# INFA723 Cryptography and Network Security

# Lab2 Use OpenSSL Command Line

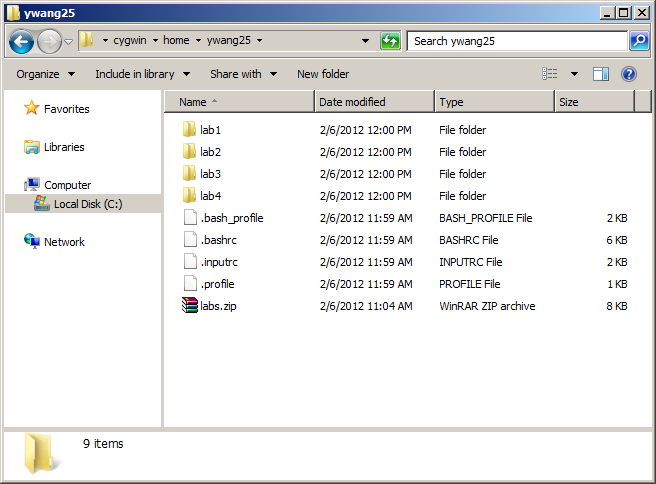
# 1 Introduction

OpenSSL toolkit ships with a command line tool which can be used for a wide range of crypto operations. In this exercise, you are going to learn how to use Openssl Command line. After the lab, answer all the questions in the end of the lab.

The lab procedure assumes the Cygwin is used. If you have a Linux OpenSSL environment, open a terminal and the lab procedure is similar.

# 2 Setup

A Lab package (labs1-4.zip) has been posted on the class’s website. Go to class’s website and download the package to your local computer. Assume you use the default Cygwin installation directory “C:\cygwin”, copy labs.zip to “C:\cygwin\home” directory. Unzip the file to your home directory and you will see four Labs listed here: lab1, lab2, lab3, and lab4.



# 3 OpenSSL Command Line Tool

Double click Cygwin icon on your desktop and open a new Cygwin terminal. Follow the steps below to finish the lab.

1. **Find out OpenSSL version number**

$ openssl version

OpenSSL 0.9.8t 18 Jan 2012

Get more information with the version –a option

$ openssl version -a

OpenSSL 0.9.8t 18 Jan 2012

built on: Wed Jan 18 16:32:51 CET 2012

platform: Cygwin

options: bn(64,32) md2(int) rc4(idx,int) des(ptr,risc1,16,long) blowfish(idx)

compiler: gcc -D\_WINDLL -DOPENSSL\_PIC -DZLIB -DOPENSSL\_THREADS -DDSO\_DLFCN -DHAVE\_DLFCN\_H -DTERMIOS -DL\_ENDIAN -fomit-frame-pointer -O3 -march=i486 -Wall -DOPENSSL\_BN\_ASM\_PART\_WORDS -DOPENSSL\_IA32\_SSE2 -DSHA1\_ASM -DMD5\_ASM -DRMD160\_ASM -DAES\_ASM

OPENSSLDIR: "/usr/ssl"

1. **Check online help**

$ openssl help

openssl:Error: 'help' is an invalid command.

Standard commands

asn1parse ca ciphers crl crl2pkcs7

dgst dh dhparam dsa dsaparam

ec ecparam enc engine errstr

gendh gendsa genrsa nseq ocsp

passwd pkcs12 pkcs7 pkcs8 prime

rand req rsa rsautl s\_client

s\_server s\_time sess\_id smime speed

spkac verify version x509

Message Digest commands (see the `dgst' command for more details)

md2 md4 md5 rmd160 sha

sha1

Cipher commands (see the `enc' command for more details)

aes-128-cbc aes-128-ecb aes-192-cbc aes-192-ecb aes-256-cbc

aes-256-ecb base64 bf bf-cbc bf-cfb

bf-ecb bf-ofb cast cast-cbc cast5-cbc

cast5-cfb cast5-ecb cast5-ofb des des-cbc

des-cfb des-ecb des-ede des-ede-cbc des-ede-cfb

des-ede-ofb des-ede3 des-ede3-cbc des-ede3-cfb des-ede3-ofb

des-ofb des3 desx rc2 rc2-40-cbc

rc2-64-cbc rc2-cbc rc2-cfb rc2-ecb rc2-ofb

rc4 rc4-40

1. **Check command options**

$ openssl enc -h

unknown option '-h'

options are

-in <file> input file

-out <file> output file

-pass <arg> pass phrase source

-e encrypt

-d decrypt

-a/-base64 base64 encode/decode, depending on encryption flag

-k passphrase is the next argument

-kfile passphrase is the first line of the file argument

-md the next argument is the md to use to create a key

from a passphrase. One of md2, md5, sha or sha1

-K/-iv key/iv in hex is the next argument

-[pP] print the iv/key (then exit if -P)

-bufsize <n> buffer size

-engine e use engine e, possibly a hardware device.

Cipher Types

-aes-128-cbc -aes-128-cfb -aes-128-cfb1

-aes-128-cfb8 -aes-128-ecb -aes-128-ofb

-aes-192-cbc -aes-192-cfb -aes-192-cfb1

-aes-192-cfb8 -aes-192-ecb -aes-192-ofb

-aes-256-cbc -aes-256-cfb -aes-256-cfb1

-aes-256-cfb8 -aes-256-ecb -aes-256-ofb

-aes128 -aes192 -aes256

-bf -bf-cbc -bf-cfb

-bf-ecb -bf-ofb -blowfish

-cast -cast-cbc -cast5-cbc

-cast5-cfb -cast5-ecb -cast5-ofb

-des -des-cbc -des-cfb

-des-cfb1 -des-cfb8 -des-ecb

-des-ede -des-ede-cbc -des-ede-cfb

-des-ede-ofb -des-ede3 -des-ede3-cbc

-des-ede3-cfb -des-ede3-cfb1 -des-ede3-cfb8

-des-ede3-ofb -des-ofb -des3

-desx -desx-cbc -rc2

-rc2-40-cbc -rc2-64-cbc -rc2-cbc

-rc2-cfb -rc2-ecb -rc2-ofb

-rc4 -rc4-40

1. **Get a list of available ciphers**

$ openssl ciphers -v

DHE-RSA-AES256-SHA SSLv3 Kx=DH Au=RSA Enc=AES(256) Mac=SHA1

DHE-DSS-AES256-SHA SSLv3 Kx=DH Au=DSS Enc=AES(256) Mac=SHA1

AES256-SHA SSLv3 Kx=RSA Au=RSA Enc=AES(256) Mac=SHA1

EDH-RSA-DES-CBC3-SHA SSLv3 Kx=DH Au=RSA Enc=3DES(168) Mac=SHA1

EDH-DSS-DES-CBC3-SHA SSLv3 Kx=DH Au=DSS Enc=3DES(168) Mac=SHA1

DES-CBC3-SHA SSLv3 Kx=RSA Au=RSA Enc=3DES(168) Mac=SHA1

DES-CBC3-MD5 SSLv2 Kx=RSA Au=RSA Enc=3DES(168) Mac=MD5

DHE-RSA-AES128-SHA SSLv3 Kx=DH Au=RSA Enc=AES(128) Mac=SHA1

DHE-DSS-AES128-SHA SSLv3 Kx=DH Au=DSS Enc=AES(128) Mac=SHA1

AES128-SHA SSLv3 Kx=RSA Au=RSA Enc=AES(128) Mac=SHA1

RC2-CBC-MD5 SSLv2 Kx=RSA Au=RSA Enc=RC2(128) Mac=MD5

RC4-SHA SSLv3 Kx=RSA Au=RSA Enc=RC4(128) Mac=SHA1

RC4-MD5 SSLv3 Kx=RSA Au=RSA Enc=RC4(128) Mac=MD5

RC4-MD5 SSLv2 Kx=RSA Au=RSA Enc=RC4(128) Mac=MD5

EDH-RSA-DES-CBC-SHA SSLv3 Kx=DH Au=RSA Enc=DES(56) Mac=SHA1

EDH-DSS-DES-CBC-SHA SSLv3 Kx=DH Au=DSS Enc=DES(56) Mac=SHA1

DES-CBC-SHA SSLv3 Kx=RSA Au=RSA Enc=DES(56) Mac=SHA1

DES-CBC-MD5 SSLv2 Kx=RSA Au=RSA Enc=DES(56) Mac=MD5

EXP-EDH-RSA-DES-CBC-SHA SSLv3 Kx=DH(512) Au=RSA Enc=DES(40) Mac=SHA1 export

EXP-EDH-DSS-DES-CBC-SHA SSLv3 Kx=DH(512) Au=DSS Enc=DES(40) Mac=SHA1 export

EXP-DES-CBC-SHA SSLv3 Kx=RSA(512) Au=RSA Enc=DES(40) Mac=SHA1 export

EXP-RC2-CBC-MD5 SSLv3 Kx=RSA(512) Au=RSA Enc=RC2(40) Mac=MD5 export

EXP-RC2-CBC-MD5 SSLv2 Kx=RSA(512) Au=RSA Enc=RC2(40) Mac=MD5 export

EXP-RC4-MD5 SSLv3 Kx=RSA(512) Au=RSA Enc=RC4(40) Mac=MD5 export

EXP-RC4-MD5 SSLv2 Kx=RSA(512) Au=RSA Enc=RC4(40) Mac=MD5 export

1. **Benchmarking**

Note: It takes couple of minutes to get the summary results. Don’t rush to stop the command before you see the summary results.

$ openssl speed

The 'numbers' are in 1000s of bytes per second processed.

type 16 bytes 64 bytes 256 bytes 1024 bytes 8192 bytes

md2 1512.19k 3012.73k 4169.62k 4629.15k 4769.63k

mdc2 0.00 0.00 0.00 0.00 0.00

md4 21952.81k 78823.56k 243158.11k 500561.72k 705794.01k

md5 19488.74k 67485.36k 185593.02k 326634.00k 431331.44k

hmac(md5) 27362.71k 88025.76k 221557.74k 353277.16k 438393.95k

sha1 19001.98k 62921.44k 162443.47k 259820.45k 311007.10k

rmd160 15654.49k 46735.81k 104655.20k 147882.06k 168627.60k

rc4 304157.71k 351774.67k 329447.38k 299007.65k 276963.82k

des cbc 54838.98k 59191.75k 58773.08k 58489.30k 58765.71k

des ede3 21030.06k 21664.73k 21506.68k 21401.10k 21096.79k

idea cbc 0.00 0.00 0.00 0.00 0.00

seed cbc 0.00 0.00 0.00 0.00 0.00

rc2 cbc 24226.45k 24889.46k 25033.30k 25634.56k 25076.55k

rc5-32/12 cbc 0.00 0.00 0.00 0.00 0.00

blowfish cbc 89924.65k 97387.20k 97778.20k 96552.28k 97290.12k

cast cbc 69312.05k 74375.70k 75653.38k 76953.67k 75215.47k

aes-128 cbc 68327.24k 108385.98k 126082.66k 130378.75k 132351.21k

aes-192 cbc 62943.09k 92226.16k 108364.20k 110146.16k 113302.80k

aes-256 cbc 57198.24k 81931.09k 92997.17k 95298.01k 96215.60k

camellia-128 cbc 0.00 0.00 0.00 0.00 0.00

camellia-192 cbc 0.00 0.00 0.00 0.00 0.00

camellia-256 cbc 0.00 0.00 0.00 0.00 0.00

sha256 17728.84k 40418.61k 69714.25k 84932.03k 92616.97k

sha512 11228.52k 45084.52k 73001.37k 105132.90k 123083.32k

aes-128 ige 93953.10k 111965.96k 117672.32k 121286.72k 119296.52k

aes-192 ige 83440.56k 95980.43k 104247.92k 103956.27k 101667.58k

aes-256 ige 75842.95k 87193.86k 88471.93k 90818.28k 89349.95k

sign verify sign/s verify/s

rsa 512 bits 0.000435s 0.000038s 2296.5 26472.9

rsa 1024 bits 0.001964s 0.000091s 509.1 10993.0

rsa 2048 bits 0.010683s 0.000278s 93.6 3598.8

rsa 4096 bits 0.066667s 0.000914s 15.0 1093.6

sign verify sign/s verify/s

dsa 512 bits 0.000336s 0.000396s 2973.4 2524.8

dsa 1024 bits 0.000875s 0.001063s 1143.4 940.7

dsa 2048 bits 0.002710s 0.003288s 369.1 304.2

1. **Benchmark DES**

$ openssl speed des-cbc

Doing des cbc for 3s on 16 size blocks: 10392632 des cbc's in 2.98s

Doing des cbc for 3s on 64 size blocks: 2742524 des cbc's in 3.00s

Doing des cbc for 3s on 256 size blocks: 693056 des cbc's in 2.93s

Doing des cbc for 3s on 1024 size blocks: 174074 des cbc's in 2.96s

Doing des cbc for 3s on 8192 size blocks: 21684 des cbc's in 2.95s

OpenSSL 0.9.8t 18 Jan 2012

built on: Wed Jan 18 16:32:51 CET 2012

options:bn(64,32) md2(int) rc4(idx,int) des(ptr,risc1,16,long) aes(partial) blowfish(idx)

compiler: gcc -D\_WINDLL -DOPENSSL\_PIC -DZLIB -DOPENSSL\_THREADS -DDSO\_DLFCN -DHAVE\_DLFCN\_H -DTERMIOS -DL\_ENDIAN -fomit-frame-pointer -O3 -march=i486 -Wall -DOPENSSL\_BN\_ASM\_PART\_WORDS -DOPENSSL\_IA32\_SSE2 -DSHA1\_ASM -DMD5\_ASM -DRMD160\_ASM -DAES\_ASM

available timing options: TIMES TIMEB HZ=1000 [sysconf value]

timing function used: times

The 'numbers' are in 1000s of bytes per second processed.

type 16 bytes 64 bytes 256 bytes 1024 bytes 8192 bytes

des cbc 55818.10k 58585.29k 60512.39k 60138.93k 60235.78k

1. **Benchmark RC4**

$ openssl speed rc4

Doing rc4 for 3s on 16 size blocks: 58319962 rc4's in 2.98s

Doing rc4 for 3s on 64 size blocks: 16475375 rc4's in 2.95s

Doing rc4 for 3s on 256 size blocks: 3894563 rc4's in 2.96s

Doing rc4 for 3s on 1024 size blocks: 861598 rc4's in 2.95s

Doing rc4 for 3s on 8192 size blocks: 98846 rc4's in 2.90s

OpenSSL 0.9.8t 18 Jan 2012

built on: Wed Jan 18 16:32:51 CET 2012

options:bn(64,32) md2(int) rc4(idx,int) des(ptr,risc1,16,long) aes(partial) blowfish(idx)

compiler: gcc -D\_WINDLL -DOPENSSL\_PIC -DZLIB -DOPENSSL\_THREADS -DDSO\_DLFCN -DHAVE\_DLFCN\_H -DTERMIOS -DL\_ENDIAN -fomit-frame-pointer -O3 -march=i486 -Wall -DOPENSSL\_BN\_ASM\_PART\_WORDS -DOPENSSL\_IA32\_SSE2 -DSHA1\_ASM -DMD5\_ASM -DRMD160\_ASM -DAES\_ASM

available timing options: TIMES TIMEB HZ=1000 [sysconf value]

timing function used: times

The 'numbers' are in 1000s of bytes per second processed.

type 16 bytes 64 bytes 256 bytes 1024 bytes 8192 bytes

rc4 313232.42k 357553.07k 336372.51k 299279.63k 279030.47k

1. **Encryption a file using DES-CBC mode**
   1. Go to folder lab2 and a plaintext.txt has been saved in the folder.

$ cd lab2

$ ls

plaintext.txt

* 1. Encrypt plaintext.txt using DES CBC mode and save output to a text file des-cipher.txt.

$ openssl enc -des-cbc -in plaintext.txt -out cipher-des.txt

enter des-cbc encryption password:

Verifying - enter des-cbc encryption password:

* 1. Decrypt des-cipher.txt using DES CBC mode and save output to a text file p1.txt

$ openssl enc -d -des-cbc -in cipher-des.txt -out p1.txt

enter des-cbc decryption password:

* 1. Compare plaintext.txt and p1.txt

$ diff plaintext.txt p1.txt

1. **Encryption plaintext using RC4**
   1. Encrypt plaintext.txt using rc4 and save output to a text file rc4-cipher.txt

$ openssl enc -rc4 -in plaintext.txt -out cipher-rc4.txt

enter rc4 encryption password:

Verifying - enter rc4 encryption password:

* 1. Decrypt rc4-cipher.txt and save output to a text file p2.txt

$ openssl enc -d -rc4 -in cipher-rc4.txt -out p2.txt

enter rc4 decryption password:

* 1. Compare plaintext.txt and p2.txt

$ diff plaintext.txt and p2.txt

# 4 Questions

1. (5pts) What is Heartbleed bug? How do you find out if your OpenSSL library is vulnerable to the Hearbleed bug? How to fix the vulnerability? (Hint: find more information about Heartbleed bug at http://heartbleed.com/)
2. (5pts) In Lab2 folder, des-cipher-cbc.txt is encrypted using DES in CBC mode. The password used for encryption is password. Decrypt the file and enclose the plaintext in your solution document.
3. (5pts) In Lab2 folder, rc4-cipher.txt is encrypted using RC4 algorithm. The password used for encryption is madison. Decrypt the file and enclose the plaintext in your solution document.
4. (10pts) In Lab2, you benchmarked DES CBC and RC4 ciphers. For 64 bits blocks, what’s the encryption/decryption throughput of DES CBC and RC4 ciphers in your testing environment? Which cipher is faster? Why?