

## 实验 2: 数字签名与加密

Lecturer: 钱宸 (Chen Qian)

**免责声明:** 该实验材料仅用于山东大学网络空间安全学院课程教学, 尚未经过通常用于正式出版物的审查。仅在获得讲师的许可的情况下, 可以在课堂外部分发。

### 2.1 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/ecash2024.html> 中下载。

### 2.2 数字签名算法

在课堂中, 我们学习了应用陷门单向函数构造签名算法的过程, 那么我们先基于 RSA 和 DDH 假设来实现陷门单向函数吧!

Warm up!

**问题 1:** 基于 RSA 假设, 实现陷门单向函数。

**提示:**。陷门单向函数由以下三个不同的函数组成:

- $\text{TrapGen}() \rightarrow (K, td)$ : 生成计算密钥与陷门
- $\text{Eval}(K, x) \rightarrow y$ : 利用计算密钥和信息生成单向函数输出
- $\text{Invert}(td, y) \rightarrow x$ : 根据单向函数输出以及陷门, 计算信息

问题 2 将分为三个小题:

**问题 2:** 基于 DDH 假设, 实现陷门单向函数。

我们知道在离散对数假设中, 生成一个随机的  $x \in \mathcal{X}$ , 根据  $g^x$  计算  $x$  是一件很困难的事情。然而如果定义域  $\mathcal{X}$  足够小, 离散对数问题实际是简单的。

**问题 2.1:** 假设  $\mathcal{X} = \{0, 1, \dots, 2^{15} - 1\}$ , 计算离散对数问题。

函数形式:

- $\text{BitInvert}(y) \rightarrow x$  计算  $x \in \mathcal{X}$  使得  $g^x = y$ 。

令  $n = \lceil \log q/15 \rceil$ , 其中  $q$  为群  $G$  的阶。已知给定任意一个  $n \times n$  的可逆矩阵  $M \in \mathbb{Z}_q^{n \times n}$  以及一个随机的向量  $\vec{x} \in \{0, \dots, 2^{15} - 1\}^n$ 。在 DDH 假设下  $f_{g^M}(\vec{x}) = g^{M\vec{x}}$  为单向函数 [1]。

**说明:**

```
def createMatrix(rows,cols,x):
    arr=[]
    for i in range(rows):
        col = [x for _ in range(cols)]
        arr.append(col)
    return arr
```

Figure 2.1: 生成矩阵的函数

- 这里 15 可以为小于  $\log q$  的任意整数，这里为了是的矩阵算法与 BitInvert 同时高效，所以选取 15。
- Python 中的矩阵生成稍微有些复杂，可用 fig. 2.1 中的算法生成初始值为  $x$  的矩阵。
- 矩阵的求逆是在  $\text{mod } n$  中计算，可以使用 sympy 库中 `M.inv_mod(n)` 来实现。

**问题 2.2:** 编写单向函数  $f_{g^M}(\bar{x})$ 。

函数形式为：

- $\text{Eval}(K, x) \rightarrow y$

问题 2.2 中的单向函数实际是一个单向陷门函数，其陷门为  $M$ 。

**问题 2.3:** 为问题 2.2 中的函数添加陷门生成与求逆运算。

函数形式为：

- $\text{TrapGen}() \rightarrow (K, td)$
- $\text{Invert}(td, y) \rightarrow x$

**问题 3:** 利用上述编写的 RSA 陷门单向函数，构造签名算法。并验证算法正确性。

## 2.3 加密算法

首先我们先实现 El-Gammal 加密算法。

**问题 4:** 基于 DDH 假设，实现 El-Gammal 加密算法加密一个群元素。

函数形式为：

- $\text{KGen}() \rightarrow \text{pk}, \text{sk}$
- $\text{Enc}(\text{pk}, m; r) \rightarrow \text{ct}$
- $\text{Dec}(\text{sk}, \text{ct}) \rightarrow m$

我们课上介绍了 CCA1 安全的加密算法, Naor-Yung 加密。就是将同样的信息加密两遍, 并且用零知识证明证明加密的是相同的信息。其中 [2] 证明了对于 El-Gammal 加密而言, 如果两遍加密使用同样的随机数仍然是 CCA1 安全的。使用这种特殊加密的转化称为 twisted Naor-Yung。

**问题 5:** 利用 El-Gammal 实现 CCA1 安全的加密算法。

**提示:**

- 两个随机数相同的 El-Gammal 加密  $ct_1 = (g^a, g^r, g^{ar} \cdot m)$ ,  $ct_2 = (g^b, g^r, g^{br} \cdot m)$  的明文相同等价于将两者相除以后得到的  $(g, g^{(a-b)}, g^r, g^{(a-b)r})$  为一个 DDH 对。
- 证明  $(g, f, G, F)$  为一个 DDH 对, 可分别证明知道  $\log_g G$  和  $\log_f F$ , 并且在证明的过程中使用同样的挑战, 且最终的应答相同。即最终证明形式为  $(R, h_1, h_2, s_1, s_2)$  其中  $h = H(g, f, G, F, R_1, R_2)$ 。

(以下为附加题, 非必做可选做)

**问题 6:** 结合问题 3 与问题 5 编写 CCA2 加密安全的加密算法。

**提示:** 仍然使用所有加密公用随机数的思路。

## References

- [1] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, Gil Segev, **More Constructions of Lossy and Correlation-Secure Trapdoor Functions**, Journal of Cryptology 2013
- [2] Silvio Biagioni, Daniel Masny, Daniele Venturi, **Naor-Yung Paradigm with Shared Randomness and Applications**, SCN 2016