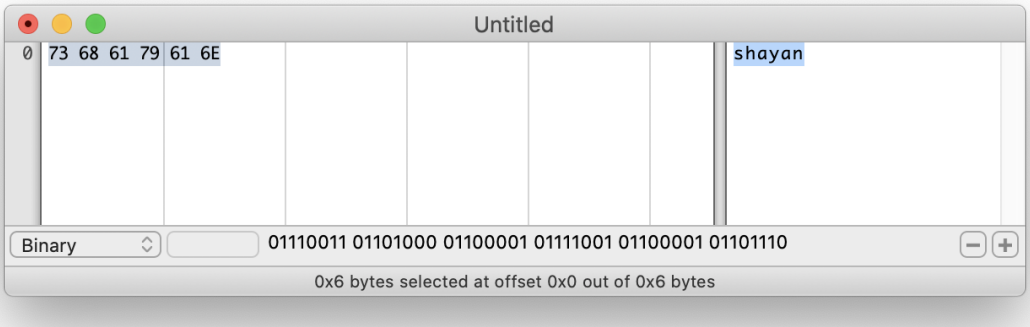
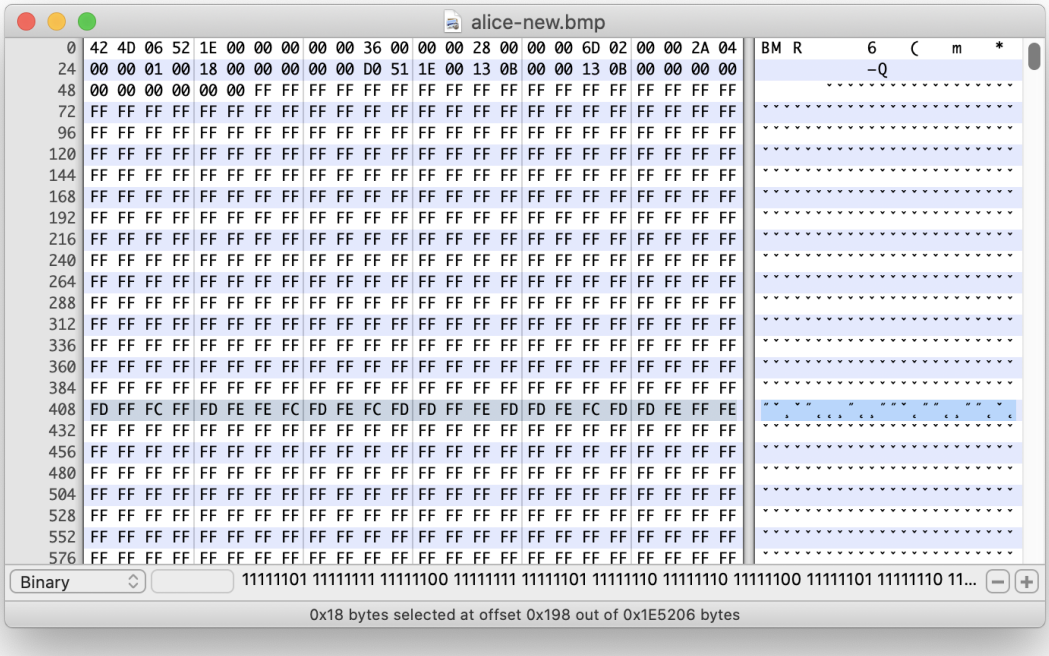
CS 827: Lab 1

Due on Wednesday, March 11, 2020

Shayan Amani (sa1149)

1., **2.**, and **3.** I have downloaded and changed the hexadecimal information in the provided image file. There is no human-eye-noticable difference between the tampered picture and the original one:





4. By comparison, the digest for the steganography-manipulated file is different than the one for the original file. The avalanche effect is obvious as by slight change in the original file, the digest is changing in high degree:

```
[ shi-on@MacBook-Pro7 /Volumes/DevCamp/CLionProjects/ComputerSecurity/lab1/res (master) $ openssl sha1 alice.bmp
SHA1(alice.bmp)= 315fb91db33c19491e28f9866af89e2737091f46
[ shi-on@MacBook-Pro7 /Volumes/DevCamp/CLionProjects/ComputerSecurity/lab1/res (master) $ openssl sha256 alice.bmp
SHA256(alice.bmp)= eca132e831160e2d461c1978baa6b7a45371d233020ca4b2b42073f3e2c40fb8
[ shi-on@MacBook-Pro7 /Volumes/DevCamp/CLionProjects/ComputerSecurity/lab1/res (master) $ openssl sha1 alice-new.bmp
SHA1(alice-new.bmp)= d38462d6dabe3e8a8ad4eb2fecc23d4485d8c1d7
[ shi-on@MacBook-Pro7 /Volumes/DevCamp/CLionProjects/ComputerSecurity/lab1/res (master) $ openssl sha256 alice-new.bmp
SHA256(alice-new.bmp)= 01afbc53e3dc14c3e889e3a539483fd7d9e12e6063c128bf2d8ded864826804a
```

5. I have genertated a private and a public key:

```
$ openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048
-out private-key.pem

$ openssl pkey -in private-key.pem -out public-key.pem -pubout
```

6. Using the generated private key, I signed the digest of the new image file (`alice-new.bmp.sha256`):

```
$ openssl dgst -sha256 -sign private-key.pem -out
alice-new.bmp.sha256 alice-new.bmp
```

7. 8. The implementation of TEA algorithm in two modes, ECB and CBC has been provided with the uploaded files. The code implementation is compatible with Java 9 and above. The input file should be placed in the `res` directory and you can run the corresponding Java classes out of `ECBEncrypt.java, ECBDecrypt.java, CBCEncrypt.java, or CBCDecrypt.java` with the following program input arguments:

1. Encryption:

   `<KEY> <PLAINTEXT> <CIPHERTEXT>`

   example:

   `shayanam alice-new.bmp cbc-alice-enc.bmp`

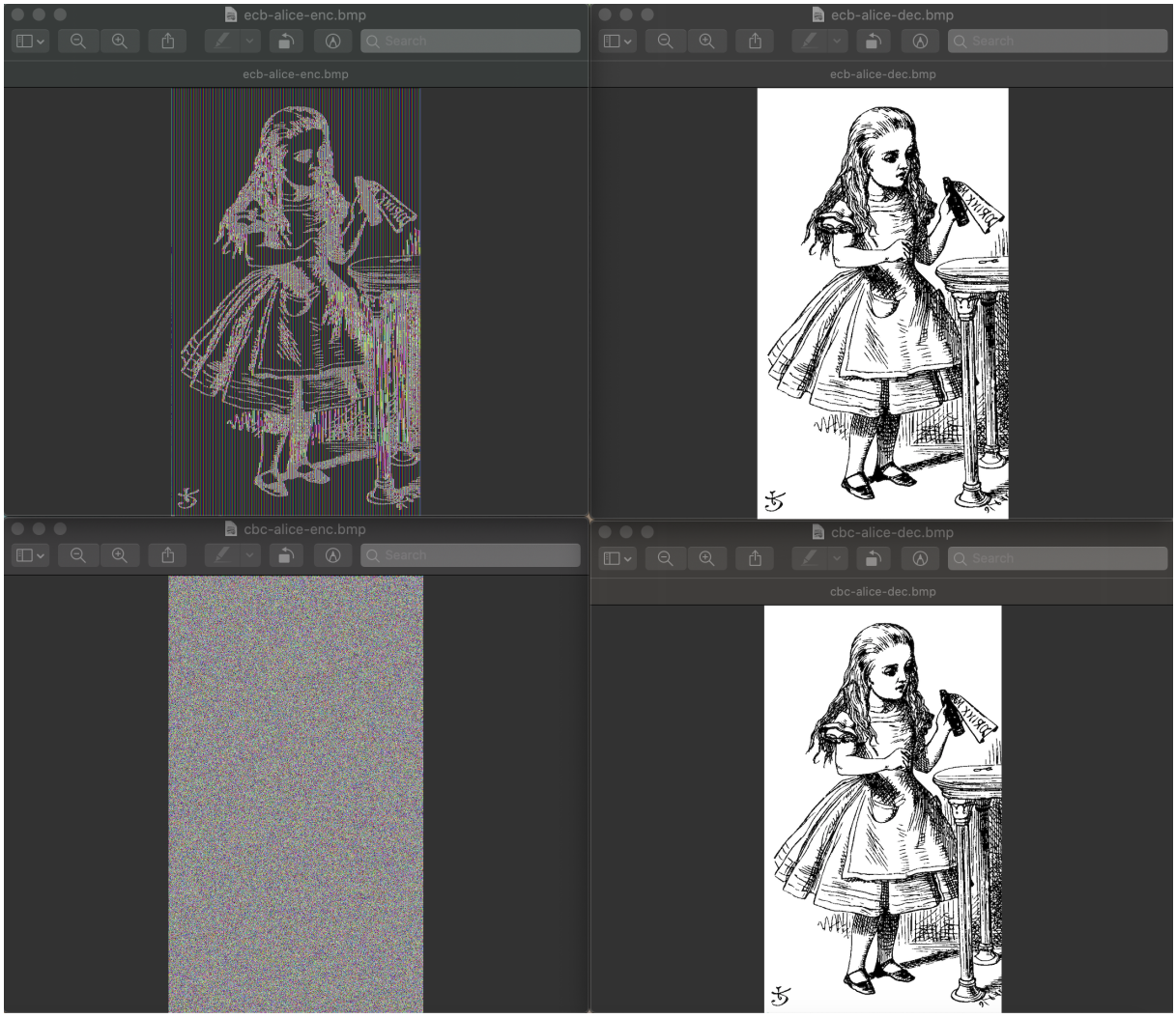
2. Decryption:

   `<KEY> <CIPHERTEXT> <PLAINTEXT>`

   example:

   `shayanam cbc-alice-enc.bmp cbc-alice-dec.bmp`

The utilized key is a 8 character-long string.
Note: The key used in enc/dec in my submission is my first and last name: **`shayanam`**



Note: I left the header readable to figure out the encrypted version and decrypted version visually.