openssl

强大的安全套接字层密码库

补充说明

OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及SSL协议，并提供丰富的应用程序供测试或其它目的使用。在OpenSSL被曝出现严重安全漏洞后，发现多数通过SSL协议加密的网站使用名为OpenSSL的开源软件包。由于这是互联网应用最广泛的安全传输方法，被网银、在线支付、电商网站、门户网站、电子邮件等重要网站广泛使用，所以该漏洞影响范围广大。

OpenSSL有两种运行模式：交互模式和批处理模式。

直接输入openssl回车进入交互模式，输入带命令选项的openssl进入批处理模式。

OpenSSL整个软件包大概可以分成三个主要的功能部分：密码算法库、SSL协议库以及应用程序。OpenSSL的目录结构自然也是围绕这三个功能部分进行规划的。

对称加密算法

OpenSSL一共提供了8种对称加密算法，其中7种是分组加密算法，仅有的一种流加密算法是RC4。这7种分组加密算法分别是AES、DES、Blowfish、CAST、IDEA、RC2、RC5，都支持电子密码本模式（ECB）、加密分组链接模式（CBC）、加密反馈模式（CFB）和输出反馈模式（OFB）四种常用的分组密码加密模式。其中，AES使用的加密反馈模式（CFB）和输出反馈模式（OFB）分组长度是128位，其它算法使用的则是64位。事实上，DES算法里面不仅仅是常用的DES算法，还支持三个密钥和两个密钥3DES算法。

非对称加密算法

OpenSSL一共实现了4种非对称加密算法，包括DH算法、RSA算法、DSA算法和椭圆曲线算法（EC）。DH算法一般用户密钥交换、RSA算法既可以用于密钥交换，也可以用于数字签名，当然，如果你能够忍受其缓慢的速度，那么也可以用于数据加密。DSA算法则一般只用于数字签名。

信息摘要算法

OpenSSL实现了5种信息摘要算法，分别是MD2、MD5、MDC2、SHA（SHA1）和RIPEMD。SHA算法事实上包括了SHA和SHA1两种信息摘要算法，此外，OpenSSL还实现了DSS标准中规定的两种信息摘要算法DSS和DSS1。

密钥和证书管理

密钥和证书管理是PKI的一个重要组成部分。OpenSSL为之提供了丰富的功能，支持多种标准。

首先，OpenSSL实现了ASN.1的证书和密钥相关标准，提供了对证书、公钥、私钥、证书请求以及CRL等数据对象的DER、PEM和BASE64的编解码功能。OpenSSL提供了产生各种公开密钥对和对称密钥的方法、函数和应用程序，同时提供了对公钥和私钥的DER编解码功能。并实现了私钥的PKCS#12和PKCS#8的编解码功能。OpenSSL在标准中提供了对私钥的加密保护功能，使得密钥可以安全地进行存储和分发。

在此基础上，OpenSSL实现了对证书的X.509标准编解码、PKCS#12格式的编解码以及PKCS#7的编解码功能。并提供了一种文本数据库，支持证书的管理功能，包括证书密钥产生、请求产生、证书签发、吊销和验证等功能。

事实上，OpenSSL提供的CA应用程序就是一个小型的证书管理中心（CA），实现了证书签发的整个流程和证书管理的大部分机制。

实例

1、使用 openssl 生成密码

几乎所有 Linux 发行版都包含 openssl，我们可以利用它的随机功能来生成可以用作密码的随机字母字符串。

```
openssl rand -base64 10
# nU9LlHO5nsuUvw==
```

nU9LlHO5nsuUvw==

2、消息摘要算法应用例子

用SHA1算法计算文件file.txt的哈西值，输出到stdout。

```
# openssl dgst -sha1 file.txt
```

用SHA1算法计算文件file.txt的哈西值，输出到文件digest.txt。

```
# openssl sha1 -out digest.txt file.txt
```

用DSS1(SHA1)算法为文件file.txt签名，输出到文件dsasign.bin，签名的privatekey必须为DSA算法产生的，保存在文件dsakey.pem中。

```
# openssl dgst -dss1 -sign dsakey.pem -out dsasign.bin file.txt
```

用dss1算法验证file.txt的数字签名dsasign.bin，验证的private key为DSA算法产生的文件dsakey.pem。

```
# openssl dgst -dss1 -prverify dsakey.pem -signature dsasign.bin file.txt
```

用sha1算法为文件file.txt签名,输出到文件rsasign.bin，签名的private key为RSA算法产生的文件rsaprivate.pem。

```
# openssl sha1 -sign rsaprivate.pem -out rsasign.bin file.txt
```

用sha1算法验证file.txt的数字签名rsasign.bin，验证的public key为RSA算法生成的rsapublic.pem。

```
# openssl sha1 -verify rsapublic.pem -signature rsasign.bin file.txt
```

3、对称加密应用例子

对称加密应用例子，用DES3算法的CBC模式加密文件plaintext.doc，加密结果输出到文件ciphertext.bin。

```
# openssl enc -des3 -salt -in plaintext.doc -out ciphertext.bin
```

用DES3算法的OFB模式解密文件ciphertext.bin，提供的口令为trousers，输出到文件plaintext.doc，注意：因为模式不同，该命令不能对以上的文件进行解密。

```
# openssl enc -des-ede3-ofb -d -in ciphertext.bin -out plaintext.doc -pass
pass:trousers
```

用Blowfish的CFB模式加密plaintext.doc口令从环境变量PASSWORD中取，输出到文件ciphertext.bin

```
# openssl bf-cfb -salt -in plaintext.doc -out ciphertext.bin -pass
env:PASSWORD
```

给文件ciphertext.bin用base64编码，输出到文件base64.txt

```
# openssl base64 -in ciphertext.bin -out base64.txt
```

用RC5算法的CBC模式加密文件plaintext.doc输出到文件ciphertext.binsaltkey和初始化向量(iv)在命令行指定。

```
# openssl rc5 -in plaintext.doc -out ciphertext.bin -S C62CB1D49F158ADC -iv
E9EDACA1BD7090C6 -K 89D4B1678D604FAA3DBFFD030A314B29
```

4Diffie-Hellman应用例子

使用生成因子2和随机的1024-bit的素数产生D0ffie-Hellman参数，输出保存到文件dhparam.pem

```
# openssl dhparam -out dhparam.pem -2 1024
```

从dhparam.pem中读取Diffie-Hell参数，以C代码的形式，输出到stdout

```
# openssl dhparam -in dhparam.pem -noout -C
```

5DSA应用例子应用例子

生成1024位DSA参数集，并输出到文件dsaparam.pem

```
# openssl dsaparam -out dsaparam.pem 1024
```

使用参数文件dsaparam.pem生成DSA私钥匙，采用3DES加密后输出到文件dsaprivatekey.pem

```
# openssl gendsa -out dsaprivatekey.pem -des3 dsaparam.pem
```

使用私钥匙dsaprivatekey.pem生成公钥匙，输出到dsapublickey.pem

```
# openssl dsa -in dsaprivatekey.pem -pubout -out dsapublickey.pem
```

从dsaprivatekey.pem中读取私钥匙，解密并输入新口令进行加密，然后写回文件dsaprivatekey.pem

```
# openssl dsa -in dsaprivatekey.pem -out dsaprivatekey.pem -des3 -passin
```

6RSA应用例子

产生1024位RSA私匙，用3DES加密它，口令为trousers输出到文件rsaprivatekey.pem

```
# openssl genrsa -out rsaprivatekey.pem -passout pass:trousers -des3 1024
```

从文件rsaprivatekey.pem读取私匙，用口令trousers解密，生成的公钥匙输出到文件rsapublickey.pem

```
# openssl rsa -in rsaprivatekey.pem -passin pass:trousers -pubout -out
rsapubckey.pem
```

用公钥匙rsapublickey.pem加密文件plain.txt，输出到文件cipher.txt

```
# openssl rsautl -encrypt -pubin -inkey rsapublickey.pem -in plain.txt -out
cipher.txt
```

使用私钥匙rsaprivatekey.pem解密密文cipher.txt，输出到文件plain.txt

```
# openssl rsautl -decrypt -inkey rsaprivatekey.pem -in cipher.txt -out
plain.txt
```

用私钥匙rsaprivatekey.pem给文件plain.txt签名，输出到文件signature.bin

```
# openssl rsautl -sign -inkey rsaprivatekey.pem -in plain.txt -out
signature.bin
```

用公钥匙rsapublickey.pem验证签名signature.bin，输出到文件plain.txt

```
# openssl rsautl -verify -pubin -inkey rsapublickey.pem -in signature.bin -
out plain
```

从X.509证书文件cert.pem中获取公钥匙，用3DES加密mail.txt，输出到文件mail.enc

```
# openssl smime -encrypt -in mail.txt -des3 -out mail.enc cert.pem
```

从X.509证书文件cert.pem中获取接收人的公钥匙，用私钥匙key.pem解密S/MIME消息mail.enc，结果输出到文件mail.txt

```
# openssl smime -decrypt -in mail.enc -recip cert.pem -inkey key.pem -out
mail.txt
```

cert.pem为X.509证书文件，用私匙key,pem为mail.txt签名，证书被包含在S/MIME消息中，输出到文件mail.sgn

```
# openssl smime -sign -in mail.txt -signer cert.pem -inkey key.pem -out
mail.sgn
```

验证S/MIME消息mail.sgn，输出到文件mail.txt，签名者的证书应该作为S/MIME消息的一部分包含在mail.sgn中

```
# openssl smime -verify -in mail.sgn -out mail.txt
```

更多实例:

```
openssl version -a
openssl help
openssl genrsa -aes128 -out fd.key 2048 # pem format
```

```
openssl rsa -text -in fd.key
```