SD04630230: E-Cash and Blockchain **Dec 5, 2024**

实验 3: 中心化数字货币与 Dolev-Strong 共识协议

Lecturer: 钱宸 (Chen Qian)

3.1 实验说明

本次实验基于 Python 以及 ecc-pycrypto 库来实现。其中我们将使用 ecc-pycrypto 库中的 P256 椭圆曲线的一个简单实现。实验说明文档和所需文件在课程主页`https://qianchen92.github.io/teaching/ecash` 中下载。

3.2 中心化数字货币

在课堂中，我们接触到了 Chaum 的中心化数字货币，主要基于 RSA 盲签名系统与抗双支付攻击的序列号两部分组成。那么我们来编写一下具体的货币。

问题 1: 基于 RSA 假设，实现盲签名系统。

提示:。盲签名算法由以下三个不同的函数组成:

- $\mathsf{KeyGen}() \to (vk, sk, ek)$: 生成一对签名密钥和一个盲化密钥
- $\mathsf{Blind}(ek, m) \to \bar{m}$: 算出盲化的信息
- $\mathsf{Sig}(sk, \bar{m}) \to \bar{\sigma}$: 计算盲化后的签名
- $\mathsf{UnBlind}(ek, \bar{\sigma}) \to \sigma$: 对盲化后的签名进行脱盲

问题 2: 基于离散对数假设设计抗双支付攻击的序列号。

函数形式如下:

- $\mathsf{f_{SN}}(pk_a, pk_b, sk_a) \to C_{a,b}$: 计算抗双支付攻击的序列号

问题 3: 假设 Alice 分别将同样的货币支付给 Bob 和 Charlie，在银行收到两个序列号 $C_{a,b}$ 和 $C_{a,c}$ 后，请找出 Alice 的公钥。

函数形式如下:

- $\mathsf{DS}(C_{a,b}, C_{a,c}, pk_b, pk_c) \to pk_a$: 计算出双支付作弊者的公钥。

3.3 Dolve-Strong 协议

问题 4: 利用 python 的多进程编程，模拟多个进程之间实现 Dolev-Strong 共识协议。

提示:

- 我们主要会用到 Python 中的多进程模块 “import multiprocessing as mp”
- 可以使用 Barrier 来进行多进程之间的同步
- 进程之间的消息传递可以通过共享数组方式实现（模拟公开的广播信道）
- Python 关于多进程编程的说明请参见:
  `https://docs.python.org/zh-cn/3/library/multiprocessing.html`
- 使用 RSA 签名算法来对每一个消息进行签名

问题 5: 基于 Dolev-Strong 协议时，我们知道要保证协议的安全，当存在 k 个恶意节点时，需要进行至少 $k+2$ 轮。编程给出 $k+1$ 时候产生的攻击。