**Task 1: Become a Certificate Authority (CA)**

This task began with making a copy of the file from /usr/lib/ssl/openssl.cnf. From here, I made a folder “demoCA”, and inside of that folder were the following folders: “certs”, “crl”, and “newcerts”. It also had the files “index.txt” and “serial” inside it.

Next, I generated a self-signed certificate to serve as the Certificate Authority. Some of the details are below:

Country: US  
State: Pennsylvania

Locality: Erie

Organization: MPopovich LLC.

Organizational Unit: Engineering

Common Name: Matt Popovich

Email Address: [thegreatest@psu.edu](mailto:thegreatest@psu.edu)

Password: mattisawesome

**Task 2: Create a Certificate for PKILabServer.com**

First, a 1024 bit RSA public/private key pair for PKI Lab Server was generated. The password for the pair was “PKILabServer.com”.

Next, a certificate signing request (CSR) was created with the following information:

Country: US

State: Pennsylvania

Locality: Erie

Organization: MPopovich LLC.

Organizational Unit: Software

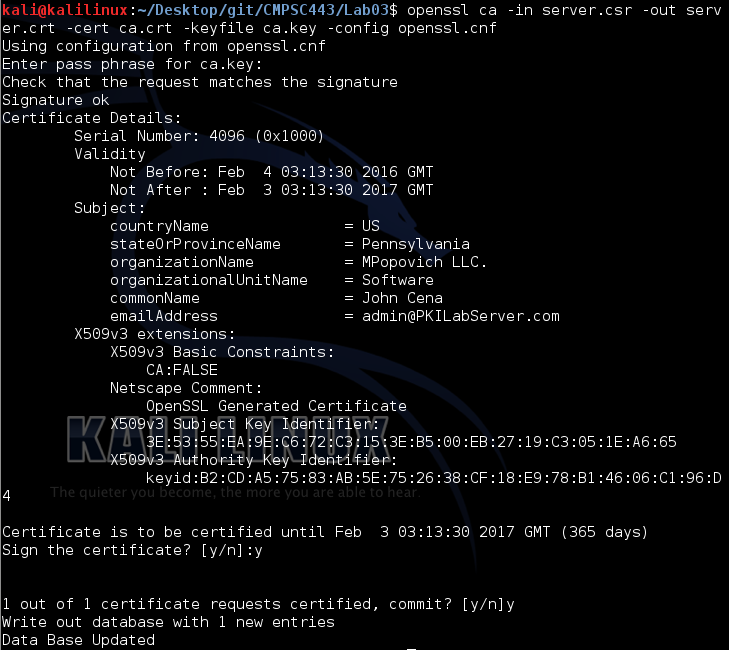
Common Name: John Cena

Email Address: [admin@PKILabServer.com](mailto:admin@PKILabServer.com)

Challenge Password: .

Company Name: .

Next, a certificate was generated. To do this, the CSR file needed to be signed by the CA. A screenshot of this is below in Figure 1.



**Figure 1:** Signing the CSR file with the CA

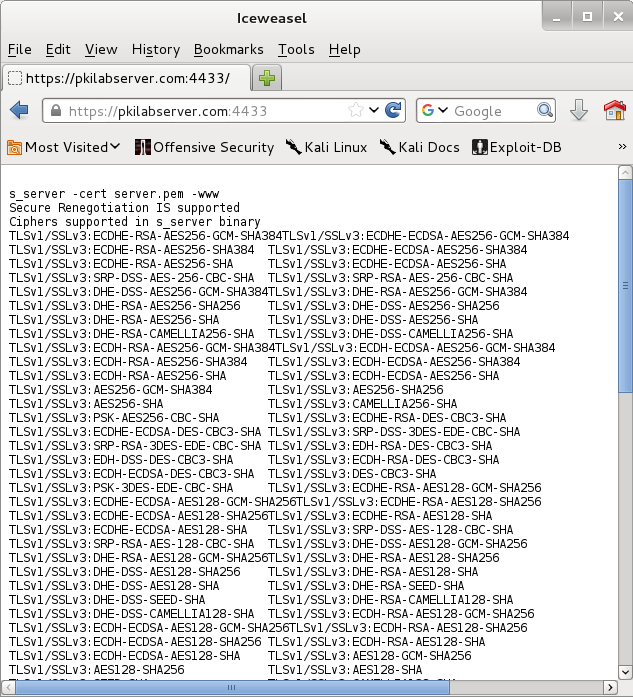
**Task 3: Use PKI for Web Sites**

First, I mapped the domain name “PKILabServer.com” to 127.0.0.1 in the file “/etc/hosts”.

Next, a “server.pem” file was created which is simply server.key and server.crt concatenated into one file.

Next, openssl was used to launch a simple web server with the “server.pem” certificate.

However, although everything is set up to view PKILabServer.com, our browser will not allow it. Our browser doesn’t have our CA preloaded into it. Before viewing PKILabServer.com, our CA (MPopovich LLC.) will need to be added. After adding the CA, the website can now be viewed. A view of the website is below in Figure 2.



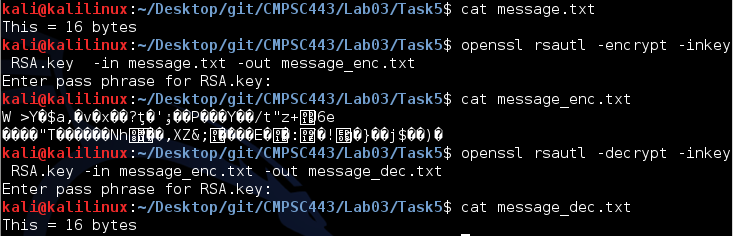
**Figure 2:** View of the Website After Installing the CA

As expected after working with last week’s encryption and decryption labs, if a bit is modified in the server’s certificate, the webpage is unable to be viewed because the certificate is invalid.

We can also connect to the server via <https://localhost:4433>. However, because the certificate is only for PKILabServer.com, we must tell our browser to trust the localhost domain as well.

**Task 5: Performance Comparison: RSA versus AES**

First, a 16 byte text file was created, followed by a 1024-bit RSA key pair. Figure 3 below demonstrates encrypting and decrypting a file with the generated key pair.



**Figure 3:** Encryption and Decryption with RSA

In my tests, AES was much quicker to encrypt and decrypt, and those agreed with the commands of “openssl speed rsa” and “openssl speed aes”. My estimation for the reason of this is because RSA is operating with many more bits than AES.

**Task 6: Create Digital Signature**

First, a text file was created along with a 1024-bit RSA key pair using “vi RSA.key”.

Next, a 1024-bit RSA key pair was generated via “openssl genrsa –des3 –out RSA.key 1024”.

Next, an example file was created using “vi example.txt”.

Next, the file was signed using “openssl dgst –sha256 –sign RSA.key example.sha256”.

Next, the signature was verified via “openssl –verify RSA.key –signature example.sha256 example.txt”. The terminal responded with “Verify OK”.

However, after modifying a bit of the signature, the signature verification failed.

Digital signatures are very useful and important. They allow a user to verify that a document or file was actually from the expected recipient.