign(sk,userInfo),

Hash(userInfo))

验证节点验证注册身份信息。验证节点根据ID，通过调用智能合约获取用户链上对应已注册凭证信息，包含用户身份infoHash、ownerAddr 等。根据凭证信息验证注册身份信息的真实性、完整性以及发起验证请求的用户是否为注册身份的拥有者本人。验证机制如下：

1.验证节点线下验证用户提交的身份信息ueserInfo 的格式以及真实性。

2.验证签名。

验证verfySignature(sign(sk,userInfo),Hash(userInfo))? = owerAddr，若等式成立，则验证通过；否则验证失败。由于在区块链中私钥是用户账户拥有权的证明。同样的，在本项目公钥地址代表了链上用户实体，而拥有私钥代表线下用户对链上身份的所有权象征。通过验证签名来确保当前用户拥有对应私钥，实现线下身份和链上身份的关联。

3.验证节点本地计算公式Hash(userInfo)? = infoHash。若等式成立，则表明用户注册身份信息是真实且完整的。

(3) verifyNode:confirmIdentity(ID,expiryDate)验证节点修改凭证状态和设置过期时间expiryDate。通过(2)中验证后，验证节点通过修改用户凭证状态达到标识链上身份通过验证的目的。验证节点调用智能合约修改凭证状态，由注册待验证（pendding）修改为已通过验证（pass），代表用户成功发布一个身份凭证，实现了可信身份上链。

通过身份注册和身份验证，完成用户可信身份凭证的发布。用户可以选择发布多种类型的身份信息。生成的凭证可用于在其他服务商处进行身份认证，该凭证是信任用户身份的可信依据。

**3.用户信息隐私保护**

(1) 身份信息隐私性

区块链作为一个去中心化的分布式账本，使得链上交易信息公开可验证，但随之也带来了身份隐私安全性问题。为了保证用户身份信息隐私安全性，本项目采取不以明文形式存储用户身份信息，防止用户链上身份信息泄露，保证用户身份信息安全。由于infoHash=Hash( userInfo) ，区块链凭证中存储的是用户身份信息数据哈希运算后的值 infoHash，由于哈希函数单向性的特点，使得通过凭证 infoHash 反向推导出身份明文信息 userInfo，在计算上具有一定的难度。除此以外，由于在 IPFS 中存储的是身份信息加密密文encriptInfo,能够在一定程度上保护用户的隐私安全。

(2) 不可伪造性

主要是针对链上可信身份的不可伪造，分为主动伪造和被动伪造两个方面。

第一种是主动伪造，主要是指身份拥有者（用户）注册虚假身份。系统采用如下方式进行防范：

① 信息验证：验证节点采取线下本地验证方式对用户请求的身份信息真实性和

完整性进行确认。假设用户提供虚假身份信息 userInfo\*，那么将无法通过验证节点本地验证；假设用户提供真实身份信息 userInfo，但注册身份阶段提供虚假身份信息，那么 Hash (userInfo)= \* infoHash 将不成立（infoHash\*是身份注册后凭证中存储的身份信息哈希值），同样将无法通过验证。

② 签名验证：验证节点验证Dec(sign(sk,userInfo ),Hash(userInfo ))?= ownerAddr 。由于 ownerAddr 与指定身份凭证相关联，若恶意分子试图伪造用户私钥对指定身份信息的签名值，将无法通过该签名验证计算。

第二种是被动伪造，是指在用户不知情的情况下第三方恶意伪造用户链上身份或修改身份信息。为了保证用户对其链上身份自主可控，避免其他以第三方机构为中心的身份管理模型带来的安全隐患，项目采取以用户为中心的设计思想，链上身份由用户自己注册，实现对其链上身份的自主可控。由于链上身份（凭证）与区块链账户地址、私钥等相关联，注册身份机制保证了仅有账户拥有者本人才能执行该操作。除此以外，智能合约中定义了验证节点只能对凭证状态、创建时间以及过期时间进行修改，并无修改用户链上身份相关信息的权限。

用户信息隐私保护执行流程图如图6所示：

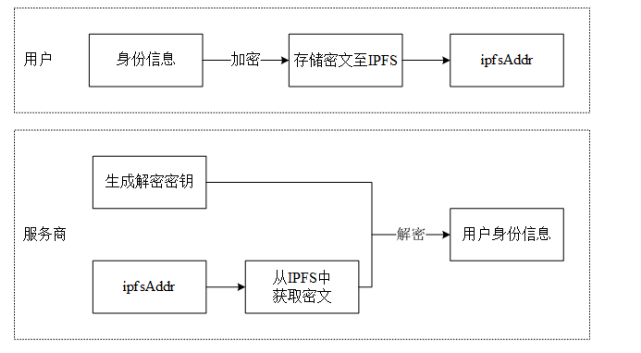


图6 用户隐私信息保护

具体执行过程为：

(1) encrypt(pk,userInfo)->encryptInfo

系统通过注册身份时初始化生成的公共参数 pk、 以及用户身份信息数据 userInfo，使用RSA加密算法加密用户信息，将得到的用户身份密文信息 encryptInfo 存储到 IPFS，并返回计算得到的哈希结果 ipfsAddr。IPFS 能够进一步实现信息共享，并且能够弥补区块链无法存储大规模数据的不足的问题。

1. 解密密文：请求验证身份的用户将私钥发送给验证方或者服务方， decript(sk,encryptInfo →userInfo

**系统介绍和展示**

系统主要是基于以太坊利用Ganache实现系统开发。通过Ganache 可以搭建一个以太坊私有链进行功能实现和测试。智能合约的编写主要是运用了Solidity 语言，利用Remix 部署、编译和测试合约。系统前端的开发主要运用了html和JavaScript。

系统主要由三部分组成：

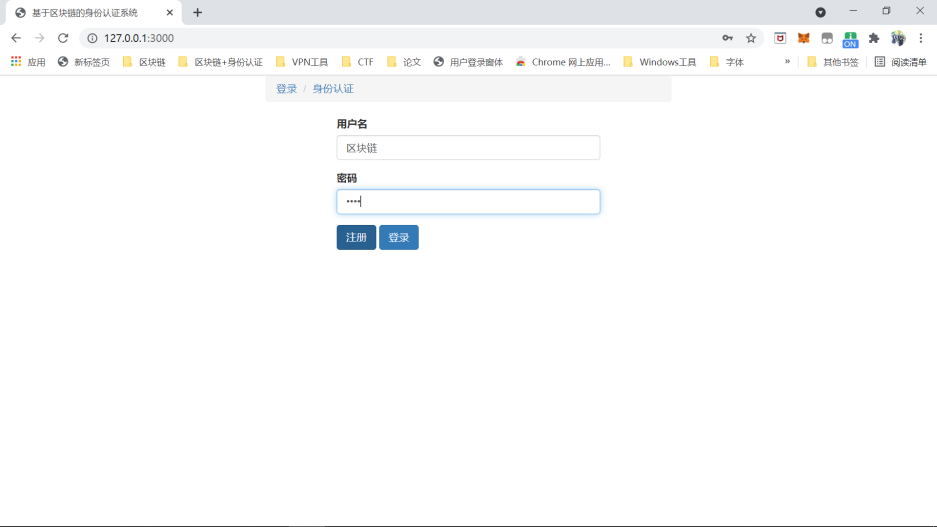
(1) 前端页面：前端框架采用的是html和JavaScript，采用的Web 浏览器是Chrome。

(2) 智能合约部分：主要是用Solidity 语言编写的合约内容。

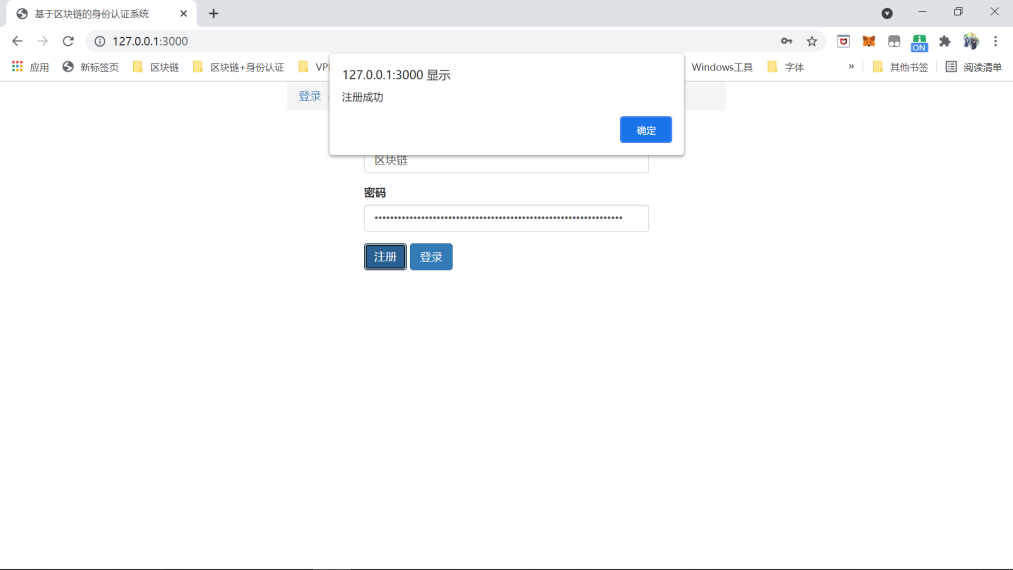
(3) 接口部分：主要是封装了身份注册、加解密等算法接口。

非系统用户通过账户注册功能成为系统用户。通过输入密码实现账号注册。首先用户填写密码表单；然后计算密码哈希值passwordHash，通过查询账号是否已经存在。如果已存在则提示已注册，否则通过调用创建账号实现注册功能，返回用户账户地址address、用户私钥privateKey 等信息；接着调用智能合约中的registerAccount 接口函数存储用户passwordHash 与账户地址address之间的映射关系。最后提示注册成功，返回账户地址和私钥，用户可复制保存。系统

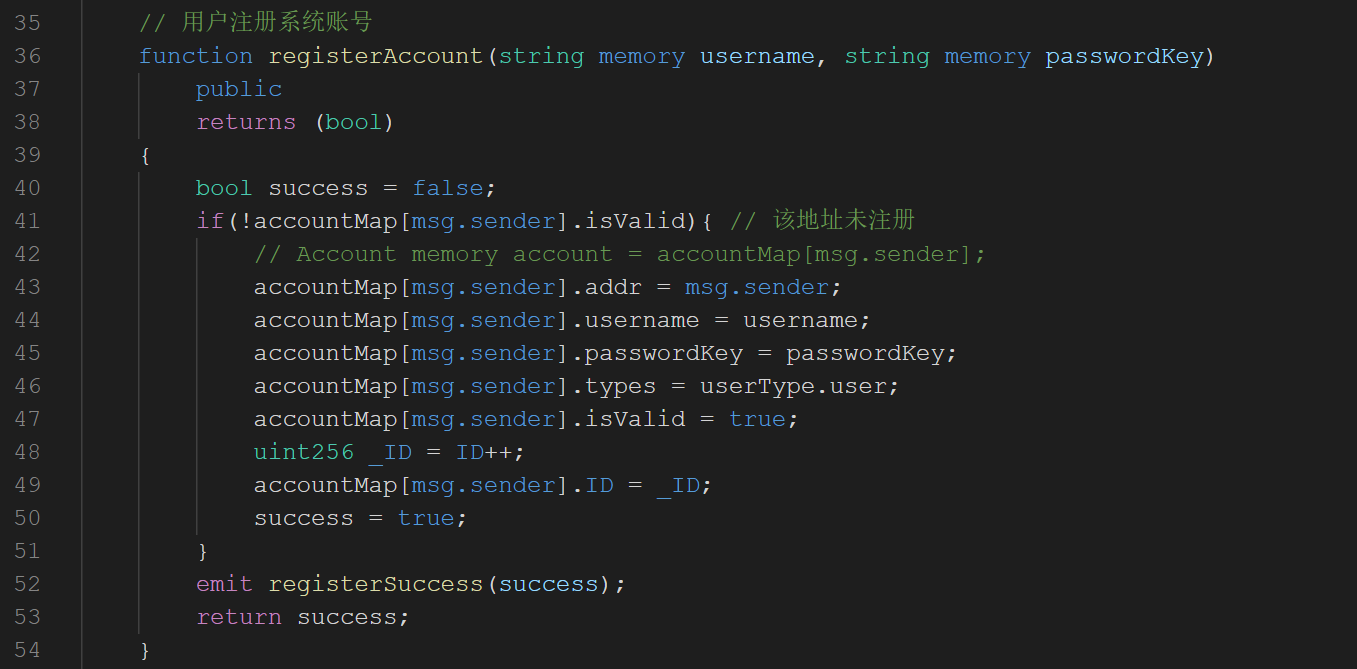
注册和注册成功Web 页面分别如图所示。



注册

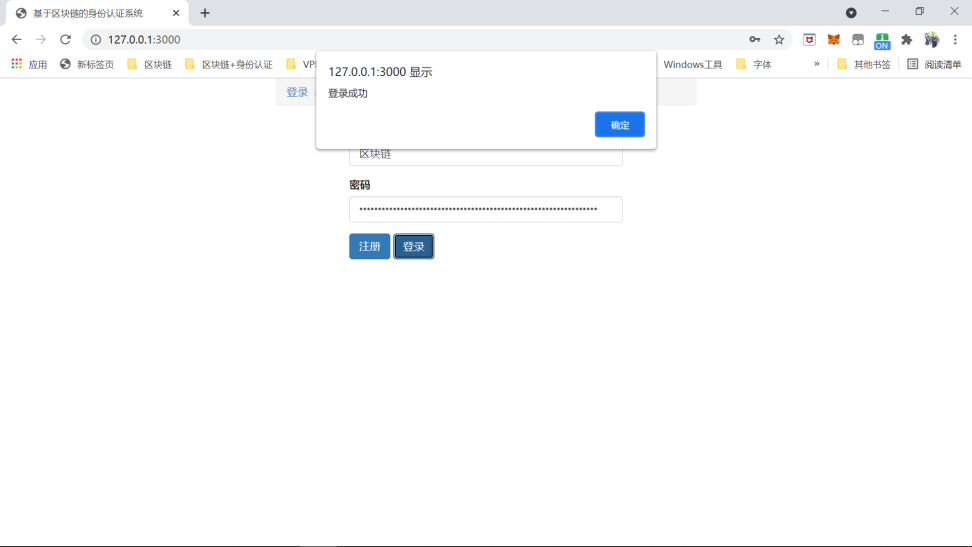


注册成功

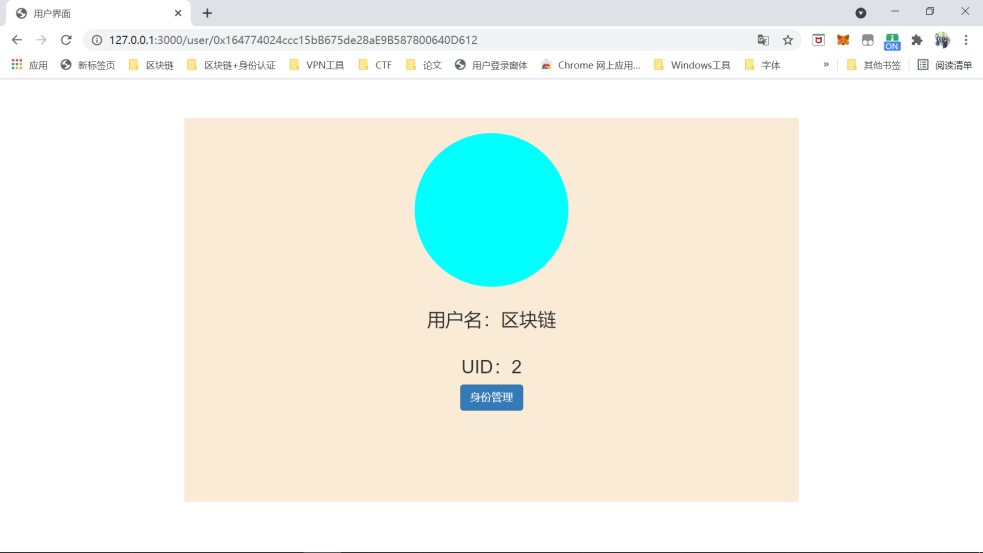


注册代码

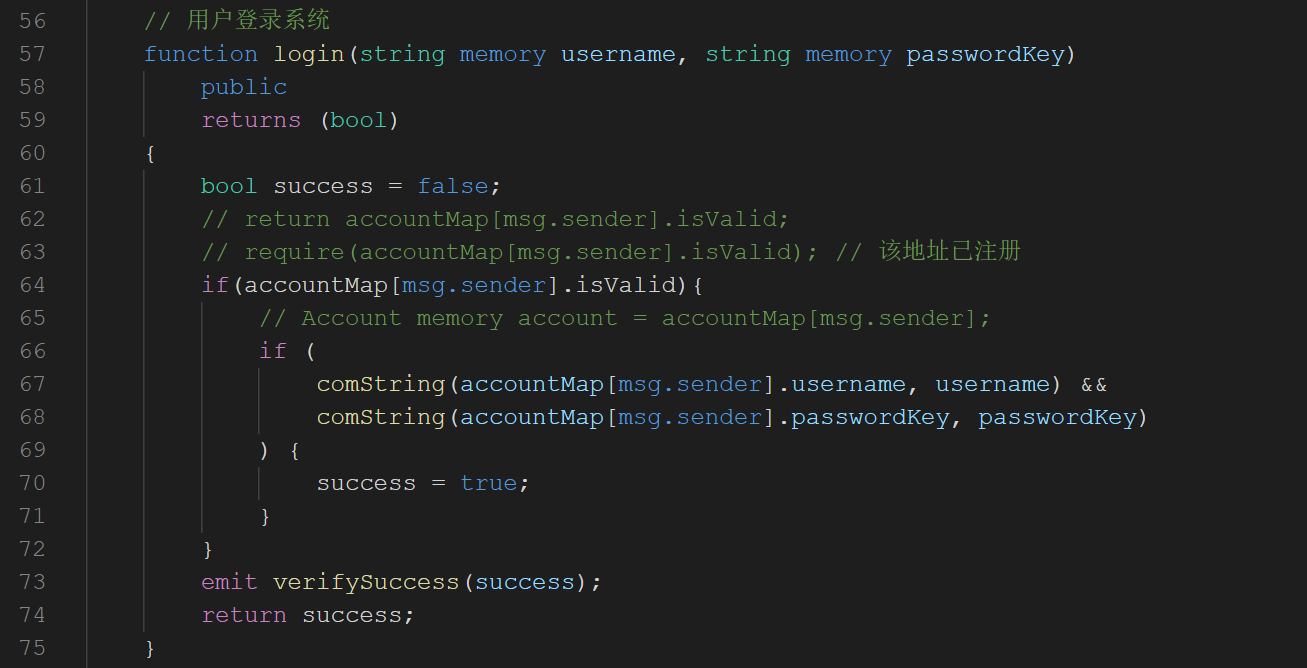
已有账号用户直接可以登录系统，用户在登录页面填写账户地址和密码表单信息。登录时，系统根据密码计算哈希值passwordHash，查询“address-passwordHash”映射表，若存在表明填写信息无误，提示登录成功。登录Web 页面如图所示。



登录

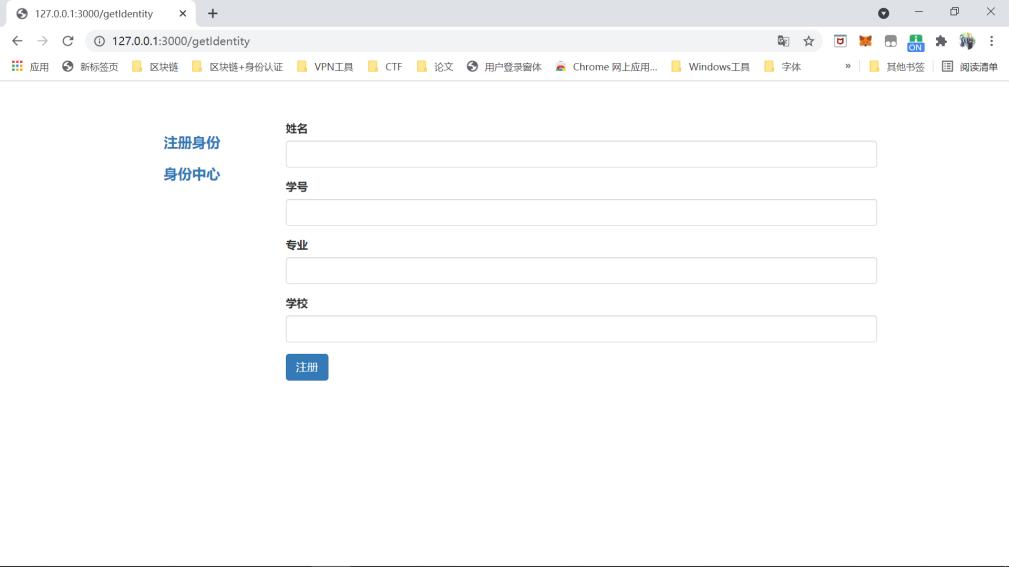


登录成功

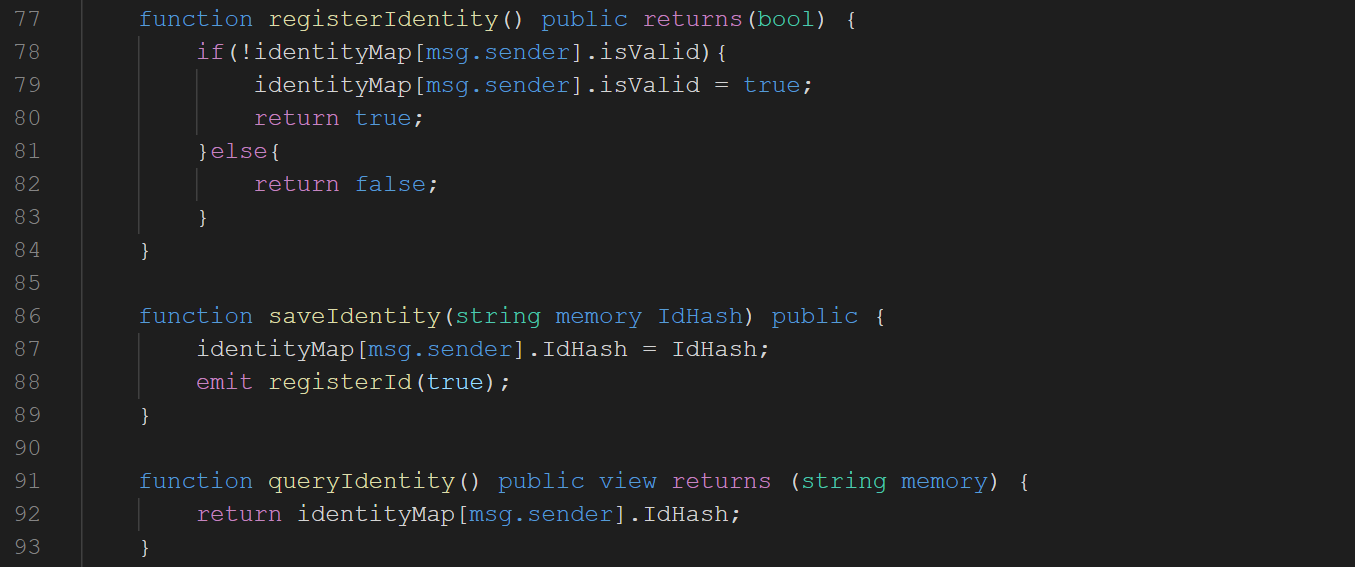


登录代码

登录成功后展示系统页面，系统内展示“注册身份”和“身份中心”两个页面。身份注册主要实现是用户注册身份功能。系统Web 界面表现为用户登录系统后，前往注册身份页面提交身份信息，如图所示。

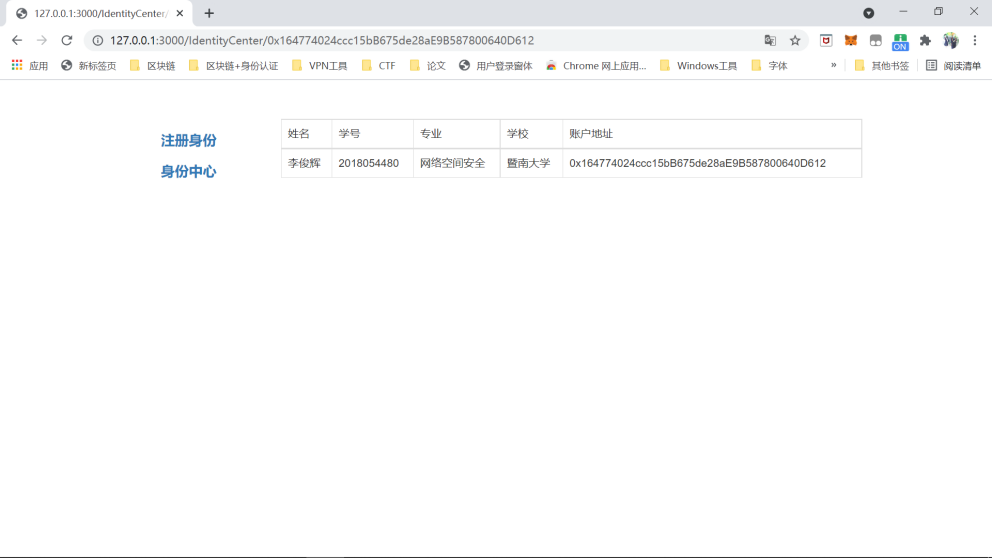


用户身份注册

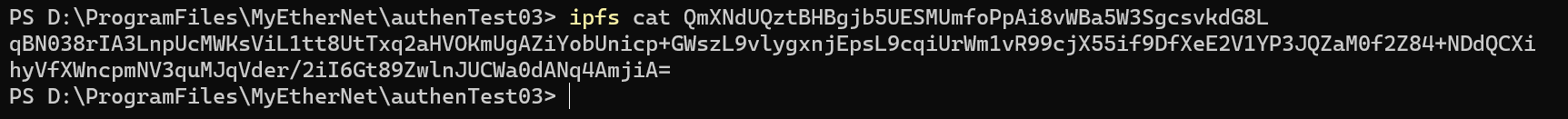


用户注册身份凭证代码

在系统内配置了“高校学生基本信息”模板，首先用户在注册身份页面填写用户身份表单信息（实际生活中考虑到需要注册更为复杂的信息，可以通过直接上传文件的形式）。填写需要存储身份信息（至IPFS 中），利用属性加密算法加密身份明文信息，并将密文存储在IPFS 中，最后将IPFS 返回哈希值同身份信息一同写入身份凭证中。注册成功后用户可以选择前往“身份中心”查看已注册身份信息。Web 页面如图所示。



身份凭证中心



IPFS存储系统



IPFS加密存储代码

**系统测试**

系统测试使用的计算机配置以及实验环境如表所示

|  |  |  |
| --- | --- | --- |
| 表1 实验环境配置表 | | |
| 名称 | 具体信息 | |
| 操作系统 | Windows 10 |  |
| CPU | Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz |  |
| 内存 | 8GB |  |
| 浏览器版本 | Chrome 91.0.4472.114 (64位) |  |
| 开发语言 | Solidity, JavaScript等 |  |

5.1.账户注册和登录功能测试

表2 注册

|  |  |
| --- | --- |
| 用例编号 | Test01 |
| 测试模块 | 系统账号注册 |
| 测试目的 | 测试系统账号注册功能是否正常 |
| 用户类型 | 普通用户 |
| 操作步骤 | 1. 填写注册相关表单信息 2. 点击注册按钮 |
| 预期结果 | 若账号已经存在，提示注册失败，否则注册成功 |
| 测试数据 | Example1：  注册用户名：区块链  注册密码：1234  Example2：  注册用户名：区块链  注册密码：1234 |
| 实际结果 | 1. Example1中，提示注册成功 2. Example2中，提示账号重复注册，注册失败 |

表3 登录

|  |  |
| --- | --- |
| 用例编号 | Test02 |
| 测试模块 | 系统账号注册 |
| 测试目的 | 测试系统登录功能是否正常 |
| 用户类型 | 普通用户 |
| 操作步骤 | 1. 填写登录相关表单信息（用户名和地址） 2. 点击登录按钮 |
| 预期结果 | 1. 若正确输入表单信息（用户名和密码），提示登录成功 2. 若错误输入表单信息（用户名和密码），提示信息输入有误 |
| 测试数据 | Example1：  用户名：区块链  密码：1234  Example2：  用户名：区块链  密码123456 |
| 实际结果 | 1. Example1中，提示登录成功，系统页面跳转正常，进入用户主页面 2. Example2中，提示信息有误，页面无跳转 |

表4 注册凭证

|  |  |
| --- | --- |
| 用例编号 | Test03 |
| 测试模块 | 用户身份凭证注册 |
| 测试目的 | 测试身份凭证注册是否正常 |
| 用户类型 | 普通用户 |
| 操作步骤 | 1. 填写基本身份信息 2. 点击提交按钮 |
| 预期结果 | 注册完成后，将在“身份中心”列表中出现一条身份凭证信息 |
| 测试数据 | Example1：  姓名：张三  学号：2018050000  专业：网络空间安全  学校：暨南大学 |
| 实际结果 | 注册后生成一笔交易，同时在“身份列表”中新增一条身份凭证信息 |

表5 IPFS存储

|  |  |
| --- | --- |
| 用例编号 | Test04 |
| 测试模块 | 身份凭证存储 |
| 测试目的 | 1. 测试身份凭证是否正常 2. 测试加密是否已在IPFS存储中应用 |
| 用户类型 | 普通用户 |
| 操作步骤 | 注册身份凭证Test03 |
| 预期结果 | 注册完身份凭证后，在IPFS中能够查看到该条身份凭证的密文存储 |
| 测试数据 | 注册身份凭证Test03 |
| 实际结果 | 注册完身份凭证后，在IPFS中能查看到该条身份凭证的密文存储 |