Introduction to Computer Security Module – G6077

-

Introduction to OpenSSL

**Objectives:**

1) Understand OpenSSL

2) Encrypting and decrypting files with ciphers

3) Generate public and privates keys using OpenSSL

Contents

[OpenSSL Tutorial: An Introduction to Internet Security 2](#_Toc23495756)

[Abbreviations Key 2](#_Toc23495757)

[Task 1 start the openssl 3](#_Toc23495758)

[Task 2 version of openssl 3](#_Toc23495759)

[Task 3 standard commands 4](#_Toc23495760)

[Task 4 Encryption with AES algorithm 4](#_Toc23495761)

[Task 5 – Other ciphers 5](#_Toc23495762)

[Task 6 Public and private keys. 5](#_Toc23495763)

[Task 7 Optional but highly recommended tasks. 6](#_Toc23495764)

OpenSSL is among the most popular cryptography libraries. It is most commonly used to implement the Secure Sockets Layer and Transport Layer Security ([SSL and TLS](https://www.keycdn.com/support/what-is-ssl-tls)) protocols to ensure secure communications between computers. In recent years, SSL has become basically obsolete since TLS offers a higher level of security, but some people have gotten into the habit of referring to both protocols as “SSL.”

Cryptography is tricky business, and OpenSSL has too many features to cover in one article, but this OpenSSL tutorial will help you get started creating keys and certificates.

## **OpenSSL Tutorial: An Introduction to Internet Security**

When a client requests a secure connection to a server, the server, in turn, requests information to figure out which types of cryptographic security the client can support. Once it determines the most secure option, the following takes place:

1. The server sends a security certificate that is signed with the server’s public key.
2. Once the client verifies the certificate, it generates a secret key and sends it to the server encrypted with the public key.
3. Next, both sides use the secret key to create two sets of public-private keys. At last, secure communication can commence.

SSL and TLS are two of many security protocols used to accomplish these steps. To implement these protocols, we need software like OpenSSL.

## **Abbreviations Key**

You’ll come across tons of abbreviations in this guide and other OpenSSL tutorials. For quick reference, here is a short list of some terms you might encounter:

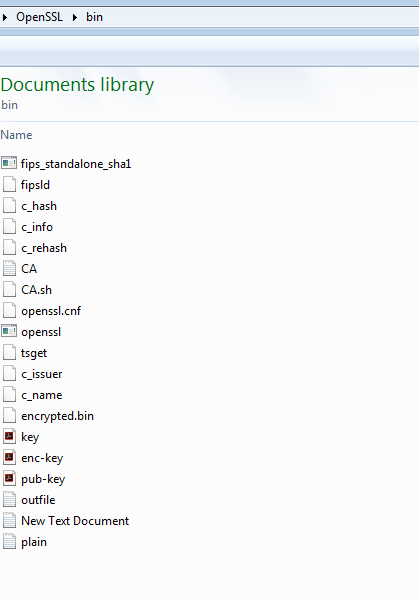
* **CSR:** Certificate Signing Request
* **DER:** Distinguished Encoding Rules
* **PEM:** Privacy Enhanced Mail
* **PKCS:** Public-Key Cryptography Standards
* **SHA:** Secure Hash Algorithm
* **SSL:** Secure Socket Layer
* **TLS:** Transport Layer Security

## **Task 1 start the openssl**

Download the OpenSSL folder from Canvas Week 6 into your student drive. Unzip this and find openssl application file in the bin folder. Double click on this, you will notice a command prompt like interface will open. In this window, you will write your openssl commands.

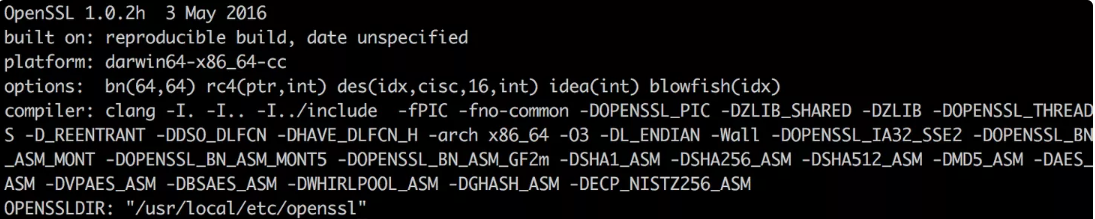
Important note:

In my examples, you will notice the word openssl. It refers to the openssl prompt. You do not need to type openssl while writing commands.



## **Task 2 version of openssl**

openssl version -a



Is any of these words familiar to you? Note the word blowfish in the output above. Goto lecture5a and slide 25 to find what is blowfish.

## **Task 3 standard commands**

openssl list-standard-commands

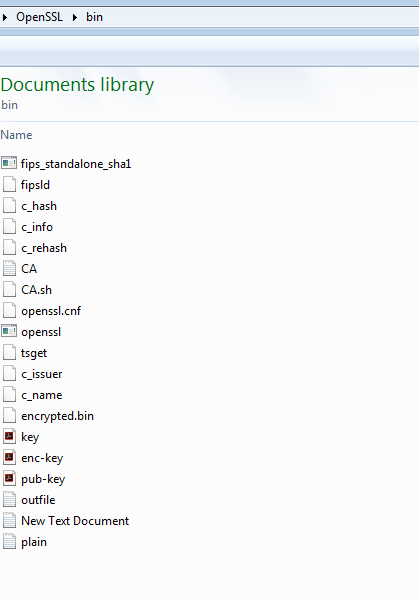
Find further information about the following commands from the list by clicking on [link](https://www.openssl.org/docs/man1.0.2/man1/openssl.html)

a) enc b)x509 c) genpkey d) genrsa

If you want to have career in computer security, you should look at all the standard commands.

## **Task 4** **Encryption with AES algorithm**

In bin folder, create a text file called it plain.txt. Enter the text Hello World in it.



Run the following command

openssl enc -aes-256-cbc -in plain.txt -out encrypted.bin

What does CBC and 256 represent in the command above? Are there any other options?

Enter the password as 1234

//enter aes-256-cbc encryption password: 1234

//Verifying - enter aes-256-cbc encryption password: 1234

Check the file encrypted.bin and see what is inside. Then run the following command and notice the output.

openssl enc -aes-256-cbc -d -in encrypted.bin -pass pass:1234

// Hello World!

## **Task 5 – Other ciphers**

Before going into the next task, think about what you have done and how you can do it differently by using another cipher.

If you are struggling with this task, here is a useful resource, click on this [link](https://www.tutonics.com/2012/10/easy-file-encryption-using-openssl.html).

## **Task 6 Public and private keys.**

For the sake of example, we can demonstrate how OpenSSL manages public keys using the [RSA algorithm](https://simple.wikipedia.org/wiki/RSA_%28algorithm%29). You can use other algorithms of course, and the same principles will apply. The first step is to generate public and private pairs of keys. Enter the following command to create an RSA key of 1024 bits:

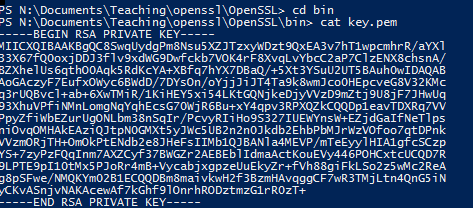
openssl genrsa -out key.pem 1024

You should now have a file called key.pem containing a public key and private key. As the file’s name suggests, the private key is coded using the Privacy Enhanced Email, or PEM, standard. Use the following code to display it:

In your command prompt or Powershell, run the following the command. The .pem file will be in the bin folder, first you need to be in that folder in your command prompt.

cat key.pem

Path to the file



openssl rsa -in key.pem -text -noout

This command should return information about the public and private exponents, the modulus and the other methods and numbers used to optimize the algorithm. In this context, the -noout option prevents display of the key in base 64 format, which means that only hexadecimal numbers are visible. The public exponent is an exception, of course, since it is always 65537 for 1024 bit keys.

To encrypt our private key, we use the following code:

openssl rsa -in key.pem -des3 -out enc-key.pem

Once the key file has been encrypted, you will then be prompted to create a password. Next, we can extract the public key from the file key.pem with this command:

openssl rsa -in key.pem -pubout -out pub-key.pem

Finally, we are ready to encrypt a file using our keys. Use the following format:

openssl pkeyutl -encrypt -in <input\_file> -inkey <key.pem> -out <output\_file>

In the above context, <input\_file> is the file you want to encrypt. Since we’re using RSA, keep in mind that the file can’t exceed 116 bytes. The <key.pem> is the file containing the public key. If that file doesn’t also include the private key, you must indicate so using -pubin. The <output\_file> is the encrypted file name.

Now, to decrypt the file, you can simply flip the equation: Change -encrypt to -decrypt, and switch the input and output files

## **Task 7 Optional but highly recommended tasks.**

The link below is a very useful tutorial about openssl. If you scroll up and down, you will notice that there are six parts in total. If you have completed upto task 5, then you have completed the first two parts in the tutorial on the given link. I will recommend strongly that you should complete the rest of the four parts in this tutorial.

<https://www.keycdn.com/blog/openssl-tutorial>