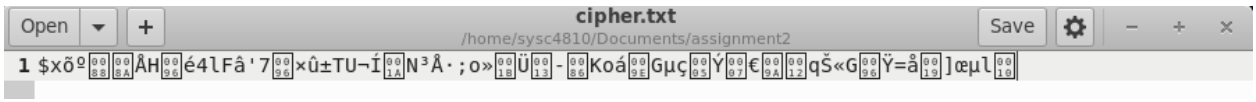
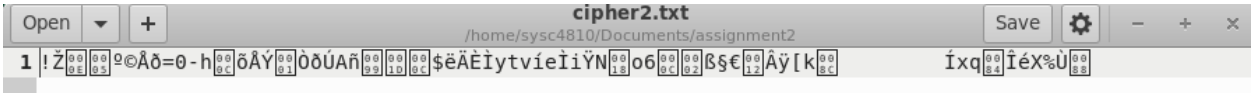
1. A) My plaintext file is plain.txt and it contains the contents “This is a secret message intended only for my closest friend.”
   B) Key: 00112233445566778899AABBCCDDEEFF, Initialization Vector: 01020304050607080102030405060708

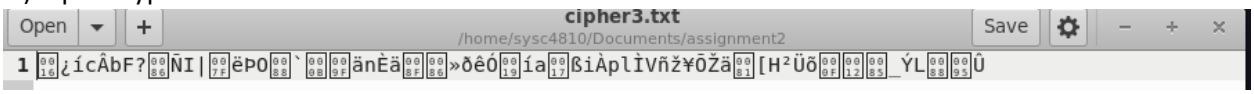
   i) Cipher type: -aes-128-cbc



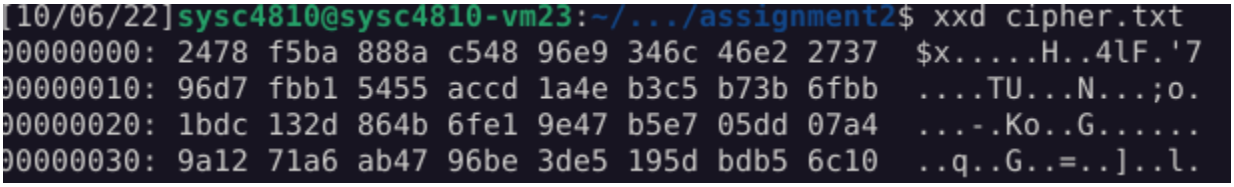
   ii) Cipher type: -bf-cbc



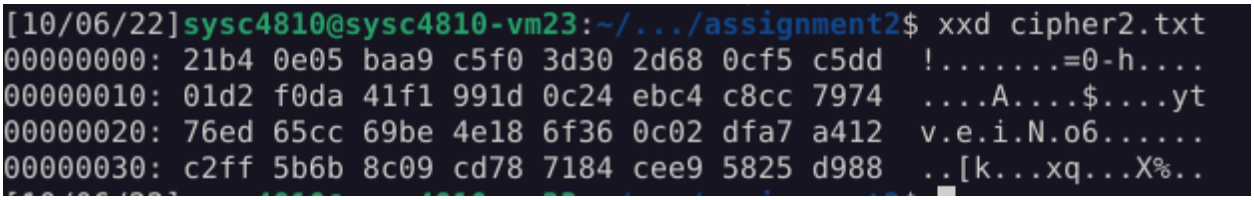
   iii) Cipher type: -aes-128-cfb



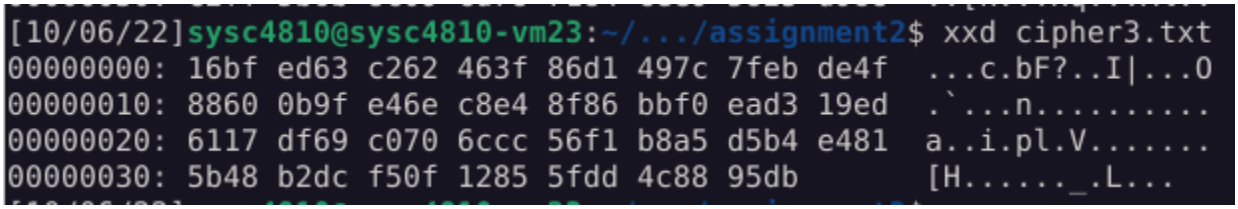
   C) i) Cipher type: -aes-128-cbc



   ii) Cipher type: -bf-cbc



   iii) Cipher type: -aes-128-cfb



   Based on these outputs, the plaintext has been encrypted in all 3 forms.

2. A)

```
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ openssl enc -aes-128-ecb -e -in plate1.bmp -out
cipher4.bmp -K 00112233445566778899AABBCCDDEEFF -iv 01020304050607080102030405060708
```

```
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ openssl enc -aes-128-cbc -e -in plate1.bmp -out
cipher5.bmp -K 00112233445566778899AABBCCDDEEFF -iv 01020304050607080102030405060708
```

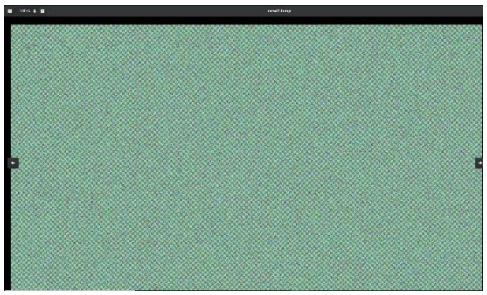
B) **ECB**

```
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ head -c 54 plate1.bmp > header
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ tail -c +55 cipher4.bmp > body
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ cat header body > new.bmp
```

CBC

```
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ head -c 54 plate1.bmp > header
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ tail -c +55 cipher5.bmp > body
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ cat header body > new2.bmp
```

c) **ECB** **CBC**



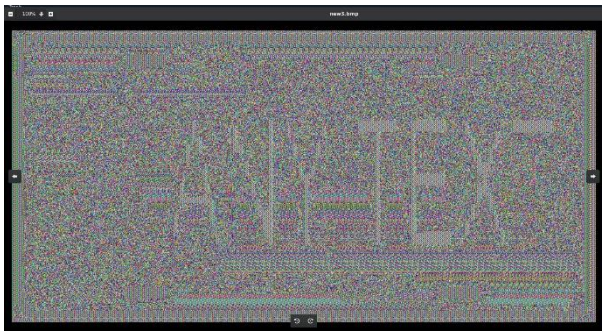
We can see the license plate information from the ECB encryption but nothing from the CBC encryption.

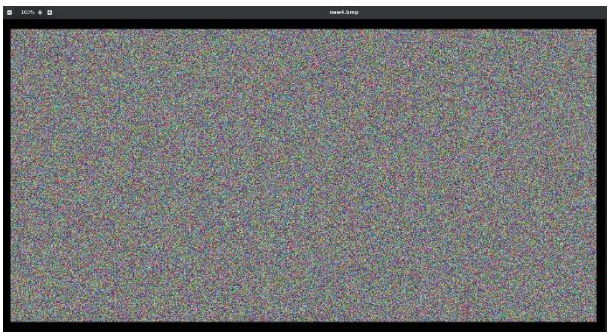
d)

```
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ openssl enc -aes-128-ecb -e -in plate2.bmp -out
cipher6.bmp -K 00112233445566778899AABBCCDDEEFF -iv 01020304050607080102030405060708
warning: iv not used by this cipher
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ openssl enc -aes-128-cbc -e -in plate2.bmp -out
cipher7.bmp -K 00112233445566778899AABBCCDDEEFF -iv 01020304050607080102030405060708
```

```
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ head -c 54 plate2.bmp > header
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ tail -c +55 cipher6.bmp > body
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ cat header body > new3.bmp
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ head -c 54 plate2.bmp > header
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ tail -c +55 cipher7.bmp > body
[10/06/22]sysc4810@sysc4810-vm23:~/.../assignment2$ cat header body > new4.bmp
```

ECB **CBC**





Results are similar to plate 1, however they are even less clear or leave less trace of the original. ECB still shows some information while CBC shows none. Colour has also been removed.

3. Our d = 1FD7DE2CBA5826EF69A76D29000E2BE5CFC0479623A758A008462A5EFED5CB19 with the values given. We have used 128 bits here for p and q for simplicity's sake where as in practice they are at least 512 bits. Refer to file q3.c.

```
n = a*b =  4B54D1DD84ACC560ED99562D4676E58CB09538FC1B981CCD43C3E659D11B14C1
totient of n =  4B54D1DD84ACC560ED99562D4676E58B8196EA045031B2FA85259F5FA5F12FD4
d*e mod tn where d =  1FD7DE2CBA5826EF69A76D29000E2BE5CFC0479623A758A008462A5EFED5CB19
Prvaite Key = ( 1FD7DE2CBA5826EF69A76D29000E2BE5CFC0479623A758A008462A5EFED5CB19,  4B54D1DD84ACC560ED99562D4676E58CB09538FC1B981CCD43C3E659D11B14C1)
```

4. ASCII string of message to hexadecimal is: 506c6174653a204c53524520383435202d3e204661696c65642053746f70.

```
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ python3 -c 'print("Plate: LSRE 845 -> Failed Stop".encode().hex())'
506c6174653a204c53524520383435202d3e204661696c65642053746f70
```

C is the encrypted value, the ciphertext. Original M is what we got from our python code above. Res is the M plaintext value we vallate with the private key decrypting the ciphertext verifying that it was encrypted properly. Refer to q4.c.

```
C = M^e mod n =  725B1BC90197737A4DCC09FA1E18912E70440C67723B6EB7BDD1FB2AEDEB2A
Orignial M is  506C6174653A204C53524520383435202D3E204661696C65642053746F70
res = C^d mod n =  506C6174653A204C53524520383435202D3E204661696C65642053746F70
```

5. Refer to q5.c.

```
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ ./a.out
M = C^d mod n =  506C6174653A20504845592037323920 2D3E205370656564696E67
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ python3 -c 'print(bytes.fromhex("506C6174653A20504845592037323920 2D3E205370656564696E67").decode("ASCII"))'
Plate: PHEY 729 -> Speeding
```

6. A) Refer to q6.c.

```
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ python3 -c 'print("Plate: YYXR 481 -> Illegal Turn".encode().hex())'
506c6174653a205959585220343831202d3e20496c6c6567616c205475726e
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ ./a.out
S = M^d mod n =  870E8C1C96F5340AF41D2919E7B60C5FF791F584EFCDD6377FFFB82C365B45
```

B) Changing even a single characher produces a completely different signature.

```
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ python3 -c 'print("Plate: YYXR 482 -> Illegal Turn".encode().hex())'
506c6174653a205959585220343832202d3e20496c6c6567616c205475726e
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ ./a.out
S = M^d mod n =  87B921572D5065E174B2E59F37ECF18F829DC331587A01BD40329F839E4547
```

7. a) The messages are the same. Refer to q7.c.

```
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ python3 -c 'print("Plate: AUHW 090 -> Clear Record".encode().hex())'
506c6174653a204155485720303930202d3e20436c656172205265636f7264
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ a.out
M = S^e mod n =  506C6174653A204155485720303930202D3E20436C656172205265636F7264
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ python3 -c 'print(bytes.fromhex("506C6174653A204155485720303930202D3E20436C656172205265636F7264").decode("ASCII"))'
Plate: AUHW 090 -> Clear Record
```

b) With the corrupted signature, we get an error and when put into an online translator the message is corrupted.

```
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ ./a.out
M = S^e mod n =  0579BC46BDB85885901F51A8B905D0ADBE4ACEBFA455E7C98F964B248B18B156
[10/15/22]sysc4810@sysc4810-vm23:~/.../assignment2$ python3 -c 'print(bytes.fromhex("0579BC46BDB85885901F51A8B905D0ADBE4ACEBFA455E7C98F964B248B18B156").decode("ASCII"))'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
UnicodeDecodeError: 'ascii' codec can't decode byte 0xbc in position 2: ordinal not in range(128)
```

□y¼F½¸X□□□Q¨¹□Ð¾JÎ¿¤UçÉ□□K$□□±V

8. The AES-512-CBC ciphertype should be used rather than ECB to encrypt all the license plates. ECB leaves the least amount of infromation when being encrypted allowing for safer more secure encrypted messages, as seen in question 2d. The information of the licence plate can still be seen in the ECB encyption. Additionally a key length of 512 should be used to provide the adequate amount of security without compromising too much performance.

   The use of digital signatures is encouraged as it will ensure that license plates are coming from a trusted roadside unit and that the contents of the message have not been altered. However, is there is even the slightest change of a single charcter will cause a corrupted licene plate to be sent back to the company. Therefore it is vital that the signature is not altered in any way. This can be seen from the reults in question 7.