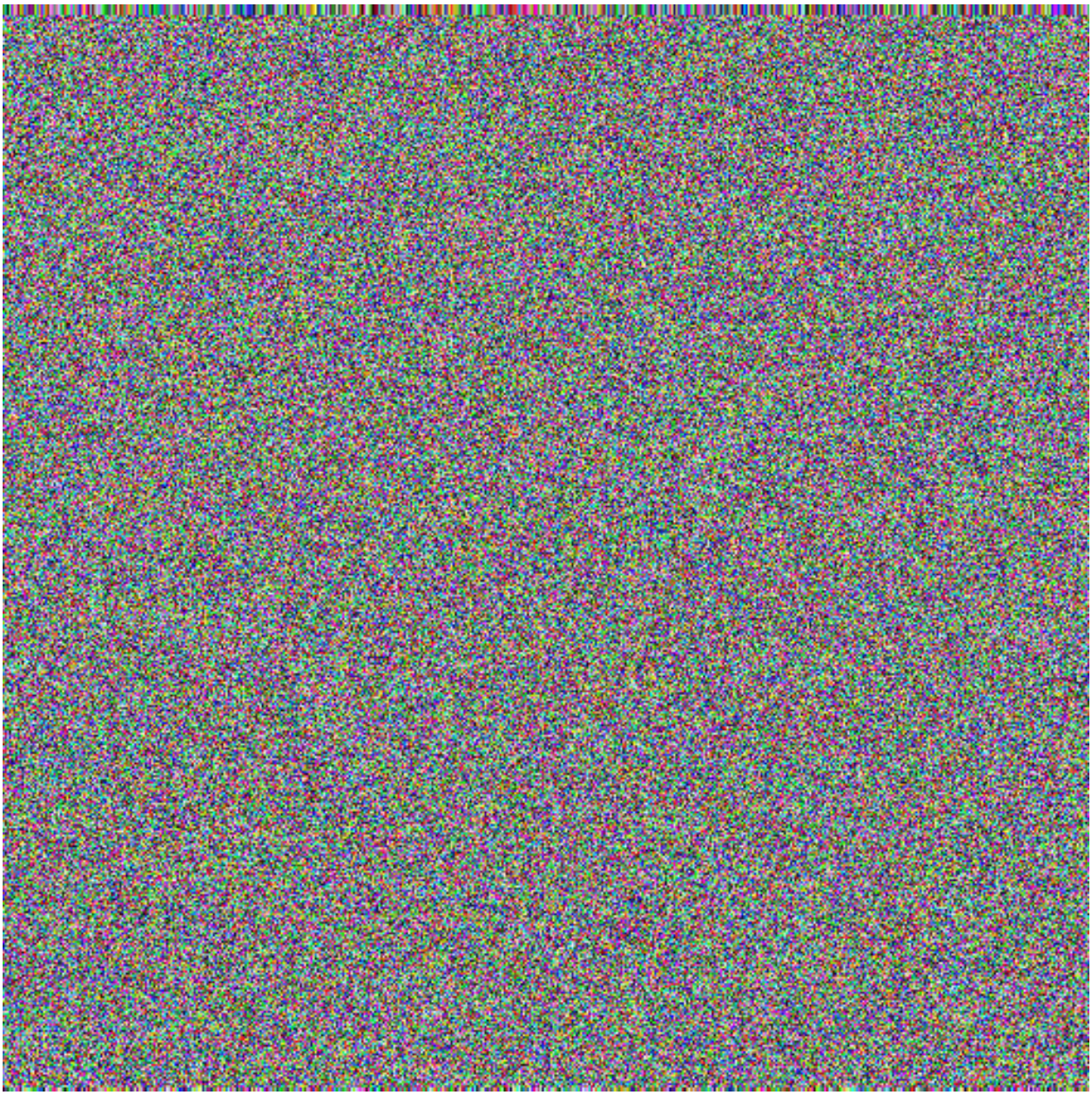
Symmetric Encryption

Original Image

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.

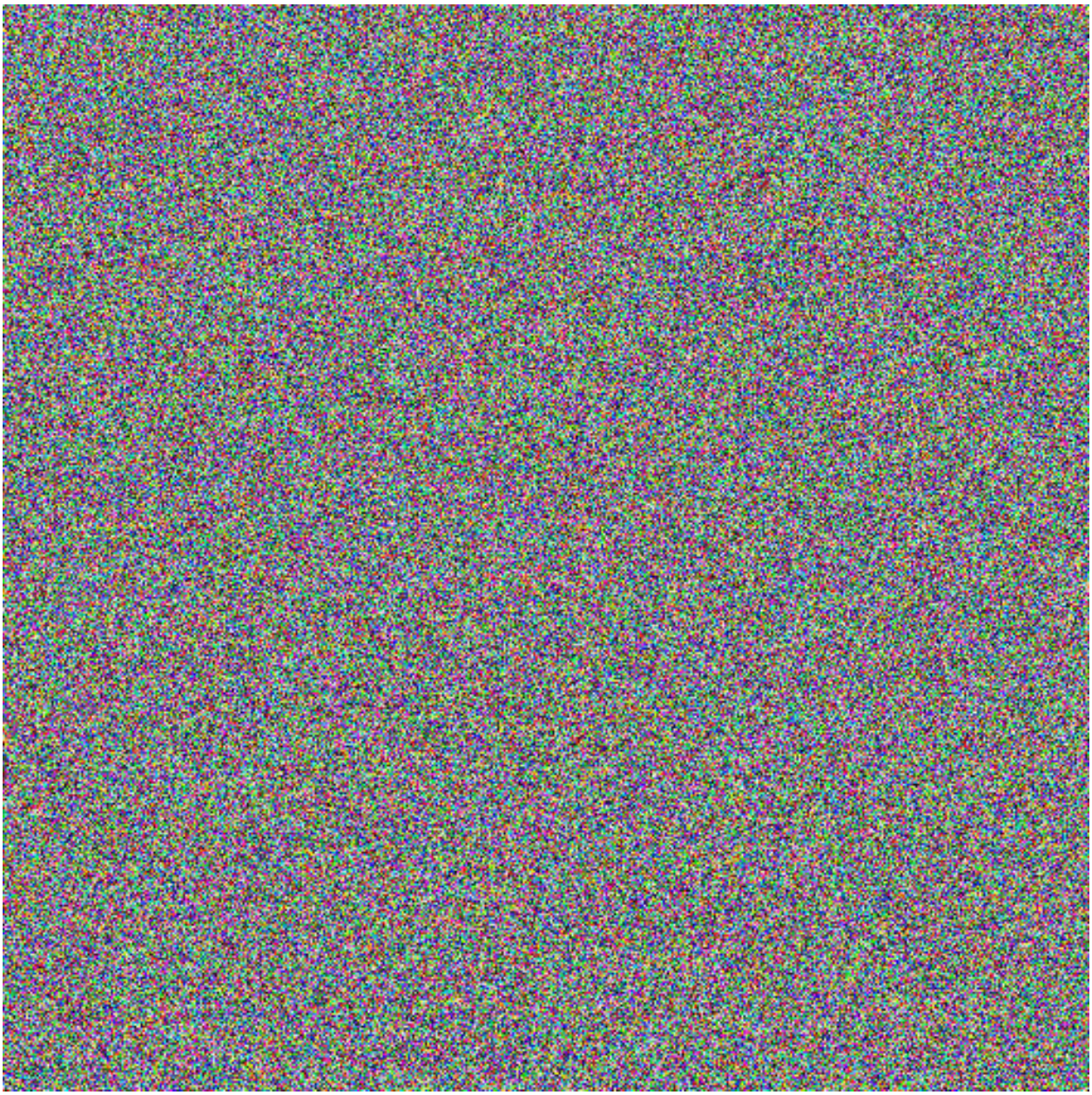
```
head -n 3 image.ppm > header.txt
tail -n +4 image.ppm > body.bin
openssl enc -aes-128-ecb -e -in  body.bin -out body.ecb.bin -K
1001011
cat header.txt body.ecb.bin > image.ecb.ppm
```

ECB Image



```
openssl enc -aes-128-cbc -e -in  body.bin -out body.cbc.bin -K
1001011 -iv 0010011
cat header.txt body.cbc.bin > image.cbc.ppm
```

CBC Image



Hashing

Using OpenSSL, hash the same Lena 512x512 image using the following hash functions: SHA-1, SHA-256, SHA-512.

```
openssl dgst -sha1 -out sha1.txt image.tiff
SHA1(image.tiff)= e647d0f6736f82e498de8398eccc48cf0a7d53b9

openssl dgst -sha256 -out sha256.txt image.tiff
SHA256(image.tiff)=
c056da23302d2fb0d946e7ffa11e0d94618224193ff6e2f78ef8097bb8a3569b

openssl dgst -sha512 -out sha512.txt image.tiff
SHA512(image.tiff)=
2cb9d7df53eb8640dc48d736974f472a98d9c7186de7a972490455f5f3ed29df
c5b75c95ccb3ed4596bc2bfc4b1e52cf4d76bcee27d334dd155bb426617392dc
```

Public Key Encryption

```
#!/bin/sh

#RSA-AES Encryption
openssl genrsa -out private.pem 2048
openssl rsa -in private.pem -outform PEM -pubout -out public.pem

openssl rand -base64 32 > key.bin

openssl rsautl -encrypt -inkey public.pem -pubin -in key.bin -
out key.bin.enc
openssl enc -aes-256-cbc -salt -in image.tiff -out
image.tiff.enc -pass file:./key.bin


#ECDSA Signature
openssl dgst -sha256 -sign private.pem image.tiff >
imagesignature.der
openssl dgst -sha256 -hex -sign private.pem image.tiff >
rsasha.txt

hexdump imagesignature.der

openssl dgst -sha256 -verify public.pem -signature
imagesignature.der image.tiff
```

RSA-SHA256(image.tiff)=
3665a5bbca98713f421d9de67b97ea91083b82799333a2d9c88c7ce4cd304bf8
619054ffa011f21bf6f6ad52564331d208f7af85bebafa08d2b8dfba1332a443
b1b1ec67f31c0ce77b1e2538e84397b528690ca00d59bd6d8b0dd0d4d47e9ce4
2326c1765a1fa34b599d78d7b32a988721b142e3000ad866df04293cdef8e550
ee4e7be5e3ea16a761320bf4fc3ce67cf5a60fd3314e85b5edcddc357128e7cf
eaec1cbb640feaa8ac679d7d6fed8d7e1bccabea11daca91f2d2d472489acd5b
521ac085aaadc04ed4665bfbbe459988f467a898e79e58149cd934762312e31a
50a3ee68b5249b8e0a11ea6b171e21381f25af08c631a94ea87d7b3e980bb549

Reference:

[1] https://blog.filippo.io/the-ecb-penguin/
[2] https://users.dcc.uchile.cl/~pcamacho/tutorial/crypto/openssl/openssl_intro.html
[3] http://davidederosa.com/basic-blockchain-programming/elliptic-curve-digital-signatures/