浙江工商大学计算机与信息工程学院

上机实验报告 （）

课程名称： 密码货币与区块链技术 姓 名： 梁宇航 沈林杰 黄尧 学号：2212190506 2212190519 2212190512

指导教师：邵俊 班 级： 安全 2201 日 期： 2024 年 9 月 26 日

【一】实验内容及要求

实验名称：基于 RSA 盲签名的交易系统实现

实验目的

设计基于盲签名的货币交易系统

实验环境

Windows 操作系统，python

实验内容

1. 设计盲签名核心代码
2. 设计电子货币的标准
3. 实现角色为中央银行的服务端
4. 实现角色为储户的客户端(命令行)
5. 设计并实现中央银行和客户端的交互

【二】实验过程及结果

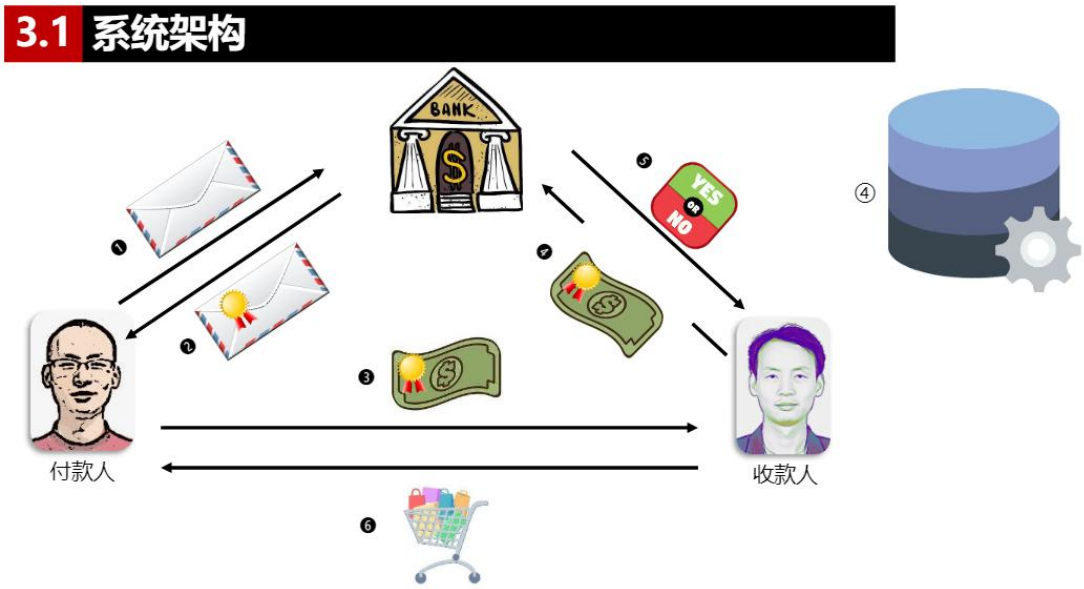
实验内容

一．RSA 盲签名

使用第三方库 cryptography 进行 rsa 密钥生成

https://cryptography.io/en/latest/hazmat/primitives/asymmetric/rsa/

二．架构



根据这个架构，我们设计了如下所述的交易系统，该系统旨在将现实货币交易转化为不可追溯的虚拟货币交易

三．关键角色：

中央银行

作为现实货币与虚拟货币之间的桥梁，用户可以在中央银行将现实货币转换为“签名机会”，在本系统中，每次签名机会相当于一枚 Coin。Coin 即为系统中的虚拟货币。

中央银行 并不直接分发 Coin，而是提供对 Coin 的签名与验证服务。用户通过将一定数量的现实货币兑换为签名机会，实现现实货币与虚拟货币之间的价值关联。

付款方

Coin 是由付款方生成的，生成后，付款方会对其进行盲化处理，然后将盲化后的 Coin 发送给 中央银行 进行签名。每次签名操作消耗一次签名机会。经过签名后的数据与其签名组合在一起，最终形成一枚有效的 Coin。

付款过程即是付款方通过任意方式将生成的 Coin 发送给收款方。

收款方

收款方在接收到付款方发送的 Coin 后，可以使用 中央银行 提供的公钥来验证货币的有效性。收款方还可以选择将验证通过的 Coin 直接发送回 中央银行 ，兑换为签名机会或现实货币。

这套系统通过盲化签名技术，确保了交易的不可追溯性，同时保持了虚拟货币与现实货币的价值联动。

四．Coin 实现及设计规范

pickle 序列化 Coin （model/coin.py） 类后的数据 Coin 货币、Coin 中的签名的传输过程中均为 base64 编码后的字符串

公钥的分发中传输的是 RSAPublicKey 对象 pickle 序列化 后经过 base64 编码的数据

五．功能

```
PS G:\dev\cryptocurrency\task1> python .\client\client_app.py -h
usage: Doin Client [-h] -bh BANK_HOST -bp BANK_PORT {generate,view,exchange,register} ...

Doin trading client

positional arguments:
  {generate,view,exchange,register}
    generate            生成coin
    view                查看coin信息
    exchange            交换货币为签名机会
    register            用户注册

options:
  -h, --help            show this help message and exit
  -bh BANK_HOST, --bank-host BANK_HOST
                        Bank 服务器地址
  -bp BANK_PORT, --bank-port BANK_PORT
                        Bank 服务器端口
PS G:\dev\cryptocurrency\task1>
```

用户注册：用户注册后拥有一定的签名次数唯一的 uid，有些功能的使用需要校验 uid

```
Coin 验证成功
PS G:\dev\cryptocurrency\task1> python .\client\client_app.py -bh localhost -bp 8008 register -p 123456
Namespace(bank_host='localhost', bank_port='8008', sub_command_name='register', password='123456')
账号注册成功: tVE35
PS G:\dev\cryptocurrency\task1>
```

申请签名：消耗签名次数给无签名的电子货币签名

```
PS G:\dev\cryptocurrency\task1> python .\client\client_app.py -bh localhost -bp 8008 generate -u NbE9B -p 123458
Namespace(bank_host='localhost', bank_port='8008', sub_command_name='generate', userid='NbE9B', password='123458', expiry_date=None, out_path=None)
{'userid': 'NbE9B', 'passwd': 'e6757959da8eff84c42d4df125b44eb40143dff452afd56aea5cfa058f245028'}
[-] 身份验证失败
PS G:\dev\cryptocurrency\task1> python .\client\client_app.py -bh localhost -bp 8008 generate -u NbE9B -p 123456
Namespace(bank_host='localhost', bank_port='8008', sub_command_name='generate', userid='NbE9B', password='123456', expiry_date=None, out_path=None)
{'userid': 'NbE9B', 'passwd': '8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92'}
coin 生成成功
```

查看货币：用户可以查看自己文件中电子货币的信息

```
PS G:\dev\cryptocurrency\task1> python .\client\client_app.py -bh localhost -bp 8008 view -c Coin_1727359133.7211185.coin
Namespace(bank_host='localhost', bank_port='8008', sub_command_name='view', coin='Coin_1727359133.7211185.coin')
UUID:0baf5ad7-93e8-4e58-b77f-f4772050e29c
过期时间:Never
签名Hash: 0ba9da707c1ac8655cc423b35f857f176b9298f08596969a6e26ea0e4fcbe08c
```

存储电子货币：用户可以用自己的电子货币与银行交换签名次数，本质上完成了对电子货币的存储

```
PS G:\dev\cryptocurrency\task1> python .\client\client_app.py -bh localhost -bp 8008 exchange -u NbE9B -p 123456 -c Coin_1727359416.741983.coin
Namespace(bank_host='localhost', bank_port='8008', sub_command_name='exchange', userid='NbE9B', password='123456', coin='Coin_1727359416.741983.coin')
Coin 验证成功
```

```
[-] 错误: 400
PS G:\dev\cryptocurrency\task1> python .\client\client_app.py -bh localhost -bp 8008 exchange -u NbE9B -p 123456 -c Coin_1727359133.7211185.coin
Namespace(bank_host='localhost', bank_port='8008', sub_command_name='exchange', userid='NbE9B', password='123456', coin='Coin_1727359133.7211185.coin')
[-] Coin 验证失败
```

六．核心代码

测试盲签名是否正常签名并能够验证签名

```python
import unittest
import sys
import os

sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), "..")))
from bank.utils import *
from client.utils import *
class TestRSABlindSignature(unittest.TestCase):
    """RSA 盲签名相关单元测试"""
    def test_sign(self):
        signature_machine = SignatureMachine()
        blind_device = BlindingDevice(signature_machine.get_pub_key())
        message = b"Test message for RSA signature"
        # 盲化
        blinded_message = blind_device.blind_message(message)
        # 对盲化消息签名
        signature_b = signature_machine.sign_message(blinded_message)
        # 签名去盲化
        signature = blind_device.unblind_signature(signature_b)
        # 签名验证
        self.assertTrue(signature_machine.verify_signature(message, signature))
        # 尝试验证一个伪造的签名（应当返回 False）
        forged_signature = signature[:-1] + bytes(
            [signature_b[-1] ^ 0x01]
        )  # 修改签名的最后一位
        self.assertFalse(signature_machine.verify_signature(message, forged_signature))
```

git 仓库地址：[Ly4hm/cryptocurrency (github.com)]