加密介绍

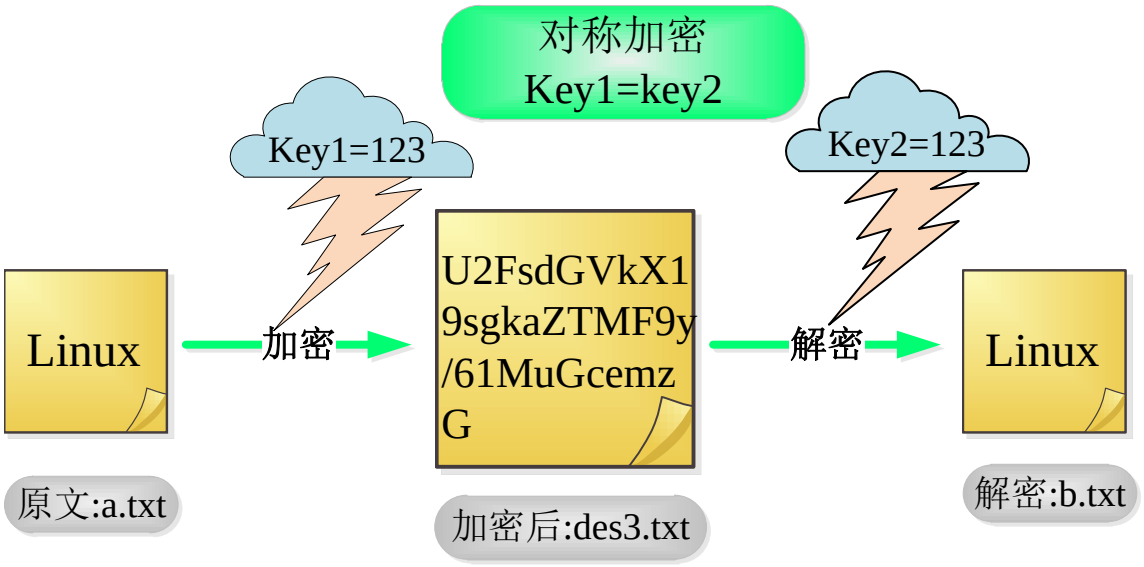
常见的加密技术：

对称加密；非对称加密；单向加密；SSL/TLS；秘钥交换

1、对称加密

采用单钥密码系统的加密方法，同一个密钥可以同时用作信息的加密和解密，这种加密方法称为对称加密，也称为单密钥加密。

对称加密的常见算法：DES、3DES、AES、IDEA、RC6、CAST5 等



(1)优点

加密、解密使用同一个密钥，效率高；

(2)缺点

必须商定秘钥：数据传送前，双方必须商定好秘钥。线上商定，存在被窃取风险；

来源无法确认：如果钥匙被窃取，窃取者就可以冒充另一方进行通讯，接收者无法确认来源；

密钥管理麻烦：每对用户都需要唯一秘钥，这会使得双方所拥有的钥匙数量巨大。

(3)实现简单对称加密：

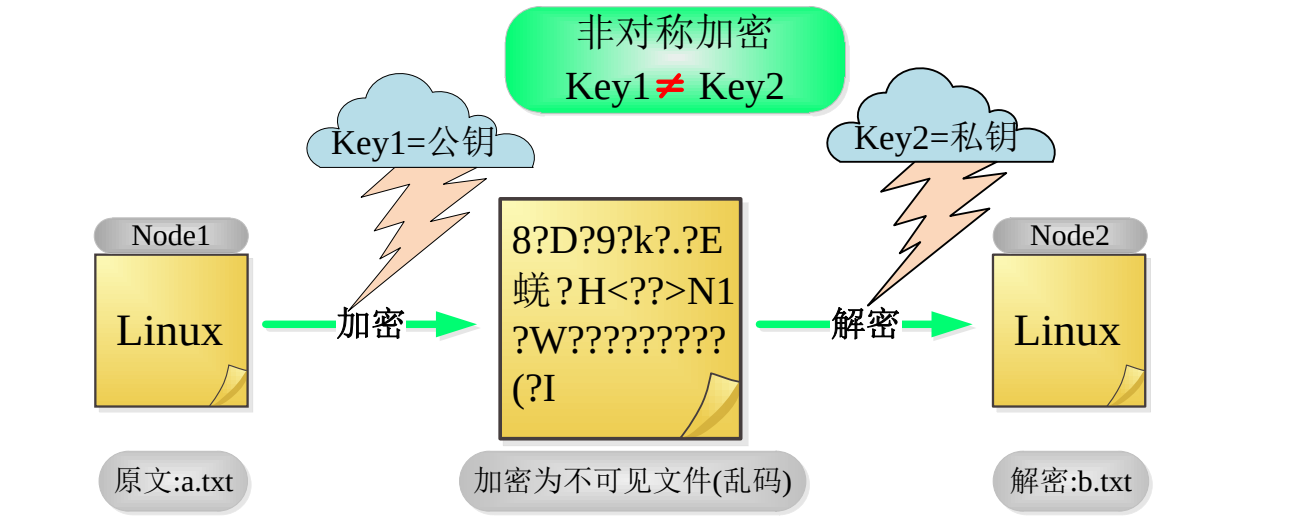
```
[root@node1 ~]# echo maohua > a.txt
[root@node1 ~]# openssl enc -e -des3 -a -salt -in a.txt -out aa.txt    #采用 des3 加密算法
enter des-ede3-cbc encryption password:                                #秘钥为 123
Verifying - enter des-ede3-cbc encryption password:                    #秘钥为 123
[root@node1 ~]# cat a.txt aa.txt
maohua                                                                 #加密前的数据内容
U2FsdGVkX19necsoBCPCTiyGxhr3h6eX                                       #加密后的数据内容
```

```
[root@node1 ~]# openssl enc -d -des3 -a -salt -in aa.txt -out b.txt    #用原加密算法解密加密文件
enter des-ede3-cbc decryption password:                                #输入加密时的秘钥 123
[root@node1 ~]# cat a.txt aa.txt b.txt
maohua
U2FsdGVkX19necsoBCPCTiyGxhr3h6eX
maohua
```

2、非对称加密

非对称加密算法使用两把完全不同但又是完全匹配的一对钥匙——公钥和私钥。在使用非对称加密算法加密文件时，只有使用匹配的一对公钥和私钥，才能完成对明文的加密和解密。

非对称加密常见的算法：RSA（既可以用来加密解密，又可以用来实现用户认证）、DSA（只能用来加密解密）、ELGamal 等



注：key2是Node2生成的私钥;key1是由Node2私钥生成的公钥;key1和key2是成对且不相同的秘钥

(1)优特点:

密钥是成对出现的，用公钥加密数据，只能使用与之配对的私钥解密；反之亦然；

公钥：（public key）公开给所有主机；

私钥：（secret key）主机留存，必须保证其私密性；

(2)缺点:

密钥长， 加密解密效率低下

(3)非对称加密技术举例：

```
[root@node1 ~]# openssl genrsa -out private.key 1024
#private.key 包含了公钥和密钥两部分，该文件即可用来加密也可以用来解密，1024 是密钥长度。
[root@node1 ~]# openssl rsa -in private.key -pubout -out pub.key  #由密钥 private.key 生成公钥
[root@node1 ~]# echo "123456" | openssl rsautl -encrypt -inkey pub.key -pubin >encode.result
```

```
#使用公钥 pub.key 对字符"123456"进行加密
[root@node1 ~]#cat encode.result | openssl rsautl -decrypt -inkey private.key
#使用私钥 private.key 对公钥加密文件 encode.result 进行解密得到原始字符"123456"
```

(4)非对称加密应用场景：ssh 无秘钥远程登录的实现就是利用了非对称加密技术

```
[root@node1 ~]#ssh-keygen -t rsa -f /root/.ssh/id_rsa -P "" #非对称加密输出私钥和公钥
[root@node1 ~]#ls /root/.ssh/
id_rsa id_rsa.pub       #id_rsa 私钥，id_rsa.pub 为公钥,将公钥拷贝到远程主机上，则可实现无密码登录。
```

3、单向加密

单向加密算法，又称 hash 函数（也称杂凑函数或杂凑算法）就是把任意长的输入消息串变化成固定长的输出串的一种函数。这个输出串称为该消息的杂凑值。一般用于产生消息摘要，密钥加密等。

单向加密的常见算法：md5、sha1、sha224、sha256、sha384、sha512

(1)单向加密加密特点

数据相同，摘要相同；数据不同，摘要不同；不可通过摘要反推源数据；相同算法，摘要长度相同。

(2)单项散列的简单实践：

```
[root@node1 ~]# echo maohua | md5sum
8de05c05333e0d7897bd7989ec054fbd  -    #字符"maohua"的摘要
[root@node1 ~]# echo mao hua | md5sum
29f86ecfc01cb2f584143118b58dc22e  -    #字符"mao hua"的摘要，与"maohua"摘要完全不同，仅多一个空格
[root@node1 ~]# echo 1 | md5sum
b026324c6904b2a9cb4b88d6d61c81d1  -    #一个字符"1"的摘要也是 128bits
```

(3)密码加密方面

可以通过加‘盐’提高密码的安全性，同一算法相同数据的条件下：盐相同，结果相同；盐不同，结果必不相同；数据不同，结果肯定也不同。以下是简单实践：

```
##对"linux"进行 MD5 加密算法：
[root@node1 ~]# grub-md5-crypt     #使用 md5 加密并且自动向加密对象内加入 salt
Password:                          #输入 linux，进行 md5 加密
Retype password:
$1$VyRD5/$Zk92HqQ/loWEfWpUwxwYy1  #加密结果，有三个字段以"$"分割，第二个字段为盐(salt)
[root@node1 ~]# openssl passwd -1 -salt "VyRD5/"     #相同待加密字符指定相同加密算法并添加相同 salt
Password:                                            #输入"linux"
$1$VyRD5/$Zk92HqQ/loWEfWpUwxwYy1                     #输出相同的加密结果
[root@node1 ~]# openssl passwd -1 -salt "VyRD5?"     #相同待加密字符指定相同加密算法并添加不同 salt
Password:                                            #输入"linux"
```

```
$1$VyRD5?$fxhBzsEq38F6s8W7mpzfS0                    #输出完全不同的加密结果
```

4、SSL/TLS

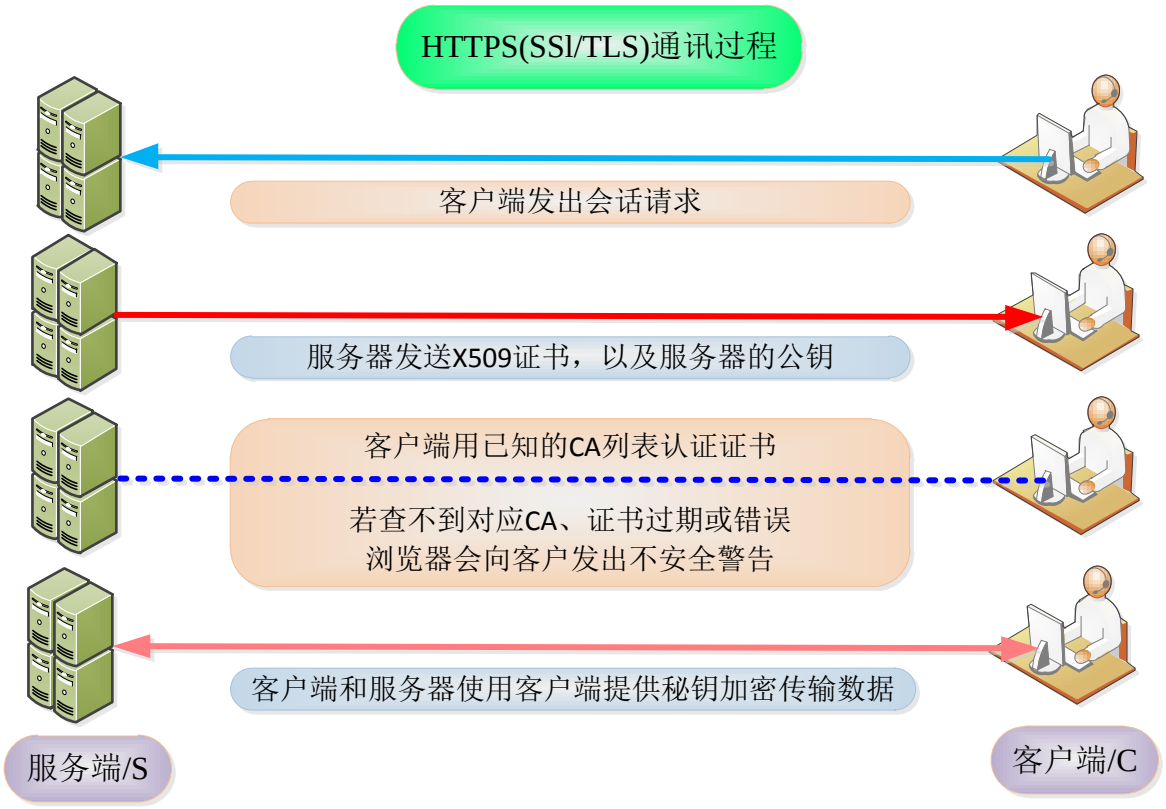
Handshake 协议：包括协商安全参数和密码套件、服务器身份认证（客户端身份认证可选）、密钥交换

ChangeCipherSpec 协议：一条消息表明握手协议已经完成

Alert 协议：对握手协议中一些异常的错误提醒，分为 fatal 和 warning 两个级别， fatal 类型错误会直接中断 SSL 链接，而 warning 级别的错误 SSL 链接仍可继续，只是给出错误警告。

Record 协议：包括对消息的分段、压缩、消息认证和完整性保护、加密等

HTTPS 协议：就是”HTTP 协议”和”SSL/TLS 协议”的组合。”HTTP over SSL”或”HTTP over TLS”，对 http 协议的文本数据进行加密处理后，成为二进制形式传输。



5、密钥交换

IPsec IKE：网络密钥交换协议（IPsec IKE: Internet Key Exchange Protocol）

网络密钥交换（IPsec IKE）是 IPsec 体系结构中的一种主要协议。它是一种混合协议，使用部分 Oakley 和部分 SKEME，并协同 ISAKMP 提供密钥生成材料和其它安全连系，比如用于 IPsec DOI 的 AH 和 ESP 。

DH：(Diffie-Hellman)一种确保共享 KEY 安全穿越不安全网络的方法，它是 Oakley 的一个组成部分。

这个机制的巧妙在于需要安全通信的双方可以用这个方法确定对称密钥。然后可以用这个密钥进行加密和解密。DH 只能用于密钥的交换，而不能进行消息的加密和解密。

密钥协商过程如下：

第一步：A 和 B 协商生成公开的整数 a，大素数 p；

第二步：A 生成隐私数据 $x,(x<p)$，计算得出 $(a^x)\%p$ 发送给 B；

B 生成隐私数据 $y,(y<p)$，计算得出 $(a^y)\%p$ 发送给 A；

第三步：A 计算得出 $((a^y)\%p)^x==(a^{xy})\%p$ 生成为秘钥；

B 计算得出 $((a^x)\%p)^y==(a^{xy})\%p$ 生成为秘钥。

附加、openssl 使用详解

openssl 中有如下后缀名的文件

(1)*.key 格式：私有的密钥

(2)*.crt 格式：证书文件，certificate 的缩写

(3)*.csr 格式：证书签名请求（证书请求文件），含公钥信息，certificate signing request 的缩写

(4)*.crl 格式：证书吊销列表，Certificate Revocation List 的缩写

(5)*.pem 格式：用于导出，导入证书时候的证书的格式。

openssl 的使用：

#----------加密----------#

```
##对 fstab 文件进行 des3 加密，生成加密文件 fstab.des3，参数：-d
[root@stu2~]# openssl enc -des3 -in fstab -e -out fstab.des3
### openssl enc 指定加密的类型,des3 表示使用对称加密,-in 指定要加密的文件,-e 表示加密,-out 保存的文件名
```

#----------解密----------#

```
##对 fstab.des3 文件进行 des3 解密，生成解密文件 fstab.txt，参数：-e
[root@stu2~]# openssl enc -des3 -in fstab.des3 -d -out fstab.txt
### openssl enc 指定加密的类型,des3 表示对称加密,-in 表示加密后的文件,-d 表示解密,-out 要保存的文件名
```

#----------单向加密(或 md5sum)获取文件的特征码----------#

```
[root@stu2~]# openssl dgst -md5 -hex fstab          #获取 16 进制的特征码，默认为 16 进制的可以省略
##除了 openssl 命令获取文件特征码，还可利用 md5sum 命令获取文件特征码：
[root@stu2~]# md5sum fstab                          #获取文件特征码
```

#----------复习签署 CA 证书签署----------#

```
###为 CA 生成一个私钥
```

```
[root@stu2 CA]# (umask 077;openssl genrsa -out private/cakey.pem 2048)
###由 CA 负责签署证书
[root@stu2 CA]# openssl req -new -x509 -key private/cakey.pem -out cacert.pem -days 3656
###添加证书签署的序列号
[root@stu2CA]# touch serial index.txt
[root@stu2CA]# echo 01 > serial
################先生成证书的私钥和公钥文件，以及证书，然后 CA 进行证书的签署：
[root@stu2~]# (umask 077;openssl genrsa -out httpd_private.key 2048) #生成私钥文件
[root@stu2~]# openssl req -new -key httpd.key -out httpd.csr          #生成证书
[root@stu2~]# openssl ca -in httpd.csr -out httpd.crt -days 3656      #在 CA 主机上签署生成的证书
```

参考链接：http://lanlian.blog.51cto.com/6790106/1281720/