大地之光

openssl 摘要和签名验证指令dgst使用详解

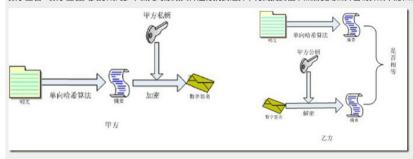
1、信息摘要和数字签名概述

信息摘要:对数据进行处理,得到一段固定长度的结果,其特点输入:

- 1、输出长度固定。即输出长度和输入长度无关。
- 2、不可逆。即由输出数据理论上不能推导出输入数据
- 4、对输入数据敏感。当输入数据变化极小时,输出数据也会发生明显的变化
- 5、防碰撞。即不同的数据数据得到相同输出数据的可能性极低。

由于信息摘要有上述特点,一般保证数据的完整性,对一个大文件进行摘要运算,得到其摘要值。通过网络或者其他渠道传输后,通过验证其摘要值,确定大文件本身有没有发生变化。

数字签名:数字签名其实分成两步,首先对原始文件进行摘要运算,得到摘要值,然后使用公开密钥算法中的私钥对摘要值进行加密。其签名和验证过程如下图所示



有数字签名的过程可以知道,对发送信息进行数字签名,可以保证数字签名的完整性、真实性、不可抵赖性。即接收者可以确认消息的来源、消息的真实,发送者不可以抵赖自己发送的消息,与现实生活中签名的作用大致相同。

2、摘要算法和数字签名相关指令及用法

目前openssl提供的摘要算法有md4、md5、ripemd160、sha、sha1、sha224、sha256、sha512、sha384、wirlpool。可以通过openssl dgst -命令查看。

上面我们已经提到了,数字签名分为摘要和加密两部分。在openss提供的指令中,并没有区分两者。而是在摘要算法指令中包含了签名和校验参数。例如我们适用openssI md5-命令可以看到它提供的选项有签名和验证等参数。

在openssl中单独使用摘要算法指令完成摘要或者签名操作,也可以通过dgst完成相同的操作。在签名的时候多数使用RSA私钥或者DSA私钥,当使用RSA私钥的时候,我们可以使用单独的摘要算法指令指定摘要算法进行签名,但当使用DSA使用签名的时候,就必 须使用dast指令,因为使用DSA签名的时候必须使用DSA自身的摘要算法,而openssl没有为它提供相应的指令。



可以看到md5和dgst完成相同的功能。不过让人纠结的是使用md5进行签名的时候可以指定其他摘要算法,笔者觉得太别扭了。所以建议做摘要和签名验证时使用dgst指令,忘记其他.....

dgst指令用法介绍如下

```
xlzh@cmos:~/test$ openssl dgst -
unknown option '-'
```

```
2021年10月
      = =
H
           四
               五
  27
     28
26
         29
            30
               1
   4
     5
         6
           7
               8
10
  11 12 13 14 15
                 16
17
  18
     19 20
           21
              22
                  23
24
  25 26 27 28 29 30
  1
        3
           4
```

31 导航 博客园 首页 新随笔 联系 订阅 ண 管理

统计 随笔 - 9 文章 - 0 评论 - 5 阅读 - 5605

公告 昵称: 哈哈稻草人 **园龄:5年10个月**

粉丝: 0 关注: 1 +加关注

3

搜索

找找看

谷歌搜索

常用链接 我的随笔 我的评论 我的参与 最新评论 我的标签

我的标签 mfc 窗口风格(1) ECC 软件保护(1) openssl(1) phpstorm webstorm(1)

随笔分类 C++(1)

随笔档案 2018年11月(1)

2018年10月(1) 2018年9月(3) 2018年8月(1)

2018年7月(2) 2018年6月(1)

libeven

```
options are
             to output the digest with separating colons
                                                           //输出的摘要信息以分号隔离. 和-hex同时使用
-c
                                                           //指定输出的格式
-r
             to output the digest in coreutils format
                                                           //输出BIO调试信息
-d
             to output debug info
                                                           //以16进制打印输出结果
-hex
             output as hex dump
                                                           //输出二进制结果
-binary
             output in binary form
-hmac arg
             set the HMAC key to arg
                                                           //指定hmac的key
-non-fips-allow allow use of non FIPS digest
                                                           //允许使用不符合fips标准的摘要算法
                                                           //执行签名操作, 后面指定私钥文件
-sign file sign digest using private key in file
                                                           //执行验证操作,后面指定公钥文件,与prverfify不能同时使用
-verify file verify a signature using public key in file
                                                           //执行验证操作,后面指定密钥文件,与verfify不能同时使用
-prverify file verify a signature using private key in file
-keyform arg key file format (PEM or ENGINE)
                                                           //指定密钥文件格式, pem或者engine
-out filename output to filename rather than stdout
                                                           //指定输出文件. 默认标准输出
                                                           //指定签名文件, 在验证签名时使用
-signature file signature to verify
-sigopt nm:v signature parameter
                                                           //祭名参数
                                                           //制作一个hmac 使用key
             create hashed MAC with key
-hmac key
-mac algorithm create MAC (not neccessarily HMAC)
                                                           //制作一个mac
                                                           //mac复法参数或者kev
-macopt nm:v MAC algorithm parameters or key
                                                           //使用硬件或者三方加密库
-engine e
             use engine e, possibly a hardware device.
             to use the md4 message digest algorithm
                                                           //摘要算法使用md4
-md4
-md5
             to use the md5 message digest algorithm
                                                           //摘要算法使用md5
-ripemd160
                                                          //摘要算法使用ripemd160
             to use the ripemd160 message digest algorithm
             to use the sha message digest algorithm
-sha
                                                          //摘要算法使用sha
             to use the shal message digest algorithm
                                                          //摘要算法使用sha1
-sha1
-sha224
             to use the sha224 message digest algorithm
                                                          // 摘要算法使用 sha 223
-sha256
             to use the sha256 message digest algorithm
                                                          //摘要算法使用sha256
-sha384
             to use the sha384 message digest algorithm
                                                           //摘要算法使用sha384
             to use the sha512 message digest algorithm
-sha512
                                                           //摘要算法使用sha512
             to use the whirlpool message digest algorithm
                                                           //摘要算法使用whirlpool
-whirlpool
```

3、dgst使用示例

1、仅做摘要运算而不做签名操作

```
/*对file.txt文件使用sha1算法讲行hash运算*/
xlzh@cmos:~/test$ openssl dgst -sha1 file.txt
SHA1(file.txt) = c994aec2a9007221a9b9113b8ab60a60144740c9
/*指定-non-fips-allow参数, 与fips标准有关, 尚待研究*/
xlzh@cmos:~/test$ openssl dgst -sha1 -non-fips-allow file.txt
SHA1(file.txt)= c994aec2a9007221a9b9113b8ab60a60144740c9
/*指定-d参数,打印调试消息*/
xlzh@cmos:~/test$ openssl dgst -sha1 -d file.txt
BIO[02469910]:ctrl(6) - FILE pointer
BIO[02469910]:ctrl return 0
BIO[02469910]:ctrl(108) - FILE pointer
BIO[02469910]:ctrl return 1
BIO[02469910]:read(0,8192) - FILE pointer
BIO[02469910]:read return 37
BIO[02469910]:read(0,8192) - FILE pointer
BIO[02469910]:read return 0
SHA1(file.txt) = c994aec2a9007221a9b9113b8ab60a60144740c9
BIO[02469910]:ctrl(1) - FILE pointer
BIO[02469910]:ctrl return 0
BIO[02469910]:Free - FILE pointer
/*指定-c -hex参数, 以16进制打印结果*/
xlzh@cmos:~/test$ openssl dgst -sha1 -c -hex file.txt
SHA1(file.txt) = c9:94:ae:c2:a9:00:72:21:a9:b9:11:3b:8a:b6:0a:60:14:47:40:c9
/*指定-r参数, 输出结果如下所示, 然并卵.....*/
xlzh@cmos:~/test$ openssl dgst -sha1 -r file.txt
c994aec2a9007221a9b9113b8ab60a60144740c9 *file.txt
/*指定-binary参数, 输入结果为二进制*/
xlzh@cmos:~/test$ openssl dgst -shal -binary file.txt
G@xlzh@cmos:~/test$
```

libevent

阅读排行榜

- 1. openssl 摘要和签名验证指令dgst 使用详解(2435)
- 2. windows openssl-1.1.1 编译静态 库和动态库(1381)
- 3. ECC算法软件保护中的应用(636)
- 4. JAVA同步锁机制 wait() notify() no tifvAll()(468)
- 5. MFC窗口风格 WS_style/WS_EX_sty le(263)

评论排行榜

- 1. windows openssl-1.1.1 编译静态 库和动态库(4)
- 2. PhpStorm激活方式(1)

最新评论

- 1. Re:windows openssl-1.1.1 编译静态库和动态库
- dmake下载链接
- --哈哈稻草人 2. Re:windows openssl-1.1.1 编译静 态库和动态库
- 4、安裝dmake, ppm install dmake 很可能会失败,可以手动安装dmake 解压dmake,将解压后大目录放在 Perl64\site\lib\auto中, auto目录不 存在就自己建立...
 - --哈哈稻草人
- 3. Re:windows openssl-1.1.1 编译静态库和动态库

如果不使用汇编就这样: perl configure no-asm VC-WIN32 -- prefix="e:\mylib"perl configure no-asm no-shared VC-WIN3...

--哈哈稻草人

4. Re:windows openssl-1.1.1 编译静 态库和动态库

编译debug库 perl configure VC-WIN32 --debug --prefix="e:\mylib" --哈哈稻草人

5. Re:PhpStorm激活方式 以上方式同时可以激活 webstorm

2018.2版本,但是有闪退的现象解决办法:链接:密码:rblr替换新的破解包...

--人地之九

```
2、使用RSA密钥进行签名验证操作
 /*摘要算法选取sha256, 密钥RSA密钥, 对file.txt进行签名*/
xlzh@cmos:~/test$ openssl dgst -sign RSA.pem -sha256 -out sign.txt file.txt
 /*使用RSA密钥验证签名(prverify参数),验证成功*/
xlzh@cmos:~/test$ openssl dgst -prverify RSA.pem -sha256 -signature sign.txt file.txt
Verified OKt
/*从密钥中提取公钥*/
xlzh@cmos:~/test$ openssl rsa -in RSA.pem -out pub.pem -pubout
writing RSA key
 /*使用RSA公钥验证签名(verify参数),验证成功*/
 xlzh@cmos:~/test$ openssl dgst -verify pub.pem -sha256 -signature sign.txt file.txt
 Verified OK
 3、使用DSA密钥进行签名验证操作
 /*使用DSA算法, 摘要算法sha256, 对file.txt进行签名*/
 xlzh@cmos:~/test$ openssl dgst -sign DSA.pem -sha256 -out sign.txt file.txt
 /*使用DSA密钥验证签名*/
xlzh@cmos:~/test$ openssl dgst -prverify DSA.pem -sha256 -signature sign.txt file.txt
Verified OK
/*使用DSA算法, 摘要算法dss1, 对file.txt进行签名*/
xlzh@cmos:~/test$ openssl dgst -sign DSA.pem -dss1 -out sign1.txt file.txt
```

根据dgst man手册的定义,如果使用DSA算法进行签名验证,必须使用dss1摘要算法,但是本实验证明使用其他摘要算法也可以签名验证。此处不明白,希望大牛指点……

4、HMAC的使用

/***使用**DSA密钥验证签**名***/

Verified OK /*提取公钥*/

read DSA key writing DSA key /*使用DSA公钥验证签名*/

Verified OK /*使用DSA公钥验证签名*/

Verified OK xlzh@cmos:~/test\$

MAC 消息认证码,构造方法可以基于hash,也可以基于对称加密算法,HMAC是基于hash的消息认证码。数据和密钥作为输入,摘要信息作为输出,常用于认证。

```
xlzh@cmos:~/test$ openssl dgst -sha256 -hmac 123456 file.txt
HMAC-SHA256(file.txt)= b8e92990b9fc2ac9b58fde06f4738dceb4fb1fc47b4d2234a9c3f152907b333a
```

例如用户登录服务器

- 1、服务器给客户端发送一个随机数
- 2、客户端使用随机数作为密钥和用户密码做HMAC,结果发送给服务器
- 3、服务器去除存储的用户密码,也是用随机数与用户密码做HMAC,根据HMAC结果是否一样确认用户身份。

xlzh@cmos:~/test\$ openssl dgst -prverify DSA.pem -dss1 -signature sign1.txt file.txt

xlzh@cmos:~/test\$ openssl dgst -verify pub.pem -dss1 -signature sign1.txt file.txt

xlzh@cmos:~/test\$ openssl dgst -verify pub.pem -sha256 -signature sign.txt file.txt

xlzh@cmos:~/test\$ openssl dsa -in DSA.pem -out pub.pem -pubout

4、遗留问题

dqst中sigopt、mac、macopt参数的含义即使用方法,因为doc都没给出具体例子,待研究openssl源码后进行补充

为什么使用DSA签名的时候可以选择其他hash算法(man 手册说只能使用dss1)

还有dgst的hmac和hmac参数,没错,你没看错,它的确提供了两个完全一样的参数,给出的解释还不一样,还是研究源码去吧.

可恶的openssl......

转自:https://www.cnblogs.com/gordon0918/p/5382541.html













« 上一篇: PhpStorm激活方式

» 下一篇: windows openssl-1.1.1 编译静态库和动态库

posted on 2018-09-14 14:49 哈哈稻草人 阅读(2435) 评论(0) 编辑 收藏 举报

录 登录后才能查看或发表评论,立即登录或者逛逛博客园首页



- ·聊聊我在微软外服的工作经历及一些个人见解
- · 死磕 NIO Reactor 模式就一定意味着高性能吗?
- ·消息队列那么多,为什么建议深入了解下RabbitMQ?
- · 技术管理进阶——管人还是管事?
- ·以终为始:如何让你的开发符合预期

最新新闻:

- ·LSTM一败涂地!男生发表4页最离谱论文,用时序模型预测女友情绪(2021-10-25 17:09)
- · AI杀手终成现实?美国陆军「杀人机器狗」引发恐慌(2021-10-25 17:02)
- · AI学会灌水和造假!Google新研究揭露了AI现实应用的陷阱(2021-10-25 16:50)
- ·可口可乐宣布推出由100%植物性塑料制成的瓶子(2021-10-25 16:40)
- · 抢先Win11! 华为移动应用引擎第二批众测开启: 在PC上玩安卓App (2021-10-25 16:30)
- » 更多新闻...

0 ●推荐

0

印反对

刷新评论 刷新页面 返回顶部

Powered by: 博客园 Copyright © 2021 哈哈稻草人