Cheryl M. Siy
2010-25696
CS 253

I. Symmetric Encryption

For this exercise, the command line of Mac OS Sierra version:10.12.4 was used. A *.bmp file was used to easier see the output. [1,2] was used as a reference on how to encrypt *.bmp files. Note that encrypted *.bmp files needs to be manually overwritten with 54 bits of header file(from the original photo). Thus, ‘dd command’ [3] was used.

ECB

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl enc -aes-128-ecb -
a -in lena.bmp -out ECB128lena.bmp -k test1234
```

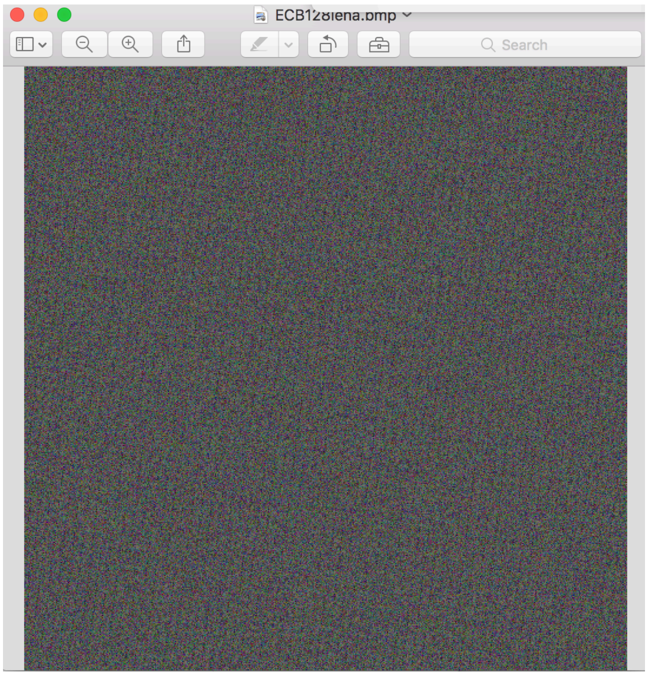
```
Cheryls-MacBook:CS253openssl cherylsiy$ dd if=lena.bmp
of=ECB128lena.bmp bs=1 count=54 conv=notrunc
54+0 records in
54+0 records out
54 bytes transferred in 0.000125 secs (432237 bytes/sec)
```

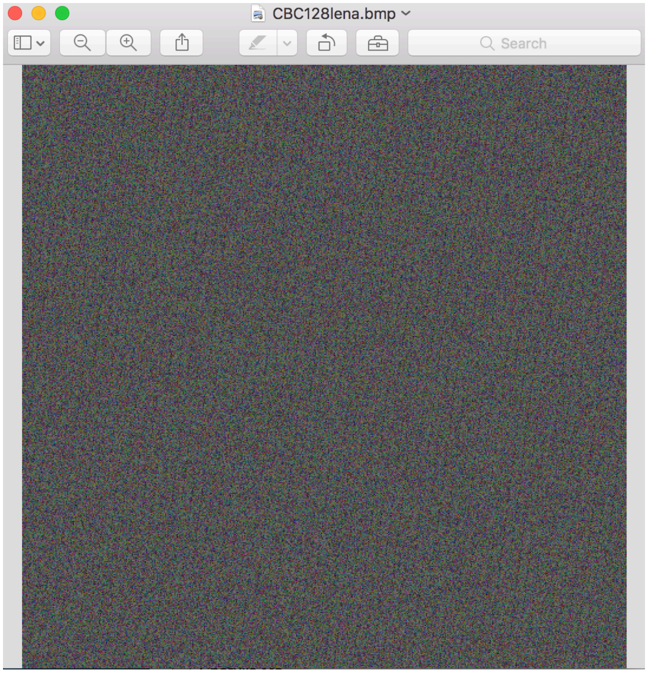
CBC

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl enc -aes-128-cbc -
a -in lena.bmp -out CBC128lena.bmp -k test1234
```

```
Cheryls-MacBook:CS253openssl cherylsiy$ dd if=lena.bmp
of=CBC128lena.bmp bs=1 count=54 conv=notrunc
54+0 records in
54+0 records out
54 bytes transferred in 0.000121 secs (446731 bytes/sec)
```

Output:





II. Hashing

Hashing was done via terminal. Syntax was guided by the openssl dgst manual [4].

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl dgst -sha1 lena.tiff
SHA1(lena.tiff)= e647d0f6736f82e498de8398eccc48cf0a7d53b9
Cheryls-MacBook:CS253openssl cherylsiy$ openssl dgst -sha256 lena.tiff
SHA256(lena.tiff)=
c056da23302d2fb0d946e7ffa11e0d94618224193ff6e2f78ef8097bb8a3569b
Cheryls-MacBook:CS253openssl cherylsiy$ openssl dgst -sha512 lena.tiff
SHA512(lena.tiff)=
2cb9d7df53eb8640dc48d736974f472a98d9c7186de7a972490455f5f3ed29dfc5b75c95ccb3ed45
96bc2bfc4b1e52cf4d76bcee27d334dd155bb426617392dc
```

III. Public Keys

First create a public/private key pair[5]

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl genrsa -out
key.pem 2048
Generating RSA private key, 2048 bit long modulus
...................................+++
.............................+++
e is 65537 (0x010001)
```

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl rsa -in key.pem -
text -noout
RSA Private-Key: (2048 bit)
```

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl rsa -in key.pem -
pubout -out pub.pem
writing RSA key
Cheryls-MacBook:CS253openssl cherylsiy$ openssl rsa -in pub.pem -
pubin -text -noout
RSA Public-Key: (2048 bit)
```

Then encrypt the *.tiff file.

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl rsautl -encrypt -
inkey pub.pem -pubin -in lena.tiff -out lenarsa.tiff
RSA operation error
140736890479552:error:0406D06E:rsa
routines:RSA_padding_add_PKCS1_type_2:data too large for key
size:crypto/rsa/rsa_pk1.c:125:
```

However, an error occurred(data too large for key size). Smime[6] was used to encrypt the file.

First create a private key.

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl req -newkey rsa:
2048 -keyout privkey.pem -out req.pem
Cheryls-MacBook:CS253openssl cherylsiy$ openssl x509 -req -in
req.pem -signkey privkey.pem -out cert.pem
```

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl smime -encrypt -
aes256 -in lena.tiff -binary -outform DER -out enc.tiff cert.pem
```

To decrypt the file:

```
openssl smime -decrypt -in enc.tiff -inform DER -inkey privkey.pem
-out lena2.tiff
```

Elliptic curve:

First step is to check the different elliptic curve parameters. [7]

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl ecparam -
list_curves
  secp112r1 : SECG/WTLS curve over a 112 bit prime field
  secp112r2 : SECG curve over a 112 bit prime field
  secp128r1 : SECG curve over a 128 bit prime field
  secp128r2 : SECG curve over a 128 bit prime field
  secp160k1 : SECG curve over a 160 bit prime field
  secp160r1 : SECG curve over a 160 bit prime field
  secp160r2 : SECG/WTLS curve over a 160 bit prime field
  secp192k1 : SECG curve over a 192 bit prime field
  secp224k1 : SECG curve over a 224 bit prime field
  secp224r1 : NIST/SECG curve over a 224 bit prime field
  secp256k1 : SECG curve over a 256 bit prime field
  secp384r1 : NIST/SECG curve over a 384 bit prime field
  secp521r1 : NIST/SECG curve over a 521 bit prime field
  prime192v1: NIST/X9.62/SECG curve over a 192 bit prime field
  prime192v2: X9.62 curve over a 192 bit prime field
  prime192v3: X9.62 curve over a 192 bit prime field
  prime239v1: X9.62 curve over a 239 bit prime field
  prime239v2: X9.62 curve over a 239 bit prime field
  prime239v3: X9.62 curve over a 239 bit prime field
  prime256v1: X9.62/SECG curve over a 256 bit prime field
  sect113r1 : SECG curve over a 113 bit binary field
  sect113r2 : SECG curve over a 113 bit binary field
  sect131r1 : SECG/WTLS curve over a 131 bit binary field
  sect131r2 : SECG curve over a 131 bit binary field
  sect163k1 : NIST/SECG/WTLS curve over a 163 bit binary field
  sect163r1 : SECG curve over a 163 bit binary field
  sect163r2 : NIST/SECG curve over a 163 bit binary field
  sect193r1 : SECG curve over a 193 bit binary field
  sect193r2 : SECG curve over a 193 bit binary field
  sect233k1 : NIST/SECG/WTLS curve over a 233 bit binary field
  sect233r1 : NIST/SECG/WTLS curve over a 233 bit binary field
  sect239k1 : SECG curve over a 239 bit binary field
  sect283k1 : NIST/SECG curve over a 283 bit binary field
  sect283r1 : NIST/SECG curve over a 283 bit binary field
  sect409k1 : NIST/SECG curve over a 409 bit binary field
  sect409r1 : NIST/SECG curve over a 409 bit binary field
  sect571k1 : NIST/SECG curve over a 571 bit binary field
  sect571r1 : NIST/SECG curve over a 571 bit binary fie
```

For this example, I chose ***ecparam secp 256k1***. From here, a private and public key is generated. [7]

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl ecparam -name secp256k1 -genkey -noout -out secp256k1-key.pem
Cheryls-MacBook:CS253openssl cherylsiy$ openssl ec -in secp256k1-key.pem -pubout -out secp256k1-pub.pem
read EC key
writing EC key
```

Next is to sign and verify the file [8]

```
Cheryls-MacBook:CS253openssl cherylsiy$ openssl dgst -sha1 -sign secp256k1-key.pem lena.tiff > signature.bin
Cheryls-MacBook:CS253openssl cherylsiy$ openssl dgst -sha1 -verify secp256k1-pub.pem -signature signature.bin lena.tiff
Verified OK
```

Sources:

[1] https://notsoprogrammer.wordpress.com/2013/10/18/encrypting-pictures-aes-ecb-and-cbc/
[2] https://wiki.openssl.org/index.php/Manual:Enc(1)
[3] http://man7.org/linux/man-pages/man1/dd.1.html
[4] https://www.openssl.org/docs/man1.0.2/apps/dgst.html
[5]http://stackoverflow.com/questions/29010967/openssl-unable-to-load-public-key
[6]http://stackoverflow.com/questions/18924715/encrypt-a-big-file-using-openssl-smime
[7]https://wiki.openssl.org/index.php/Command_Line_Elliptic_Curve_Operations
[8]https://superuser.com/questions/737574/openssl-ecdsa-sign-and-verify-file