

# **CRYPTOGRAPHY PROGRAM USING PYTHON AND OPENSLL**

Submitted by:

Rahino C. Qujano

## Machine's Operating System during Development and Testing

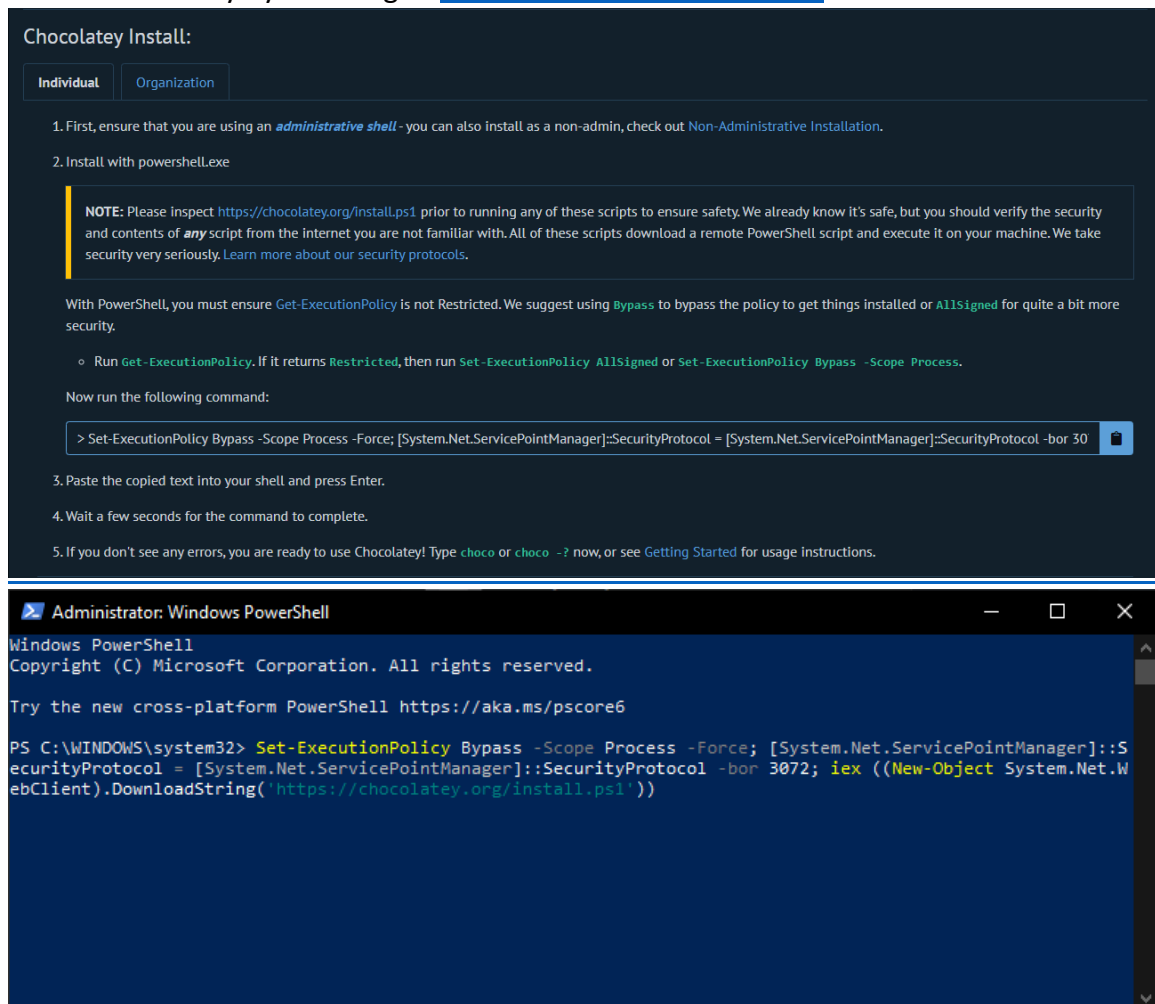
- Windows 10 Home Single Language

## Machine's Essential Software to install dependent programs and to run the code itself

- Python 3.9.1 – For running the program source code
- Chocolatey v0.10.15 – For installing OpenSSL
- OpenSSL 1.1.1i 8 Dec 2020 – For providing command line cryptography tools

## Installing program dependencies

1. Install Python 3.9.x from <https://www.python.org/downloads/>
2. Verify Python from shell using the command (do not include ">>")  
`>> python --version`
3. Install Chocolatey by referring to <https://chocolatey.org/install>



The image shows the Chocolatey installation instructions and a terminal window. The top part is a screenshot of the Chocolatey website's installation guide, which includes steps for ensuring administrative shell access, installing PowerShell, and running a command to bypass execution policy. The bottom part is a screenshot of a Windows PowerShell terminal window where the command from the guide has been executed.

**Chocolatey Install:**

**Individual** | **Organization**

1. First, ensure that you are using an **administrative shell** - you can also install as a non-admin, check out [Non-Administrative Installation](#).

2. Install with powershell.exe

**NOTE:** Please inspect <https://chocolatey.org/install.ps1> prior to running any of these scripts to ensure safety. We already know it's safe, but you should verify the security and contents of **any** script from the internet you are not familiar with. All of these scripts download a remote PowerShell script and execute it on your machine. We take security very seriously. [Learn more about our security protocols](#).

With PowerShell, you must ensure **Get-ExecutionPolicy** is not **Restricted**. We suggest using **Bypass** to bypass the policy to get things installed or **AllSigned** for quite a bit more security.

- Run **Get-ExecutionPolicy**. If it returns **Restricted**, then run **Set-ExecutionPolicy AllSigned** or **Set-ExecutionPolicy Bypass -Scope Process**.

Now run the following command:

```
> Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
```

3. Paste the copied text into your shell and press Enter.

4. Wait a few seconds for the command to complete.

5. If you don't see any errors, you are ready to use Chocolatey! Type **choco** or **choco -?** now, or see [Getting Started](#) for usage instructions.

**Administrator: Windows PowerShell**

Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/powershell>

```
PS C:\WINDOWS\system32> Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
```

4. Verify Chocolatey from shell using the command (do not include ">>")  
 >> choco --version
5. Restart your admin shell and install OpenSSL by typing the given command on an administrative shell (do not include ">>")  
 >> choco install openssl
6. If OpenSSL cannot be installed using choco, open the link to download the installer  
<https://slproweb.com/products/Win32OpenSSL.html?fbclid=IwAR2PCdPsUL0J3ZgJlLeEUICOY0qq6rg7yyFy2BRW6LkiMiJuytPK6aTyWYEdownload%20python> and choose the 63 MB EXE installer depending on your System Architecture

Download Win32/Win64 OpenSSL		
Download Win32/Win64 OpenSSL today using the links below!		
File	Type	Description
<a href="#">Win64 OpenSSL v1.1.1 Light EXE   MSI</a>	3MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v1.1.1. (Recommended for users by the creators of <a href="#">OpenSSL</a> ). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
<a href="#">Win64 OpenSSL v1.1.1 EXE   MSI</a>	63MB Installer	Installs Win64 OpenSSL v1.1.1. (Recommended for software developers by the creators of <a href="#">OpenSSL</a> ). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
<a href="#">Win32 OpenSSL v1.1.1 Light EXE   MSI</a>	3MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v1.1.1. (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
<a href="#">Win32 OpenSSL v1.1.1 EXE   MSI</a>	54MB Installer	Installs Win32 OpenSSL v1.1.1. (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

7. Install the OpenSSL and after the installation, add the OpenSSL directory to path in the system environment variables
8. Verify OpenSSL from shell using the command (do not include ">>")  
 >> openssl version

## Code Tour

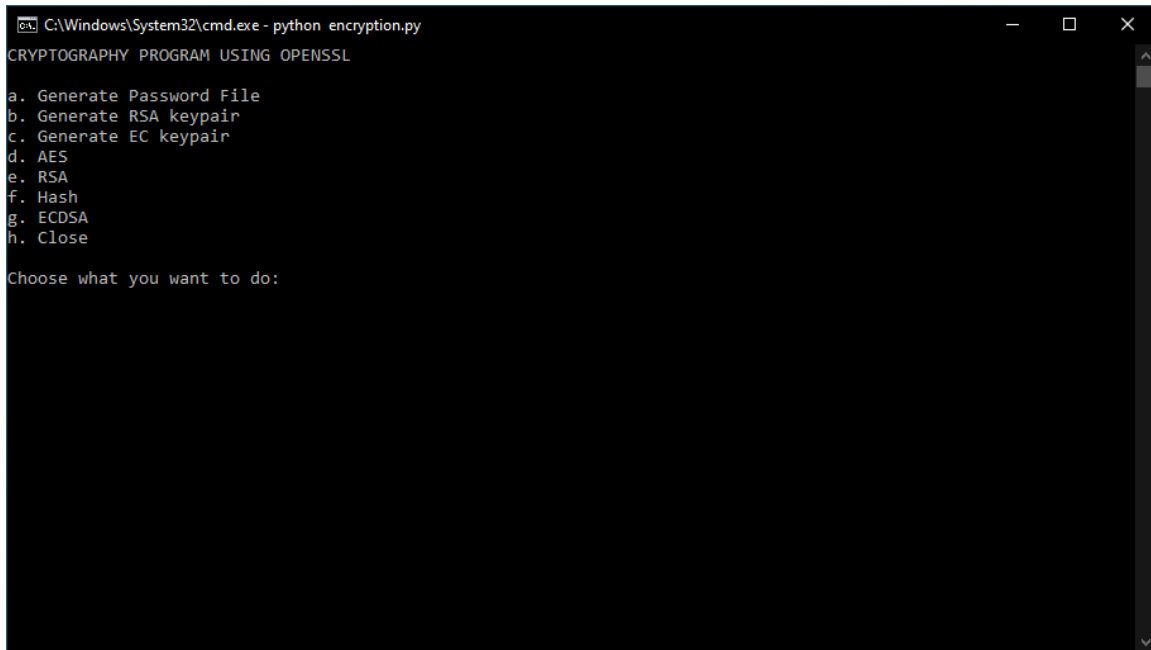
The program's source code was organized into two main sections. The first section is the Primitive Operations designated by a single line comment which contains multiple functions that encapsulates a particular OpenSSL cryptographic operation like AES encryption, RSA encryption, Hashing and more. The second section is the Simplified Operations which is primarily dominated by a single function called "interface" that provides a user interface and rudimentary error handling. Interface is then called at the main function which is then called after a standalone-use checking. The program imports two standard python modules namely "os" and "subprocess".

## Program Usage

The program can either be use as a module to be imported to other files or it can be used as a standalone program. Program usage will describe how the program will be used as a standalone python program.

The program provides four main encryption services, Symmetric Encryption using AES, Asymmetric Encryption using RSA, Hashing and Digital Signature using Elliptic Curve Digital Signature Algorithm but three additional optionalities were added to provide more controls for the user, these are Password File, RSA key pair and Elliptic Curve keypair generation.

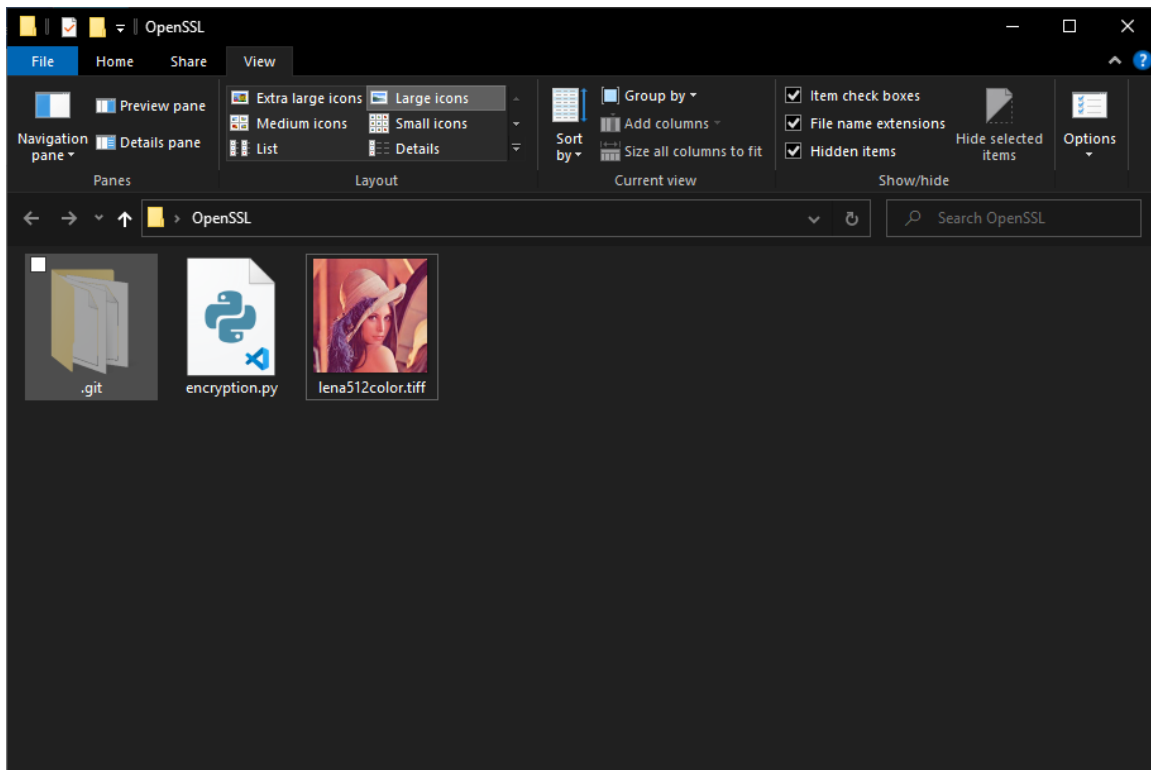
To launch the program, type “python encryption.py” in the command prompt or any shell and upon execution, you will see the display of the program.



```
C:\Windows\System32\cmd.exe - python encryption.py
CRYPTOGRAPHY PROGRAM USING OPENSSL

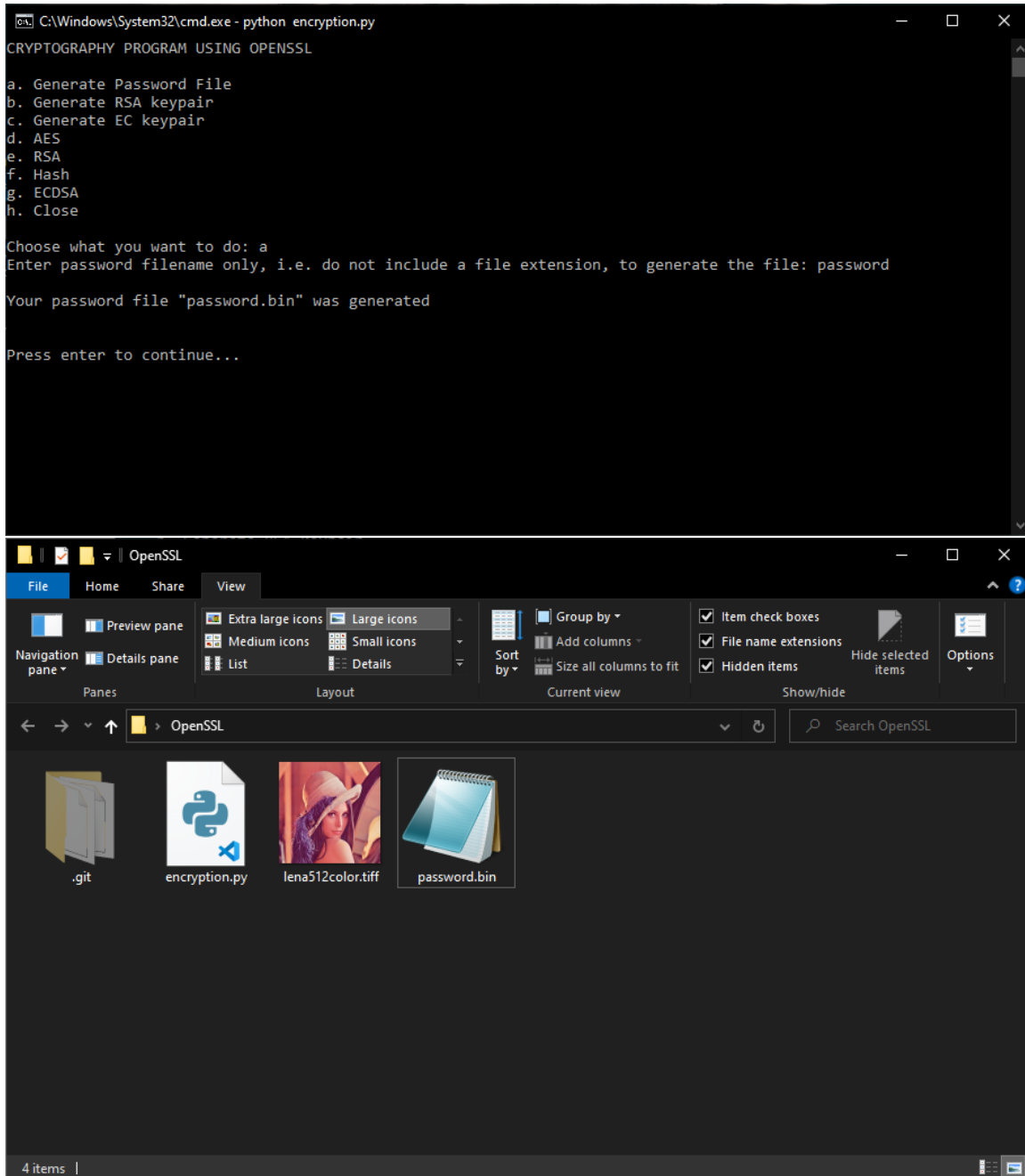
a. Generate Password File
b. Generate RSA keypair
c. Generate EC keypair
d. AES
e. RSA
f. Hash
g. ECDSA
h. Close

Choose what you want to do:
```



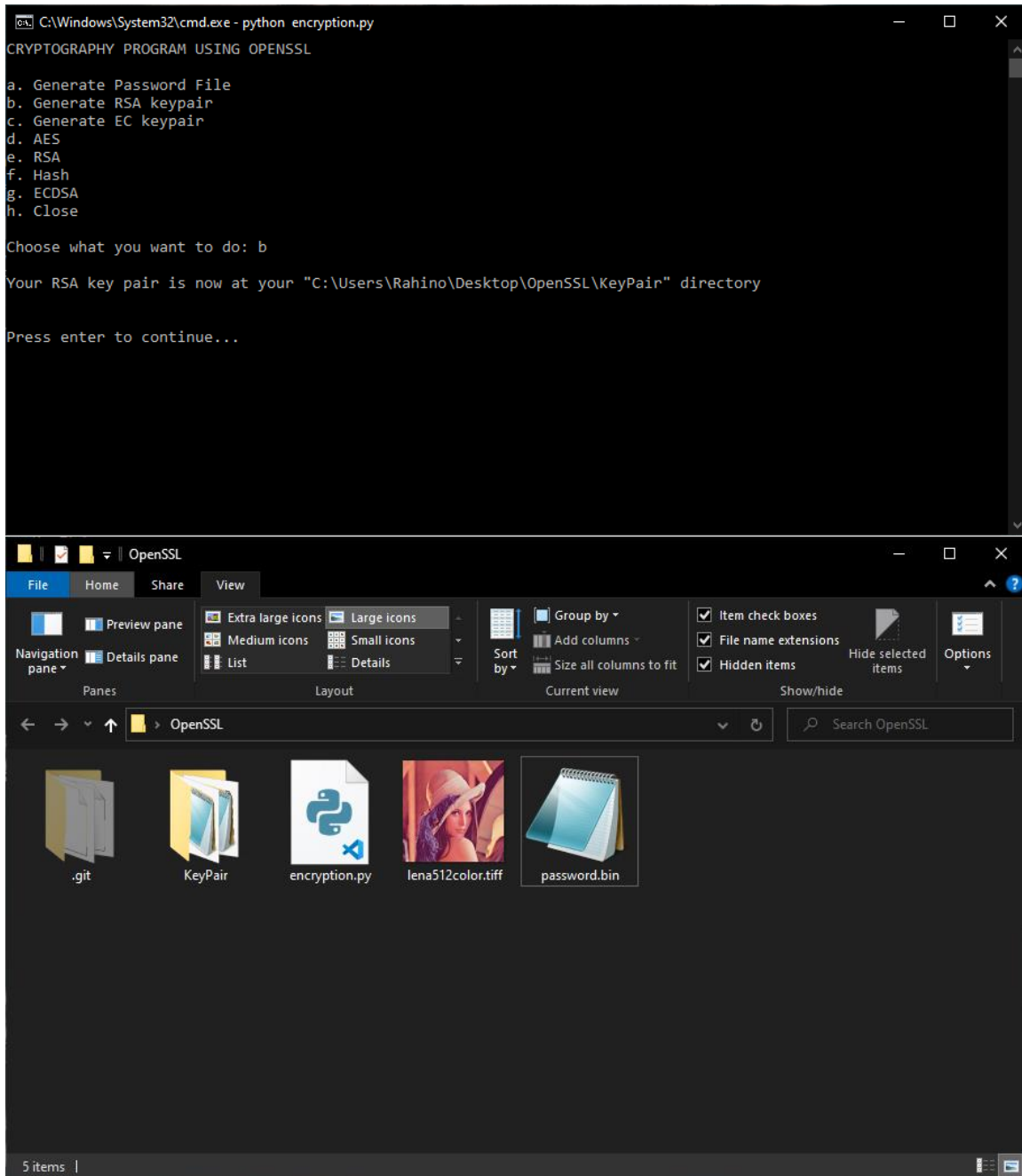
a. **Generate Password File** – option a. provides you a way to generate a password based on 64-bit random hexadecimal number.

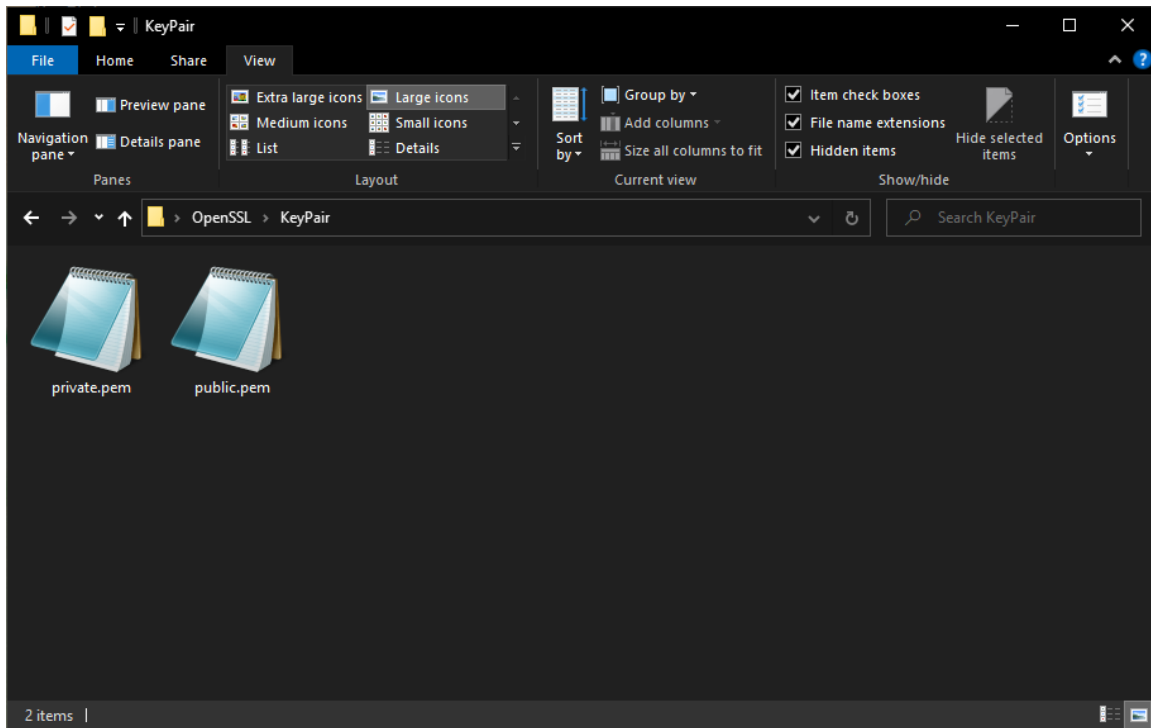
1. Press “A” key and hit Enter key
2. Type your desired filename for your password file and hit Enter
3. Check your Directory where the encryption.py file is located and you will see your file created with .bin file extension



b. **Generate RSA keypair** – option b. provides you a way to Generate 2048-bit private key and its corresponding public key

1. Press “B” key and hit Enter key
2. The RSA key pair will be automatically generated inside a folder name KeyPair





- c. **Generate EC keypair** – option c. provides you a way to Generate elliptic curve key pair using secp384r1 EC curve
1. Press “C” key and hit Enter key
  2. The EC key pair will be automatically generated inside a folder name ECpair

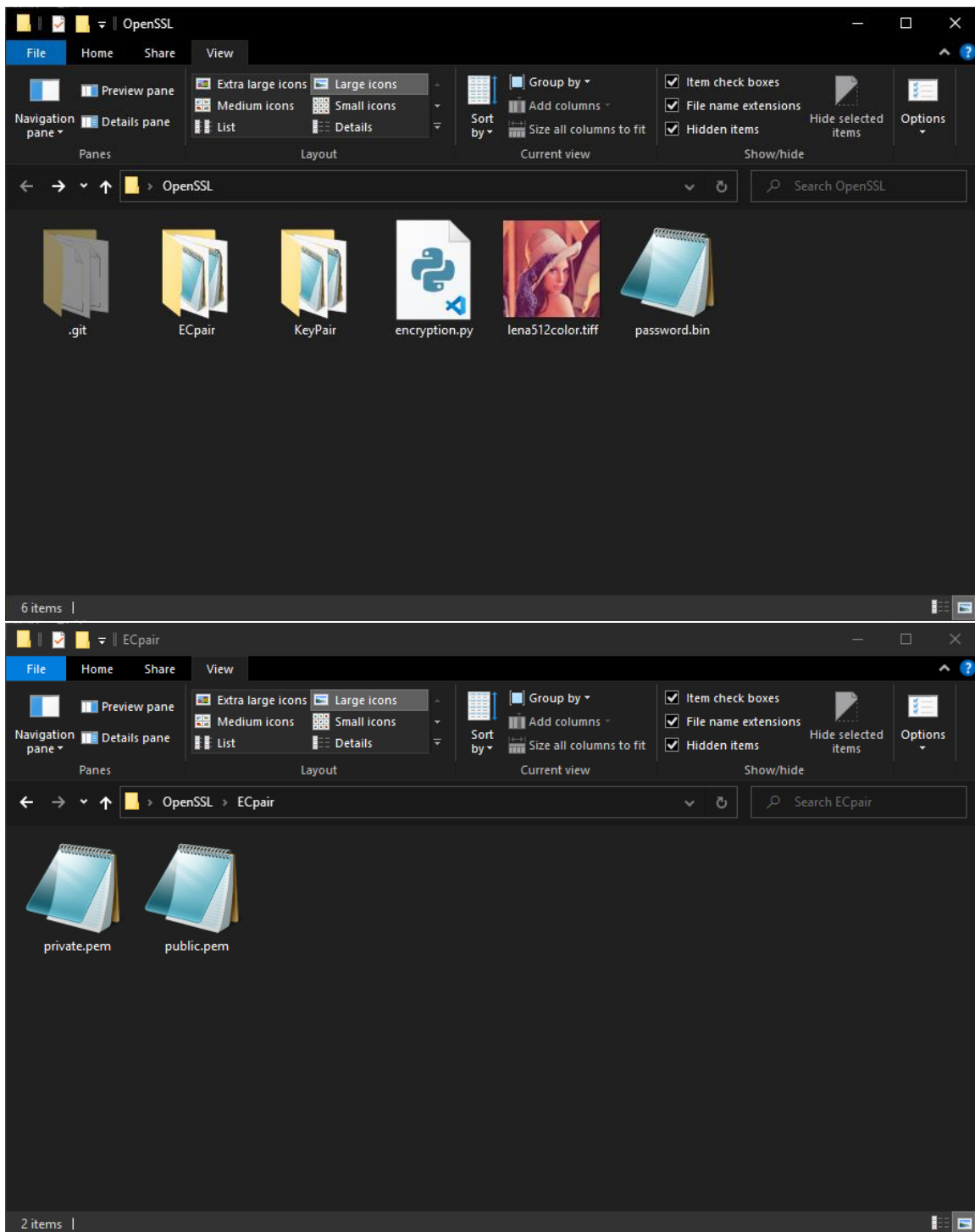
```
C:\Windows\System32\cmd.exe - python encryption.py
CRYPTOGRAPHY PROGRAM USING OPENSSL

a. Generate Password File
b. Generate RSA keypair
c. Generate EC keypair
d. AES
e. RSA
f. Hash
g. ECDSA
h. Close

Choose what you want to do: c

Your EC key pair is now at your "C:\Users\Rahino\Desktop\OpenSSL\ECpair" directory

Press enter to continue...
```





- d. **AES** – d. option provides you tools for symmetric encryption and decryption using Advance Encryption Standard defaulted to 128-bit block encryption

**ENCRYPTION**

1. Press “D” key and hit Enter key
2. Press “A” key and hit Enter key to choose “Encrypt” option
3. Enter the filename you want to encrypt and include the file extension  
“lena512color.tiff”
4. Specify the mode of operation either “ecb” or “cbc”. The sample specifies “ecb”.
5. Specify whether you want to use file-based password or type-in password. The sample uses file-based password.
6. Specify the password file’s filename including its file extension. Sample uses  
“password.bin” the one generated earlier. If the user chooses not to use file-base password, this step will ask the type-in password from the user
7. Encrypted version of the file was generated with an extension of “.enc”

```
C:\Windows\System32\cmd.exe - python encryption.py
CRYPTOGRAPHY PROGRAM USING OPENSAL

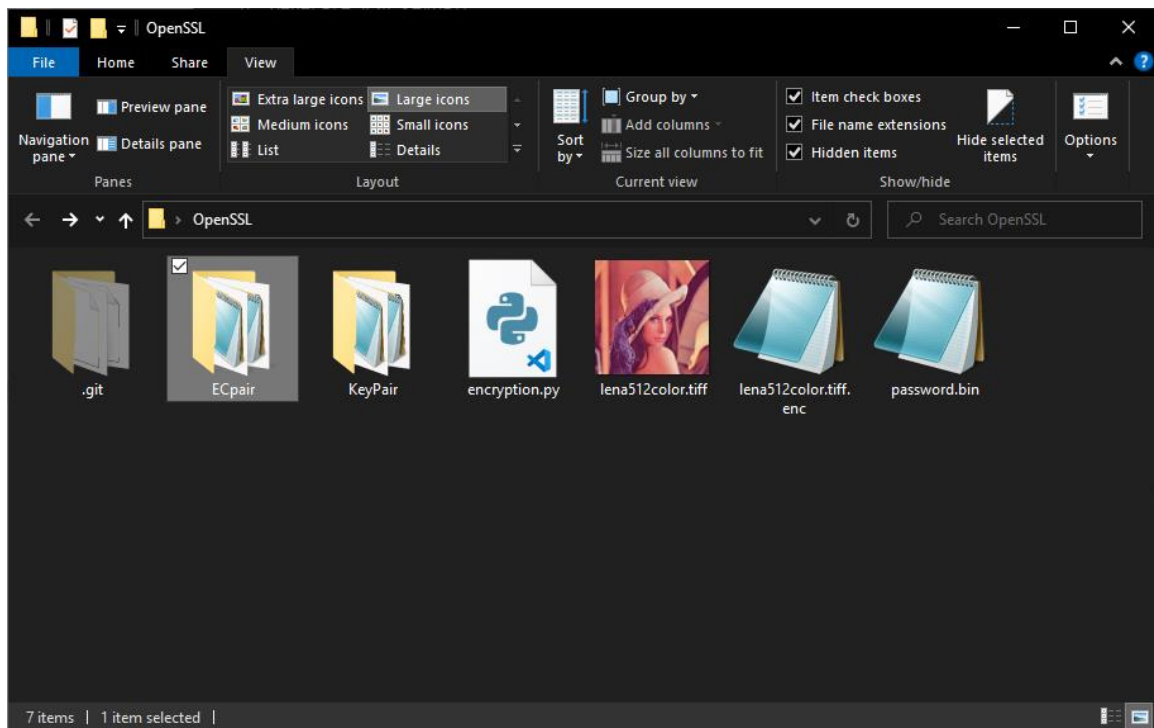
a. Generate Password File
b. Generate RSA keypair
c. Generate EC keypair
d. AES
e. RSA
f. Hash
g. ECDSA
h. Close

Choose what you want to do: d
a. Encrypt
b. Decrypt
Choose one: a
Enter filename to be encrypted: lena512color.tiff
Enter AES mode ecb / cbc : ecb
Do you want to use a password file? y/n : y
Enter password filename: password.bin

"lena512color.tiff.enc" generated and can now be sent

*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

Press enter to continue...
```



## **DECRYPTION**

1. Press “D” key and hit Enter key
2. Press “B” key and hit Enter key to choose “Decrypt” option
3. Enter the filename you want to decrypt and include the file extension  
“lena512color.tiff.enc”
4. Specify the mode of operation either “ecb” or “cbc”. The sample specifies “ecb”.  
Take note, mode of decryption should match with mode of encryption.
5. Specify whether you want to use file-based password or type-in password. The sample uses file-based password.
6. Specify the password file’s filename including its file extension. Sample uses  
“password.bin” the one generated earlier. If the user chooses not to use file-  
base password, this step will ask the type-in password from the user
7. Decrypted version of the file was generated with “\_new” appended on its  
filename

```
C:\Windows\System32\cmd.exe - python encryption.py
CRYPTOGRAPHY PROGRAM USING OPENSLL

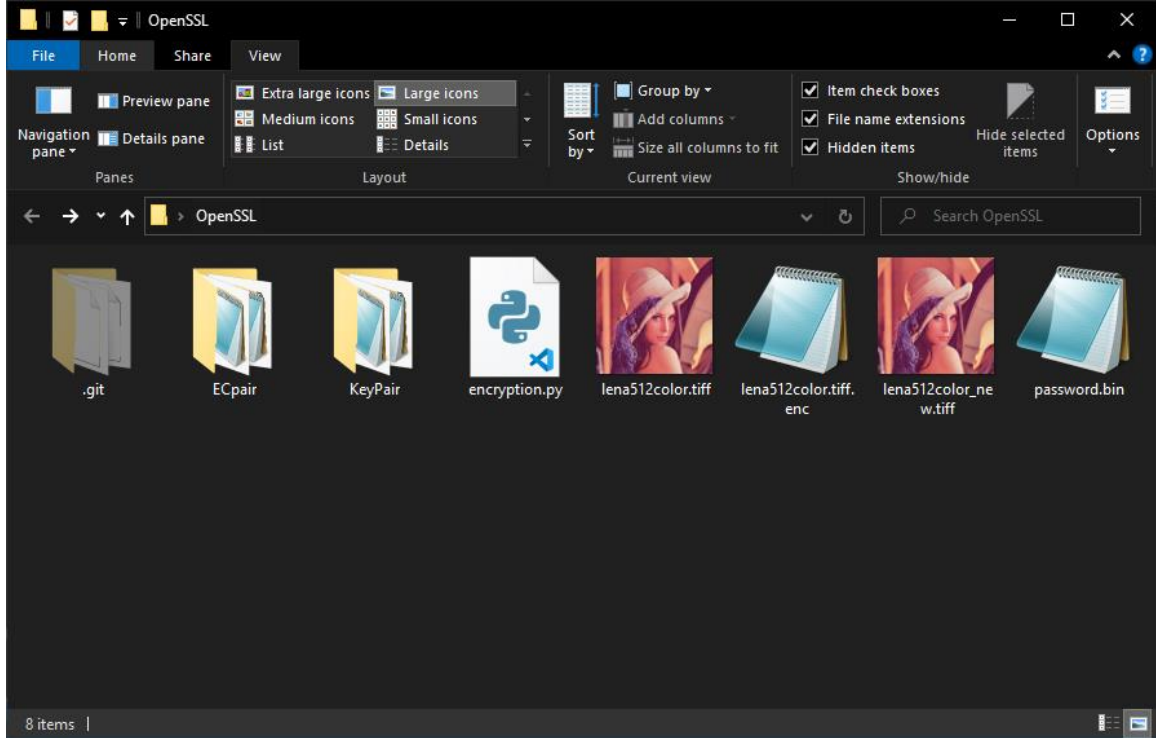
a. Generate Password File
b. Generate RSA keypair
c. Generate EC keypair
d. AES
e. RSA
f. Hash
g. ECDSA
h. Close

Choose what you want to do: d
a. Encrypt
b. Decrypt
Choose one: b
Enter filename to be decrypted: lena512color.tiff.enc
Enter AES mode ecb / cbc : ecb
Has password file? y/n : y
Enter password filename: password.bin

"lena512color.tiff.enc" now decrypted into a new file

*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

Press enter to continue...
```



- e. **RSA** – option e. provides you tools for public key encryption using both AES 128-bit CBC encryption and decryption for the main file and Rivest–Shamir–Adleman (RSA) encryption and decryption for the 64-bit password used as a key for the preceding AES encryption. AES was used for the main file because RSA can only encrypt files with sizes smaller than the RSA keypair bit, hence RSA public key was only used for the 64-bit key used in AES.

### ENCRYPTION

1. Press “E” key and hit Enter key
2. Press “A” key and hit Enter key to choose “Encrypt” option
3. Specify if you already have a public key. In this case sample already has the public key. In case you do not have, program will automatically generate RSA keypair.
4. Specify if you already have a password. In this case the password that the program generates is actually a 64-bit key for symmetric encryption. The sample already generates it, if you do not have, program will prompt you a filename and automatically generates the password file.
5. Specify the filename of the file to be encrypted including its file extension “lena512color.tiff”
6. Specify the password file’s filename including its file extension. Sample uses “password.bin” the one generated earlier
7. Specify the public key’s filename including its file extension. Sample uses “keypair/public.pem” inside the KeyPair folder.
8. Symmetrically encrypted target file and Asymmetrically encrypted AES key is now ready to be sent over given that the receiving end has the private key for the public key you used in AES key asymmetric encryption

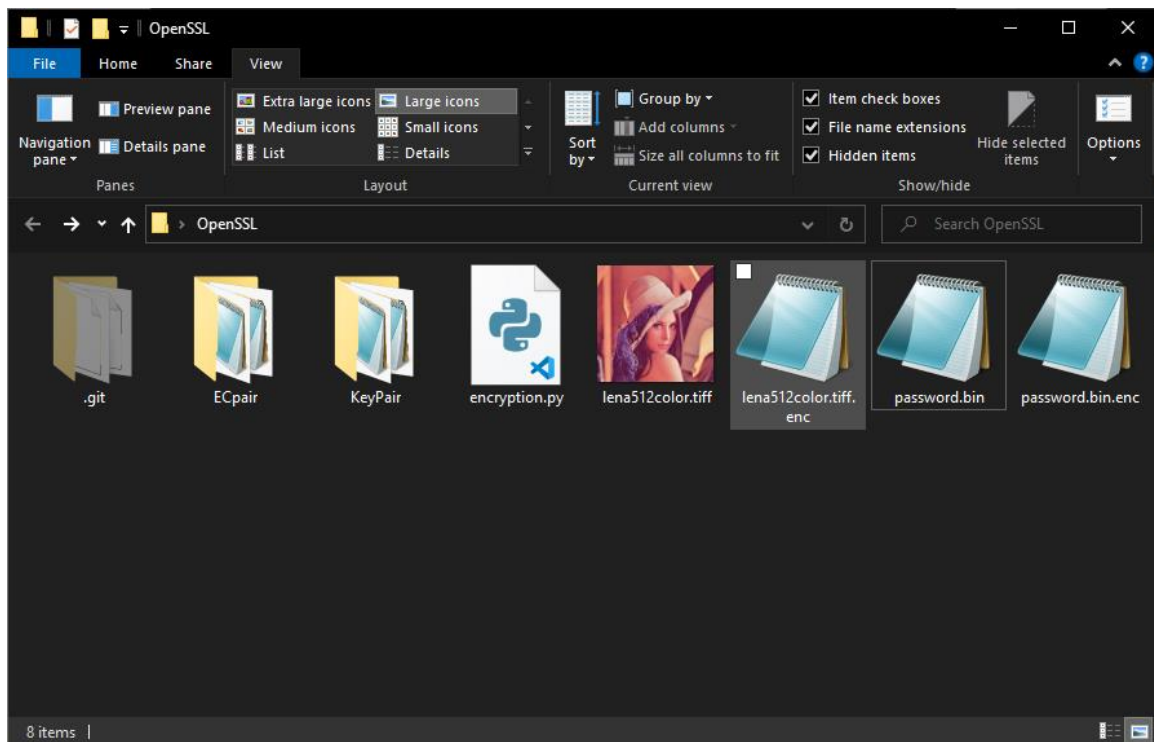
```
C:\Windows\System32\cmd.exe - python encryption.py
a. Generate Password File
b. Generate RSA keypair
c. Generate EC keypair
d. AES
e. RSA
f. Hash
g. ECDSA
h. Close

Choose what you want to do: e
a. Encrypt
b. Decrypt
Choose one: a
Do you have public key? y/n : y
Do you have password? y/n : y
Enter filename to be encrypted: lena512color.tiff
Enter generated password filename: password.bin
Enter public key filename: keypair/public.pem

You can now send back the "lena512color.tiff.enc" and "password.bin.enc"
given that the receiving end has the key pair and you only have the public key

*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

Press enter to continue...
```



## **DECRYPTION**

1. Press “E” key and hit Enter key
2. Press “B” key and hit Enter key to choose “Decrypt” option
3. Specify the filename of the file to be decrypted including its file extension  
“lena512color.tiff.enc”
4. Specify the encrypted password’s (AES key) filename including its file extension  
“password.bin.enc”
5. Specify the private key’s filename including its file extension. Sample uses  
“keypair/private.pem” inside the KeyPair folder.
6. Symmetrically encrypted target file and Asymmetrically encrypted AES key is  
now ready to be sent over given that the receiving end has the private key for  
the public key you used in AES key asymmetric encryption
7. Decrypted version of the file was generated with “\_new” appended on its  
filename

```
C:\Windows\System32\cmd.exe - python encryption.py
CRYPTOGRAPHY PROGRAM USING OPENSSSL

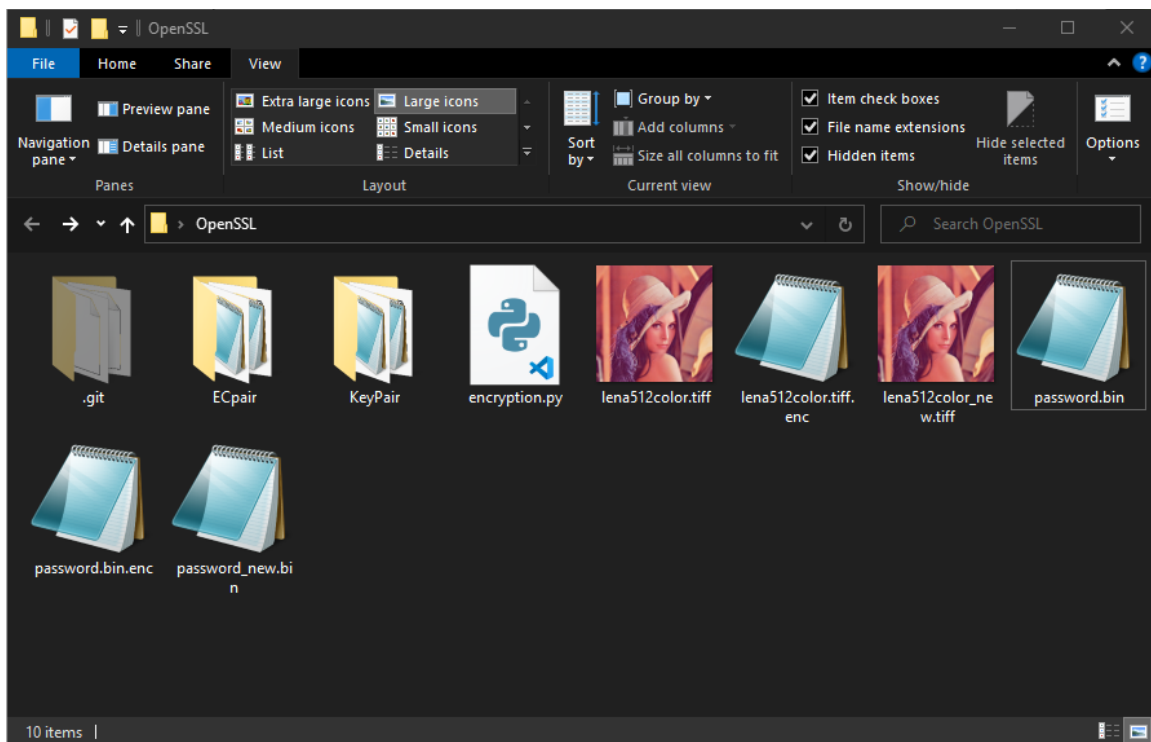
a. Generate Password File
b. Generate RSA keypair
c. Generate EC keypair
d. AES
e. RSA
f. Hash
g. ECDSA
h. Close

Choose what you want to do: e
a. Encrypt
b. Decrypt
Choose one: b
Enter filename to be decrypted: lena512color.tiff.enc
Enter encrypted password filename: password.bin.enc
Enter private key filename: keypair/private.pem

"lena512color.tiff.enc" now decrypted into a new file

*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

Press enter to continue...
```



f. **Hash** – option f. provides you tools for hashing a file using Secure Hash Algorithm

1. Press “F” key and hit Enter key
2. Enter the filename you want to hash including its file extension  
“lena512color.tiff”
3. Specify SHA version “1”, “256” or “512”. Sample uses “256”
4. Hash was generated with “\_sha256” appended on its filename

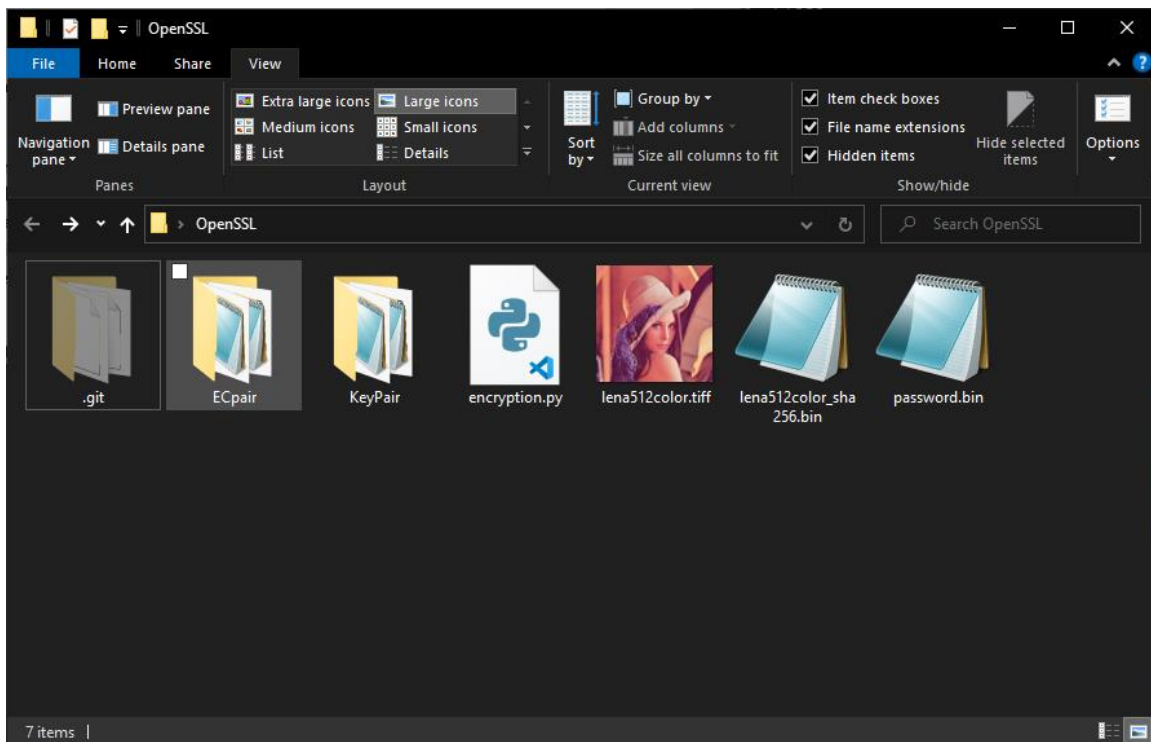
```
C:\Windows\System32\cmd.exe - python encryption.py
CRYPTOGRAPHY PROGRAM USING OPENSSL

a. Generate Password File
b. Generate RSA keypair
c. Generate EC keypair
d. AES
e. RSA
f. Hash
g. ECDSA
h. Close

Choose what you want to do: f
Enter filename: lena512color.tiff
Enter version 1 / 256 / 512 : 256

"lena512color.tiff" has was generated using sha256

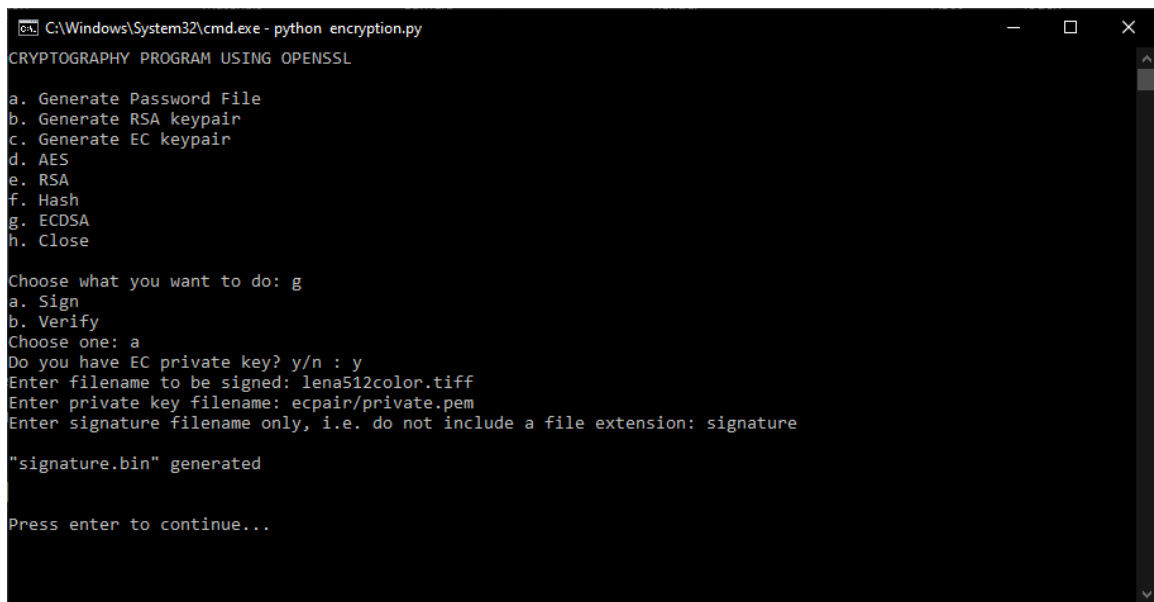
Press enter to continue...
```



- g. **ECDSA** – option g. provides you tools for signing and verifying files using Elliptic Curve Digital Signature Algorithm

### ***SIGNING***

1. Press “G” key and hit Enter key
2. Press “A” key and hit Enter key to choose “Sign” option
3. Specify if you have EC private key. If not, the program will automatically generate EC keypair inside ECpair folder.
4. Specify the filename of the file to be signed including its file extension  
“lena512color.tiff”
5. Specify the EC private key’s filename including its file extension  
“ecpair/private.pem”
6. Specify the filename for your signature file. Sample used “signature”
7. “signature.bin” is now generated and can be used for file verification



```
C:\Windows\System32\cmd.exe - python encryption.py
CRYPTOGRAPHY PROGRAM USING OPENSSL

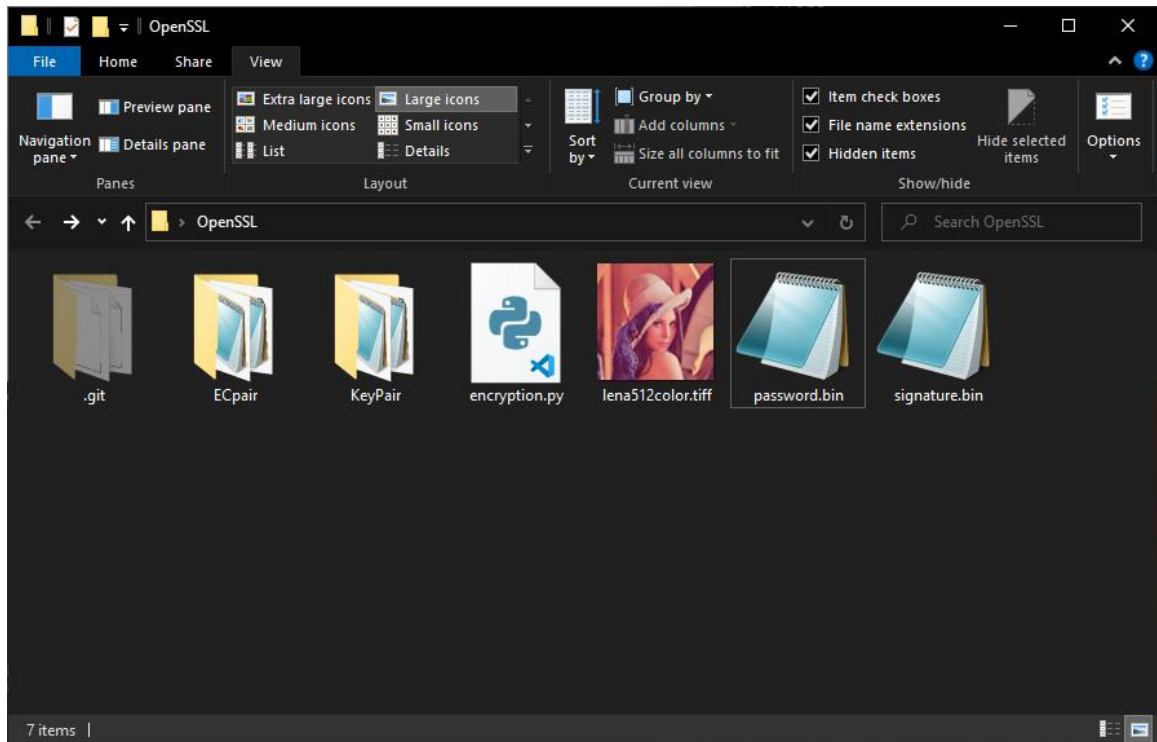
a. Generate Password File
b. Generate RSA keypair
c. Generate EC keypair
d. AES
e. RSA
f. Hash
g. ECDSA
h. Close

Choose what you want to do: g
a. Sign
b. Verify
Choose one: a
Do you have EC private key? y/n : y
Enter filename to be signed: lena512color.tiff
Enter private key filename: ecpair/private.pem
Enter signature filename only, i.e. do not include a file extension: signature

"signature.bin" generated

Press enter to continue...
```





## VERIFICATION

1. Press "G" key and hit Enter key
2. Press "B" key and hit Enter key to choose "Verify" option
3. Specify the filename of the file to be verified including its file extension  
"lena512color.tiff"
4. Specify the EC public key's filename including its file extension  
"ecpair/public.pem"
5. Specify the filename for your signature file including its file extension  
"signature.bin"
6. Program prints "lena512color.tiff" Verified OK meaning that the file is authentic.  
Program will show an alternate print when the file is inauthentic.

```
C:\Windows\System32\cmd.exe - python encryption.py
CRYPTOGRAPHY PROGRAM USING OPENSLL

a. Generate Password File
b. Generate RSA keypair
c. Generate EC keypair
d. AES
e. RSA
f. Hash
g. ECDSA
h. Close

Choose what you want to do: g
a. Sign
b. Verify
Choose one: b
Enter filename to be verified: lena512color.tiff
Enter public key filename: ecpair/public.pem
Enter signature filename: signature.bin

"lena512color.tiff" Verified OK

Press enter to continue...
```

