**Name:** AGUILAR, Paul Christian D. **Section:** BS CpE-3C

**DDC Cryptography Project.**

*Due Saturday 20 February 2021.*

Follow the detailed submission instructions at the end of this specs sheet.

Cryptography Exercises using OpenSSL

1. Download and install OpenSSL, if you do not have it yet.

[https://www.openssl.org](https://www.openssl.org/)

For each of the following exercises, you are given the freedom on how you would use OpenSSL (i.e., accessed via a shell script, or called from within your own program, etc.).

Reference all sources that you use in your answers.

2. SYMMETRIC ENCRYPTION.

Study the OpenSSL Library and use it to perform symmetric AES encryption on the 512x512 Color (24-bit) Lena image (<http://www.ece.rice.edu/~wakin/images/lena512color.tiff>)

Use both ECB and CBC mode, for AES-128.

Fully document the process (in a document) of how you performed the encryption using OpenSSL, and the results of the encryption.

3. HASHING.

Using OpenSSL, hash the same Lena 512x512 image using the following hash functions:

SHA-1, SHA-256, SHA-512.

Again, fully document the process and the results of the hash.

4. PUBLIC KEY ENCRYPTION.

Using OpenSSL, perform an RSA encryption on the Lena 512x512 image, using RSA-2048.

Using OpenSSL, generate an ECDSA signature on the same Lena image.

If you need to use a hash function, use SHA-256.

For all other details not outlined in this spec sheet, you have the freedom to choose or decide on the design detail. For example, you can define your own passwords or passphrases as basis for key generation.

TO BE SUBMITTED:

1. All source code, scripts, and documentation (in PDF) are to be housed in a git repository. You may use a public git repository (e.g., create an account on [github.com](http://github.com/) or [bitbucket.com](http://bitbucket.com/)) or your own private git repository.

2. Submit an accessible link to your git repository via e-mail to [roselia,delacruz@bulsu.edu.ph](mailto:spfestin@dcs.upd.edu.ph) by the deadline. I should be able to clone your git repo given the link and run (compile) your code/scripts from my machine. You should provide sufficient documentation for me to replicate your environment, e.g., what operating system, programming language, etc.

**DDC Cryptography Project Documentation**

Cryptography Exercises using OpenSSL

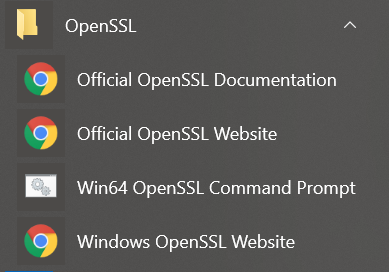
1. Download and install OpenSSL, if you do not have it yet.

[https://www.openssl.org](https://www.openssl.org/)

For each of the following exercises, you are given the freedom on how you would use OpenSSL (i.e., accessed via a shell script, or called from within your own program, etc.).

Reference all sources that you use in your answers.

* This is the OpenSSL application used during the Project duration:



* Upon clicking the Win64 OpenSSL Command Prompt software, this will show up:
* The window above is OpenSSL using shell script and it is ready for Cryptography operations

2. SYMMETRIC ENCRYPTION.

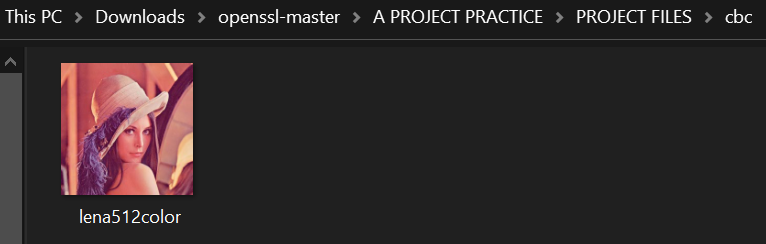
Study the OpenSSL Library and use it to perform symmetric AES encryption on the 512x512 Color (24-bit) Lena image (<http://www.ece.rice.edu/~wakin/images/lena512color.tiff>)

Use both ECB and CBC mode, for AES-128.

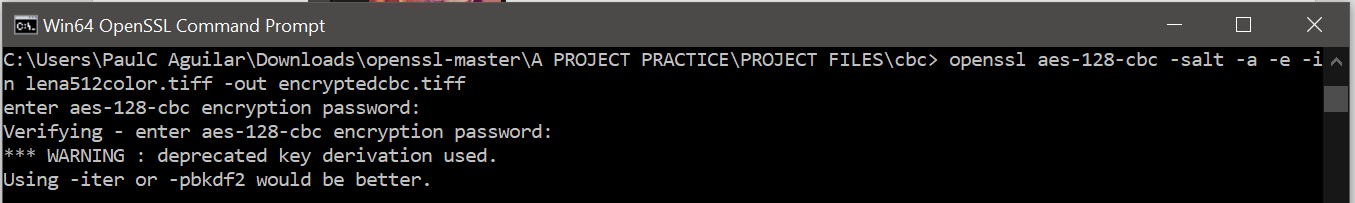
Fully document the process (in a document) of how you performed the encryption using OpenSSL, and the results of the encryption.

**Steps done to perform Symmetric Encryption in CBC Mode**

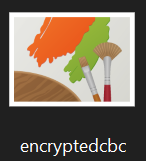
* Download the Lena Image



* It was transferred to the “cbc” folder for easy access
* Then using the OpenSSL Command Prompt, use the following codes for Symmetric Encryption via CBC mode



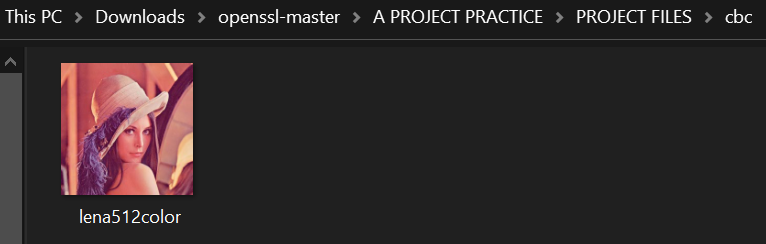
* Once the command was executed, a new file will be created. This means the code worked and the picture was encrypted. Nothing will show up when the file is opened.



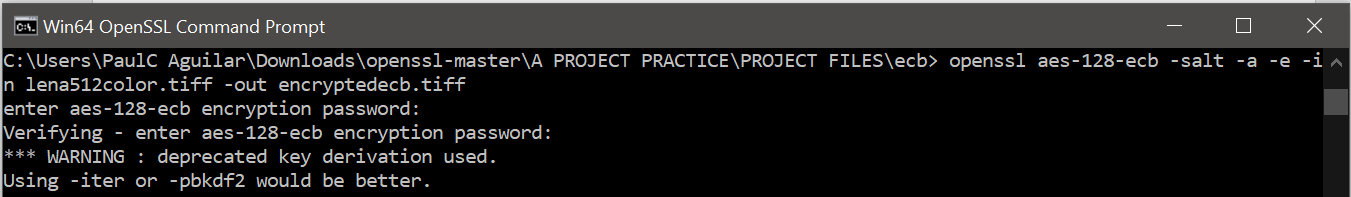


**Here is the process of Symmetric Encryption using ECB Mode**

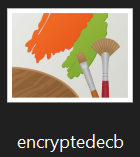
* Download another Lena Image



* The file was relocated to the “ecb” folder for convenience
* Enter the Symmetric Encryption code in ECB Mode using OpenSSL Command Prompt



* Once the command was executed, a new file will be created. This means the code worked and the picture was encrypted. Due to the encryption, the original content of the picture will not show up when opened.





Reference:

<https://gist.github.com/dreikanter/c7e85598664901afae03fedff308736b>

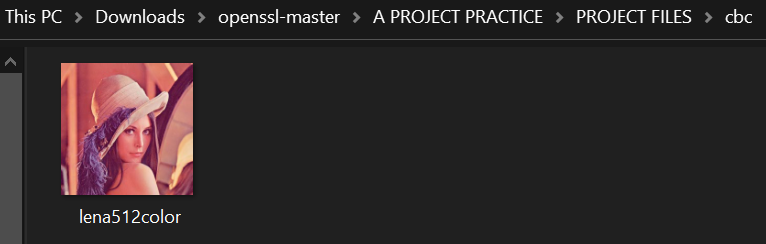
3. HASHING.

Using OpenSSL, hash the same Lena 512x512 image using the following hash functions:

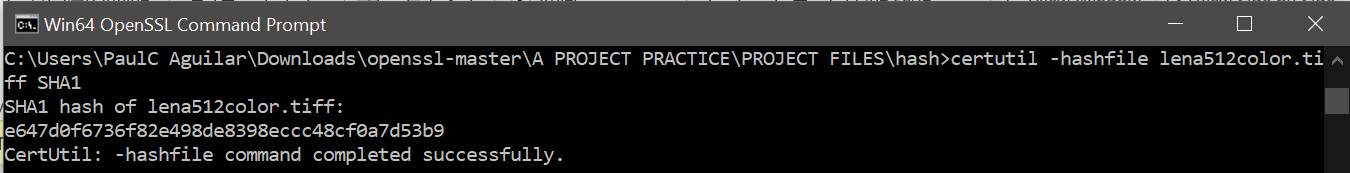
SHA-1, SHA-256, SHA-512.

Again, fully document the process and the results of the hash.

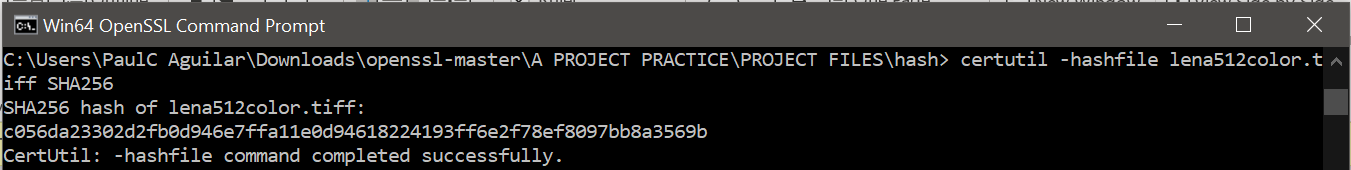
* Provide a Lena Image for hashing



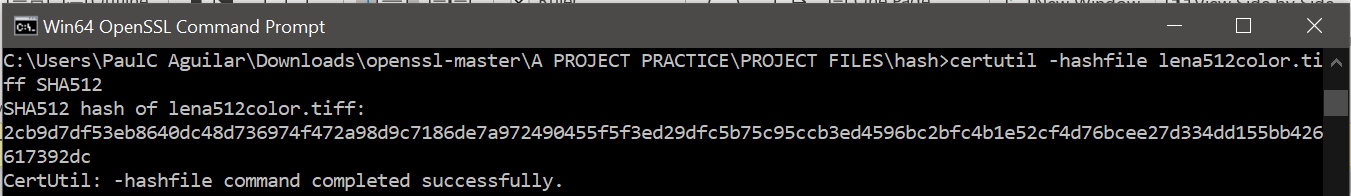
* There was a designated “hash” folder for this portion of the activity
* To create a HASH-1 data, enter the following code:



* The given line with a mixture of alphanumeric symbols is the product of hashing and are used for other methods in encryption
* To try another type of hashing like HASH-256, run the code below:



* Like the previous product, there is a line of alphanumeric code. The only difference is that HASH-256 has longer code using the same exact file.
* Here is another mode in hashing called HASH-512 and this is the code to use it:



* Another set of alphanumeric code was created using HASH-512. However, the file used is still the same.

References:

<https://www.geeksforgeeks.org/getting-hash-of-a-file-using-cmd/>

4. PUBLIC KEY ENCRYPTION.

Using OpenSSL, perform an RSA encryption on the Lena 512x512 image, using RSA-2048.

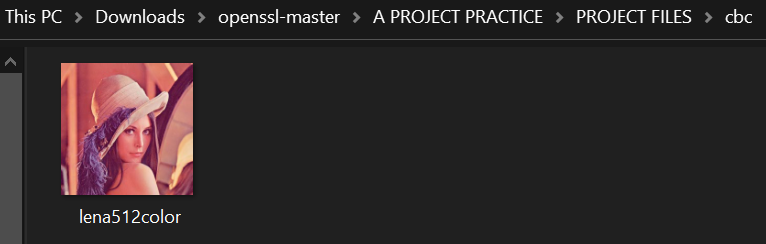
Using OpenSSL, generate an ECDSA signature on the same Lena image.

If you need to use a hash function, use SHA-256.

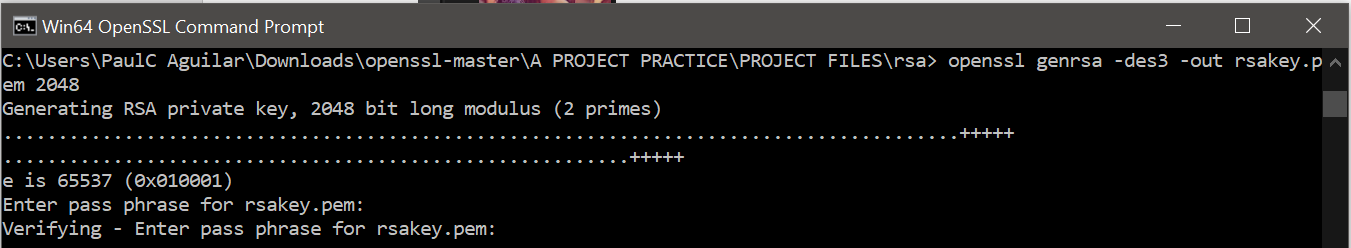
For all other details not outlined in this spec sheet, you have the freedom to choose or decide on the design detail. For example, you can define your own passwords or passphrases as basis for key generation.

**Procedures to carry out the RSA Encryption with the RSA-2048 Mode:**

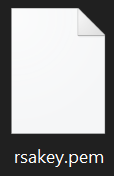
* A Lena image was provided for this type of encryption:



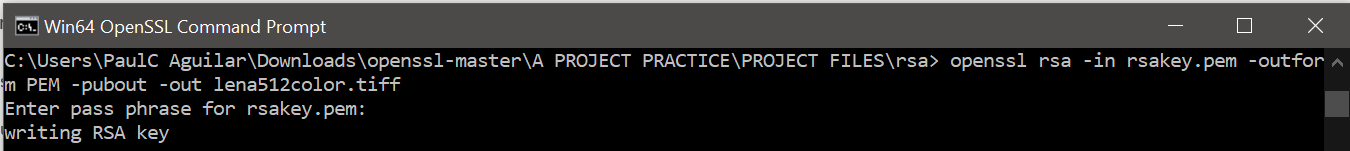
* Whereas a folder called “rsa” was created for the procedures
* The first step in RSA 2048 encryption is the generation of 2048 RSA key



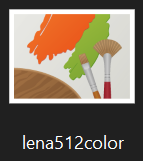
* Upon the execution of the command, this file will be created:



* Once the “rsakey.pem “become available, the next thing to do is run this code:

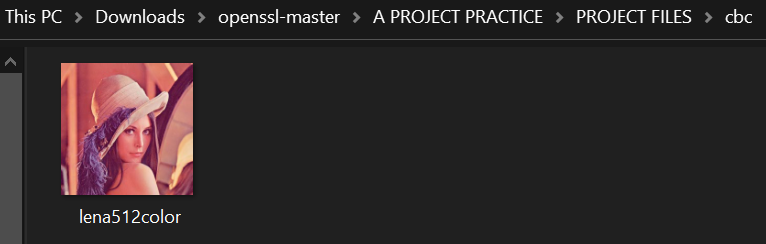


* Upon implementation, the original lena512color.tiff image will become this:

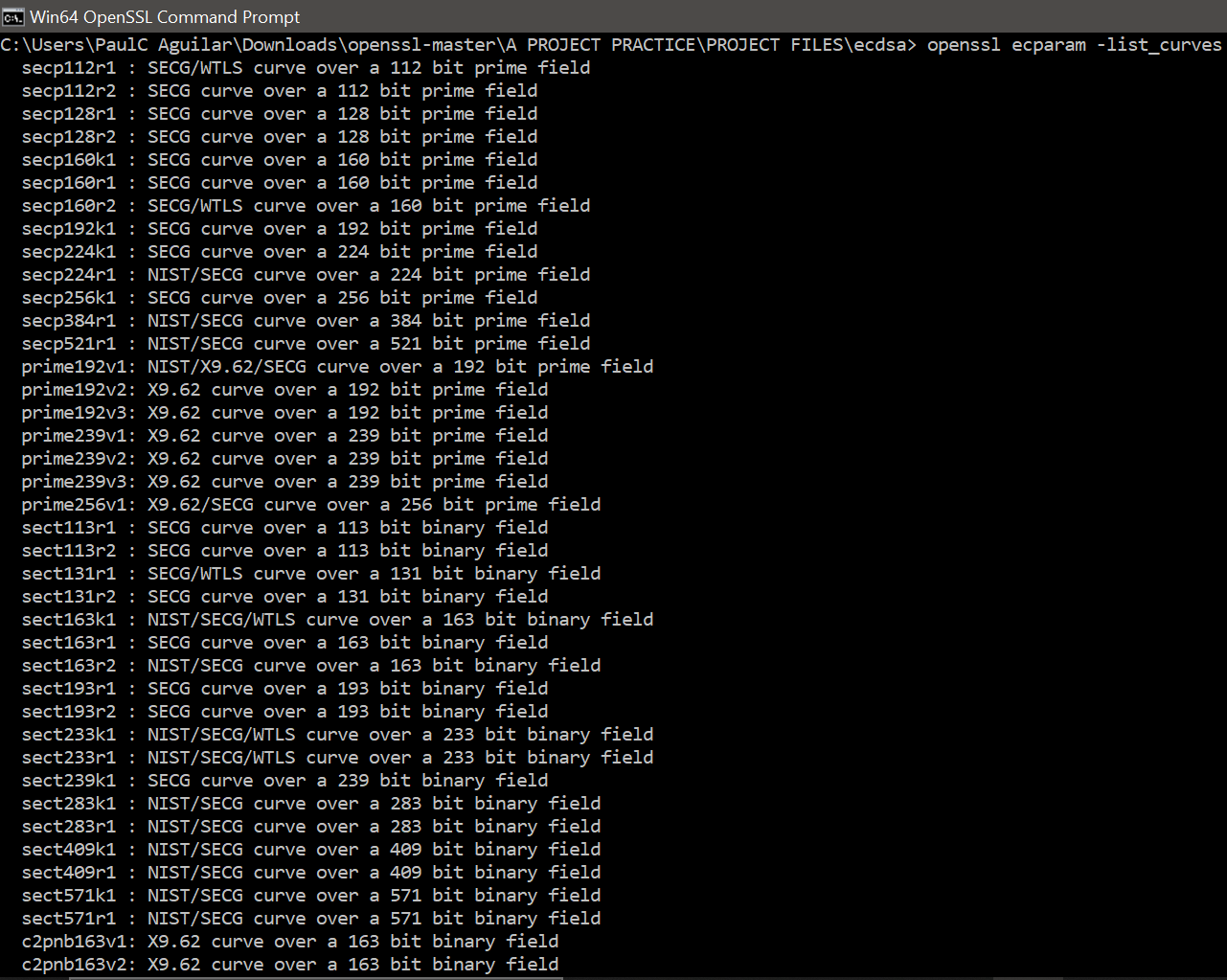


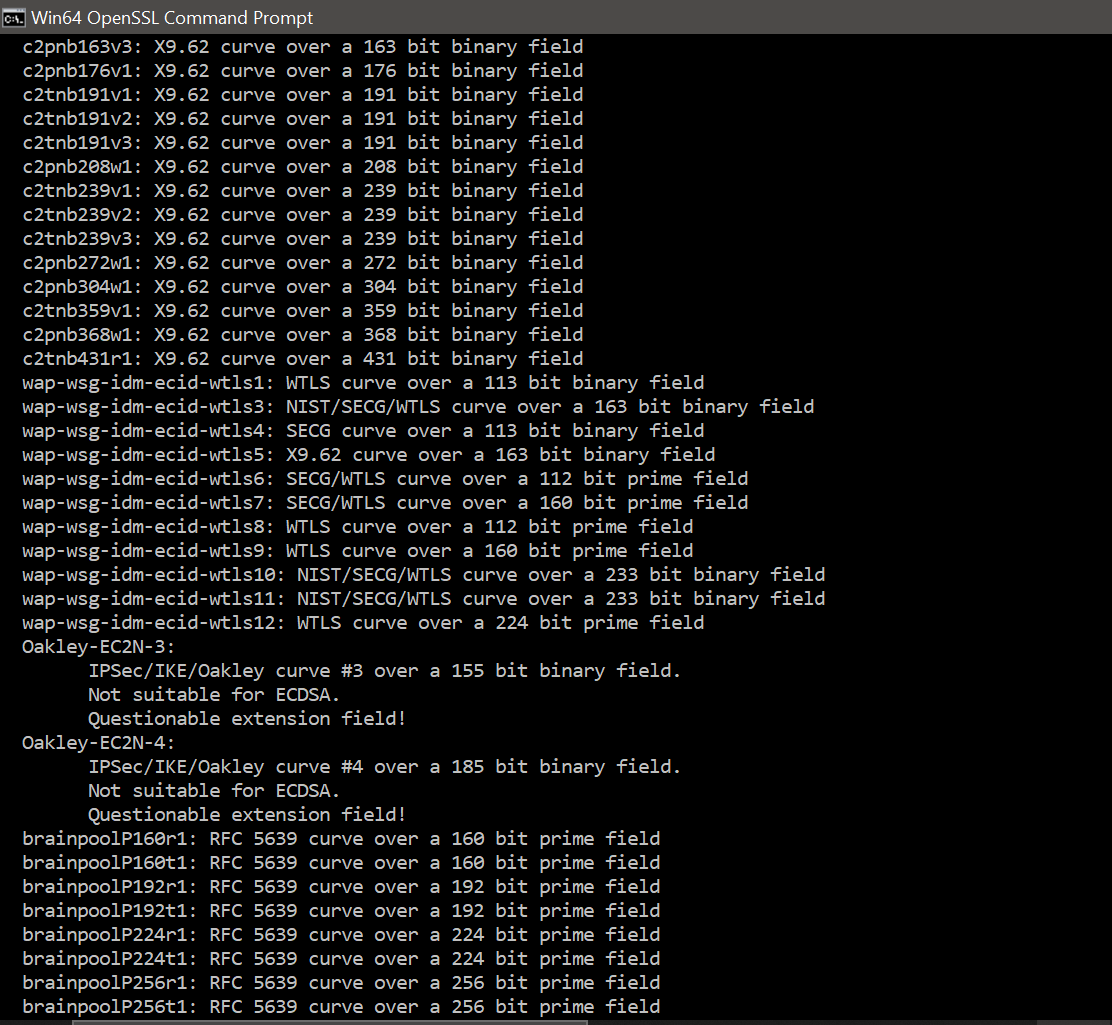
**In terms of the ECDSA encryption, these are the process taken**

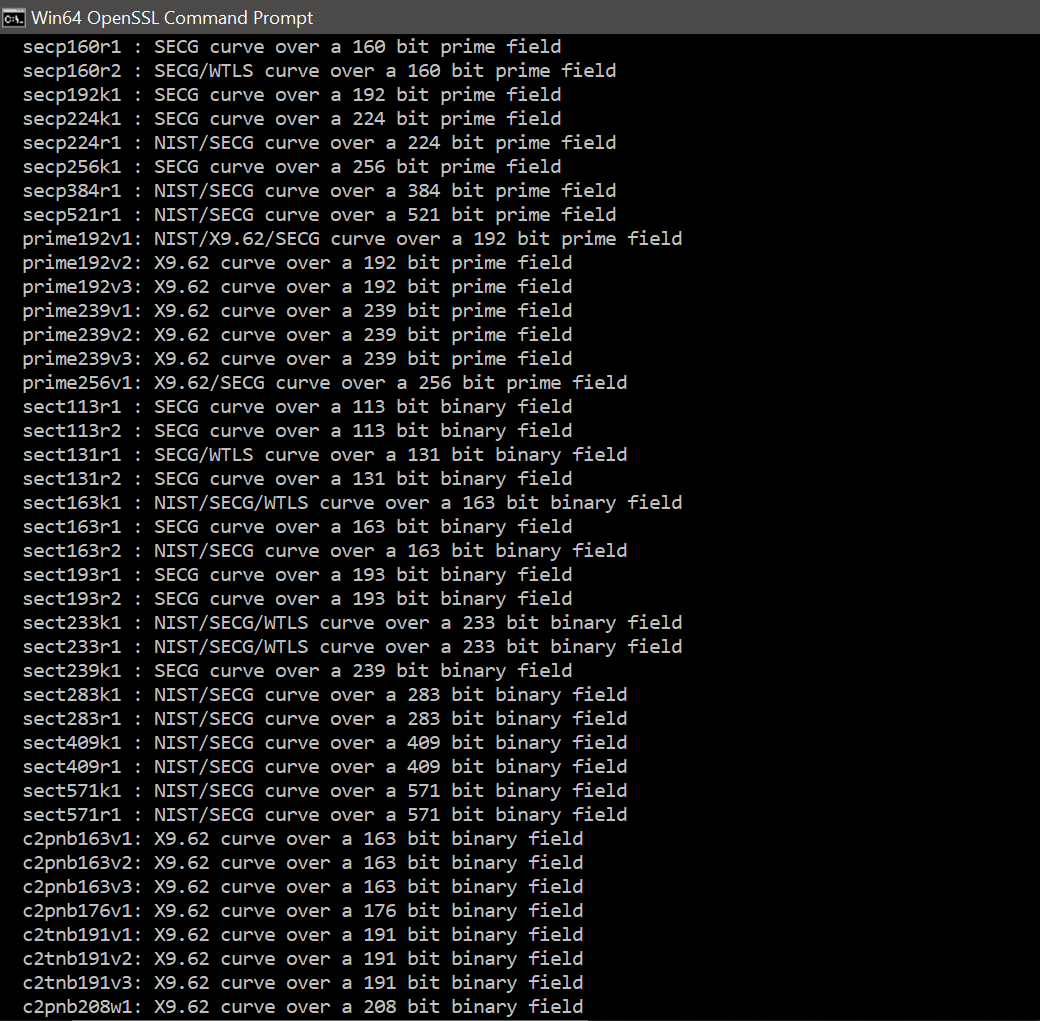
* Another Lena image was given for the ECDSA encryption

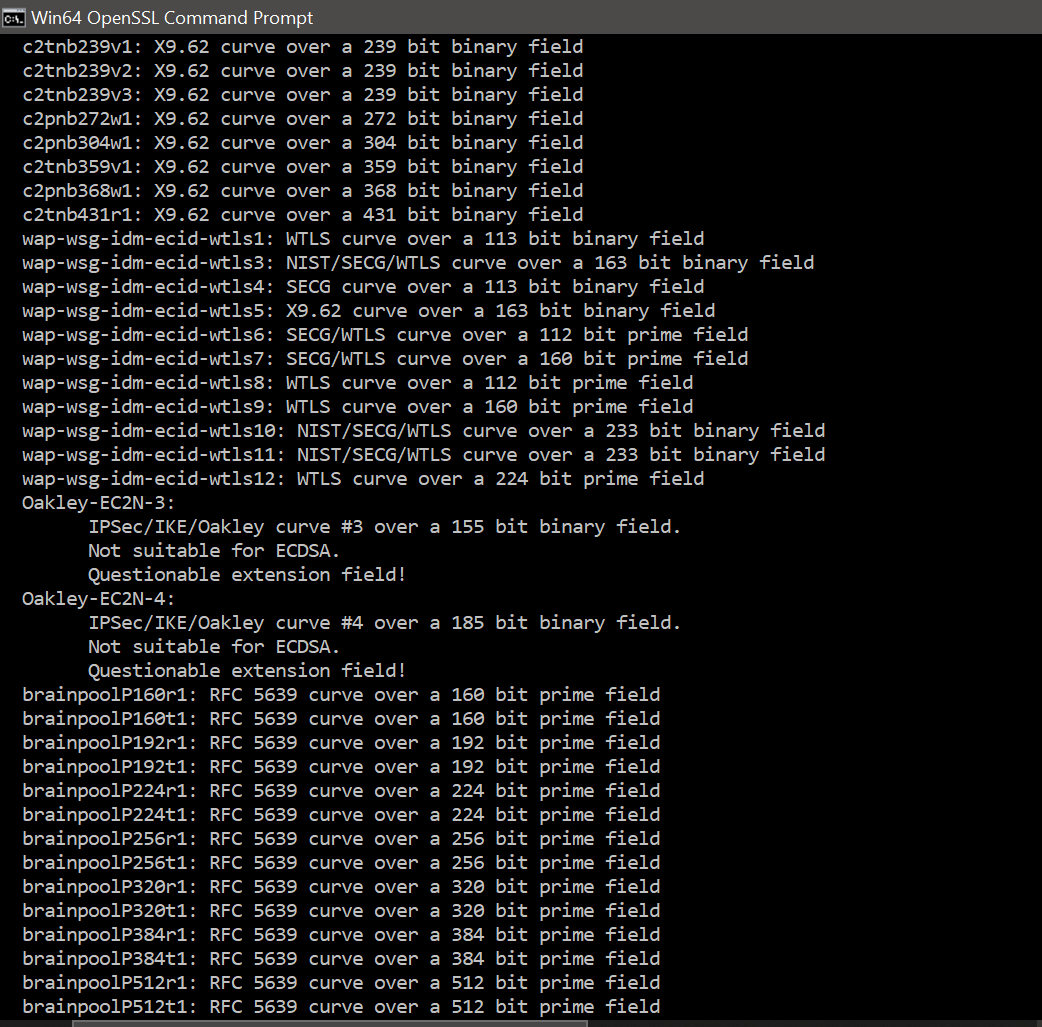


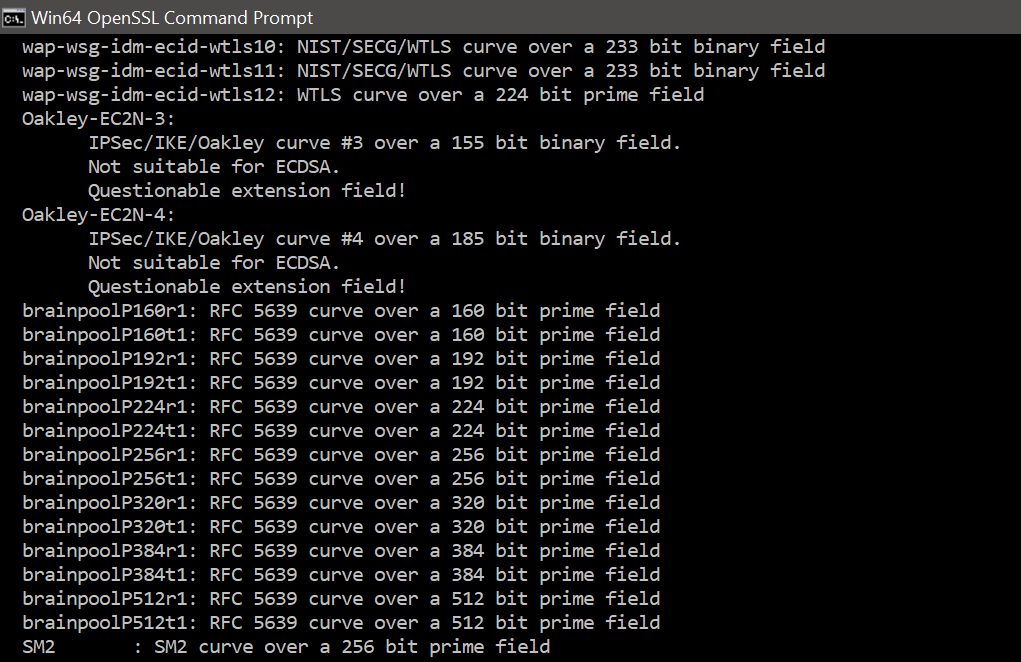
* Then, another folder was created for “ecdsa”
* It is required to choose from the following library code: “openssl ecparam -list\_curves” to perform the ecdsa process.



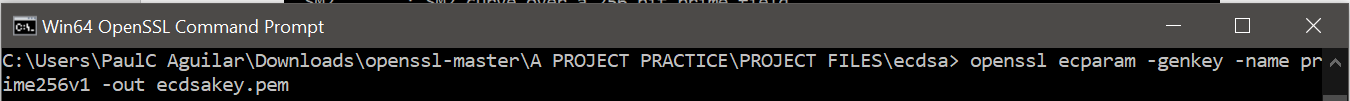








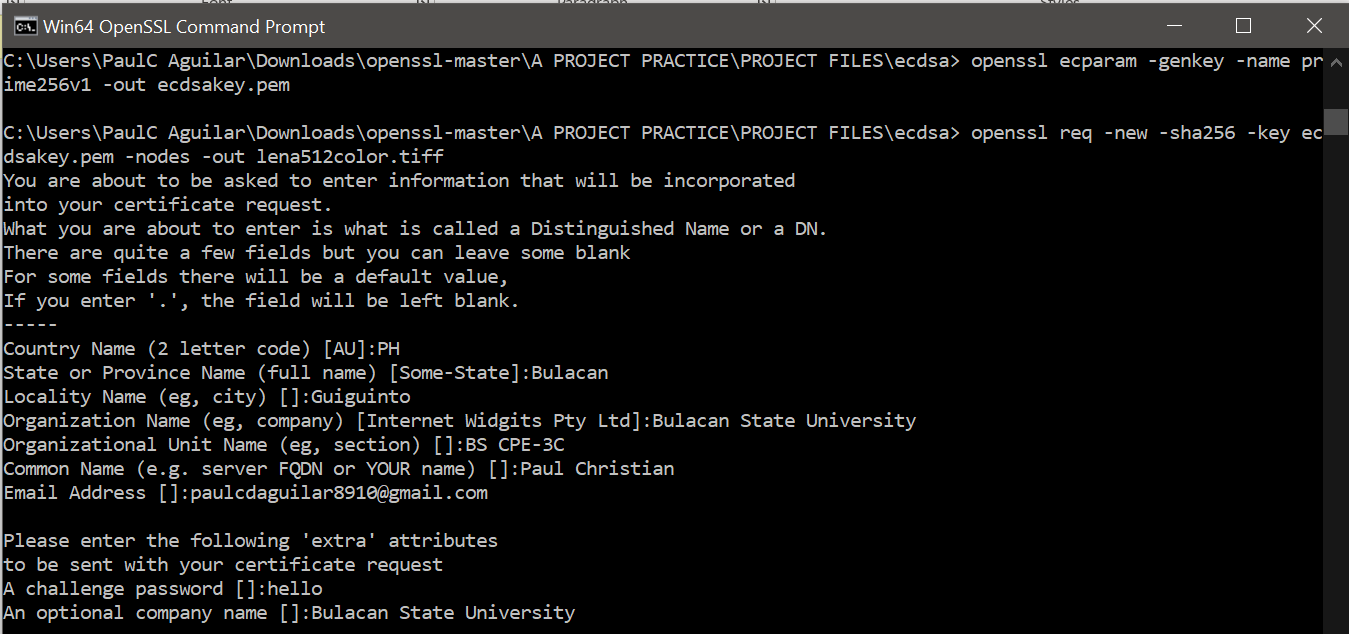
* It was indicated to use SHA-256 so the chosen one within the library is “prime256v1”.
* In this case, the following procedure can be done to do ECDSA encryption:



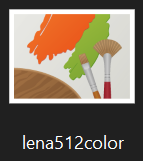
* The code will generate a private key in preparation for the main encryption.



* After that, the code below will perform the ECDSA encryption certificate:



* Supposedly, there will be an emailed CA file to the given email address. Yet, nothing came. Due to this, the ECDSA encryption process was not fully completed.
* Nonetheless, the original lena512color.tiff file was altered and hidden, so it is still successfully encrypted.



References:

<https://rietta.com/blog/openssl-generating-rsa-key-from-command/>

<https://github.com/StormWindStudios/OpenSSL-Notes/blob/master/ecdsa.md>

<https://www.digicert.com/kb/ssl-support/openssl-quick-reference-guide.htm>