**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 10月 10日**

**【一】实验内容及要求**

**实验名称：**Chaum-Fiat-Naor 数字货币系统**实现**

**实验目的**

**设计基于**Chaum-Fiat-Naor 数字货币系统**的货币交易系统**

实验环境

· **操作系统**：Windows

· **开发工具**：Python

· **所用库**：

· Flask：用于创建Web服务器，模拟银行、付款人和收款人的交互。

· Crypto：用于实现RSA加密和签名。

· hashlib：用于生成哈希值。

· pickle：用于存储交易记录。

实验内容

1. **设计双花检测的核心代码**
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. **实现角色为双花者的客户端**
5. **实现角色为收款人的客户端**

**【二】实验过程及结果**

**实验内容**

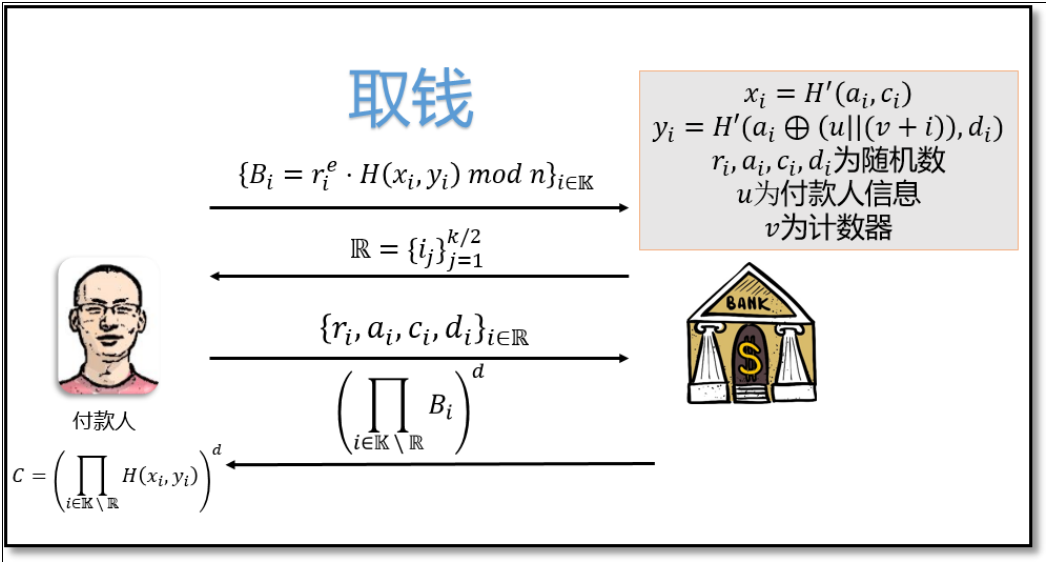
**一．双花检测**

存钱时，使用异或运算计算user名，与银行user表中存储的user字段进行比较

**二．架构**

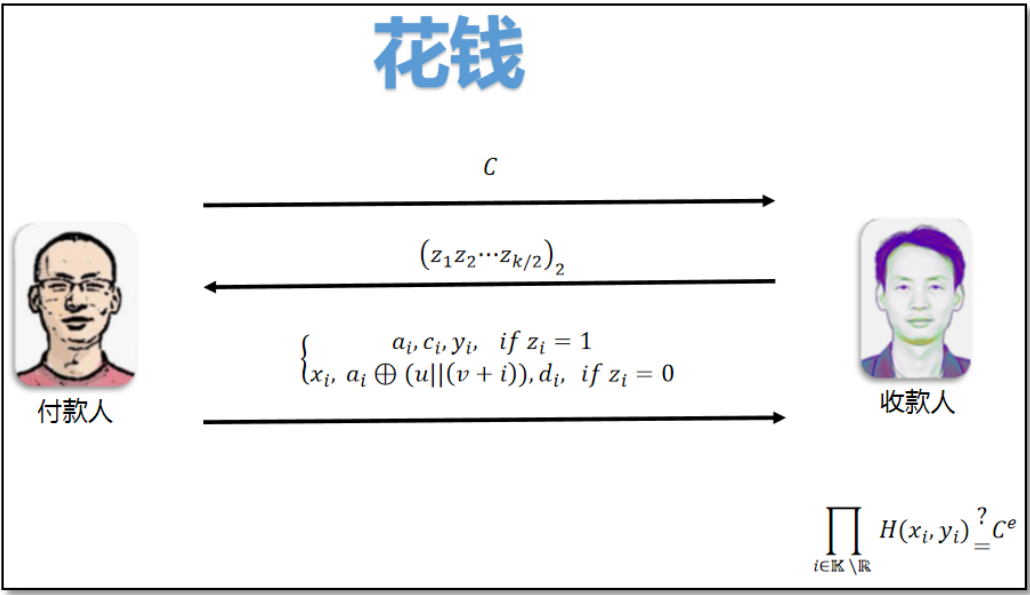
取钱环节：

付款人生成一个随机的电子货币，并对其进行盲化，然后将盲化后的货币发送给银行，银行签名后返回。



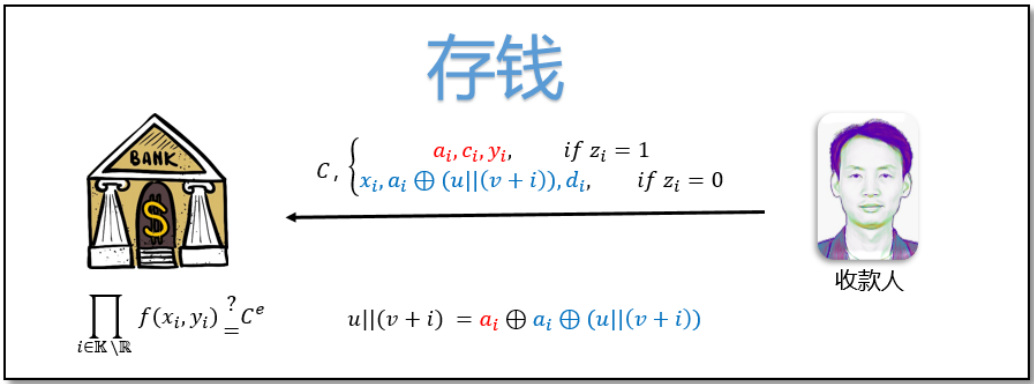
花钱环节：

付款人将经过银行签名的货币发送给收款人。收款人对货币进行验证，生成挑战并要求付款人进行响应。



存钱环节：

收款人将验证通过的货币发送给银行，银行检查是否有双花行为。



**三．关键角色：**

**中央银行**

中央银行在用户存钱阶段对电子货币进行双花检测，如果有用户进行了双花的行为，银行有最少1/4的概率根据第一次存储在银行的信息检测到双花者的用户名，由于Coin的基数较大，所以实际概率很高。

**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。签名时银行会验证付款方生成的Coin是否合法，即符合算法。

付款过程即是付款方通过客户端将签名后的 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，可以先使用公钥验证货币的有效性。收款方还可以选择将验证通过的 Coin 直接发送回 中央银行 ，验证Coin是否被双花了。

1. **Coin实现：**

Coin是一个由x，y组成的message，满足以下公式：

xi = H(a + c)

y = H(xor\_strings(a, uid\_v\_i) + d)

其中a,c,d为随机数，u为用户名，v为计数器

1. **核心代码**

电子货币的盲化签名过程（withdraw接口）

@app.route('/withdraw', methods=['POST'])

def withdraw():

    global e, n

    # 从银行获取公钥

    if not e or not n:

        response = requests.get(bank\_url + "/public\_key")

        e, n = response.json()['e'], response.json()['n']

    for i in range(k):

        r = random.randint(1, n - 1)

        a = random\_string()

        c = random\_string()

        d = random\_string()

        xi = H(a + c)

        uid\_v\_i = u + str(v + i)  # u || (v + i)

        y = H(xor\_strings(a, uid\_v\_i) + d)

        H\_xy = H(xi + y)

        S\_blind\_value = (pow(r, e, n) \* int(H\_xy, 16)) % n

        r\_i.append(r)

        a\_i.append(a)

        c\_i.append(c)

        d\_i.append(d)

        x\_i.append(xi)

        y\_i.append(y)

        S\_blind.append(S\_blind\_value)

    # 发送盲化签名请求给银行

    response = requests.post(bank\_url + "/select\_indices", json={'k': k})

    checked\_indices = response.json()['indices']

    # 返回对应的 ri, ai, ci, di

    revealed\_info = []

    for i in checked\_indices:

        info = [r\_i[i], a\_i[i], c\_i[i], d\_i[i], x\_i[i], y\_i[i], S\_blind[i]]

        revealed\_info.append(info)

    # 发送验证信息并获取签名消息

    unchecked\_indices = [i for i in range(k) if i not in checked\_indices]

    unchecked\_blinded\_messages = [S\_blind[i] for i in unchecked\_indices]

    response = requests.post(bank\_url + "/verify\_and\_sign", json={

        'revealed\_info': revealed\_info,

        'blinded\_messages': unchecked\_blinded\_messages

    })

    if response.status\_code != 200:

        return jsonify({'status': 'failed', 'error': response.json().get('error')}), 400

    signed\_messages = response.json()['signed\_messages']

    # 去盲化签名

    for idx, S\_signed in zip(unchecked\_indices, signed\_messages):

        S\_unblinded = (S\_signed \* pow(r\_i[idx], -1, n)) % n

        S.append(S\_unblinded)

    return jsonify({'status': 'withdraw successful', 'S': S})

恢复用户身份（recover\_u\_from\_data函数）

def recover\_u\_from\_data(ai\_exp\_hex, ai):

    ai\_bytes = bytes.fromhex(ai\_exp\_hex) if is\_hex\_string(ai\_exp\_hex) else ai\_exp\_hex.encode('utf-8')

    ai\_exp\_bytes = bytes.fromhex(ai) if is\_hex\_string(ai) else ai.encode('utf-8')

    # 异或恢复 u\_v\_i

    u\_v\_i\_bytes = bytes(a ^ b for a, b in zip(ai\_exp\_bytes, ai\_bytes))

    # 提取 u（假设 u 的长度为 7）

    u\_length = 7

    u\_bytes = u\_v\_i\_bytes[:u\_length]

    return u\_bytes.decode('utf-8', errors='ignore')

单元测试：

import requests

def test\_withdraw():

    response = requests.post('http://localhost:5001/withdraw')

    if response.status\_code == 200:

        print("Withdraw test successful:", response.json())

    else:

        print("Withdraw test failed:", response.json())

def test\_spend():

    spend\_data = {

        'payee\_url': 'http://localhost:5002'

    }

    response = requests.post('http://localhost:5001/spend', json=spend\_data)

    if response.status\_code == 200:

        print("Spend test successful:", response.json())

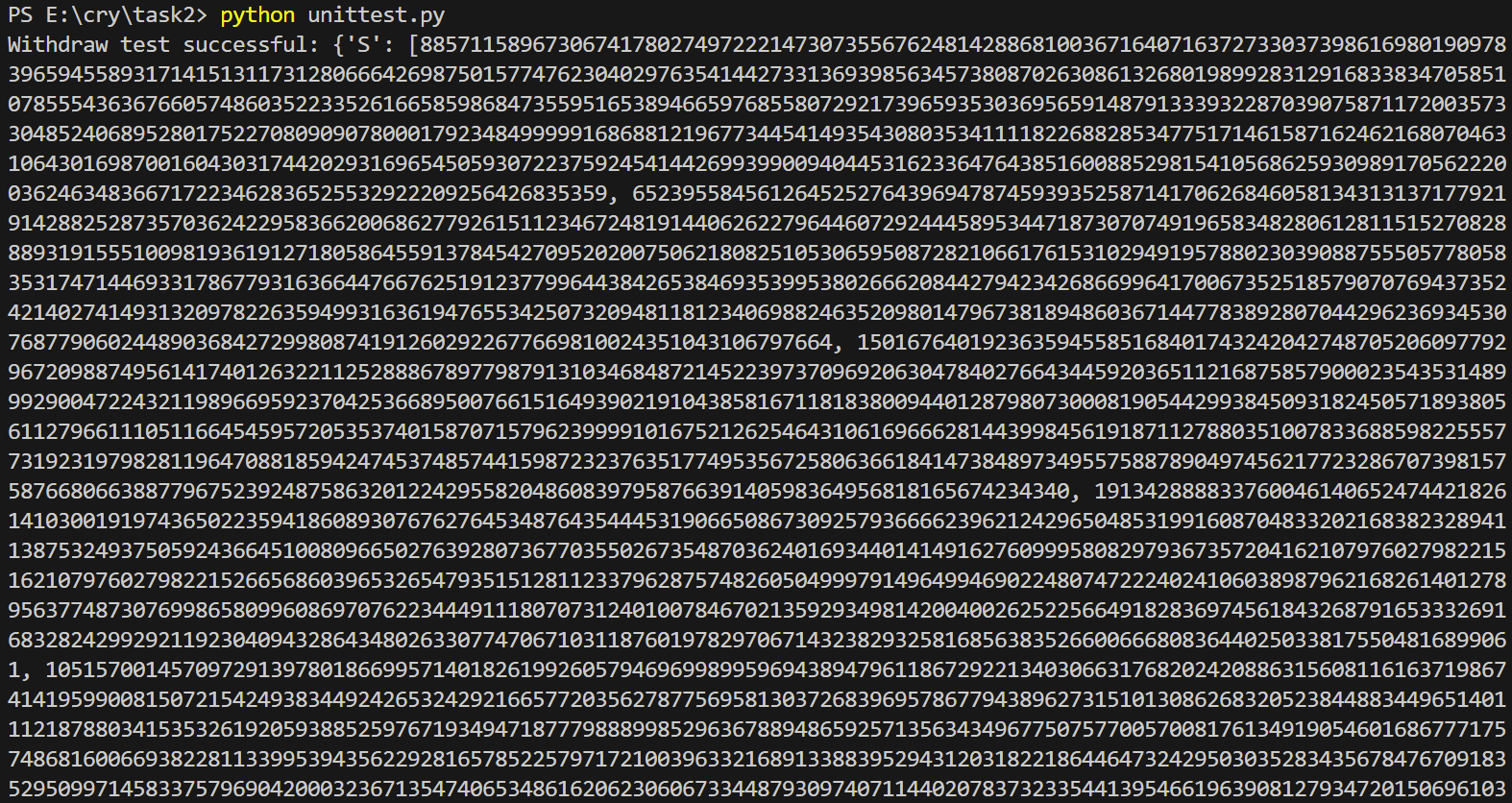
    else:

        print("Spend test failed:", response.json())

test\_withdraw()

test\_spend()

函数test\_withdraw()模拟付款人向银行申请对Coin进行签名的过程，成功则显示签名成功的货币：



函数test\_spend则测试消费功能，如果成功，显示：



同时收款人将Coin存入银行，银行检测是否有双花行为，如有，显示：



双花检测：

**双花检测**是本方案的关键部分。为了检测同一电子货币是否被重复使用，银行会检查相同的xi值下是否存在不同的zi值。简单来说，如果对于同一个xi，两次交易的zi值不同，则可以判定为双花行为。

@app.route('/verify\_transaction', methods=['POST'])

def verify\_transaction():

    transactions = request.json['transactions']

    double\_spending\_detected = False

    u\_value = None

    for transaction in transactions:

        zi = transaction['zi']

        data = transaction['data']

        if zi == 0:

            #对每个数据异或

            for \_,data\_stored in transaction\_store:

                u\_value = recover\_u\_from\_data(data[1], data\_stored[0])

                #查找是否有双花者

                if u\_value in users:

                    double\_spending\_detected = True

                    break

        else:

            for \_,data\_stored in transaction\_store:

                u\_value = recover\_u\_from\_data(data\_stored[1], data[0])

                if u\_value in users:

                    double\_spending\_detected = True

                    break

    #如有双花产生错误

    if double\_spending\_detected:

        return jsonify({'status': 'failed', 'error': 'Double spending detected', 'payer\_identity': u\_value}), 400

    #确认无双花之后进行存储

    for transaction in transactions:

        zi = transaction['zi']

        data = transaction['data']

        transaction\_store.append((zi, data))

    #if transactions["store"]:

        #pickle.dump(transaction\_store, open("transaction\_store.pickle", "wb"))

    return jsonify({'status': 'success'})

代码逻辑请见注释

源码地址：[Ly4hm/cryptocurrency (github.com)](https://github.com/Ly4hm/cryptocurrency)