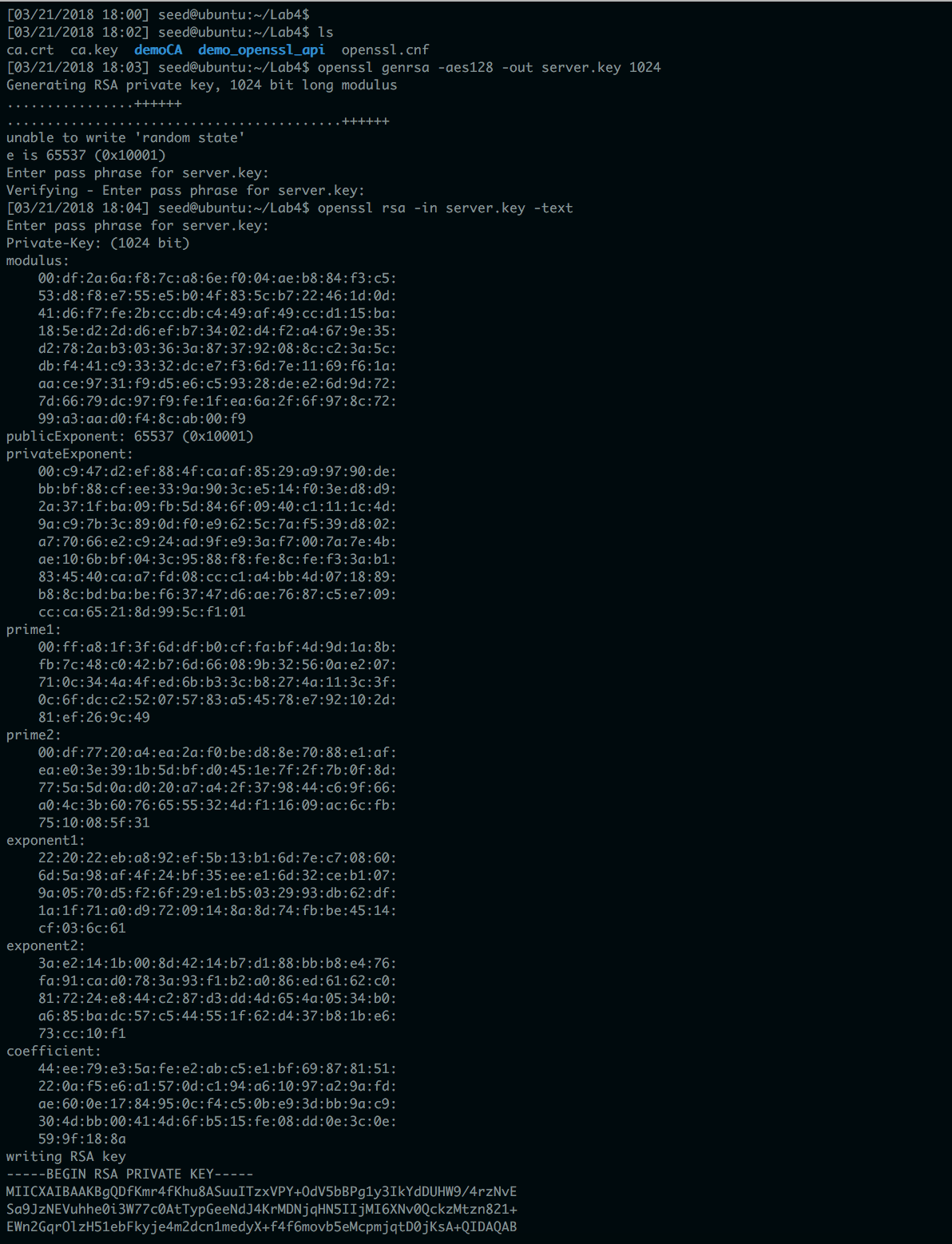
Homework #4

Crypto Lab - Public-Key Cryptography and PKI

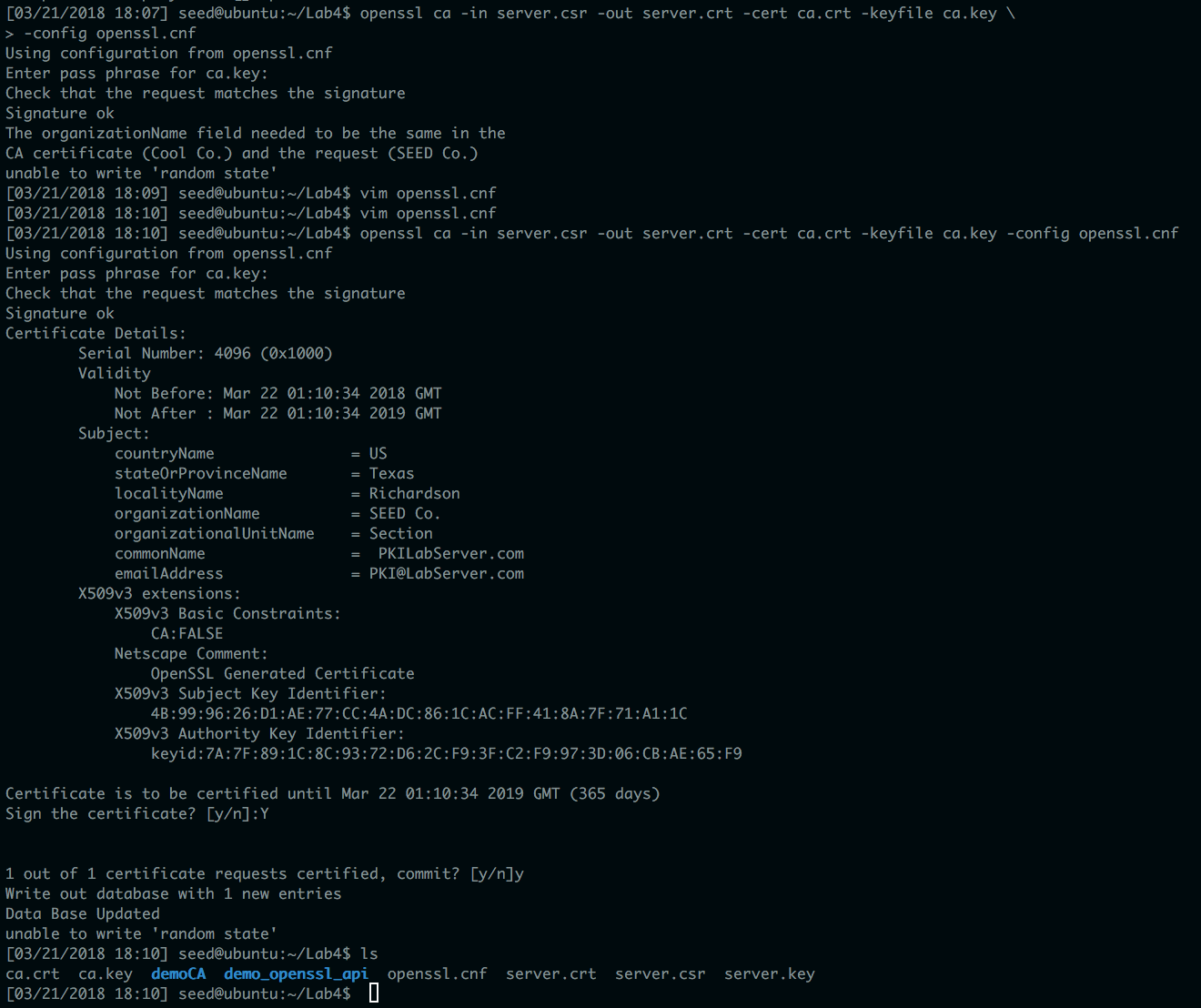
* Lab Environment:  
  The lab environment is already set up, as we installed OpenSSL on our SEED VM in the previous lab.
* Task 1:  
  Here is a screenshot of me performing the steps outlined in Task 1.  
    
  This screenshot demonstrates me setting up the directories to match the openssl.cnf file. It also demonstrates me creating my self-signed certificate as a CA.
* Task 2:
  + Part 2.1:  
    Here is a screenshot of me performing step 1 of Task 1:

  
 In this screenshot I generated a public/private key pair and viewed its  
 content.

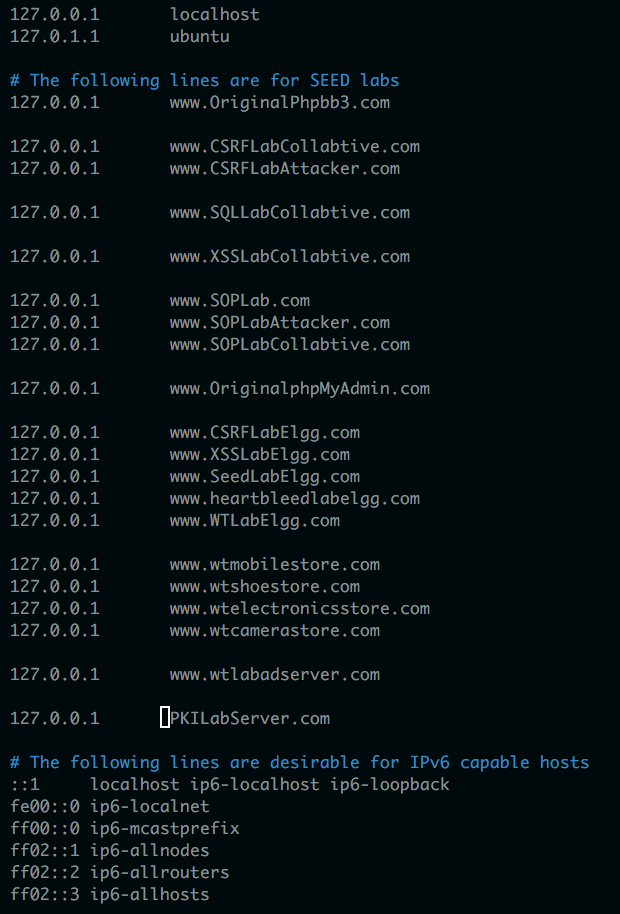
* + Part 2.2:

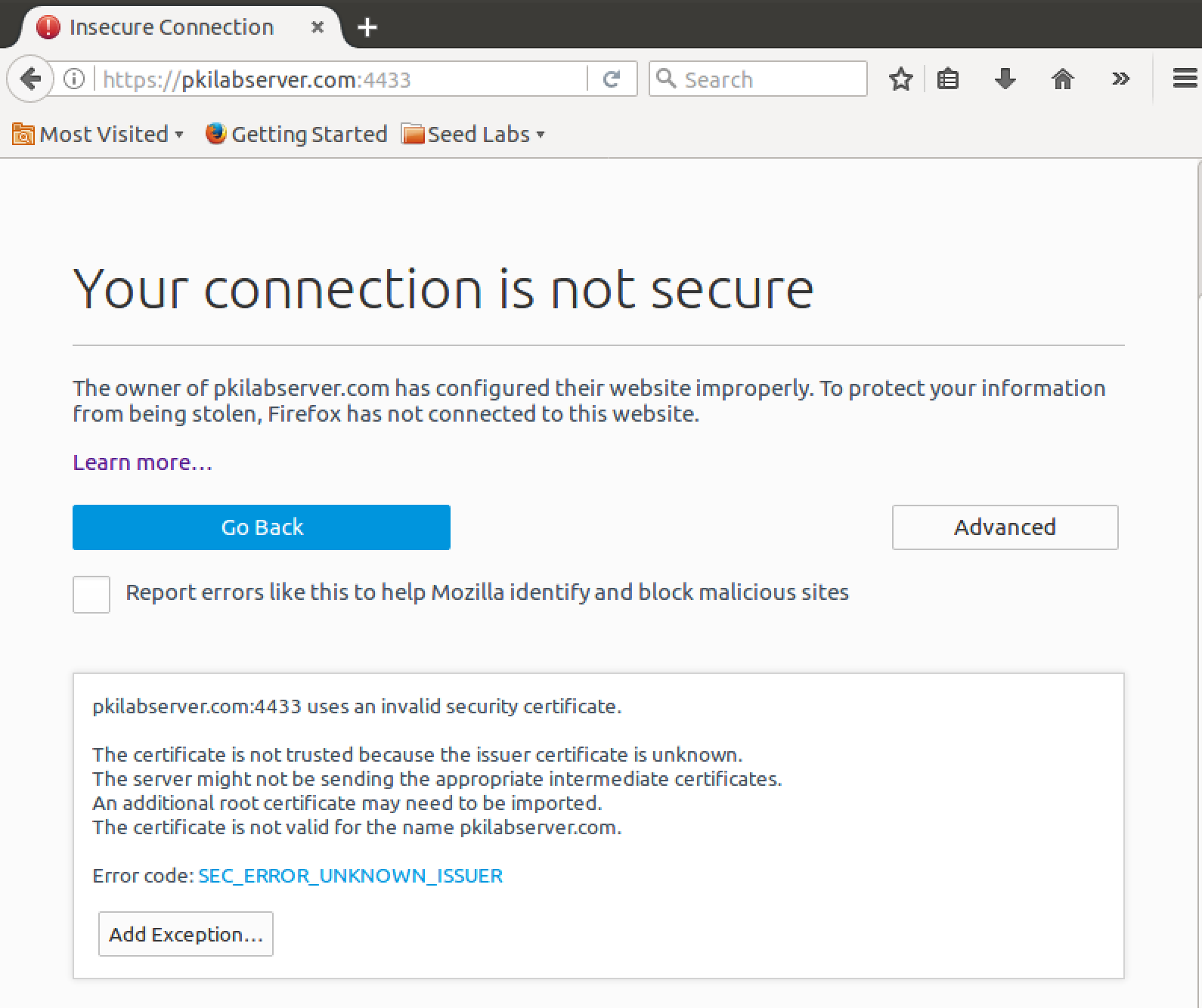
Here is a screenshot of me generating a certificate signing request:  
  
The certificate signing request is created as server.csr

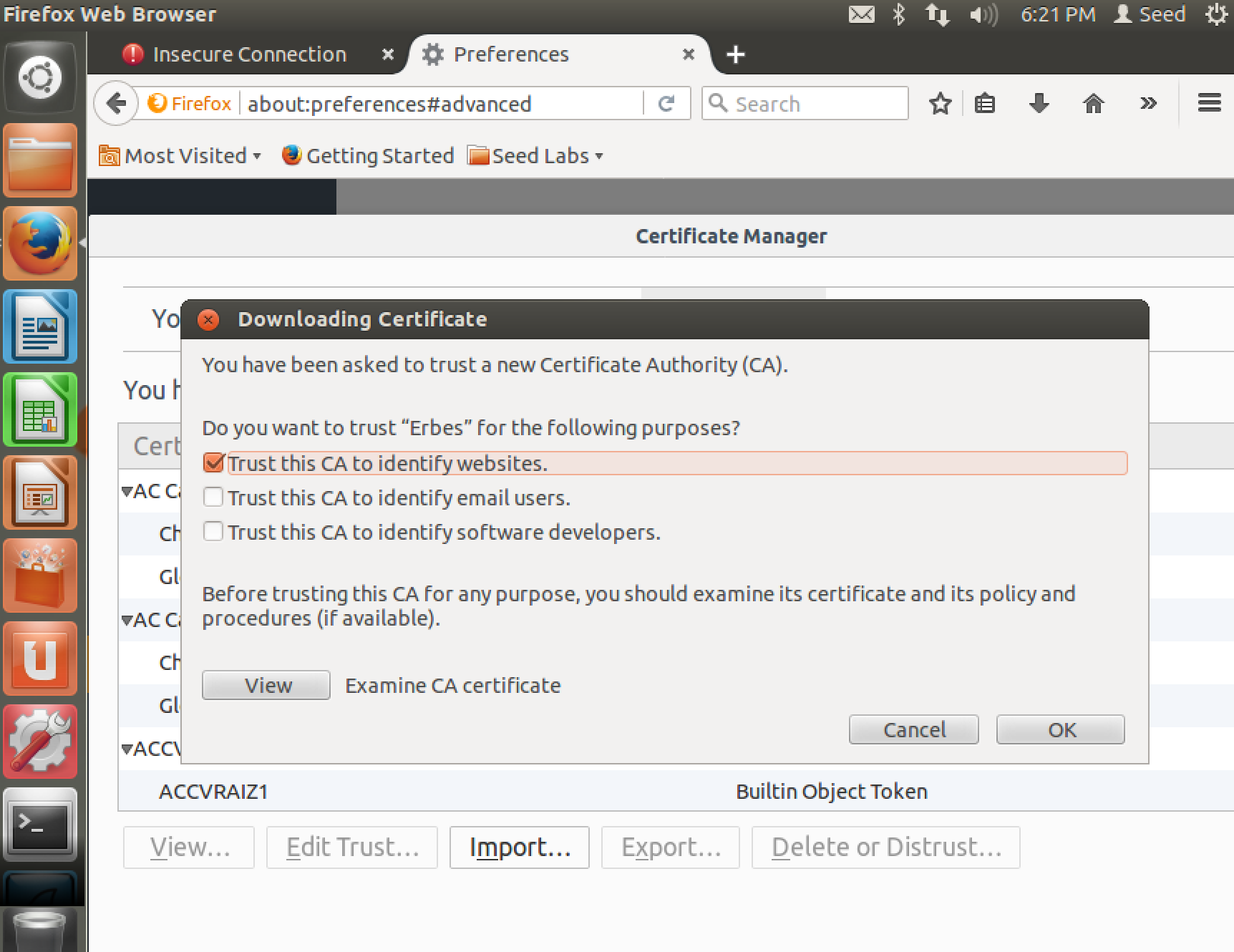
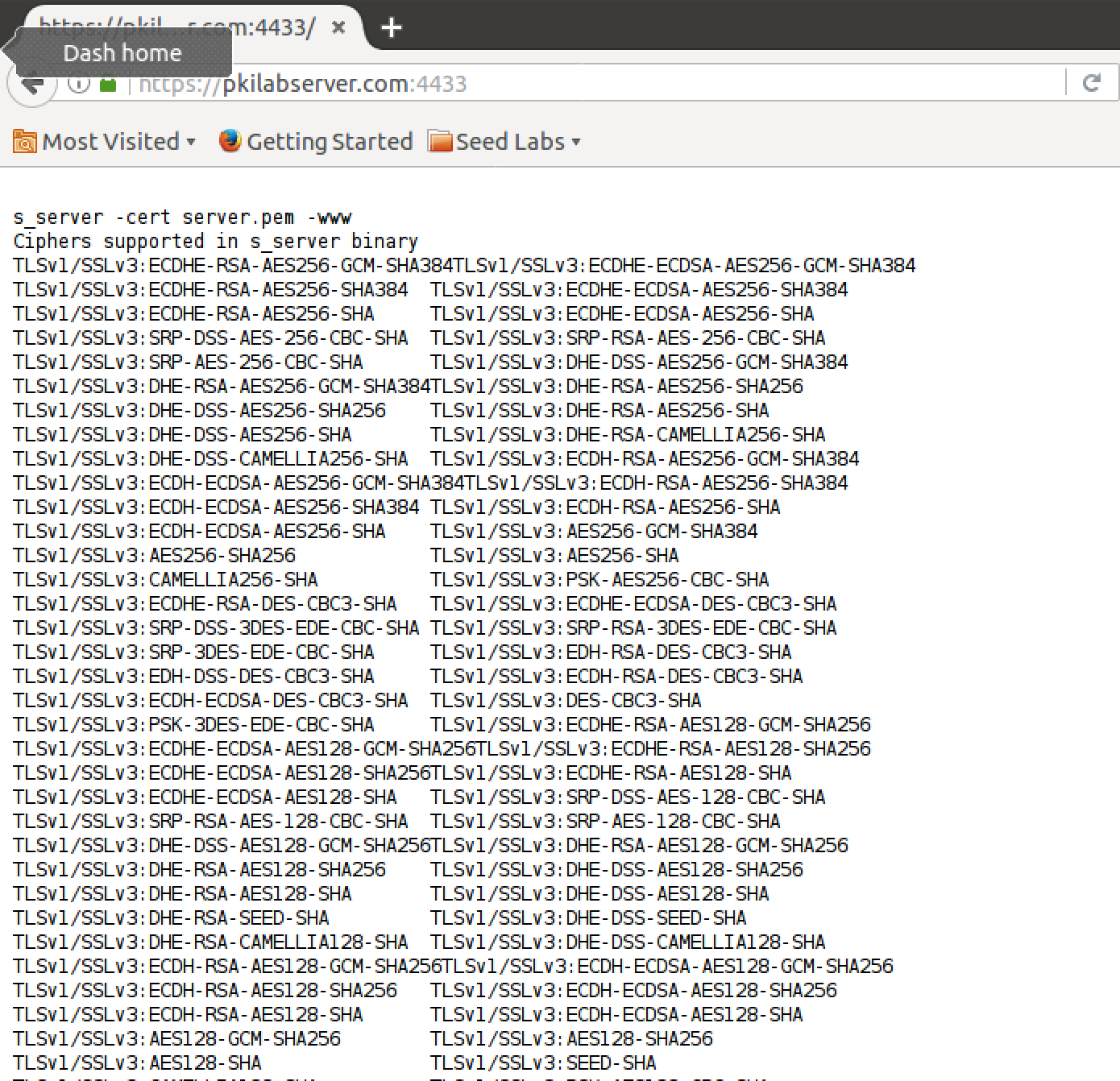
* + Part 2.3:

Here is a screenshot of me generating the certificates:  
  
I had to adjust the openssl.cnf file to use policy\_anything in order for the certificates to be generated successfully.

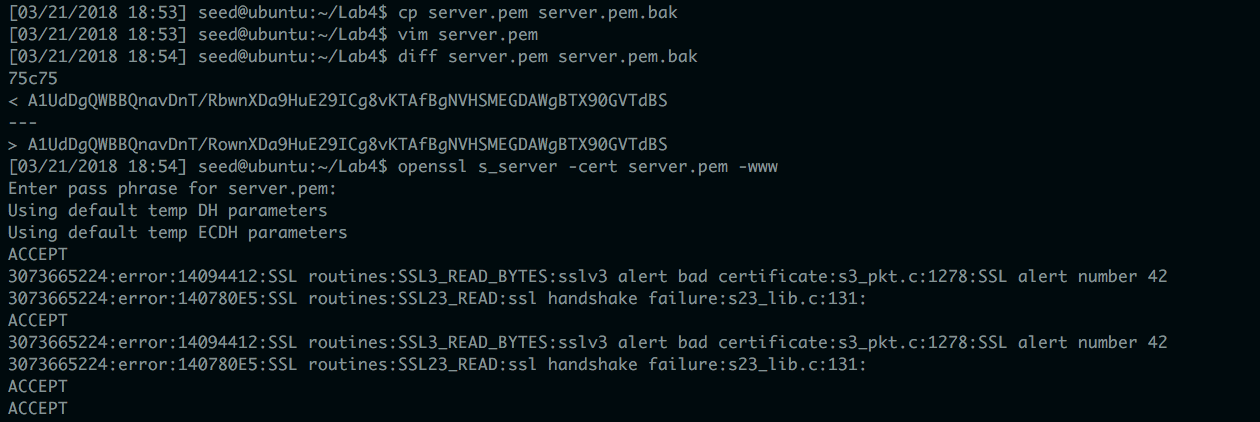
* Task 3:  
  Here is a screenshot of me adding pkilabserver.com to the /etc/hosts file:

  
Here is a screenshot of me combining the secret key and certificate into one file and then starting the server.

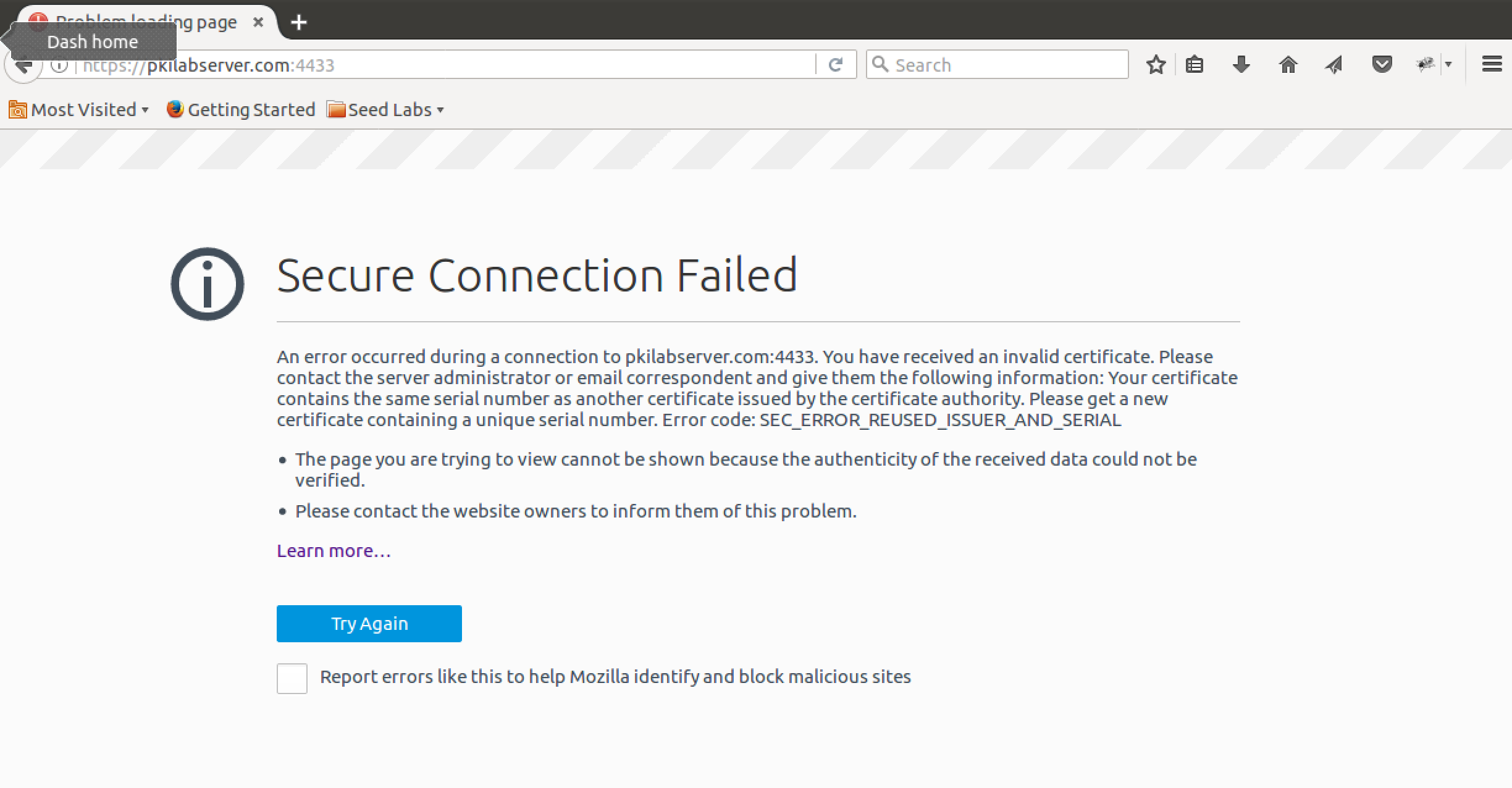
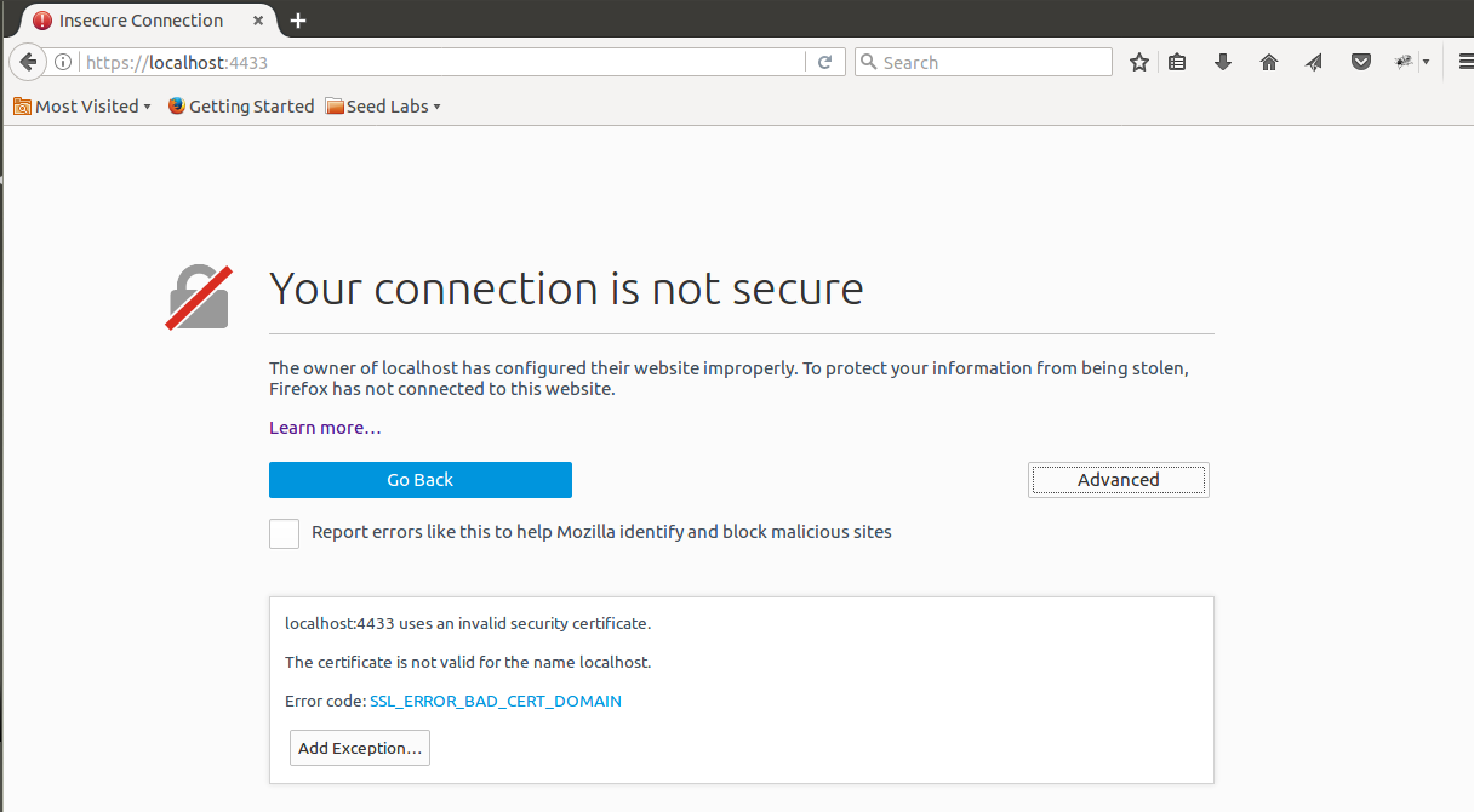
  
Here is a screenshot of what happened when I first visited the pkilabserver url:   
  
Firefox complained because my server certificate was not authorized by a trusted third party.

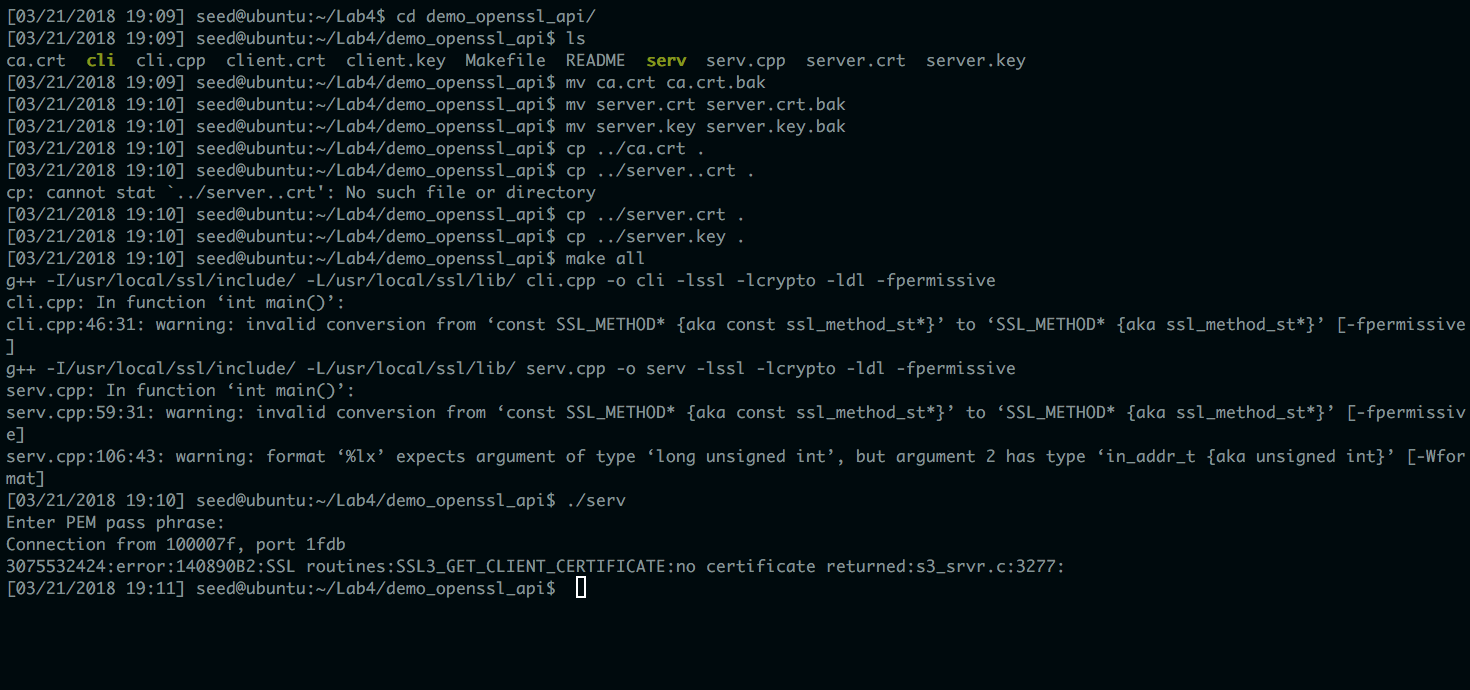
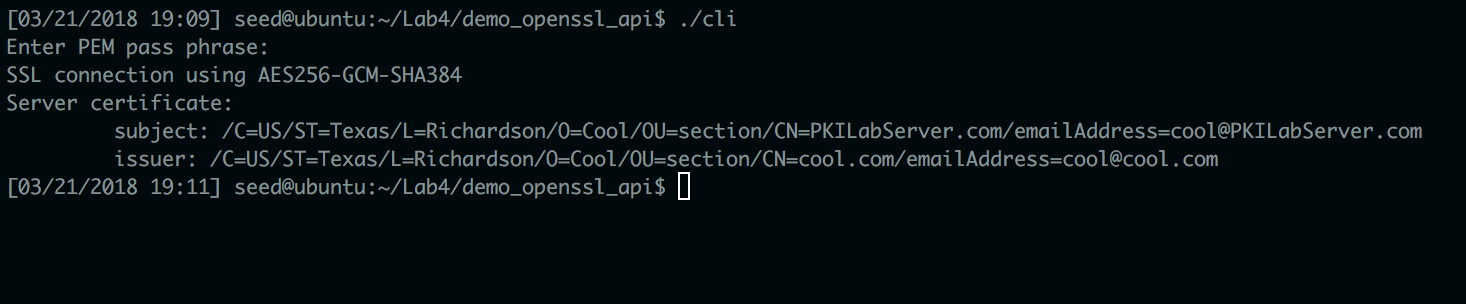
Here is a screenshot of me adding my CA to Firefox’s trusted authorities list:  
  
Here is a screenshot of what the pkilabserver website looked like after I added the CA to the list:  


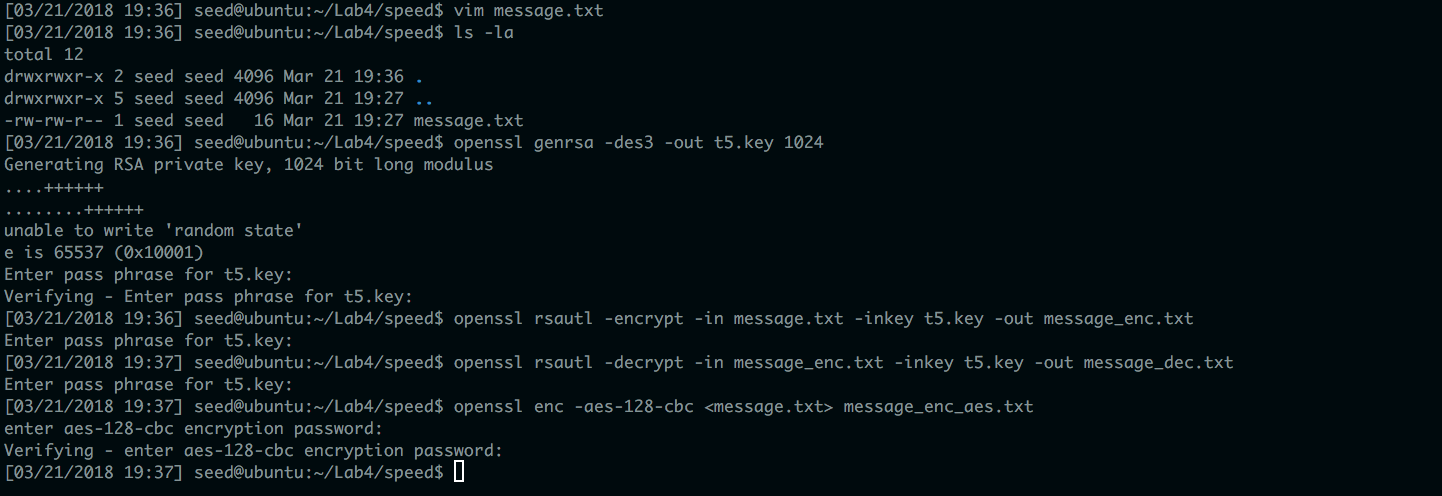
Here is a screenshot of me running the server with a slightly modified certificate, as indicated by the diff performed in the screenshot:

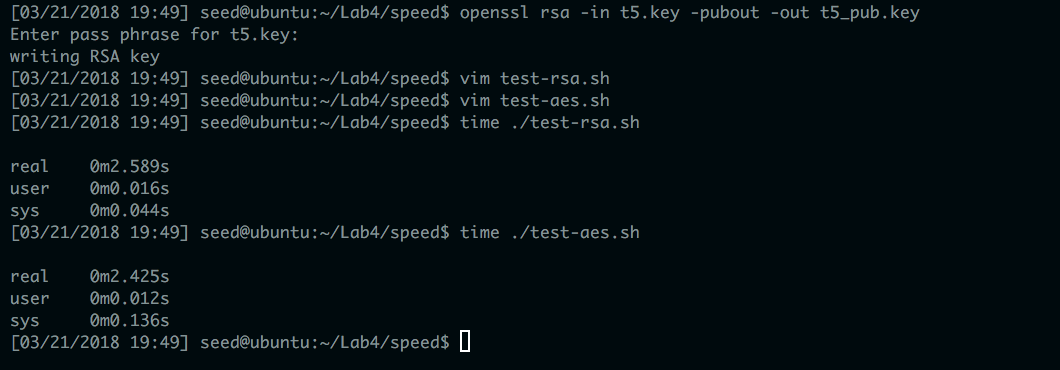


Here is a screenshot of me visiting the pkilabserver after modifying the server’s certificate slightly.

  
The certificate was detected to be invalid, because it duplicated the serial number issued to the first certificate by my CA.  
  
Here is a screenshot of me visiting https://localhost:4433:  
  
The certificate was only configured to work under the pkilabserver.com domain, so it was not able to provide a secure connection for localhost, even though both domains actually point to the same IP address.

* Task 4:  
  Here is a screenshot of me configuring the server to use the pkilabserver.com certificate, compiling, and then running the serv executable:  
    
  Here is a screenshot of me running the cli executable:  
    
  The client code detects the pkilabserver certificate being used by the server. The server code notes that the client is not using a certificate of their own.
* Task 5:  
  Here is a screenshot of me creating a RSA public/private key pair and performing the three operations outlined in Task 5:



Here is a screenshot of me comparing RSA and AES encryption time:  
  
The source code for these shell scripts will be included on the next page. Each test script encrypted the message.txt file 1000 times, and the time it took to compute each script was taken as a benchmark.  
  
The calculated RSA and AES encryption time was very close, but AES seemed to be a slight bit faster. Perhaps the message.txt file was so small that it made the choice of encryption algorithm negligible.

Source code for test-rsa.sh:  
  
#!/bin/sh

x=1

while [ $x -le 1000 ]

do

openssl rsautl -encrypt -in message.txt -inkey t5\_pub.key -pubin -out message\_enc.txt

x=$(($x+1))

done

Source code for test-aes.sh:  
  
#!/bin/sh

x=1

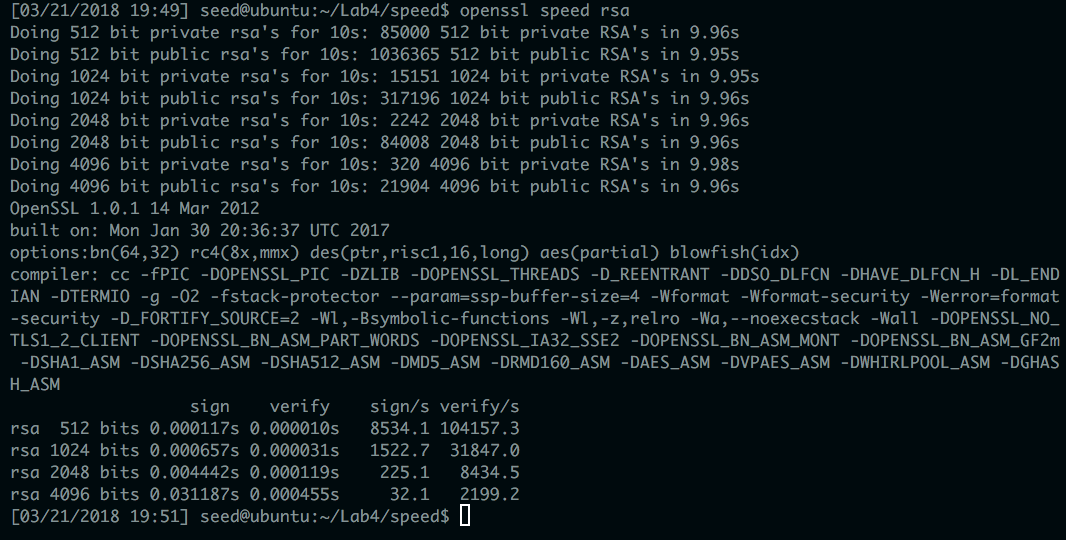
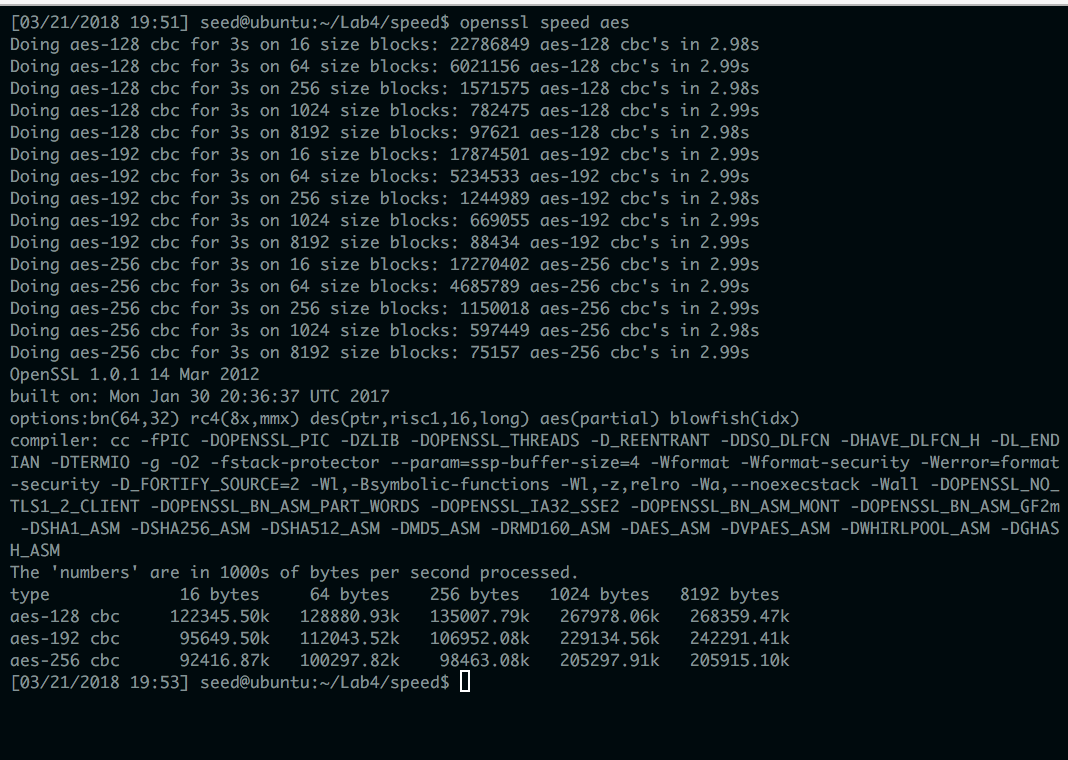
while [ $x -le 1000 ]

do

openssl enc -aes-128-cbc <message.txt> message\_enc\_aes.txt -k 123

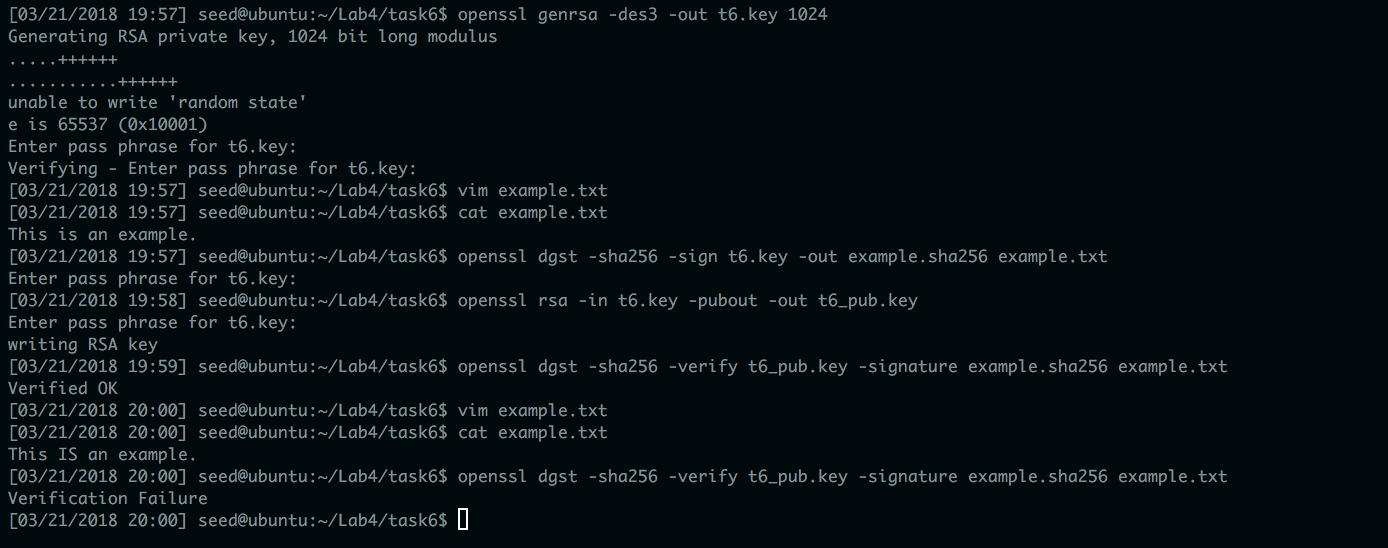
x=$(($x+1))

done

Here is the output from running openssl speed rsa:  
  
  
Here is the output from running openssl speed aes:  
  
It appears that openssl uses different metrics when comparing its results for the speed of RSA and the speed of AES. This suggests that RSA and AES maybe be faster or slower for different applications. AES is much more useful for encrypting large file sizes, while RSA is useful for signing and verifying things.

* Task 6:

Here is a screenshot of me performing Task 6:



I first generated a RAS public/private key pair, then used it to sign the example text. After signing, I was able to successfully verify the signed message with the hashed signature. After changing the message, however, the signature no longer verified the text successfully, which indicated that the current file differed from the original one that was signed.