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

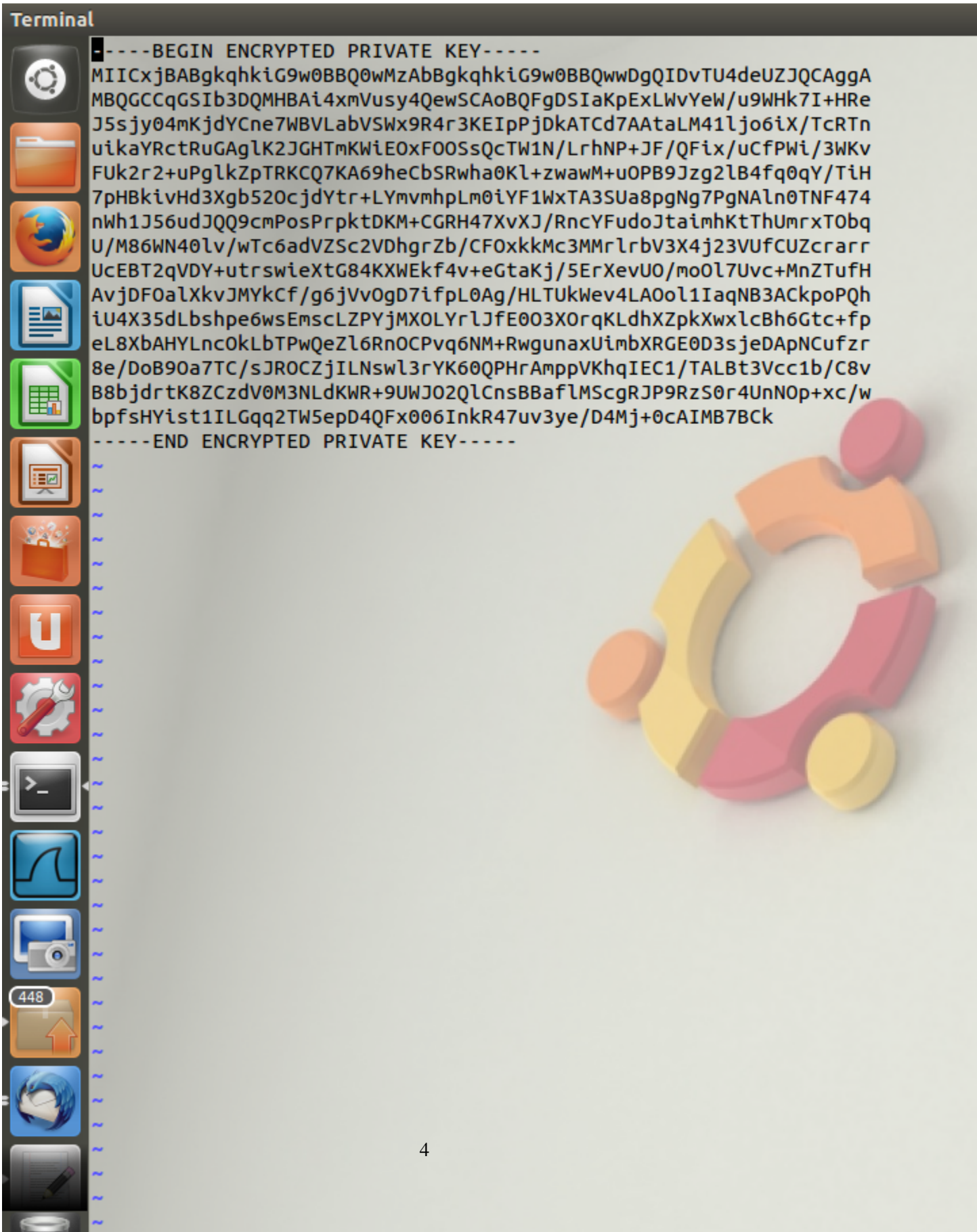
Performing 10000 times aes took 0.112sec for me and speed test showed it takes 0.11757sec. Next, for RSA I am observing a lot of difference it took me 0.10 sec for both encryption and decryption while system speed test showed that it takes 9.33sec for both encrypting and decrypting.

Task 6

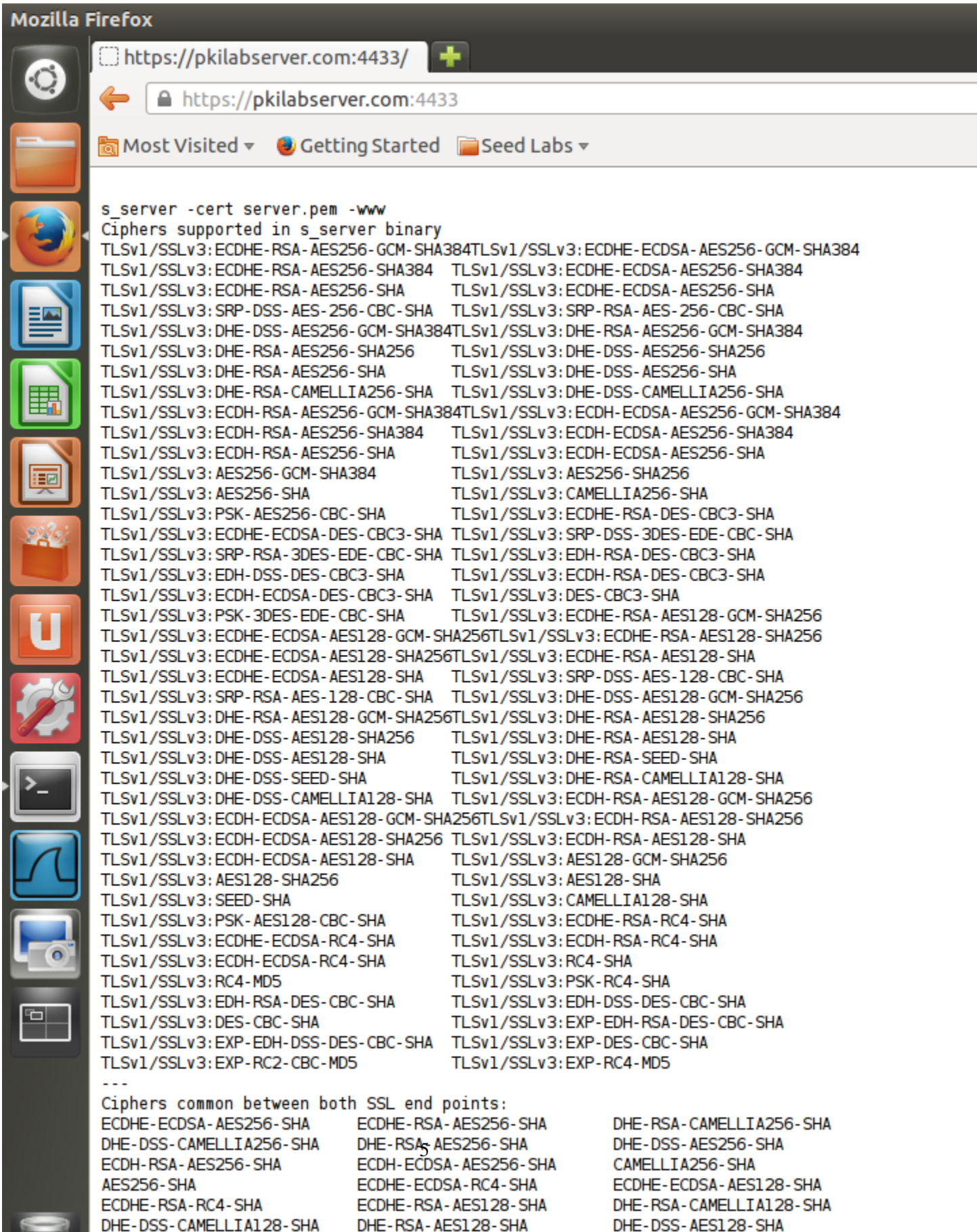
Following commands are used to sign SHA256 of example.txt
```
openssl genrsa -out my.pem 1024
openssl rsa -in my.pem -pubout > my.pub
openssl dgst -sha256 example.txt | cut -d= -f2 > hash openssl rsautl -sign
-inkey key.pem -keyform PEM -in hash > example.sha256
```
Verification is done by the following 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

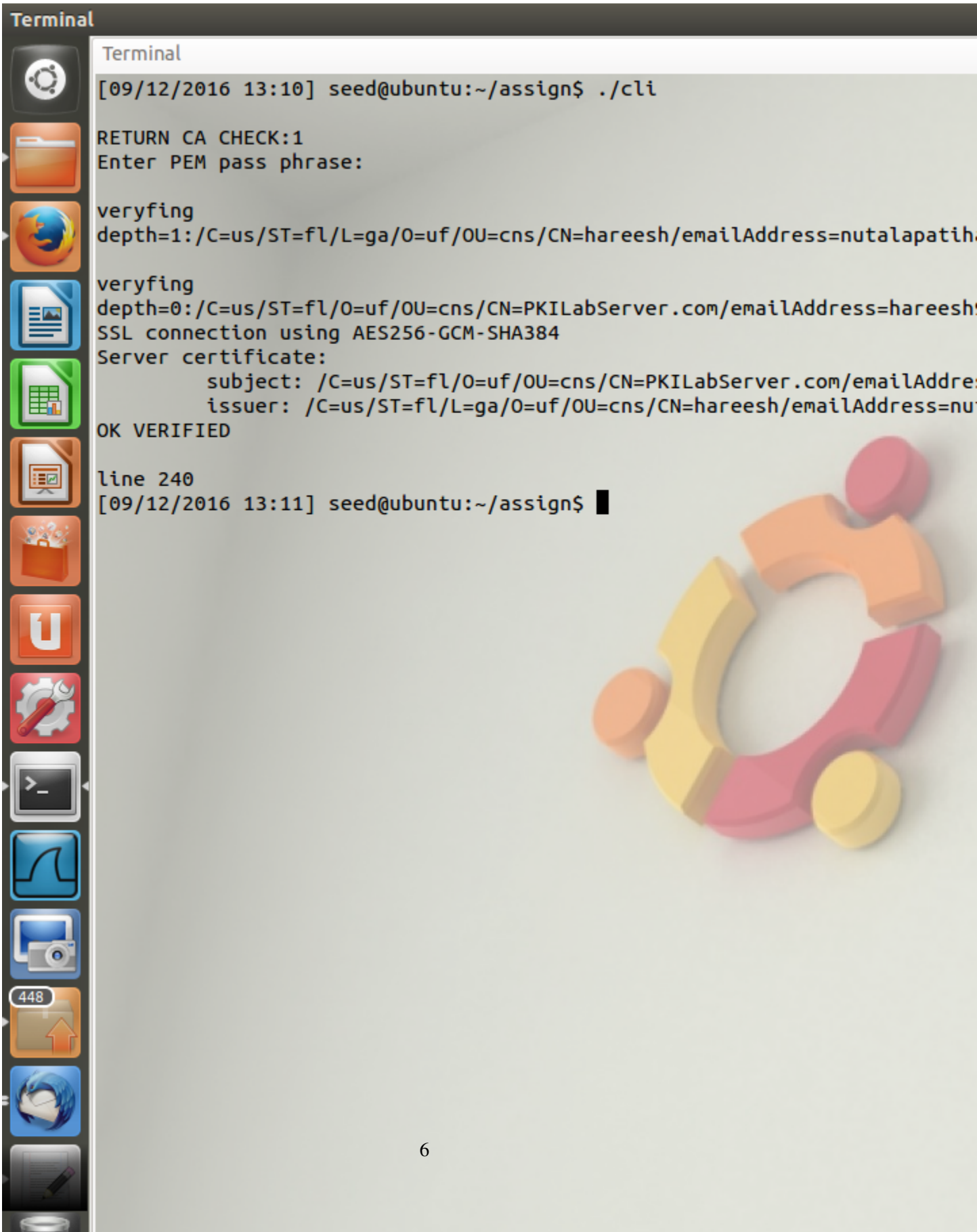
TASK 1 screenshot2



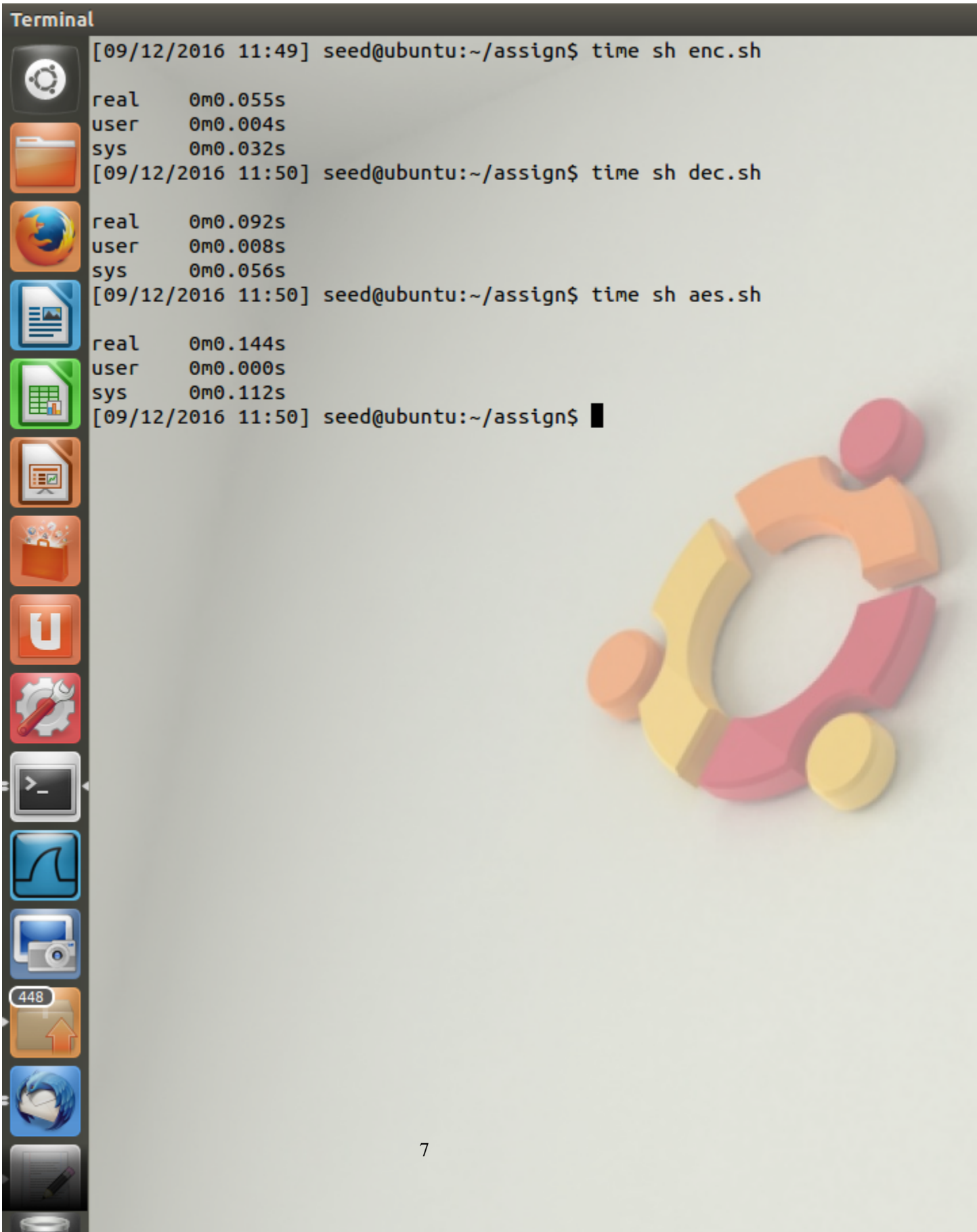
TASK 3 screenshot1



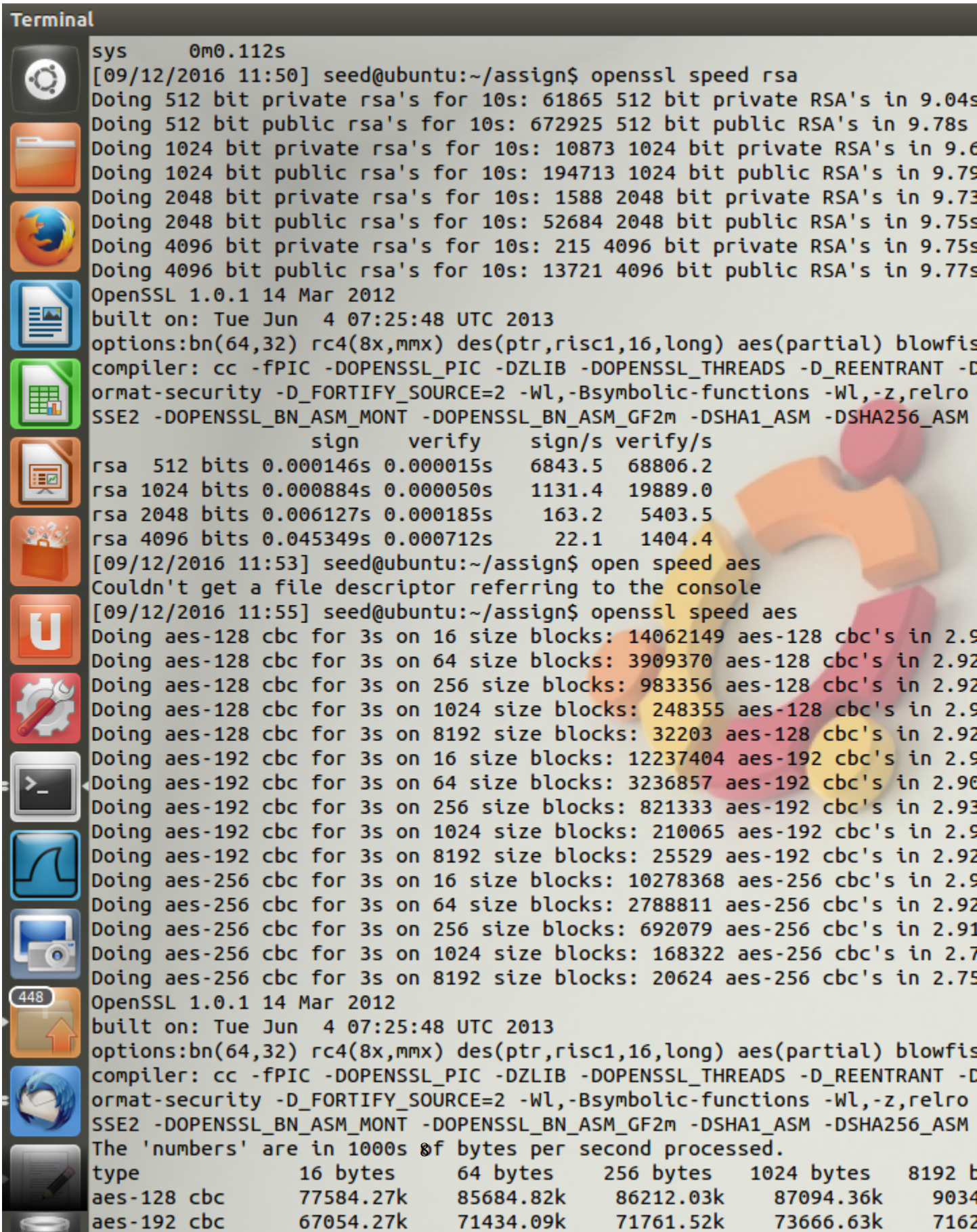
TASK 4 screenshot



TASK 5 screenshot1



TASK 5 screenshot2



TASK 6 screenshot