《密码学》实验五报告

MD5哈希函数实现

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。

实验原理

以往的文件或书信可以通过手写亲笔签名来证明其真实性，而通过计算机网络传输的信息则通过数字签字技术实现其真实性的验证。

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。

DSA数字签名算法

DSA源于Elgamal签名算法，被美国NIST采纳为DSS（Digital Signature Standard）数字签名标准，其工作原理如下：

- 全局变量的设置：
  - 素数p满足$2^{511} < p < 2^{512}$;
  - 选取$p-1$的一个素因子q，其中$2^{159} < q < 2^{160}$;
  - $g \in \mathbb{Z}_p$满足$g \equiv h^{\frac{p-1}{q}} \pmod{p}$，且$1 < h < p-1$为一个随机整数。
- 私钥：$x \leftarrow \mathsf{UniformRandom}(0, q)$.
- 公钥：$y \equiv g^x \pmod{p}$，公钥整体为$\langle p, q, g, y \rangle$.
- 用户的随机数选择：$k \leftarrow \mathsf{UniformRandom}(0, q)$为随机数或伪随机数。

签字流程如下：

1. 设待签字消息为m，生成
   - $\gamma = (g^k \bmod p) \bmod q$;
   - $\delta = (k^{-1}(H(m) + x\gamma)) \bmod q$

   作为签字。其中$H(\cdot)$是一个安全哈希函数，一般用SHA-1算法作为哈希函数。
2. 消息签字后发送给验证方，验证签名的流程如下：
   - $w = \delta^{-1} \bmod q$
   - $u_1 = H(m)w \bmod q$

- $u_2 = \gamma w \bmod q$

若计算结果满足$((g^{u_1}y^{u_2}) \bmod p) \bmod q = \gamma$，则认为消息的签字是有效的。

该算法的正确性证明如下：

首先可以注意到

$$\delta = (k^{-1}(H(m) + x\gamma)) \bmod q \Rightarrow k = (\delta^{-1}H(m) + \delta^{-1}x\gamma) \bmod q.$$

则

$$k = (wH(m) + wx\gamma) \bmod q.$$

故有

$$\begin{aligned}\gamma &= (g^k \bmod p) \bmod q \\ &= (g^{(wH(m)+wx\gamma) \bmod q} \bmod p) \bmod q \\ &= (g^{wH(m)}y^{w\gamma} \bmod p) \bmod q \\ &= (g^{u_1}y^{u_2} \bmod p) \bmod q.\end{aligned}$$

SHA-1安全哈希函数

SHA-1（英语：Secure Hash Algorithm 1，中文名：安全散列算法1）是一种密码散列函数，美国国家安全局设计，并由美国国家标准技术研究所（NIST）发布为联邦资料处理标准（FIPS）。SHA-1可以生成一个被称为消息摘要的160位（20字节）散列值，散列值通常的呈现形式为40个十六进制数。

对于任意长度的明文，SHA1首先对其进行分组，使得每一组的长度为512位，然后对这些明文分组反复重复处理。

对于每个明文分组的摘要生成过程如下：

（1） 将512位的明文分组划分为16个子明文分组，每个子明文分组为32位。

（2） 申请5个32位的链接变量，记为A、B、C、D、E。

（3） 16份子明文分组扩展为80份。

（4） 80份子明文分组进行4轮运算。

（5） 链接变量与初始链接变量进行求和运算。

（6） 链接变量作为下一个明文分组的输入重复进行以上操作。

（7） 最后，5个链接变量里面的数据就是SHA1摘要。

对于任意长度的明文，SHA1的明文分组过程与MD5相类似，首先需要对明文添加位数，使明文总长度为$448 \bmod 512$位。在明文后添加位的方法是第一个添加位是1，其余都是0。然后将真正明文的长度（没有添加位以前的明文长度）以64位表示，附加于前面已添加过位的明文后，此时的明文长度正好是512位的倍数。与MD5不同的是SHA1的原始报文长度不能超过2的64次方，另外SHA1的明文长度从低位开始填充。

经过添加位数处理的明文，其长度正好为512位的整数倍，然后按512位的长度进行分组（block），可以划分成L份明文分组，我们用Y_0，$Y_1, \cdots, Y_{L-1}$表示这些明文分组。对于每一个明文分组，都要重复反复的处理，这些与MD5是相同的。

对于512位的明文分组，SHA1将其再分成16份子明文分组（sub-block），每份子明文分组为32位，我们使用$M[k]$ $(k = 0, 1, \cdots, 15)$来表示这16份子明文分组。之后还要将这16份子明文分组扩充到80份子明文分组，我们记为$W[k](k = 0, 1, \cdots, 7)$，扩充的方法如下：

- $W_t = M_t$, 当$0 \leq t \leq 15$.
- $W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1$， 当$16 \leq t \leq 79$.

SHA1有4轮运算，每一轮包括20个步骤（一共80步），最后产生160位摘要，这160位摘要存放在5个32位的链接变量中，分别标记为A、B、C、D、E。这5个链接变量的初始值以16进制位表示如下。

```
A = 0x67452301
B = 0xEFCDAB89
C = 0x98BADCFE
D = 0x10325476
E = 0xC3D2E1F0
```

SHA1有4轮运算，每一轮包括20个步骤，一共80步，当第1轮运算中的第1步骤开始处理时，A、B、C、D、E五个链接变量中的值先赋值到另外5个记录单元A'，B'，C'，D'，E'中。这5个值将保留，用于在第4轮的最后一个步骤完成之后与链接变量A，B，C，D，E进行求和操作。SHA1的4轮运算，共80个步骤使用同一个操作程序如下：

$$\langle A, B, C, D, E\rangle \leftarrow \langle (A <<< 5) + \mathrm{ft}(B, C, D) + E + W_t + K_t], A, (B <<< 30), C, D\rangle,$$

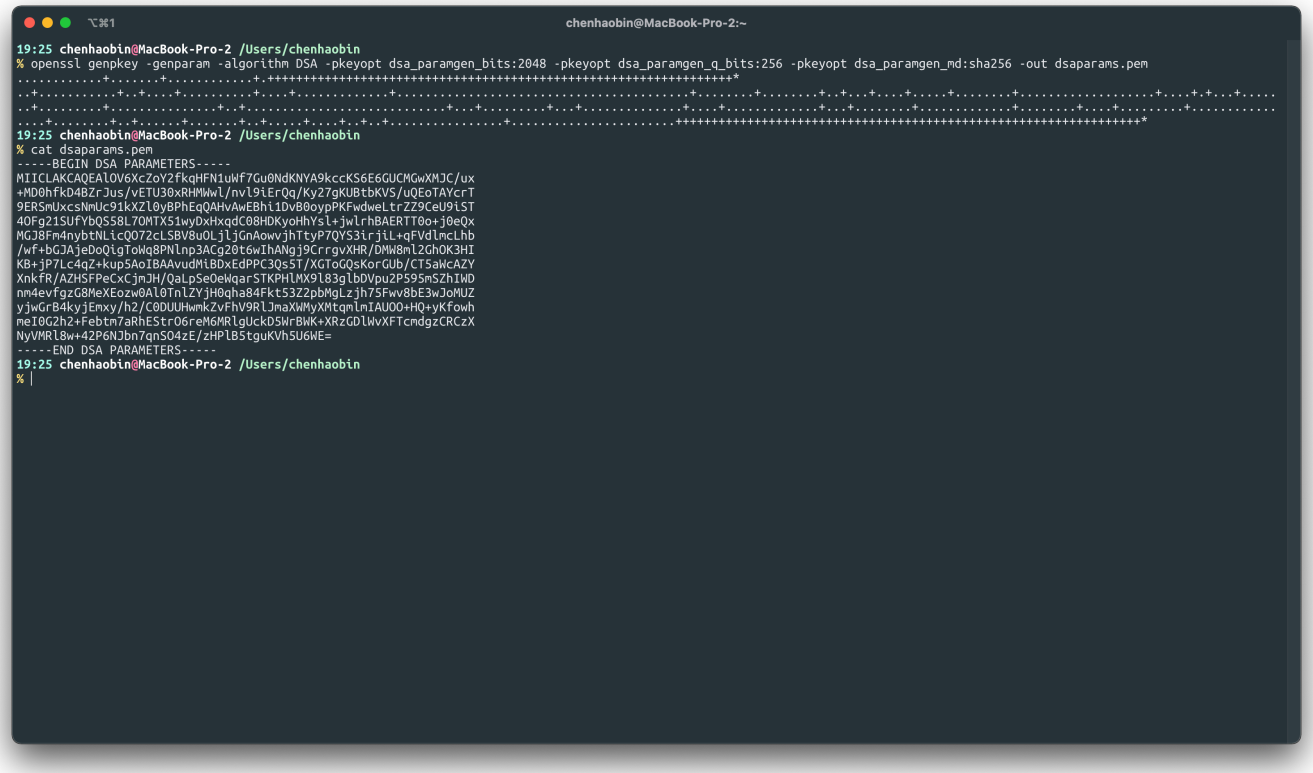
其中 $\mathrm{ft}(B, C, D)$为逻辑函数，W_t为子明文分组$W[t]$，K_t为固定常数。

综上所述，我们可以发现SHA-1的算法和MD5流程是类似的。

DSA签字演示

这里我们选取了Unix系统下面比较好用的openssl加密工具，并且运行它的cli（Command-line interface）版本。

1. 首先随机生成DSA的参数，这里我们采用2048bit长度的参数和SHA-256作为哈希函数：



2. 然后生成私钥和公钥：

```
19:25 chenhaobin@MacBook-Pro-2 /Users/chenhaobin
% openssl genpkey -paramfile dsaparams.pem -out dsaprivkey.pem
19:26 chenhaobin@MacBook-Pro-2 /Users/chenhaobin
% cat dsaprivkey.pem
-----BEGIN PRIVATE KEY-----
MIICZQIBADCCAjkGByqGSM44BAEwggIsAoIBAQCU5XpdxmhjZ+SocU3W5Z/sa7Q1
0o1gD2RxwpLoToZQIwbBcwkL+7H4wPSF+QPgFmsm6z+8RNTfTFEcxbCX+e+X2ISt
Cr8rLbuApQG1spVL+5AShMBhytP0RFKZTFyw2ZRz3WRdmXTIE+ESpAAe8DAQGGLU
O8HSjKk8oXB3B4u2tln0J5T2JJPg4WDbVJR9htBLnwvs4xNfnXDIPEfGp0LTwcMr
KgeFiyX6PCWuEEARFNPSj6PR5DEwYnwWbifJu00uJxA7vZwtIFXy44uOWMacCjC+
OFO3I/tBhLeKuOIv6oVV2WZwuFv/B/5sYkCN4OhCKBOharw82WencAKDbS3rAiEA
2CP0KuuC9cdH8MxbyaXYaE4rccgoH6M/stzipn6S6nkCggEAC+50yIEPER088LdC
zlP9cZOgZCwqisZRv8JPlpZwBlheeR9H8BkdIU94LEKOYkf9BoulJ455apqtJMo8
eUxf2XzeCVsNWm7Y/n3mZJmEhYOebh69+DMbwx5cSjPDQCXROeVliMfSqFrzgWS3
ndnalsyAvOOHvkXC/xsTfAmgxRnKPAasHiTKMSbHL+Hb8LQNRQfCaRm8WFX1GUmZ
pdYzJcy2qaWYgBQ474dD7Ip+jCGZ4jQbaHb4V5u2btpGERK2s7qt4zoxGWBRyQPl
asFYr5dHMYOVa9cVNyZ2DMJELNc3JUxGXzD7jY/o0lufuqdI7jMT/Mc+UHm2C4pW
HlTpYQQjAiEAtcSueQyWLM5uBYg5wHg5vFpdYTFBQQHtyqAzGlLvx+Y=
-----END PRIVATE KEY-----
```

然后利用私钥生成公钥：

```
19:27 chenhaobin@MacBook-Pro-2 /Users/chenhaobin
% openssl dsa -in dsaprivkey.pem -pubout > dsapubkey.pem
read DSA key
writing DSA key
19:27 chenhaobin@MacBook-Pro-2 /Users/chenhaobin
% cat dsapubkey.pem
-----BEGIN PUBLIC KEY-----
MIIDRjCCAjkGByqGSM44BAEwggIsAoIBAQCU5XpdxmhjZ+SocU3W5Z/sa7Q10o1g
D2RxwpLoToZQIwbBcwkL+7H4wPSF+QPgFmsm6z+8RNTfTFEcxbCX+e+X2IStCr8r
LbuApQG1spVL+5AShMBhytP0RFKZTFyw2ZRz3WRdmXTIE+ESpAAe8DAQGGLUO8HS
jKk8oXB3B4u2tln0J5T2JJPg4WDbVJR9htBLnwvs4xNfnXDIPEfGp0LTwcMrKgeF
iyX6PCWuEEARFNPSj6PR5DEwYnwWbifJu00uJxA7vZwtIFXy44uOWMacCjC+OFO3
I/tBhLeKuOIv6oVV2WZwuFv/B/5sYkCN4OhCKBOharw82WencAKDbS3rAiEA2CP0
KuuC9cdH8MxbyaXYaE4rccgoH6M/stzipn6S6nkCggEAC+50yIEPER088LdCzlP9
cZOgZCwqisZRv8JPlpZwBlheeR9H8BkdIU94LEKOYkf9BoulJ455apqtJMo8eUxf
2XzeCVsNWm7Y/n3mZJmEhYOebh69+DMbwx5cSjPDQCXROeVliMfSqFrzgWS3ndna
lsyAvOOHvkXC/xsTfAmgxRnKPAasHiTKMSbHL+Hb8LQNRQfCaRm8WFX1GUmZpdYz
Jcy2qaWYgBQ474dD7Ip+jCGZ4jQbaHb4V5u2btpGERK2s7qt4zoxGWBRyQPlasFY
r5dHMYOVa9cVNyZ2DMJELNc3JUxGXzD7jY/o0lufuqdI7jMT/Mc+UHm2C4pWHlTp
YQOCAQUAAoIBAFgHhRFHSRBZ9Z6citl1UDL06Gqwa5jC2jKo9sNZPlRjXvwtkRD0
EkS1E8H8/rw61bDYDoTHjcpc24WcyZUmZZJymQQxqPh3+U0eLRCx0DL4DilZu/3g
cDLewFqz6ctxL1RCIBDUj7c5kVEC9RQ7QMtGDaZM3gp8/PL7ZBz9NQRcATcRifoE
V35sTGM/DBLBVHNb42/ZeEaF/bx0TSvEVuIQR3hsKUj7WooWtT25Xjkg+yga9BdJ
44R23R9ZQbmzvm/mF8XVQQ5+YugimdZufxo/r6pwqRqE4JhQ5jicpIxj4+jAvdDI
4z2apLzyngfRpWdPXQ1IVGG+FUjw383GjSc=
-----END PUBLIC KEY-----
```

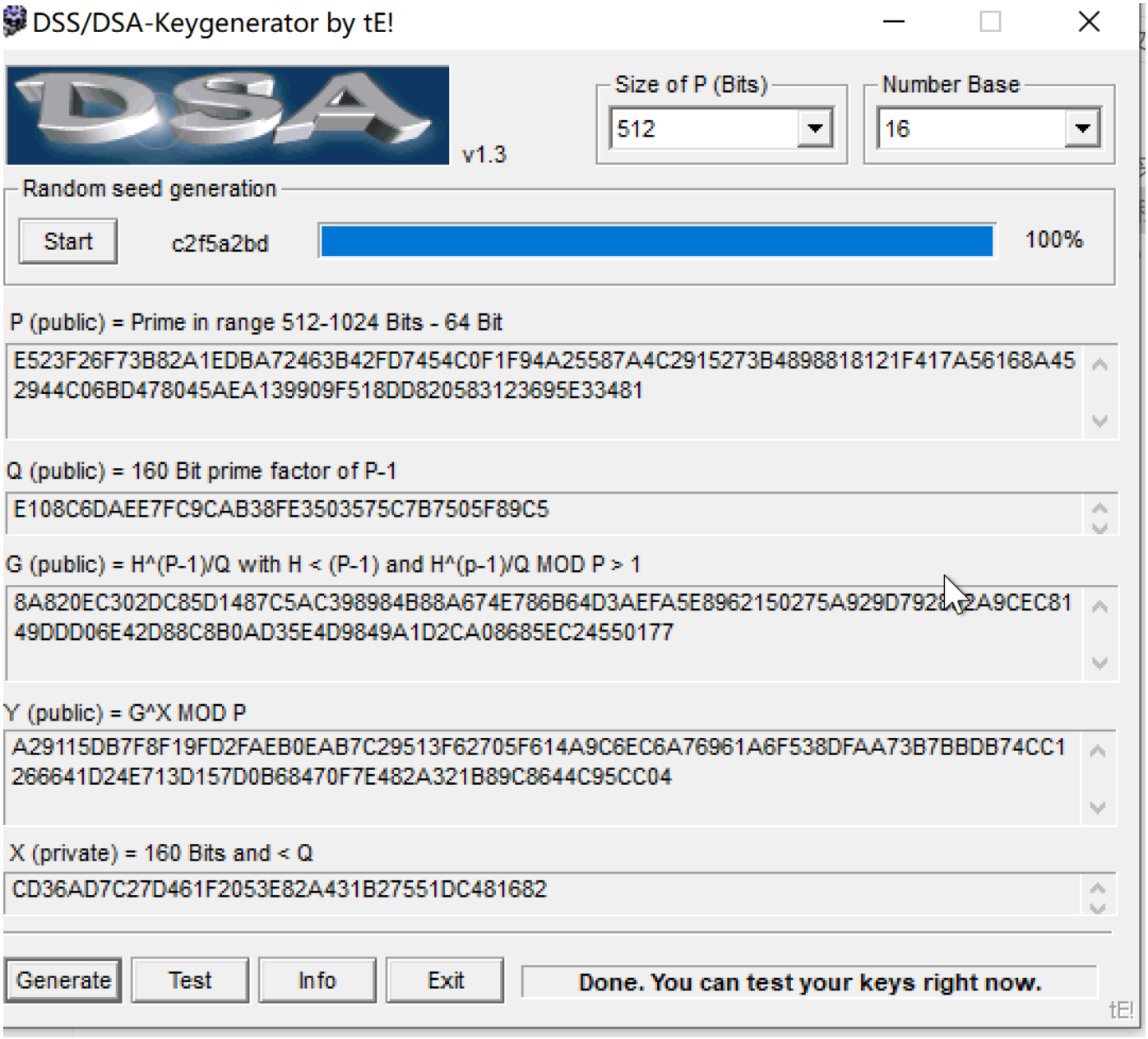
3. 对一个任意文件进行签字：

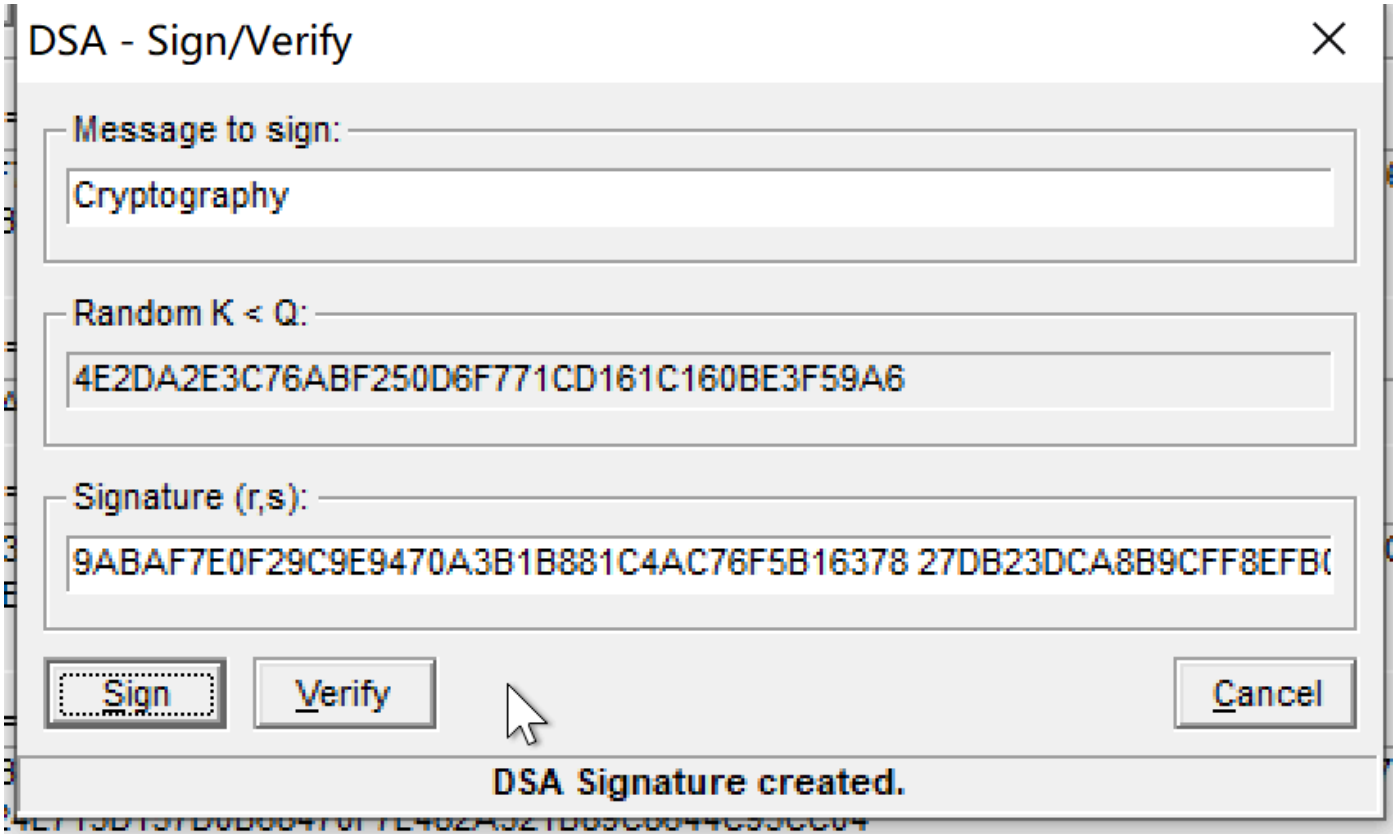
```
19:27 chenhaobin@MacBook-Pro-2 /Users/chenhaobin
% openssl dgst -sha256 -sign dsaprivkey.pem /Users/chenhaobin/Documents/NKU/Computer/Cryptography/Labs/Lab4/src/md5.cc > md5.sig
19:28 chenhaobin@MacBook-Pro-2 /Users/chenhaobin
% cat md5.sig
0E!◆◆◆◆z◆
        h◆◆◆*◆◆ ◆◆8◆   +^◆◆◆◆;t e◆◆>◆◆l◆u-◆F◆&◆>◆◆v◆◆ >aA◆%
19:28 chenhaobin@MacBook-Pro-2 /Users/chenhaobin
%
```

4. 验证文件签名是否正确：

```
19:29 chenhaobin@MacBook-Pro-2 /Users/chenhaobin
% openssl dgst -sha256 -verify dsapubkey.pem -signature md5.sig /Users/chenhaobin/Documents/NKU/Computer/Cryptography/Labs/Lab4/src/md5.cc
Verified OK
```

DSA Tools测试：


DSS/DSA-Keygenerator by tE!
DSA
v1.3
Size of P (Bits)
512
Number Base
16
Random seed generation
Start
c2f5a2bd
100%
P (public) = Prime in range 512-1024 Bits - 64 Bit
E523F26F73B82A1EDBA72463B42FD7454C0F1F94A25587A4C2915273B4898818121F417A56168A45
2944C06BD478045AEA139909F518DD820583123695E33481
Q (public) = 160 Bit prime factor of P-1
E108C6DAEE7FC9CAB38FE3503575C7B7505F89C5
G (public) = H^(P-1)/Q with H < (P-1) and H^(p-1)/Q MOD P > 1
8A820EC302DC85D1487C5AC398984B88A674E786B64D3AEFA5E8962150275A929D792
49DDD06E42D88C8B0AD35E4D9849A1D2CA08685EC24550177
Y (public) = G^X MOD P
A29115DB7F8F19FD2FAEB0EAB7C29513F62705F614A9C6EC6A76961A6F538DFAA73B7BBDB74CC1
266641D24E713D157D0B68470F7E482A321B89C8644C95CC04
X (private) = 160 Bits and < Q
CD36AD7C27D461F2053E82A431B27551DC481682
Generate
Test
Info
Exit
Done. You can test your keys right now.
tE!


DSA - Sign/Verify
Message to sign:
Cryptography
Random K < Q:
4E2DA2E3C76ABF250D6F771CD161C160BE3F59A6
Signature (r,s):
9ABAF7E0F29C9E9470A3B1B881C4AC76F5B16378 27DB23DCA8B9CFF8EFB(
Sign
Verify
Cancel
DSA Signature created.