

# 个人信息

---

学号：1911410

姓名：付文轩

专业：信息安全

## 1. 实验目的

---

通过对数字签字算法DSA的实际操作，理解DSS的基本工作原理。

## 2. 实验原理

---

数字签字目前采用较多的是非对称加密技术，其实现原理简单的说，就是由发送方利用杂凑函数对要传送的信息进行计算得到一个固定位数的消息摘要值，用发送者的私钥加密此消息的杂凑值所产生的密文即数字签字。然后将数字签字和消息一同发送给接收方。接收方收到消息和数字签字后，用同样的杂凑函数对消息进行计算得到新的杂凑值，然后用发送者的公开密钥对数字签字解密，将解密后的结果与自己计算得到的杂凑值相比较，如相等则说明消息确实来自发送方。

## 3. 实验内容和步骤

---

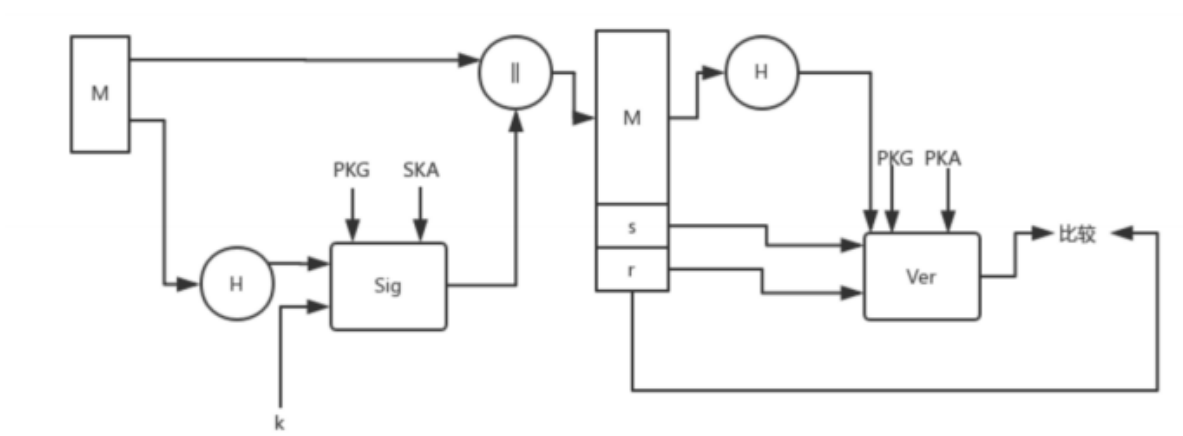
- (1) 参照教材，熟悉数字签字算法DSA；
- (2) 参照教材，熟悉杂凑函数算法SHA；
- (3) 这里给出一个可运行的DSA数字签字演示程序，运行这个程序，对一段文字进行签字和验证，了解DSA算法的签字和验证过程。

## 4. 数字签名算法DSA

---

### DSS算法基本框图

---



## DSA算法描述

### 公钥

p	512~1024位的素数，长度为L bits，且L是64的倍数
q	160位长，且为p-1的素因子
g	$g \equiv h^{(p-1)/q} \pmod{p}$ ，其中h是满足 $1 < h < p-1$ 且使 $h^{(p-1)/q} \pmod{p} > 1$ 的任一整数
y	$y = g^x \pmod{p}$ （一个p位的数）

其中：(p,q,g)为全局公开钥，y为用户公开钥

### 私钥

x	x是随机数或者伪随机数， $x < q$

### 签名过程

选取一个随机数k， $k < q$
$r(\text{签名}) = (g^k \pmod{p}) \pmod{q}$
$s(\text{签名}) = [k^{-1}(H(m) + xr)] \pmod{q}$

最终形成的签字为(r,s)

### 验证过程

$w = s^{-1} \bmod q$
$u_1 = (H(m) * w) \bmod q$
$u_2 = (rw) \bmod q$
$v = ((g^{u_1} * y^{u_2}) \bmod p) \bmod q$
if $v=r$ , accept

### 验收过程证明

若接收方收到的 $(M', r', s')$ 和发送方发送的 $(M, r, s)$ 相等，则有：

$$M = M'; r = r'; s = s'$$

那么此时就有： $v = [(g^{H(M)w} g^{xrw}) \bmod p] \bmod q = [g^{(H(M)+xr)s^{-1}} \bmod p] \bmod q = (g^k \bmod p) \bmod q = r$

### 安全性依赖

DSA依赖的是离散对数的难解性问题，敌手仅从  $r$  恢复  $k$  或者从  $s$  恢复  $x$  都是不可行的；但是需要注意的是512位的DSA不能提供长期的安全性，但是1024位可以。

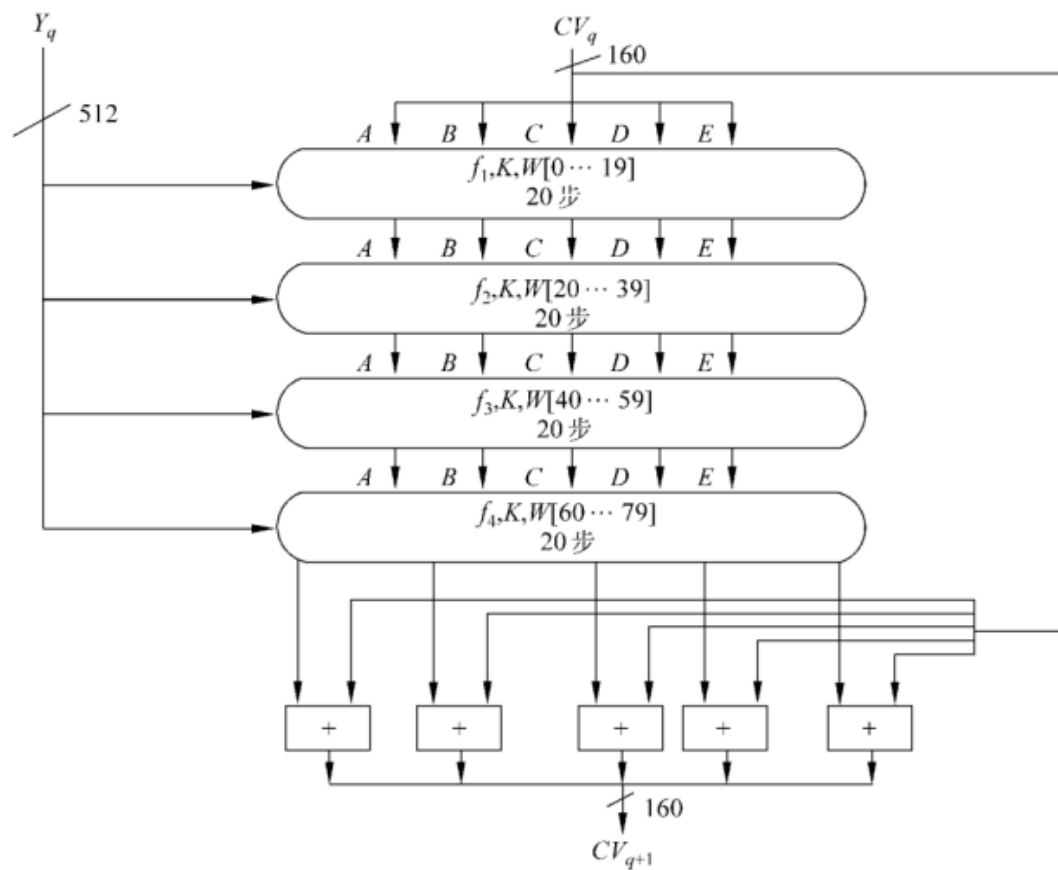
## 5. 杂凑算法SHA

SHA算法实际上是一系列的算法，有SHA-0、SHA-1、SHA-2和SHA-3，目前SHA-0和SHA-1已经被发现存在有一些缺陷，所以目前使用较多的是SHA-2和SHA-3，其中：

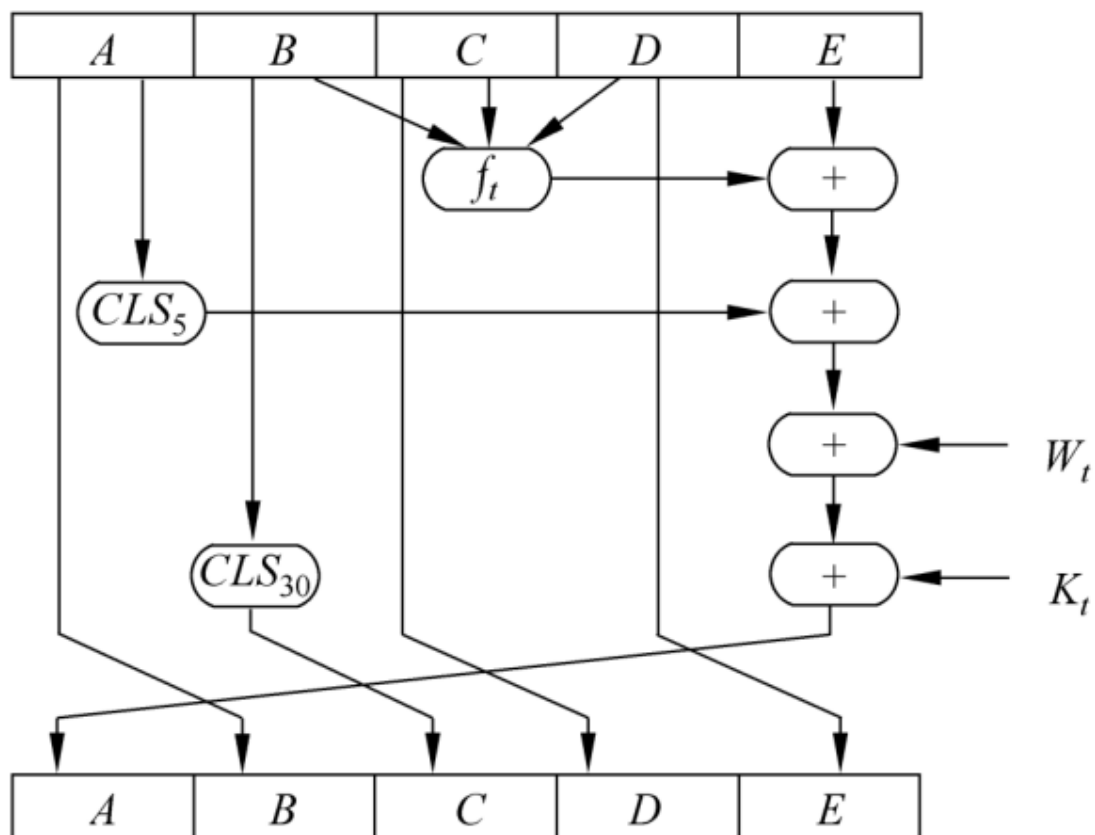
SHA-2是一系列SHA算法变体的总称，包含有如下子版本：

- SHA-256：生成长度为256bit的信息摘要
- SHA-224：生成长度为224bit的信息摘要，为SHA-256的“阉割版”
- SHA-512：生成长度为512bit的信息摘要
- SHA-384：生成长度为384bit的信息摘要，为SHA-512的“阉割版”

### SHA的分组处理框图



同样的，和MD5类似，SHA也有压缩函数，其中一次分组处理的一步迭代流程图为：




## 6. 实验过程

# 初始化全局变量

获取DSA签名算法的全局公开钥p, q, g, 并生成一个随机的用户密钥x和用户公开钥y

使用工具生成一次验证的参数

p	E0797F891A3A5DA9B9B5B43EA24C0EFCA5C1684E062E62DABE01113E9515B8CEA2544962ECB0F1DE104A9215862472D68189C4506D301056FBC54BE49B162B5
q	AACD53C264232ABA9E83220C43D4B2FDD1AB65AF
g	C7E6AD5EAF0F2D56279B7C27F95C8A2438793FEB04AF5A4AA4C4DDEA17F99C1D59A57B73E0F31CFE2FC0A9012A7EE0704D26C585E2813AFDF7115D7C4CD86180
y	769B3B9B1706033F5B550F068EEAC972F85F0494B4C8CF0160D76CF18D507EAA608497130EAB80A30F2DC7907AF54D036058681A63C529772675AD08998EAE75
x	3851C4F16E0B1D6B4B624A04D52399F7BA4912F0

 v1.3

Size of P (Bits)  
512

Number Base  
16

Random seed generation

Start

7d08f1a5

100%

P (public) = Prime in range 512-1024 Bits - 64 Bit  
E0797F891A3A5DA9B9B5B43EA24C0EFCA5C1684E062E62DABE01113E9515B8CEA2544962ECB0F1DE104A9215862472D68189C4506D301056FBC54BE49B162B5

Q (public) = 160 Bit prime factor of P-1  
AACD53C264232ABA9E83220C43D4B2FDD1AB65AF

G (public) = H^(P-1)/Q with H < (P-1) and H^(p-1)/Q MOD P > 1  
C7E6AD5EAF0F2D56279B7C27F95C8A2438793FEB04AF5A4AA4C4DDEA17F99C1D59A57B73E0F31CFE2FC0A9012A7EE0704D26C585E2813AFDF7115D7C4CD86180

Y (public) = G^X MOD P  
769B3B9B1706033F5B550F068EEAC972F85F0494B4C8CF0160D76CF18D507EAA608497130EAB80A30F2DC7907AF54D036058681A63C529772675AD08998EAE75

X (private) = 160 Bits and < Q  
3851C4F16E0B1D6B4B624A04D52399F7BA4912F0

Generate

Test

Info

Exit

Done. You can test your keys right now.

# 对消息进行签名

message	I love mi ma xue
随机数k	81B77CCCCF00EE8EF04B8A6B3863F689C4BA1E796

生成的签名(r, s)

r	2FB19D0BEFCCFF426509BAE19E22C2AD83F959F9
s	20C63E89A60EA608EA4AB9D556895E10042ECAE0

DSA - Sign/Verify

Message to sign: I love mi ma xue

Random K < Q: 81B77CCCF00EE8EF04B8A6B3863F689C4BA1E796

Signature (r,s): 2FB19D0BEFCCFF426509BAE19E22C2AD83F959F9 20C63E89A60EA608EA4A

SignVerifyCancel

DSA Signature created.

## 对消息进行验证

DSA - Sign/Verify

Message to sign: I love mi ma xue

Random K < Q: 81B77CCCF00EE8EF04B8A6B3863F689C4BA1E796

Signature (r,s): 2FB19D0BEFCCFF426509BAE19E22C2AD83F959F9 20C63E89A60EA608EA4A

SignVerifyCancel

--- Success. Signature verified ---

从工具的显示来看，验证成功

如果此时尝试修改一下消息内容

DSA - Sign/Verify

Message to sign:

Random K < Q:

Signature (r,s):

--- ERROR. SIGNATURE DOES NOT MATCH !!! ---

可以发现验证失败