


# 《现代密码学》实验报告

实验名称：DSA签名方案	实验时间：2022.12.20
学生姓名：伍建霖	学号：20337251
学生班级：20网安	成绩评定：

## 一、实验目的

1. 了解数字签名的基本原理，掌握数字签名算法DSA的实现方法
2. 实现DSA算法并验证

## 二、实验内容



### DSA 实现

- 实验要求：  
实现数字签名算法DSA（课本p.232），对消息签名并给出验证。
- 实验参数  
p: 2048比特素数  
q: 可被p-1整除的256比特素数  
Hash函数: SHA-256（复用第四次实验的实现）  
消息: Sysu+学号+名字全拼+2022（如 Sysu12345678zhangsan2022），将其使用ASCII或者UTF-8编码成字节串。  
私钥: 自己选取随机数作为私钥，并生成相应的公钥  
大数计算可以复用第5次实验RSA的代码，如果有困难可以用大数库计算。

## 三、实验原理

### DSA

已有变量为  $p, q, a, \alpha$ ，其中  $\alpha$  为私钥且在 1 到  $q-1$  之间。而通过  $\beta = \alpha^a \bmod p$  可以算出  $\beta$ 。此时得到了完整的密钥，其中  $p, q, \alpha, \beta$  为公钥， $a$  为私钥。

签名函数：随机选择一个在 1 到  $q-1$  之间的数作为随机数  $k$ ，然后对消息  $x$  进行签名：  
 $sig(x, k) = (\gamma, \delta)$ ，其中

$$\begin{aligned}\gamma &= (\alpha^k \bmod p) \bmod q, \\ \delta &= (Hash(x) + a\gamma)k^{-1} \bmod q,\end{aligned}$$

若  $\gamma$  or  $\delta$  为 0 则重新选择随机数签名。

验证函数：计算  $e_1 = Hash(x)\delta^{-1} \bmod q$  和  $e_2 = \gamma\delta^{-1} \bmod q$ ，当  $(\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q$  和  $\gamma$  相等时通过验证。

## 四、实验步骤

## 初始化参数

为方便使用，先将参数初始化为 mpz\_class 类型并赋值：

```
7 // parameter
8 mpz_class p("274563234213387494081496193530151333423409884177097320767346954066024724194421");
9 mpz_class q("104157279931481431886985012477114223269161211000653816000021509176830313880343");
10 mpz_class alpha("17074740003955525400395323583439413391693112243303604720053587149934685382");
11 mpz_class a("20337251");
12 mpz_class beta;
13 mpz_class k("2022");
14 // input
15 mpz_class hashMes("95f0fcab4a4823de89deac3045f783e6efebcbdede44d5441e76de3b1531dd9", 16);
16 // output
17 mpz_class gama;
18 mpz_class delta;
```

## 计算 $\beta$

已有  $\alpha$ ，通过  $\beta = \alpha^a \bmod p$  计算  $\beta$  的值

```
// beta = alpha^a mod p
mpz_powm(beta.get_mpz_t(), alpha.get_mpz_t(), a.get_mpz_t(), p.get_mpz_t());
cout << "beta is " << beta << '\n';
```

## 签名

随机选择一个在 1 到  $q-1$  之间的数作为随机数  $k$ ，然后对消息  $x$  进行签名： $\text{sig}(x, k) = (\gamma, \delta)$ ，其中

$$\gamma = (\alpha^k \bmod p) \bmod q,$$

$$\delta = (\text{Hash}(x) + a\gamma)k^{-1} \bmod q,$$

若  $\gamma$  or  $\delta$  为 0 则重新选择随机数签名。

这里我确认  $\gamma$  和  $\delta$  都不为 0 后，定死了  $k$  的值，方便调试。

```
void sig()
{
    // gama = (alpha^k mod p) mod q
    mpz_powm(gama.get_mpz_t(), alpha.get_mpz_t(), k.get_mpz_t(), p.get_mpz_t());
    mpz_mod(gama.get_mpz_t(), gama.get_mpz_t(), q.get_mpz_t());
    cout << "gamma is " << gama << '\n';

    // delta = (hashMes + a*gama)(k ^ -1) mod q
    mpz_class k_invert;
    mpz_invert(k_invert.get_mpz_t(), k.get_mpz_t(), q.get_mpz_t());
    mpz_mod(delta.get_mpz_t(), mpz_class((hashMes + a * gama) *
k_invert).get_mpz_t(), q.get_mpz_t());
    cout << "delta is " << delta << '\n';
}
```

## 验证

计算  $e_1 = \text{Hash}(x)\delta^{-1} \bmod q$  和  $e_2 = \gamma\delta^{-1} \bmod q$ ，当  $(\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q$  和  $\gamma$  相等时通过验证。

```
void ver()
```

```

{
    mpz_class e1, e2, delta_invert, ret, alpha_e1, beta_e2;
    mpz_invert(delta_invert.get_mpz_t(), delta.get_mpz_t(), q.get_mpz_t());
    cout << "delta_invert is " << delta_invert << '\n';

    // e1 = (hashMes * (delta ^ -1)) mod q
    mpz_mod(e1.get_mpz_t(), mpz_class(hashMes * delta_invert).get_mpz_t(),
q.get_mpz_t());
    cout << "e1 is " << e1 << '\n';

    // e2 = (gama * (delta ^ -1)) mod q
    mpz_mod(e2.get_mpz_t(), mpz_class(gama * delta_invert).get_mpz_t(),
q.get_mpz_t());
    cout << "e2 is " << e2 << '\n';

    // alpha_e1 = (alpha ^ e1) mod p
    mpz_powm(alpha_e1.get_mpz_t(), alpha.get_mpz_t(), e1.get_mpz_t(),
p.get_mpz_t());
    // beta_e2 = (beta ^ e2) mod p
    mpz_powm(beta_e2.get_mpz_t(), beta.get_mpz_t(), e2.get_mpz_t(),
p.get_mpz_t());
    // ret = ((alpha ^ e1) * (beta ^ e2) mod p) mod q
    mpz_mod(ret.get_mpz_t(), mpz_class(alpha_e1 * beta_e2).get_mpz_t(),
p.get_mpz_t());
    mpz_mod(ret.get_mpz_t(), ret.get_mpz_t(), q.get_mpz_t());
    cout << "ret is " << ret << '\n';

    if (ret == gama)
        cout << "'it is authentic signature!\n";
    else
        cout << "we are wrong qwq\n";
}

```

## 五、实验结果

这里我用实验测试网站生成了参数  $p$ ,  $q$ ,  $\alpha$ , 并把学号20337251作为私钥 $a$ , 2022作为固定的随机数 $k$ . 先生成  $\beta$ , 然后调用签名函数生成  $\gamma$  和  $\delta$ , 最后调用验证函数验证. 结果证明我们的签名算法是正确的.

问题	输出	调试控制台	终端
			<pre> beta is 1799967511981005188714918968415328253616498064017870829218686517177125345545703514934960343934427 525263569449878707490139234379435985969285320025628098125135238933347594426208122371850687188768782294987 990296499173880040100479296465734523522527395099822183869614282078883726412470094186903012857398173595312 632026974019733742955163656674576828657130293916458415961161702289257184359955550103131839145960509953659 774978993182573592094685399963048818608921489725900108321388769918680534593517947187058028956997934407038 234211396347033585254111248799409648360922383252587112891821032884874079360314586853384099006789116 gamma is 100037218867600172379540746576291372247290600849326562027423523436422725138183 delta is 94590742035079503746509016985597378844526539616821931723918648027979123857353 delta_invert is 9137859346589835890440253392229176130121073701655546929578140643354257182679 e1 is 59806913661930960942259091992370614725936638488193470994388519531127248417427 e2 is 62945115615002653615202366812464404700330442576520255718345874827480225279011 ret is 100037218867600172379540746576291372247290600849326562027423523436422725138183 'it is authentic signature! [1] + Done                               "/usr/bin/gdb" --interpreter=mi --tty=\${DbgTerm} 0&lt;"/tmp/Microsoft-MIEng ine-In-mcglpugo.w3v" 1&gt;"/tmp/Microsoft-MIEngine-Out-c5jq45t3.0kg" henry@DESKTOP-3NA4DUP:~/dsa\$ </pre>

## 六、实验总结

这次的实验直接按照书上说的实现就可以，没有难度，很快就打出来了，并且通过这次实验对于DSA也有了更加深刻的理解和记忆。比安装库好解决。DSA与之前不一样的是，在Alice这里，签名是私有的，所以需要私钥，而验证是谁都可以的，所以不需要私钥。之前是双方都需要私钥进行加解密，所以私钥需要在安全的信道中传输。签名方案中也有很多脑洞很大的算法，总而言之都是共同算出相同的变量的话验证就成功了。