

# CRYPTO LAB ASSIGNMENT

09/10/2016

## Task 1:

In this task, we made ourselves a root CA, and generated a self-signed certificate.

## Task 2:

The root CA created in the first task provides a digital signed certificate to the client called PKILAB-Server.com

## Task 3:

In this task, we came to know how public-key certificates are used by web sites to secure web browsing. After we upload our root CA certificate into the web browser we observed that the browser is able to access the server. Initially it showed us the message "pkilabserver.com:4433 uses an invalid security certificate. The certificate is not trusted because the issuer certificate is unknown .

1. Server.pem contains a)Private key b)Certificate c)Modulus d)signature algorithm. If we modify a single byte in **privte key or certificate** the web browser is unable to access the server. It is showing that unable to load the private key and if we change in certificate it is displaying unable to load certificate. But even though we change a byte in **modulus or signature algorithm** web browser is able to access the server.
2. Both the PKILabServer.com and localhost are pointing to the same address. From this we can say that we can have many number of domain names for a single address. For example we can use facebook.com or fb.com to access the facebook website.

## Task 4:

The effective date is verified by the following code

```
server_cert=SSL_get_peer_certificate(ssl);
```

Whether the server certificate is signed by the authorized CA is checked using the following code

```
void check_cert(SSL *ssl,char *host) X509 *peer;  
char peer_CN[256];  
if(SSL_get_verify_result(ssl)!=X509_V_OK)  
puts("Certificate doesn't verify");  
exit(4);
```

Whether the certificate belongs to the server is verified using the below code.

```
peer=SSL_get_peer_certificate(ssl);  
X509_NAME_get_text_by_NID (X509_get_subject_name(peer), NID_commonName, peer_CN,  
256);
```

```

if(strcasecmp(peer_CN, host))
puts("Common name doesn't match host name");
Whether the server is indeed the machine that the client wants to talk to is verified by following code
check_cert(ssl, "PKILabServer.com");
The client certificate is not necessary to be verified by the server. If client asks some confidential data then
the client certificate is to be verified. There will be so many client requests so if server begins to check each
client certificate its performance is decreased. So we could remove the below code from the server side
client_cert = SSL_get_peer_certificate (ssl);
str = X509_NAME_oneline (X509_get_subject_name (client_cert), 0, 0)
str = X509_NAME_oneline (X509_get_issuer_name (client_cert), 0, 0);
The part of the code responsible for the key exchange, i.e. for both sides to agree upon a secret key is as
follows
SSL_set_ex_data(ssl, mydata_index, &mydata);

```

## **Task 5**

Performing 10000 times aes took 0.112sec for me and speed test showed it takes 0.11757sec. Next, for RSA I am observing a lot of difference it took me 0.10 sec for both encryption and decryption while system speed test showed that it takes 9.33sec for both encrypting and decrypting.

## **Task 6**

Following commands are used to sign SHA256 of example.txt

```

openssl genrsa -out my.pem 1024
openssl rsa -in my.pem -pubout > my.pub
openssl dgst -sha256 example.txt | cut -d= -f2 > hash
openssl rsautl -sign
-inkey key.pem -keyform PEM -in hash > example.sha256
Verification is done by the following commands
openssl rsautl -verify -inkey key.pem -keyform PEM -in example.sha256
When we slightly modified the example.txt we were not able to verify the signature.

```

## TASK 1 screenshot1

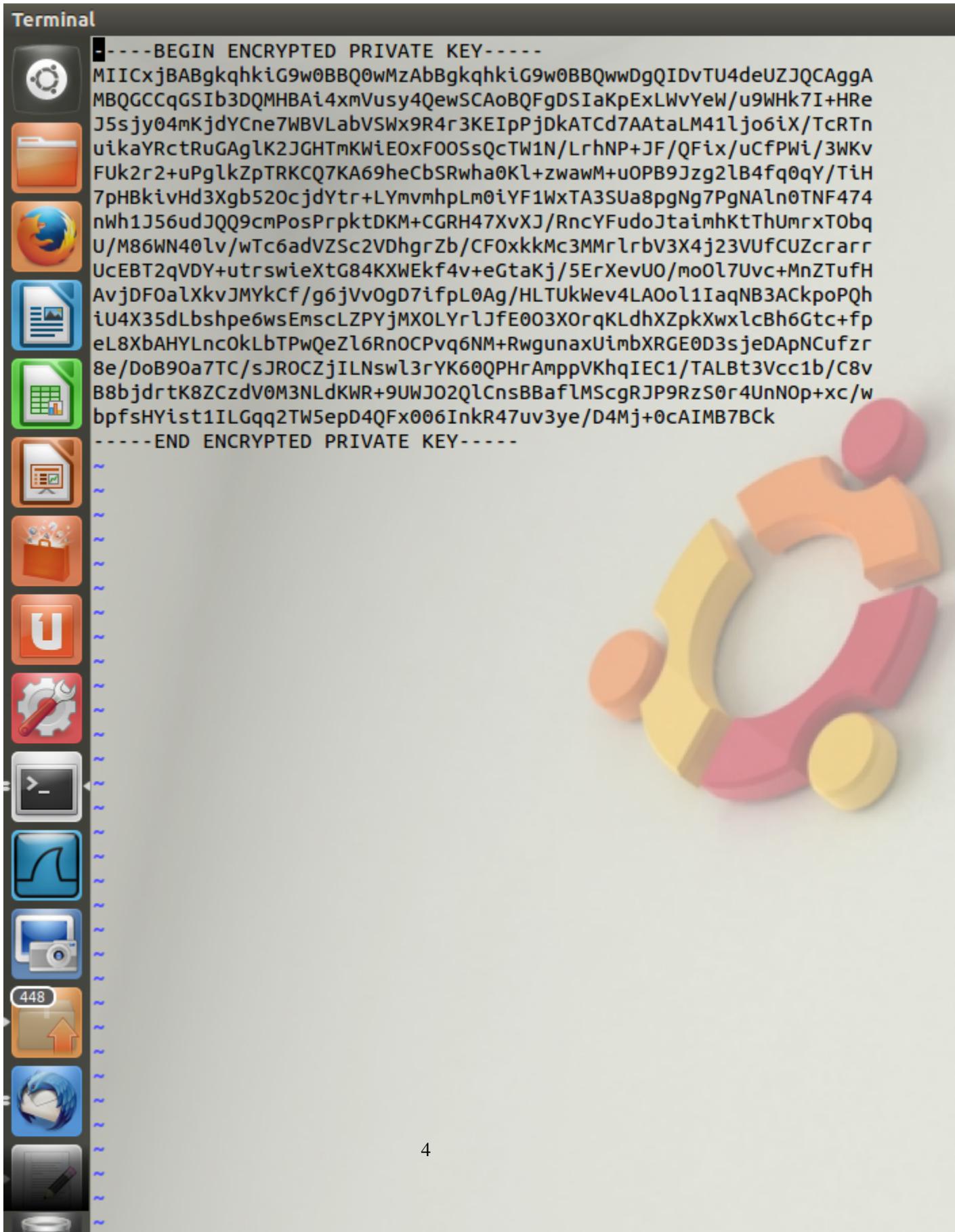
Terminal

```
-----BEGIN CERTIFICATE-----  
MIIC0DCCAjmgAwIBAgIJA0a3aVV08/X9MA0GCSqGSIb3DQEBBQUAMIGAMQswCQYD  
VQQGEwJ1czELMAkGA1UECAwCZmwxCzAJBgNVBAcMAmdhMQswCQYDVQQKDAJ1ZjEM  
MAoGA1UECwwDY25zMRAwDgYDVQQDDAdoYXJlZXNoMSowKAYJKoZIhvcNAQkBFhtu  
dXRhbGFwYXRpaGFyZWVzaEBnbWFpbC5jb20wHhcNMTYwOTEwMDUxNTIwWhcNMTYx  
MDEwMDUxNTIwWjCBgDELMAkGA1UEBhMCdxMxCzAJBgNVBAgMAMZsMQswCQYDVQQH  
DAJnYTELMAkGA1UECgwCdWYxDDAKBgNVBAsMA2NuczEQMA4GA1UEAwHaGFyZWVz  
aDEqMCgGCSqGSIb3DQEJARYbbnV0YWxhcGF0aWhhcmVlc2hAZ21haWwuY29tMIGf  
MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDb0UIf1EqppvNpqH5ggsHk47g0jbUP  
mXYMSyMnjL87vQ+Objk0D03wCTcjJMoiCKddKVjKRfdUaZRwC/P8rZ8bGevWIikx  
pQ37jHWq3cgwbQMxRteVFv10JhzqK1PyKq0ZDS73JR60/nr8Ilc22qXAUwCt1l20  
nh99a0l7QJP3RQIDAQABo1AwTjAdBgNVHQ4EFgQUl1amllHjccfuflwnFmNjEb9y  
qrQwHwYDVR0jBBgwFoAUL1amllHjccfuflwnFmNjEb9yqrQwDAYDVR0TBAtAwEB  
/zANBqkqhkiG9w0BAQUFAAOBgQCkwOKjGGbT4T0yqbaeWsl4VQ2rC2hNK6kcbSiz  
/gudLYz1zx01i247V4Ix1LNTguqAl21jka01MUioTSREaixxCjrKPFXNOF203Rvj  
s+qa8irF21f00BkZ/y20epjGxNAZAGl3Js0g6Px19mtojgtFklsDlsqPzRL30qxR  
URBmCg==  
-----END CERTIFICATE-----
```



The terminal window shows a certificate chain. The first certificate is displayed in its entirety, while subsequent certificates are shown as truncated lines starting with a tilde (~). The Ubuntu logo is visible in the background.

## TASK 1 screenshot2



TASK 3 screenshot1

Mozilla Firefox

https://pkilabserver.com:4433/

Most Visited Getting Started Seed Labs

```
s_server -cert server.pem -www
Ciphers supported in s_server binary
TLSv1/SSLv3:ECDHE-RSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDHE-ECDSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDHE-RSA-AES256-SHA384 TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA384
TLSv1/SSLv3:ECDHE-RSA-AES256-SHA TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA
TLSv1/SSLv3:SRP-DSS-AES-256-CBC-SHA TLSv1/SSLv3:SRP-RSA-AES-256-CBC-SHA
TLSv1/SSLv3:DHE-DSS-AES256-GCM-SHA384TLSv1/SSLv3:DHE-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:DHE-RSA-AES256-SHA256 TLSv1/SSLv3:DHE-DSS-AES256-SHA256
TLSv1/SSLv3:DHE-RSA-AES256-SHA TLSv1/SSLv3:DHE-DSS-AES256-SHA
TLSv1/SSLv3:DHE-RSA-CAMELLIA256-SHA TLSv1/SSLv3:DHE-DSS-CAMELLIA256-SHA
TLSv1/SSLv3:ECDH-RSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDH-ECDSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDH-RSA-AES256-SHA384 TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA384
TLSv1/SSLv3:ECDH-RSA-AES256-SHA TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA
TLSv1/SSLv3:AES256-GCM-SHA384 TLSv1/SSLv3:AES256-SHA256
TLSv1/SSLv3:AES256-SHA TLSv1/SSLv3:CAMELLIA256-SHA
TLSv1/SSLv3:PSK-AES256-CBC-SHA TLSv1/SSLv3:ECDHE-RSA-DES-CBC3-SHA
TLSv1/SSLv3:ECDHE-ECDSA-DES-CBC3-SHA TLSv1/SSLv3:SRP-DSS-3DES-EDE-CBC-SHA
TLSv1/SSLv3:SRP-RSA-3DES-EDE-CBC-SHA TLSv1/SSLv3:EDH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:EDH-DSS-DES-CBC3-SHA TLSv1/SSLv3:ECDH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:ECDH-ECDSA-DES-CBC3-SHA TLSv1/SSLv3:DES-CBC3-SHA
TLSv1/SSLv3:PSK-3DES-EDE-CBC-SHA TLSv1/SSLv3:ECDHE-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:ECDHE-ECDSA-AES128-GCM-SHA256TLSv1/SSLv3:ECDHE-RSA-AES128-SHA256
TLSv1/SSLv3:ECDHE-ECDSA-AES128-SHA256TLSv1/SSLv3:ECDHE-RSA-AES128-SHA
TLSv1/SSLv3:SRP-RSA-AES-128-CBC-SHA TLSv1/SSLv3:DHE-DSS-AES128-GCM-SHA256
TLSv1/SSLv3:DHE-RSA-AES128-GCM-SHA256TLSv1/SSLv3:DHE-RSA-AES128-SHA256
TLSv1/SSLv3:DHE-DSS-AES128-SHA256 TLSv1/SSLv3:DHE-RSA-AES128-SHA
TLSv1/SSLv3:DHE-DSS-AES128-SHA TLSv1/SSLv3:DHE-RSA-SEED-SHA
TLSv1/SSLv3:DHE-DSS-SEED-SHA TLSv1/SSLv3:DHE-RSA-CAMELLIA128-SHA
TLSv1/SSLv3:DHE-DSS-CAMELLIA128-SHA TLSv1/SSLv3:ECDH-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:ECDH-ECDSA-AES128-GCM-SHA256TLSv1/SSLv3:ECDH-RSA-AES128-SHA256
TLSv1/SSLv3:ECDH-ECDSA-AES128-SHA256 TLSv1/SSLv3:ECDH-RSA-AES128-SHA
TLSv1/SSLv3:ECDH-ECDSA-AES128-SHA TLSv1/SSLv3:AES128-GCM-SHA256
TLSv1/SSLv3:AES128-SHA256 TLSv1/SSLv3:AES128-SHA
TLSv1/SSLv3:SEED-SHA TLSv1/SSLv3:CAMELLIA128-SHA
TLSv1/SSLv3:PSK-AES128-CBC-SHA TLSv1/SSLv3:ECDHE-RSA-RC4-SHA
TLSv1/SSLv3:ECDHE-ECDSA-RC4-SHA TLSv1/SSLv3:ECDH-RSA-RC4-SHA
TLSv1/SSLv3:ECDH-ECDSA-RC4-SHA TLSv1/SSLv3:RC4-SHA
TLSv1/SSLv3:RC4-MD5 TLSv1/SSLv3:PSK-RC4-SHA
TLSv1/SSLv3:EDH-RSA-DES-CBC-SHA TLSv1/SSLv3:EDH-DSS-DES-CBC-SHA
TLSv1/SSLv3:DES-CBC-SHA TLSv1/SSLv3:EXP-EDH-RSA-DES-CBC-SHA
TLSv1/SSLv3:EXP-EDH-DSS-DES-CBC-SHA TLSv1/SSLv3:EXP-DES-CBC-SHA
TLSv1/SSLv3:EXP-RC2-CBC-MD5 TLSv1/SSLv3:EXP-RC4-MD5
---
Ciphers common between both SSL end points:
ECDHE-ECDSA-AES256-SHA ECDHE-RSA-AES256-SHA DHE-RSA-CAMELLIA256-SHA
DHE-DSS-CAMELLIA256-SHA DHE-RSA-AES256-SHA DHE-DSS-AES256-SHA
ECDH-RSA-AES256-SHA ECDH-ECDSA-AES256-SHA CAMELLIA256-SHA
AES256-SHA ECDHE-ECDSA-RC4-SHA ECDHE-ECDSA-AES128-SHA
ECDHE-RSA-RC4-SHA ECDHE-RSA-AES128-SHA DHE-RSA-CAMELLIA128-SHA
DHE-DSS-CAMELLIA128-SHA DHE-RSA-AES128-SHA DHE-DSS-AES128-SHA
```

TASK 4 screenshot

Terminal

```
Terminal
[09/12/2016 13:10] seed@ubuntu:~/assign$ ./cli
RETURN CA CHECK:1
Enter PEM pass phrase:
veryfing
depth=1:/C=us/ST=fl/L=ga/O=uf/OU=cns/CN=hareesh/emailAddress=natalapatih
veryfing
depth=0:/C=us/ST=fl/O=uf/OU=cns/CN=PKILabServer.com/emailAddress=hareesh
SSL connection using AES256-GCM-SHA384
Server certificate:
    subject: /C=us/ST=fl/O=uf/OU=cns/CN=PKILabServer.com/emailAddress=hareesh
    issuer: /C=us/ST=fl/L=ga/O=uf/OU=cns/CN=hareesh/emailAddress=natalapatih
OK VERIFIED
line 240
[09/12/2016 13:11] seed@ubuntu:~/assign$ █
```



## TASK 5 screenshot1

Terminal

```
[09/12/2016 11:49] seed@ubuntu:~/assign$ time sh enc.sh
real    0m0.055s
user    0m0.004s
sys     0m0.032s
[09/12/2016 11:50] seed@ubuntu:~/assign$ time sh dec.sh
real    0m0.092s
user    0m0.008s
sys     0m0.056s
[09/12/2016 11:50] seed@ubuntu:~/assign$ time sh aes.sh
real    0m0.144s
user    0m0.000s
sys     0m0.112s
[09/12/2016 11:50] seed@ubuntu:~/assign$ █
```

## TASK 5 screenshot2

```
Terminal
sys      0m0.112s
[09/12/2016 11:50] seed@ubuntu:~/assign$ openssl speed rsa
Doing 512 bit private rsa's for 10s: 61865 512 bit private RSA's in 9.04s
Doing 512 bit public rsa's for 10s: 672925 512 bit public RSA's in 9.78s
Doing 1024 bit private rsa's for 10s: 10873 1024 bit private RSA's in 9.64s
Doing 1024 bit public rsa's for 10s: 194713 1024 bit public RSA's in 9.79s
Doing 2048 bit private rsa's for 10s: 1588 2048 bit private RSA's in 9.73s
Doing 2048 bit public rsa's for 10s: 52684 2048 bit public RSA's in 9.75s
Doing 4096 bit private rsa's for 10s: 215 4096 bit private RSA's in 9.75s
Doing 4096 bit public rsa's for 10s: 13721 4096 bit public RSA's in 9.77s
OpenSSL 1.0.1 14 Mar 2012
built on: Tue Jun 4 07:25:48 UTC 2013
options:bn(64,32) rc4(8x,mmx) des(ptr,risc1,16,long) aes(partial) blowfish
compiler: cc -fPIC -DOPENSSL_PIC -DZLIB -DOPENSSL_THREADS -D_REENTRANT -D
format-security -D_FORTIFY_SOURCE=2 -Wl,-Bsymbolic-functions -Wl,-z,relro
SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM
sign verify sign/s verify/s
rsa 512 bits 0.000146s 0.000015s 6843.5 68806.2
rsa 1024 bits 0.000884s 0.000050s 1131.4 19889.0
rsa 2048 bits 0.006127s 0.000185s 163.2 5403.5
rsa 4096 bits 0.045349s 0.000712s 22.1 1404.4
[09/12/2016 11:53] seed@ubuntu:~/assign$ open speed aes
Couldn't get a file descriptor referring to the console
[09/12/2016 11:55] seed@ubuntu:~/assign$ openssl speed aes
Doing aes-128 cbc for 3s on 16 size blocks: 14062149 aes-128 cbc's in 2.92s
Doing aes-128 cbc for 3s on 64 size blocks: 3909370 aes-128 cbc's in 2.92s
Doing aes-128 cbc for 3s on 256 size blocks: 983356 aes-128 cbc's in 2.92s
Doing aes-128 cbc for 3s on 1024 size blocks: 248355 aes-128 cbc's in 2.92s
Doing aes-128 cbc for 3s on 8192 size blocks: 32203 aes-128 cbc's in 2.92s
Doing aes-192 cbc for 3s on 16 size blocks: 12237404 aes-192 cbc's in 2.92s
Doing aes-192 cbc for 3s on 64 size blocks: 3236857 aes-192 cbc's in 2.92s
Doing aes-192 cbc for 3s on 256 size blocks: 821333 aes-192 cbc's in 2.92s
Doing aes-192 cbc for 3s on 1024 size blocks: 210065 aes-192 cbc's in 2.92s
Doing aes-192 cbc for 3s on 8192 size blocks: 25529 aes-192 cbc's in 2.92s
Doing aes-256 cbc for 3s on 16 size blocks: 10278368 aes-256 cbc's in 2.92s
Doing aes-256 cbc for 3s on 64 size blocks: 2788811 aes-256 cbc's in 2.92s
Doing aes-256 cbc for 3s on 256 size blocks: 692079 aes-256 cbc's in 2.92s
Doing aes-256 cbc for 3s on 1024 size blocks: 168322 aes-256 cbc's in 2.75s
Doing aes-256 cbc for 3s on 8192 size blocks: 20624 aes-256 cbc's in 2.75s
OpenSSL 1.0.1 14 Mar 2012
built on: Tue Jun 4 07:25:48 UTC 2013
options:bn(64,32) rc4(8x,mmx) des(ptr,risc1,16,long) aes(partial) blowfish
compiler: cc -fPIC -DOPENSSL_PIC -DZLIB -DOPENSSL_THREADS -D_REENTRANT -D
format-security -D_FORTIFY_SOURCE=2 -Wl,-Bsymbolic-functions -Wl,-z,relro
SSE2 -DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM
The 'numbers' are in 1000s of bytes per second processed.
type          16 bytes    64 bytes   256 bytes  1024 bytes   8192 bytes
aes-128 cbc   77584.27k   85684.82k   86212.03k   87094.36k   9034
aes-192 cbc   67054.27k   71434.09k   71761.52k   73666.63k   7162
```

## TASK 6 screenshot

