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

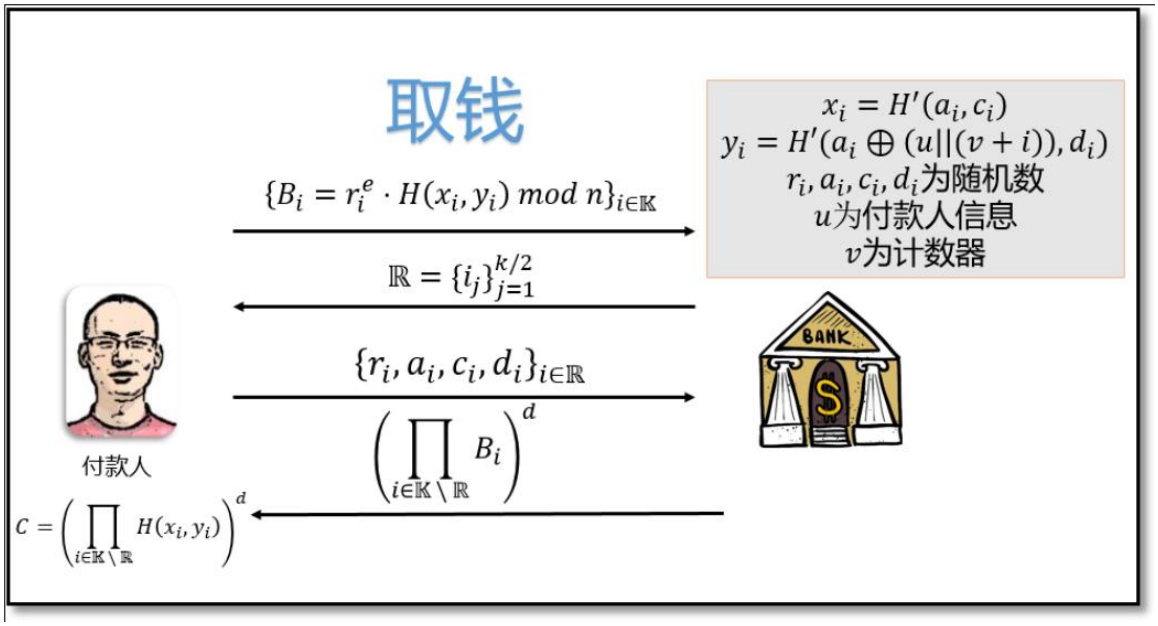
一．双花检测

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

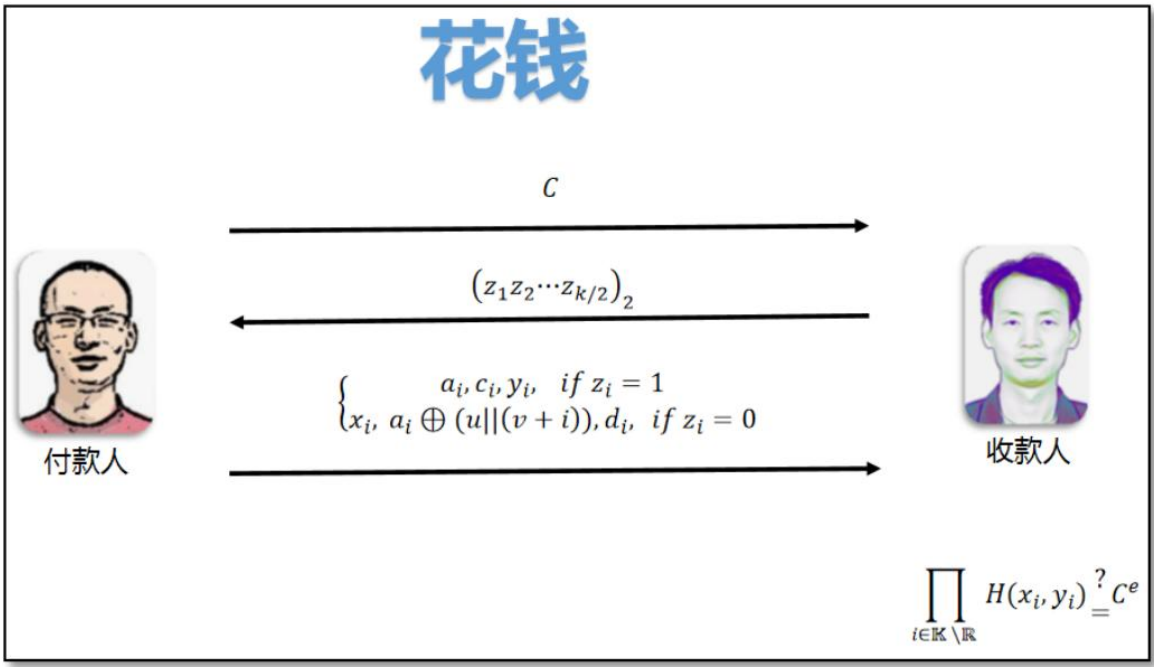
取钱环节：

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。



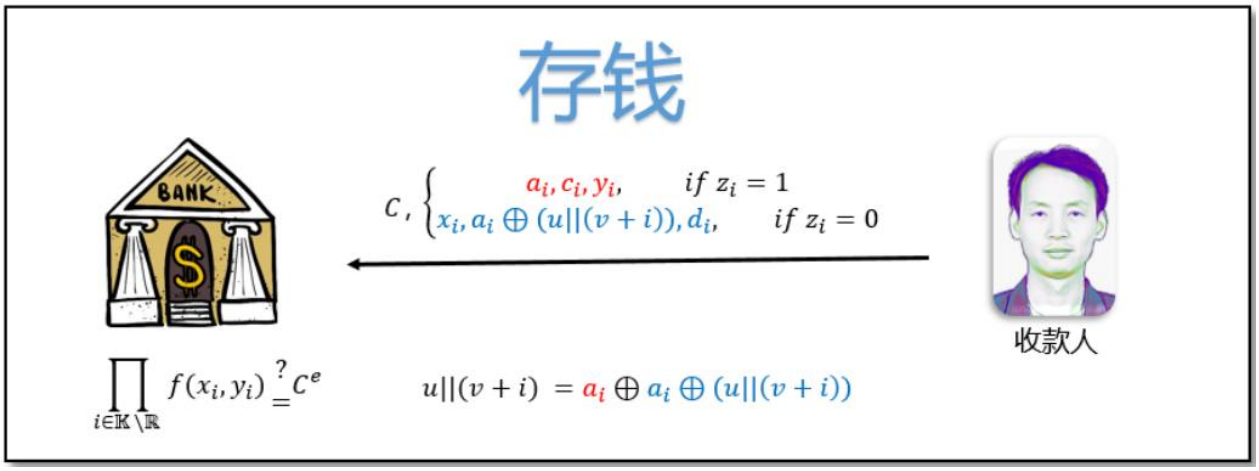
花钱环节：

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。

四. 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

五. 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：

```
PS E:\cry\task2> python unittest.py
Withdraw test successful: {'S': [885711589673067417802749722214730735567624814288681003671640716372733037398616980190978
396594558931714151311731280666426987501577476230402976354144273313693985634573808702630861326801989928312916833834705851
078555436367660574860352233526166585986847355951653894665976855807292173965935303695659148791333932287039075871172003573
304852406895280175227080909078000179234849999916868812196773445414935430803534111182268828534775171461587162462168070463
106430169870016043031744202931696545059307223759245414426993990094044531623364764385160088529815410568625930989170562220
036246348366717223462836525532922209256426835359, 652395584561264525276439647874593935258714170626846058134313137177921
914288252873570362422958366200686277926151123467248191440626227964460729244458953447187307074919658348280612811515270828
889319155510098193619127180586455913784542709520200750621808251053065950872821066176153102949195788023039088755505778058
353174714469331786779316366447667625191237799644384265384693539953802666208442794234268669964170067352518579070769437352
421402741493132097822635949931636194765534250732094811812340698824635209801479673818948603671447783892807044296236934530
768779060244890368427299808741912602922677669810024351043106797664, 150167640192363594558516840174324204274870520609779296
967209887495614174012632211252888678977987913103468487214522397370969206304784027664344592036511216875857900023543531489
992900472243211989669592370425366895007661516493902191043858167118183800944012879807300081905442993845093182450571893805
611279661110511664545957205353740158707157962399991016752126254643106169666281443998456191871127880351007833688598225557
731923197982811964708818594247453748574415987232376351774953567258063661841473848973495575887890497456217723286707398157
587668066388779675239248758632012242955820486083979587663914059836495681816567423434O, 191342888833760046140652474421826
141030019197436502235941860893076762764534876435444531906650867309257936666239621242965048531991608704833202168382328941
138753249375059243664510080966502763928073677035502673548703624016934401414916276099958082979367357204162107976027982215
162107976027982215266568603965326547935151281123379628757482605049997914964994690224807472224024106038987962168261401278
956377487307699865809960869707622344491118070731240100784670213592934981420040026252256649182836974561843268791653332691
683282429929211923040943286434802633077470671031187601978297067143238293258168563835266006668083644025033817550481689906
1, 105157001457097291397801866995714018261992605794696998995969438947961186729221340306631768202420886315608116163719867
414195990081507215424938344924265324292166577203562787756958130372683969578677943896273151013086268320523844883449651401
112187880341535326192059388525976719349471877798889985296367889486592571356343496775075770057008176134919054601686777175
748681600669382281133995394356229281657852257971721003963321689133883952943120318221864464734249505030352834356784767091833
529509971458337579690420003236713547406534861620623060673344879309740711440207837323354413954661963908127934720150696103
```

函数 test_spend 则测试消费功能，如果成功，显示：

```
Spend test successful: {'status': 'payment accepted'}
```

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：

```
Spend test successful: {'error': 'Double spending detected', 'status': 'payment failed'}
```

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)