IS2109 – Information System Security

University of Colombo, School of Computing

Index No : 22020357

1. Determine installed OpenSSL version:
   $ openssl version
   More information about OpenSSL
   $ openssl version -a

```
root@6c188931e234:/home# openssl version
OpenSSL 3.0.13 30 Jan 2024 (Library: OpenSSL 3.0.13 30 Jan 2024)
```

```
root@6c188931e234:/home#  openssl version -a
OpenSSL 3.0.13 30 Jan 2024 (Library: OpenSSL 3.0.13 30 Jan 2024)
built on: Tue Aug 20 17:05:36 2024 UTC
platform: debian-amd64
options:  bn(64,64)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -fzero-call-used-regs=used-gpr -DOPENSSL_TLS_SECURITY_LEVEL=2 -Wa,--noexecs
tack -g -O2 -fno-omit-frame-pointer -mno-omit-leaf-frame-pointer -ffile-prefix-map=/build/openssl-WgbqpY/openssl-3.0.13=. -fstack-pro
tector-strong -fstack-clash-protection -Wformat -Werror=format-security -fcf-protection -fdebug-prefix-map=/build/openssl-WgbqpY/open
ssl-3.0.13=/usr/src/openssl-3.0.13-0ubuntu3.4 -DOPENSSL_USE_NODELETE -DL_ENDIAN -DOPENSSL_PIC -DOPENSSL_BUILDING_OPENSSL -DNDEBUG -Wd
ate-time -D_FORTIFY_SOURCE=3
OPENSSLDIR: "/usr/lib/ssl"
ENGINESDIR: "/usr/lib/x86_64-linux-gnu/engines-3"
MODULESDIR: "/usr/lib/x86_64-linux-gnu/ossl-modules"
Seeding source: os-specific
CPUINFO: OPENSSL_ia32cap=0xfffab2235f8bffff:0x18405fc6f1bf27ab
```

2. List of available OpenSSL sub-commands:
   $ openssl help

```
root@6c188931e234:/home# openssl help
help:

Standard commands
asn1parse         ca                ciphers           cmp

cms               crl               crl2pkcs7         dgst

dhparam           dsa               dsaparam          ec

ecparam           enc               engine            errstr

fipsinstall       gendsa            genpkey           genrsa

help              info              kdf               list

mac               nseq              ocsp              passwd

pkcs12            pkcs7             pkcs8             pkey

pkeyparam         pkeyutl           prime             rand

rehash            req               rsa               rsautl

s_client          s_server          s_time            sess_id

smime             speed             spkac             srp

storeutl          ts                verify            version

x509

Message Digest commands (see the `dgst' command for more details)
blake2b512        blake2s256        md4               md5

rmd160            sha1              sha224            sha256

sha3-224          sha3-256          sha3-384          sha3-512
```

3. Additional information sub-commands
   $ openssl enc -h

```
root@6c188931e234:/home# openssl enc -h
Usage: enc [options]

General options:
 -help               Display this summary
 -list               List ciphers
 -ciphers            Alias for -list
 -e                  Encrypt
 -d                  Decrypt
 -p                  Print the iv/key
 -P                  Print the iv/key and exit
 -engine val         Use engine, possibly a hardware device

Input options:
 -in infile          Input file
 -k val              Passphrase
 -kfile infile       Read passphrase from file

Output options:
 -out outfile        Output file
 -pass val           Passphrase source
 -v                  Verbose output
 -a                  Base64 encode/decode, depending on encryption flag
 -base64             Same as option -a
 -A                  Used with -[base64|a] to specify base64 buffer as a single line

Encryption options:
 -nopad              Disable standard block padding
 -salt               Use salt in the KDF (default)
 -nosalt             Do not use salt in the KDF
 -debug              Print debug info
 -bufsize val        Buffer size
 -K val              Raw key, in hex
 -S val              Salt, in hex
 -iv val             IV in hex
 -md val             Use specified digest to create a key from the passphrase
 -iter +int          Specify the iteration count and force the use of PBKDF2
                     Default: 10000
 -pbkdf2             Use password-based key derivation function 2 (PBKDF2)
                     Use -iter to change the iteration count from 10000
```

4. List all available cipher algorithms
   $ openssl ciphers -v

```
root@6c188931e234:/home# openssl ciphers -v
TLS_AES_256_GCM_SHA384          TLSv1.3 Kx=any      Au=any   Enc=AESGCM(256)            Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256    TLSv1.3 Kx=any      Au=any   Enc=CHACHA20/POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256          TLSv1.3 Kx=any      Au=any   Enc=AESGCM(128)            Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384   TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(256)            Mac=AEAD
ECDHE-RSA-AES256-GCM-SHA384     TLSv1.2 Kx=ECDH     Au=RSA   Enc=AESGCM(256)            Mac=AEAD
DHE-RSA-AES256-GCM-SHA384       TLSv1.2 Kx=DH       Au=RSA   Enc=AESGCM(256)            Mac=AEAD
ECDHE-ECDSA-CHACHA20-POLY1305   TLSv1.2 Kx=ECDH     Au=ECDSA Enc=CHACHA20/POLY1305(256) Mac=AEAD
ECDHE-RSA-CHACHA20-POLY1305     TLSv1.2 Kx=ECDH     Au=RSA   Enc=CHACHA20/POLY1305(256) Mac=AEAD
DHE-RSA-CHACHA20-POLY1305       TLSv1.2 Kx=DH       Au=RSA   Enc=CHACHA20/POLY1305(256) Mac=AEAD
ECDHE-ECDSA-AES128-GCM-SHA256   TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(128)            Mac=AEAD
ECDHE-RSA-AES128-GCM-SHA256     TLSv1.2 Kx=ECDH     Au=RSA   Enc=AESGCM(128)            Mac=AEAD
DHE-RSA-AES128-GCM-SHA256       TLSv1.2 Kx=DH       Au=RSA   Enc=AESGCM(128)            Mac=AEAD
ECDHE-ECDSA-AES256-SHA384       TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(256)               Mac=SHA384
ECDHE-RSA-AES256-SHA384         TLSv1.2 Kx=ECDH     Au=RSA   Enc=AES(256)               Mac=SHA384
DHE-RSA-AES256-SHA256           TLSv1.2 Kx=DH       Au=RSA   Enc=AES(256)               Mac=SHA256
ECDHE-ECDSA-AES128-SHA256       TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(128)               Mac=SHA256
ECDHE-RSA-AES128-SHA256         TLSv1.2 Kx=ECDH     Au=RSA   Enc=AES(128)               Mac=SHA256
DHE-RSA-AES128-SHA256           TLSv1.2 Kx=DH       Au=RSA   Enc=AES(128)               Mac=SHA256
ECDHE-ECDSA-AES256-SHA          TLSv1   Kx=ECDH     Au=ECDSA Enc=AES(256)               Mac=SHA1
ECDHE-RSA-AES256-SHA            TLSv1   Kx=ECDH     Au=RSA   Enc=AES(256)               Mac=SHA1
DHE-RSA-AES256-SHA              SSLv3   Kx=DH       Au=RSA   Enc=AES(256)               Mac=SHA1
ECDHE-ECDSA-AES128-SHA          TLSv1   Kx=ECDH     Au=ECDSA Enc=AES(128)               Mac=SHA1
ECDHE-RSA-AES128-SHA            TLSv1   Kx=ECDH     Au=RSA   Enc=AES(128)               Mac=SHA1
DHE-RSA-AES128-SHA              SSLv3   Kx=DH       Au=RSA   Enc=AES(128)               Mac=SHA1
RSA-PSK-AES256-GCM-SHA384       TLSv1.2 Kx=RSAPSK   Au=RSA   Enc=AESGCM(256)            Mac=AEAD
DHE-PSK-AES256-GCM-SHA384       TLSv1.2 Kx=DHEPSK   Au=PSK   Enc=AESGCM(256)            Mac=AEAD
RSA-PSK-CHACHA20-POLY1305       TLSv1.2 Kx=RSAPSK   Au=RSA   Enc=CHACHA20/POLY1305(256) Mac=AEAD
DHE-PSK-CHACHA20-POLY1305       TLSv1.2 Kx=DHEPSK   Au=PSK   Enc=CHACHA20/POLY1305(256) Mac=AEAD
ECDHE-PSK-CHACHA20-POLY1305     TLSv1.2 Kx=ECDHEPSK Au=PSK   Enc=CHACHA20/POLY1305(256) Mac=AEAD
AES256-GCM-SHA384               TLSv1.2 Kx=RSA      Au=RSA   Enc=AESGCM(256)            Mac=AEAD
PSK-AES256-GCM-SHA384           TLSv1.2 Kx=PSK      Au=PSK   Enc=AESGCM(256)            Mac=AEAD
PSK-CHACHA20-POLY1305           TLSv1.2 Kx=PSK      Au=PSK   Enc=CHACHA20/POLY1305(256) Mac=AEAD
RSA-PSK-AES128-GCM-SHA256       TLSv1.2 Kx=RSAPSK   Au=RSA   Enc=AESGCM(128)            Mac=AEAD
DHE-PSK-AES128-GCM-SHA256       TLSv1.2 Kx=DHEPSK   Au=PSK   Enc=AESGCM(128)            Mac=AEAD
AES128-GCM-SHA256               TLSv1.2 Kx=RSA      Au=RSA   Enc=AESGCM(128)            Mac=AEAD
PSK-AES128-GCM-SHA256           TLSv1.2 Kx=PSK      Au=PSK   Enc=AESGCM(128)            Mac=AEAD
AES256-SHA256                   TLSv1.2 Kx=RSA      Au=RSA   Enc=AES(256)               Mac=SHA256
AES128-SHA256                   TLSv1.2 Kx=RSA      Au=RSA   Enc=AES(128)               Mac=SHA256
ECDHE-PSK-AES256-CBC-SHA384     TLSv1   Kx=ECDHEPSK Au=PSK   Enc=AES(256)               Mac=SHA384
ECDHE-PSK-AES256-CBC-SHA        TLSv1   Kx=ECDHEPSK Au=PSK   Enc=AES(256)               Mac=SHA1
```

5. Benchmark your computer's speed with OpenSSL
   $ openssl speed

```
root@6c188931e234:/home# openssl speed
Doing md5 for 3s on 16 size blocks: 3973542 md5's in 3.01s
Doing md5 for 3s on 64 size blocks: 2774219 md5's in 3.01s
Doing md5 for 3s on 256 size blocks: 1938029 md5's in 3.00s
Doing md5 for 3s on 1024 size blocks: 799120 md5's in 3.01s
Doing md5 for 3s on 8192 size blocks: 135879 md5's in 3.00s
Doing md5 for 3s on 16384 size blocks: 51308 md5's in 2.99s
Doing sha1 for 3s on 16 size blocks: 3010419 sha1's in 2.99s
Doing sha1 for 3s on 64 size blocks: 2705511 sha1's in 3.00s
Doing sha1 for 3s on 256 size blocks: 2008697 sha1's in 3.00s
Doing sha1 for 3s on 1024 size blocks: 1028799 sha1's in 3.01s
Doing sha1 for 3s on 8192 size blocks: 175835 sha1's in 3.01s
Doing sha1 for 3s on 16384 size blocks: 92507 sha1's in 2.99s
Doing sha256 for 3s on 16 size blocks: 3683479 sha256's in 3.00s
Doing sha256 for 3s on 64 size blocks: 3528335 sha256's in 3.00s
Doing sha256 for 3s on 256 size blocks: 1938518 sha256's in 2.98s
Doing sha256 for 3s on 1024 size blocks: 911393 sha256's in 3.00s
Doing sha256 for 3s on 8192 size blocks: 138342 sha256's in 3.01s
Doing sha256 for 3s on 16384 size blocks: ^C
```

6. How do I create an MD5 SHA1 or SHA256 digest of a file?
   I. Create a text file with dummy data as follows:

   $ echo “Hello. This is our super secret message. Keep it secret please. Goodbye.” > secret.txt

```
root@6c188931e234:/home# echo "Hello. This is our super secret message. Keep it secret please. Goodbye."> secret.txt
```

   II. Calculate the MD5 digest
   $ openssl dgst -md5 <filename>

```
root@6c188931e234:/home# openssl dgst -md5 secret.txt
MD5(secret.txt)= de230762fc89ec93a4ed12b45a9396df
```

   III. SHA1 digest
   $ openssl dgst -sha1 <filename>

```
root@6c188931e234:/home# openssl dgst -sha1 secret.txt
SHA1(secret.txt)= db130b3be3c4ee747b25126f05d1b0c503e08eda
```

   IV. SHA256 digest
   $ openssl dgst -sha256 <filename>

```
root@6c188931e234:/home# openssl dgst -sha256 secret.txt
SHA2-256(secret.txt)= dcd05a8e9cd4c29b7394404b6a4bdefdb8df374b8aaa7f43de65449d9f3bfbe4
```

7. How do I encrypt a file?
   $ openssl enc -aes-256-cbc -in <filename> -out <filename>

```
root@6c188931e234:/home# openssl enc -aes-256-cbc -in secret.txt -out secret
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

8. How do I decrypt a file?
   $ openssl enc -d -aes-256-cbc -in <filename>

```
root@6c188931e234:/home# openssl enc -d -aes-256-cbc -in secret
enter AES-256-CBC decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
Hello. This is our super secret message. Keep it secret please. Goodbye.
```

9. Generate Public/Private key pair
   $ openssl genrsa -out <keyfile>
   $ openssl rsa -in <keyfile> -pubout > <publickeyfile>

```
root@6c188931e234:/home# openssl genrsa -out private_key.pem 204
8
```

```
root@6c188931e234:/home# openssl rsa -in private_key.pem -pubout -out public_key.pem
writing RSA key
```

10. Calculating the digital signature
    $ openssl dgst -sha1 -sign <keyfile> -out <signature> <filename>

```
root@6c188931e234:/home# openssl dgst -sha1 -sign private_key.pem -out signature.bin secret.txt
```

11. Verifying the digital signature.
    $ openssl dgst -sha1 -verify <publickeyfile> -signature <signature> <filename>

```
root@6c188931e234:/home# openssl dgst -sha1 -verify public_key.pem -signature signature.bin secret.txt
Verified OK
```