Tampereen yliopisto
Tampere University

COMP.SEC.110-2021-2022-1 Cyber Security II

Secret-Key Encryption

Authors / 21.3.2022, Rev.0:

Rauli Virtanen, rauli.virtanen@tuni.fi

Henri Dyster, henri.dyster@tuni.fi

Table of Contents

1. Introduction

The learning objective of this lab was to get familiar with the concepts in the secret-key encryption and some common attacks on encryption. From this lab, we got a first-hand experience on encryption algorithms, encryption modes, paddings, and initial vector (IV). Moreover, we tried to use tools and write programs to encrypt/decrypt messages.

Many common mistakes have been made by developers in using the encryption algorithms and modes. These mistakes weaken the strength of the encryption, and eventually lead to vulnerabilities [1].

This lab covers the following topics:

• Secret-key encryption

• Substitution cipher and frequency analysis

• Encryption modes, IV, and paddings

• Common mistakes in using encryption algorithms

• Programming using the crypto library [1]

2. Task 1: Frequency Analysis

It is well-known that monoalphabetic substitution cipher is not secure, because it can be subjected to frequency analysis. In this task we have used a ciphertext which was included in Labsetup / Files folder. The ciphertext was encrypted using a monoalphabetic cipher; namely, each letter in the original text is replaced by another letter, where the replacement does not vary (i.e., a letter is always replaced by the same letter during the encryption). Our job was to use the frequency analysis to figure out the encryption key and the original plaintext. It is known that the original text is an English article [1].

Frequency information given in the assignment is presented in the following picture [1].

```
$ ./freq.py
----------------------------------
1-gram (top 20):
n: 488
y: 373
v: 348
...
----------------------------------
2-gram (top 20):
yt: 115
tn: 89
mu: 74
...
----------------------------------
3-gram (top 20):
ytn: 78
vup: 30
mur: 20
...
```

Some test cases during the Task 1:

vup = and (lots of single v:s as article a)

ytn = the (frequent)

vii = all (a + two same)

mur = ing (frequent, in the end)

nggmur ebbing (rest just checking words)

cxfncnuy = ?ove?ent -> movement

dnvhq = fears

mq -> is

vavpncd n?nd???

lmu win = win

a = c

b = f

c = m

d = f

e = p

f = v

g= b

h=r

i = l

j=

xb = of

lmii = will

pmfmpnp gnylnnu divided between>

xqavhfxymur = oscarvoting%n = e

emayzhn = picture

qmuan = since/p = d

k=

l = w

m = i

o =

enxein = ?eo?le -> people

uxcmuvymxuq = no?ia?ions ->

lvq = was

qmrumbmnp = signified

bxh = for

cmuzynq ?in?tesv = a

bxhnavqynhq = forecasters

gmiigxvhpq = ?ill?oa??su = n

pzhmur = ??ring

yx = to

mu = in

lmuq ltnu ux = wins w??n no

gmr = big

vq = as

vy = at

ynh = ter

yxx = ?t=h

mu = in

xb = ? of

tn -> no?

yt?? he?x = o

fxh = voy = t

q=s

r = g

s=

One example from tried tr command:

```
[03/21/22]seed@VM:~/.../Files$ tr 'k,l,i,a,q,f,m,p,c,b,x,t,y,n,v,u,e,h,r,z,g,d,s,j,w,o' 'X,W,L,C,
S,V,I,D,M,F,O,H,T,E,A,N,P,R,G,U,B,Y,K,Q,Z,J' < ciphertext.txt > out.txt
[03/21/22]seed@VM:~/.../Files$
```

The plaintext which has been generated from ciphertext by using the above presented tr command:

THE OSCARS TURN ON SUNDAY WHICH SEEMS ABOUT RIGHT AFTER THIS LONG STRANGE AWARDS TRIP THE BAGGER FEELS LIKE A NONAGENARIAN TOO THE AWARDS RACE WAS BOOKENDED BY THE DEMISE OF HARVEY WEINSTEIN AT ITS OUTSET AND THE APPARENT IMPLOSION OF HIS FILM COMPANY AT THE END AND IT WAS SHAPED BY THE EMERGENCE OF METOO TIMES UP BLACK-GOWN POLITICS ARMCANDY ACTIVISM AND A NATIONAL CONVERSATION AS BRIEF AND MAD AS A FEVER DREAM ABOUT WHETHER THERE OUGHT TO BE A PRESIDENT WINFREY THE SEASON DIDNT JUST SEEM EXTRA LONG IT WAS EX-TRA LONG BECAUSE THE OSCARS WERE MOVED TO THE FIRST WEEKEND IN MARCH TO AVOID CONFLICTING WITH THE CLOSING CEREMONY OF THE WIN-TER OLYMPICS THANKS PYEONGCHANG ONE BIG QUESTION SURROUNDING THIS YEARS ACADEMY AWARDS IS HOW OR IF THE CEREMONY WILL ADDRESS METOO ESPECIALLY AFTER THE GOLDEN GLOBES WHICH BECAME A JUBILANT COMINGOUT PARTY FOR TIMES UP THE MOVEMENT SPEARHEADED BY POWER-FUL HOLLYWOOD WOMEN WHO HELPED RAISE MILLIONS OF DOLLARS TO FIGHT SEXUAL HARASSMENT AROUND THE COUNTRY SIGNALING THEIR SUPPORT GOLDEN GLOBES ATTENDEES SWATHED THEMSELVES IN BLACK SPORTED LA-PEL PINS AND SOUNDED OFF ABOUT SEXIST POWER IMBALANCES FROM THE RED CARPET AND THE STAGE ON THE AIR E WAS CALLED OUT ABOUT PAY IN-EQUITY AFTER ITS FORMER ANCHOR CATT SADLER QUIT ONCE SHE LEARNED THAT SHE WAS MAKING FAR LESS THAN A MALE COHOST AND DURING THE CEREMONY NATALIE PORTMAN TOOK A BLUNT AND SATISFYING DIG AT THE ALLMALE ROSTER OF NOMINATED DIRECTORS HOW COULD THAT BE TOPPED AS IT TURNS OUT AT LEAST IN TERMS OF THE OSCARS IT PROBABLY WONT BE WOMEN INVOLVED IN TIMES UP SAID THAT ALTHOUGH THE GLOBES SIGNIFIED THE INITIATIVES LAUNCH THEY NEVER INTENDED IT TO BE JUST AN AWARDS SEASON CAMPAIGN OR ONE THAT BECAME ASSOCIATED ONLY WITH RED-CARPET ACTIONS INSTEAD A SPOKESWOMAN SAID THE GROUP IS WORKING BEHIND CLOSED DOORS AND HAS SINCE AMASSED MILLION FOR ITS LEGAL DEFENSE FUND WHICH AFTER THE GLOBES WAS FLOODED WITH THOUSANDS OF DONATIONS OF OR LESS FROM PEOPLE IN SOME COUNTRIES NO CALL TO WEAR BLACK GOWNS WENT OUT IN ADVANCE OF THE OSCARS

THOUGH THE MOVEMENT WILL ALMOST CERTAINLY BE REFERENCED BEFORE AND DURING THE CEREMONY ESPECIALLY SINCE VOCAL METOO SUPPORTERS LIKE ASHLEY JUDD LAURA DERN AND NICOLE KIDMAN ARE SCHEDULED PRE-SENTERS ANOTHER FEATURE OF THIS SEASON NO ONE REALLY KNOWS WHO

IS GOING TO WIN BEST PICTURE ARGUABLY THIS HAPPENS A LOT OF THE TIME INARGUABLY THE NAILBITER NARRATIVE ONLY SERVES THE AWARDS HYPE MACHINE BUT OFTEN THE PEOPLE FORECASTING THE RACE SOCALLED OSCAROLOGISTS CAN MAKE ONLY EDUCATED GUESSES THE WAY THE ACADEMY TABULATES THE BIG WINNER DOESNT HELP IN EVERY OTHER CATEGORY THE NOMINEE WITH THE MOST VOTES WINS BUT IN THE BEST PICTURE CATEGORY VOTERS ARE ASKED TO LIST THEIR TOP MOVIES IN PREFERENTIAL ORDER IF A MOVIE GETS MORE THAN PERCENT OF THE FIRSTPLACE VOTES IT WINS WHEN NO MOVIE MANAGES THAT THE ONE WITH THE FEWEST FIRSTPLACE VOTES IS ELIMINATED AND ITS VOTES ARE REDISTRIBUTED TO THE MOVIES THAT GARNERED THE ELIMINATED BALLOTS SECONDPLACE VOTES AND THIS CONTINUES UNTIL A WINNER EMERGES IT IS ALL TERRIBLY CONFUSING BUT APPARENTLY THE CONSENSUS FAVORITE COMES OUT AHEAD IN THE END THIS MEANS THAT ENDOFSEASON AWARDS CHATTER INVARIABLY INVOLVES TORTURED SPECULATION ABOUT WHICH FILM WOULD MOST LIKELY BE VOTERS SECOND OR THIRD FAVORITE AND THEN EQUALLY TORTURED CONCLUSIONS ABOUT WHICH FILM MIGHT PREVAIL IN IT WAS A TOSSUP BETWEEN BOYHOOD AND THE EVENTUAL WINNER BIRDMAN IN WITH LOTS OF EXPERTS BETTING ON THE REVENANT OR THE BIG SHORT THE PRIZE WENT TO SPOTLIGHT LAST YEAR NEARLY ALL THE FORECASTERS DECLARED LA LA LAND THE PRESUMPTIVE WINNER AND FOR TWO AND A HALF MINUTES THEY WERE CORRECT BEFORE AN ENVELOPE SNAFU WAS REVEALED AND THE RIGHTFUL WINNER MOONLIGHT WAS CROWNED THIS YEAR AWARDS WATCHERS ARE UNEQUALLY DIVIDED BETWEEN THREE BILLBOARDS OUTSIDE EBBING MISSOURI THE FAVORITE AND THE SHAPE OF WATER WHICH IS THE BAGGERS PREDICTION WITH A FEW FORECASTING A HAIL MARY WIN FOR GET OUT BUT ALL OF THOSE FILMS HAVE HISTORICAL OSCARVOTING PATTERNS AGAINST THEM THE SHAPE OF WATER HAS NOMINATIONS MORE THAN ANY OTHER FILM AND WAS ALSO NAMED THE YEARS BEST BY THE PRODUCERS AND DIRECTORS GUILDS YET IT WAS NOT NOMINATED FOR A SCREEN ACTORS GUILD AWARD FOR BEST ENSEMBLE AND NO FILM HAS WON BEST PICTURE WITHOUT PREVIOUSLY LANDING AT LEAST THE ACTORS NOMINATION SINCE BRAVEHEART IN THIS YEAR THE BEST ENSEMBLE SAG ENDED UP GOING TO THREE BILLBOARDS WHICH IS SIGNIFICANT BECAUSE ACTORS MAKE UP THE ACADEMYS LARGEST BRANCH THAT FILM WHILE DIVISIVE ALSO WON THE BEST DRAMA GOLDEN GLOBE AND THE BAFTA BUT ITS FILMMAKER MARTIN MCDONAGH WAS NOT NOMINATED FOR BEST DIRECTOR AND APART FROM ARGO MOVIES THAT LAND BEST PICTURE WITHOUT ALSO EARNING BEST DIRECTOR NOMINATIONS ARE FEW AND FAR BETWEEN

An original article which has been found on Internet [2]:

“New York Times

By Cara Buckley

March 1, 2018

THE CARPETBAGGER

What to Expect (and Not Expect) at the Oscars

The Oscars turn 90 on Sunday, which seems about right. After this long, strange awards trip, the Bagger feels like a nonagenarian, too.

The awards race was bookended by the demise of Harvey Weinstein at its outset and the apparent implosion of his film company at the end. And it was shaped by the emergence of #MeToo, Time’s Up, black-gown politics, arm-candy activism and a national conversation, as brief and mad as a fever dream, about whether there ought to be a President Winfrey. The season didn’t just seem extra long, it was extra long, because the Oscars were moved to the first weekend in March to avoid conflicting with the closing ceremony of the Winter Olympics. Thanks, Pyeongchang.

One big question surrounding this year’s Academy Awards is how or if the ceremony will address #MeToo, especially after the Golden Globes, which became a jubilant coming-out party for Time’s Up, the movement spearheaded by 300 powerful Hollywood women who helped raise millions of dollars to fight sexual harassment around the country.

Signaling their support, Golden Globes attendees swathed themselves in black, sported lapel pins and sounded off about sexist power imbalances from the red carpet and the stage. On the air, E! was called out about pay inequity after its former anchor Catt Sadler quit once she learned that she was making far less than a male co-host, and during the ceremony Natalie Portman took a blunt and satisfying dig at the all-male roster of nominated directors. How could that be topped?

As it turns out, at least in terms of the Oscars, it probably won’t be.

Women involved in Time’s Up said that although the Globes signified the initiative’s launch, they never intended it to be just an awards season campaign, or one that became associated only with red-carpet actions. Instead, a spokeswoman said, the group is working behind closed doors and has since amassed $21 million for its legal defense fund, which, after the Globes, was flooded with thousands of donations of $100 or less from people in some 80 countries.

No call to wear black gowns went out in advance of the Oscars, though the movement will almost certainly be referenced before and during the ceremony — especially since vocal #MeToo supporters like Ashley Judd, Laura Dern and Nicole Kidman are scheduled presenters.

Another feature of this season: No one really knows who is going to win best picture. Arguably, this happens a lot of the time. Inarguably, the nail-biter narrative only serves the awards hype machine. But often the people forecasting the race, so-called Oscarologists, can make only educated guesses.

The way the academy tabulates the big winner doesn't help. In every other category, the nominee with the most votes wins, but in the best picture category, voters are asked to list their top movies in preferential order. If a movie gets more than 50 percent of the first-place votes, it wins. When no movie manages that, the one with the fewest first-place votes is eliminated and its votes are redistributed to the movies that garnered the eliminated ballots' second-place votes, and this continues until a winner emerges.

It is all terribly confusing, but apparently the consensus favorite comes out ahead in the end. This means that end-of-season awards chatter invariably involves tortured speculation about which film would most likely be voters' second or third favorite, and then equally tortured conclusions about which film might prevail.

In 2015, it was a tossup between "Boyhood" and the eventual winner, "Birdman." In 2016, with lots of experts betting on "The Revenant" or "The Big Short," the prize went to "Spotlight." Last year, nearly all the forecasters declared "La La Land" the presumptive winner, and for two and a half minutes, they were correct, before an envelope snafu was revealed and the rightful winner, "Moonlight," was crowned.

This year, awards watchers are unequally divided between "Three Billboards Outside Ebbing, Missouri," the favorite, and, "The Shape of Water" (which is the Bagger's prediction), with a few forecasting a Hail Mary win for "Get Out."

But all of those films have historical Oscar-voting patterns against them. "The Shape of Water" has 13 nominations, more than any other film, and was also named the year's best by the producers' and directors' guilds. Yet it was not nominated for a Screen Actors Guild Award for best ensemble; and no film has won best picture without previously landing at least the actors' nomination since "Braveheart," in 1996. This year, the best ensemble SAG ended up going to "Three Billboards," which is significant because actors make up the academy's largest branch; that film, while divisive, also won the best drama Golden Globe and the Bafta. But its filmmaker, Martin McDonagh, was not nominated for best director, and apart from "Argo," movies that land best picture without also earning best director nominations are few and far between. [2]"

3. Task 2: Encryption using Different Ciphers and Modes

In this task, we played with a few encryption algorithms and modes. A target was to use at least 3 different ciphers. In this task we tried the following ciphers and commands [1]:

We used Task 1's plaintext file => plain.txt

Cipher type -aes-128-cbc / encrypt command:

openssl enc -e -aes-128-cbc -in plain.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

Result:

Padded with 0 -bytes.

Cat is nonsense as a bin gives hint as binary.

Cipher type -aes-128-cbc / decrypt command:

openssl enc -d -aes-128-cbc -in cipher.bin -out plain_dc.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708

Result:

Successfully decrypted.

Cipher type -des3 / encrypt command:

openssl enc -e -des3 -in plain.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

Cipher type -des3 / decrypt command:

openssl enc -d -des3 -in cipher.bin -out plain_de_des3.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708

Result:

diff -yt plain.txt plain_de_des3.txt: No differences.

Cipher type -base64 / encrypt command:

openssl enc -e -base64 -in plain.txt -out base_out.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708

Cipher type -base64 / decrypt command:

openssl enc -d -base64 -in base_out.txt -out base_de.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708

Result:

Success.

Double en/de with 1 digit code:

openssl enc -e -base64 -in plain.txt -out base_out1.txt -K 1 -iv 0

openssl enc -e -base64 -in base_out1.txt -out base_out2.txt -K 1 -iv 0

openssl enc -d -base64 -in base_out2.txt -out base_out1.txt -K 1 -iv 0

openssl enc -d -base64 -in base_out1.txt -out plain.txt -K 1 -iv 0

Result:

Success.

Double layer de/encoding using different iv for each layer and different ciphers:

openssl enc -e -base64 -in plain.txt -out base_aes1.txt -K 1 -iv 0

openssl enc -e -aes-256-cbc -in base_aes1.txt -out base_aes2.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708

openssl enc -d -aes-256-cbc -in base_aes2txt -out base_aes1.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708

openssl enc -d -base64 -in base_aes1.txt -out plain.txt -K 1 -iv 0

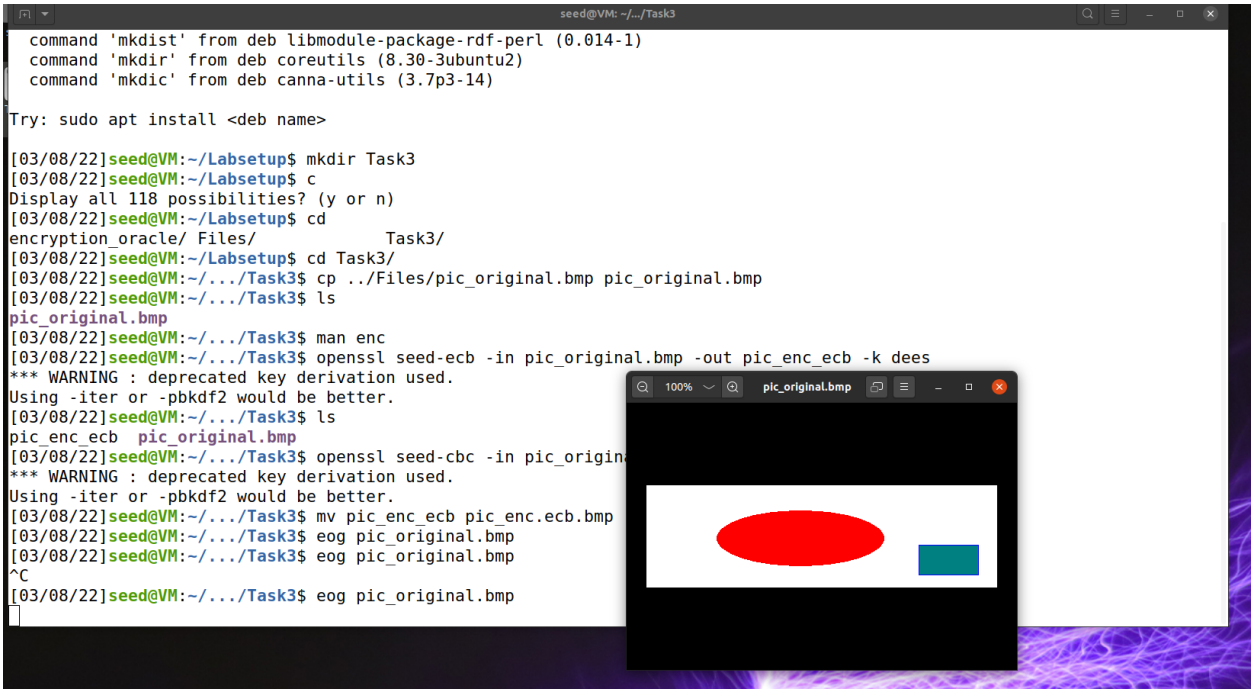
Result:

Great success!!

4. Task 3: Encryption Mode – ECB vs. CBC

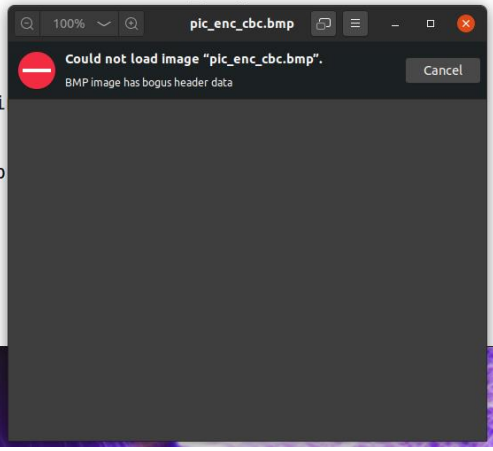
In this task we were supposed to study encryption of pictures and the differences of the results depending on the used encryption mode [1].

Making a new folder for this task, copying original picture there, encrypting it with seed-ecb and seed-cbc. Showing the original picture.



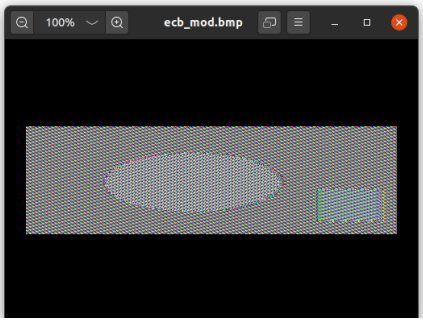
Encryption is encrypting the picture header too. That's why eog, wont be able to open it as a picture.

```
ed@VM:~/Labsetup$ c
118 possibilities? (y or n)
ed@VM:~/Labsetup$ cd
racle/ Files/             Task3/
ed@VM:~/Labsetup$ cd Task3/
ed@VM:~/.../Task3$ cp ../Files/pic_original.bmp pic_original.bmp
ed@VM:~/.../Task3$ ls
.bmp
ed@VM:~/.../Task3$ man enc
ed@VM:~/.../Task3$ openssl seed-ecb -in pic_original.bmp -out pic_enc_ecb -k dees
: deprecated key derivation used.
or -pbkdf2 would be better.
ed@VM:~/.../Task3$ ls
 pic_original.bmp
ed@VM:~/.../Task3$ openssl seed-cbc -in pic_origi
: deprecated key derivation used.
or -pbkdf2 would be better.
ed@VM:~/.../Task3$ mv pic_enc_ecb pic_enc.ecb.bmp
ed@VM:~/.../Task3$ eog pic_original.bmp
ed@VM:~/.../Task3$ eog pic_original.bmp

ed@VM:~/.../Task3$ eog pic_original.bmp

ed@VM:~/.../Task3$ eog pic_enc_cbc.bmp
```



After bringing "back" 54 first bytes from the original picture leaves encrypted part encrypted.

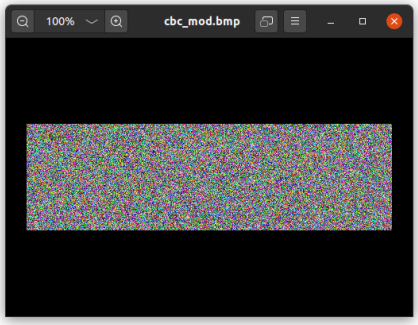
```
^C
[03/08/22]seed@VM:~/.../Task3$ eog pic_original.bmp
^C
[03/08/22]seed@VM:~/.../Task3$ eog pic_enc_cbc.bmp
[03/08/22]seed@VM:~/.../Task3$ head -c 54 pic_original.bmp > header
[03/08/22]seed@VM:~/.../Task3$ tail -c +55 pic_enc_cbc.bmp  > body
[03/08/22]seed@VM:~/.../Task3$ cat header body > cbc_mod.bmp
[03/08/22]seed@VM:~/.../Task3$ tail -c +55 pic_enc_ecb.bmp  > body
tail: cannot open 'pic_enc_ecb.bmp' for reading: No such file or directory
[03/08/22]seed@VM:~/.../Task3$ ls
body  cbc_mod.bmp  header  pic_enc_cbc.bmp  pic_enc.ecb.bmp  pic_original.bmp
[03/08/22]seed@VM:~/.../Task3$ tail -c +55 pic_enc.ecb.bmp  > body
[03/08/22]seed@VM:~/.../Task3$ cat header body > ecb_mod.bmp
[03/08/22]seed@VM:~/.../Task3$ eog ecb_mod.bmp
```



With ECB – stream without IV, one can still see the shapes, location and even colors of the objects.

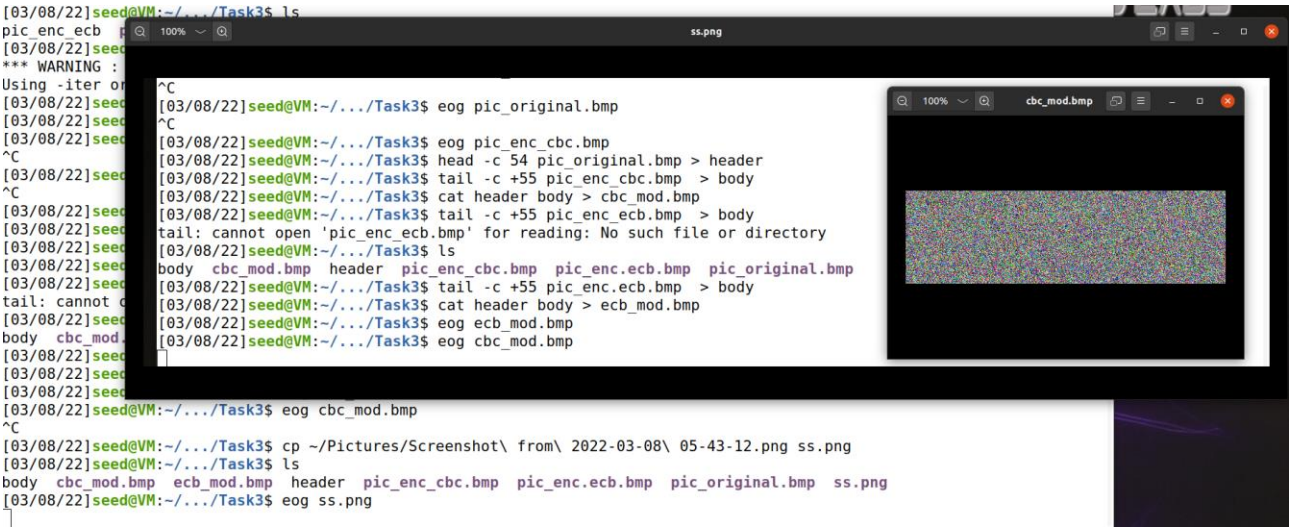
When doing the same "head swap" to the CBC -encoded picture, result looks like this:

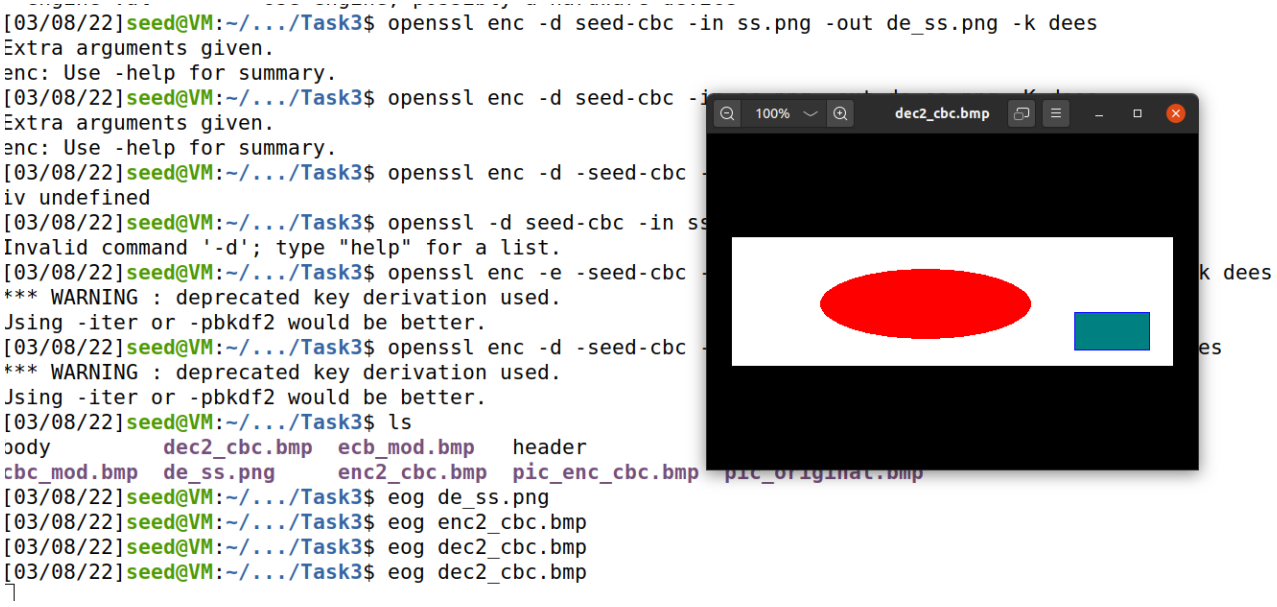
```
^C
[03/08/22]seed@VM:~/.../Task3$ eog pic_original.bmp
^C
[03/08/22]seed@VM:~/.../Task3$ eog pic_enc_cbc.bmp
[03/08/22]seed@VM:~/.../Task3$ head -c 54 pic_original.bmp > header
[03/08/22]seed@VM:~/.../Task3$ tail -c +55 pic_enc_cbc.bmp  > body
[03/08/22]seed@VM:~/.../Task3$ cat header body > cbc_mod.bmp
[03/08/22]seed@VM:~/.../Task3$ tail -c +55 pic_enc_ecb.bmp  > body
tail: cannot open 'pic_enc_ecb.bmp' for reading: No such file or directory
[03/08/22]seed@VM:~/.../Task3$ ls
body  cbc_mod.bmp  header  pic_enc_cbc.bmp  pic_enc.ecb.bmp  pic_original.bmp
[03/08/22]seed@VM:~/.../Task3$ tail -c +55 pic_enc.ecb.bmp  > body
[03/08/22]seed@VM:~/.../Task3$ cat header body > ecb_mod.bmp
[03/08/22]seed@VM:~/.../Task3$ eog ecb_mod.bmp
[03/08/22]seed@VM:~/.../Task3$ eog cbc_mod.bmp
```



Picture is completely unrecognizable. Block -cipher messes up the blocks too, when encrypting the file.

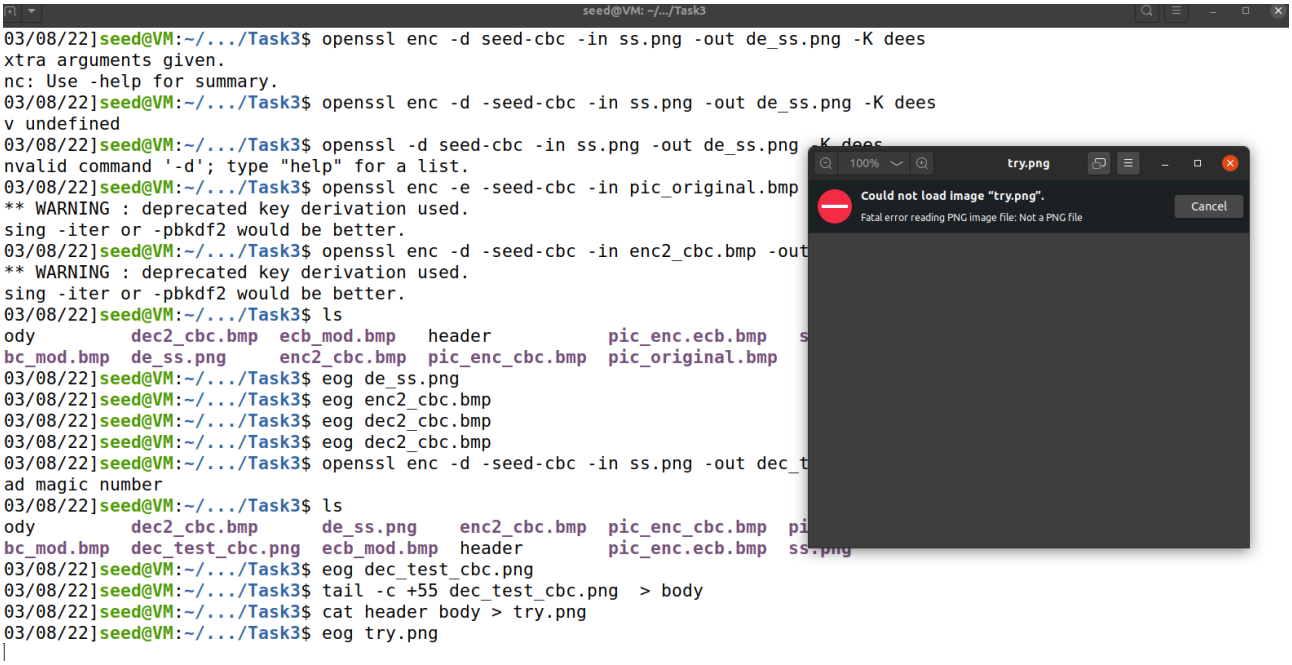
Trying different things on my own picture, screenshot from a previous task.





Encoding and decoding back successful as before.

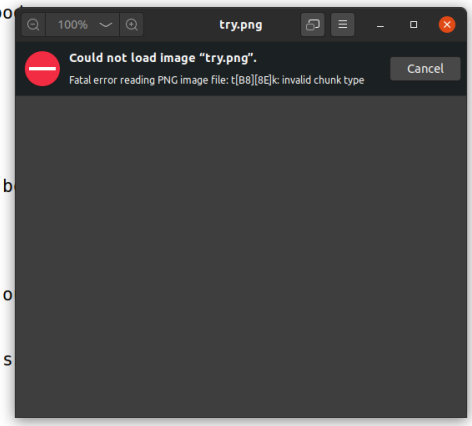
It seems that png -file header didn't work the way bmp -did.



What I tried to do here is to decrypt a clear screenshot of a encrypted picture and change the header. I was wishing to see differences in encoding pattern of one layer encoded and double encoded. No luck. It was a png -picture, maybe header isn't here 55 bytes.

Tried the same with 100 bytes header, png opens, but is blank. Maybe 100 is too much.

```
cbc_mod.bmp  dec_test_cbc.png  ecb_mod.bmp  header        pic_enc.ecb.bmp  ss.png
[03/08/22]seed@VM:~/.../Task3$ eog dec_test_cbc.png
[03/08/22]seed@VM:~/.../Task3$ tail -c +55 dec_test_cbc.png  > bo
[03/08/22]seed@VM:~/.../Task3$ cat header body > try.png
[03/08/22]seed@VM:~/.../Task3$ eog try.png
^C
[03/08/22]seed@VM:~/.../Task3$ head -c 54 ss.png  > header
[03/08/22]seed@VM:~/.../Task3$ cat header body > try.png
[03/08/22]seed@VM:~/.../Task3$ eog try.png
[03/08/22]seed@VM:~/.../Task3$ head -c 100 ss.png  > header
[03/08/22]seed@VM:~/.../Task3$ tail -c +101 dec_test_cbc.png  > b
[03/08/22]seed@VM:~/.../Task3$ cat header body > try.png
[03/08/22]seed@VM:~/.../Task3$ eog try.png
[03/08/22]seed@VM:~/.../Task3$ eog ss.png
^C
[03/08/22]seed@VM:~/.../Task3$ openssl enc seed-cbc -in ss.png -o
Extra arguments given.
enc: Use -help for summary.
[03/08/22]seed@VM:~/.../Task3$ openssl seed-cbc -in ss.png -out s
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
[03/08/22]seed@VM:~/.../Task3$ head -c 54 ss.png  > header
[03/08/22]seed@VM:~/.../Task3$ tail -c +55 ss_enc.png  > body
tail: cannot open 'ss_enc.png' for reading: No such file or directory
[03/08/22]seed@VM:~/.../Task3$ tail -c +55 ss_ec.png  > body
[03/08/22]seed@VM:~/.../Task3$ cat header body > try.png
[03/08/22]seed@VM:~/.../Task3$ eog try.png
[03/08/22]seed@VM:~/.../Task3$ eog try.png
```



Doesn’t seem to work on png. At least not in the same way it did on bmp.

5. Task 4: Padding

For block ciphers, when the size of a plaintext is not a multiple of the block size, padding may be required. The PKCS#5 padding scheme is widely used by many block ciphers. In this task, we tried to understand how this type of padding works [1].

1)

To perform encryption with a block cipher in ECB (Electronic Code Book) or CBC (Cipher Block Chaining) mode the length of the input to be encrypted must be an exact multiple of the block length B in bytes. For Triple DES the block length B is 8 bytes (64 bits) and for all AES variants it is 16 bytes (128 bits). If the length of the data to be encrypted is not an exact multiple of B, it must be padded to make it so. After decrypting, the padding needs to be removed.

For other modes of encryption, such as OFB (Output Feedback) or CFB (Cipher Feedback Mode), padding is not required. In these cases the ciphertext is always the same length as the plaintext, and a padding method is not applicable.

There are many, many conventions for padding. It is up to the sender and receiver of encrypted data to agree on the convention used. The most popular is "PKCS5" padding [3].

To make it obvious if there's a block used, we used only one byte plain text file to be encrypted.

```
[03/11/22]seed@VM:~/.../Task4$ echo t > test.txt
[03/11/22]seed@VM:~/.../Task4$ ls
test.txt
[03/11/22]seed@VM:~/.../Task4$ cat text.txt
cat: text.txt: No such file or directory
[03/11/22]seed@VM:~/.../Task4$ cat test.txt
t
[03/11/22]seed@VM:~/.../Task4$
```

ECB:

openssl enc -e -aes-256-ecb -in test.txt -out enc_test.txt -K 00112233445566778889aabbccddeeff

```
[03/11/22]seed@VM:~/.../Task4$ openssl enc -e -aes-256-ecb -in test.txt -out enc_test.txt -K  00112233445566778889aabbccddeeff
hex string is too short, padding with zero bytes to length
[03/11/22]seed@VM:~/.../Task4$ ls -la
total 16
drwxrwxr-x 2 seed seed 4096 Mar 11 01:10 .
drwxrwxr-x 6 seed seed 4096 Mar 11 01:06 ..
-rw-rw-r-- 1 seed seed   16 Mar 11 01:13 enc_test.txt
-rw-rw-r-- 1 seed seed    2 Mar 11 01:07 test.txt
```

padded -> 16 bytes (no IV)

CBC:

openssl enc -e -aes-256-cbc -in test.txt -out enc_test.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708

```
[03/11/22]seed@VM:~/.../Task4$ openssl enc -e -aes-256-cbc -in test.txt -out enc_test.txt -K  00112233445566778889aabbccddeeff -
iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/11/22]seed@VM:~/.../Task4$ ls -la
total 16
drwxrwxr-x 2 seed seed 4096 Mar 11 01:10 .
drwxrwxr-x 6 seed seed 4096 Mar 11 01:06 ..
-rw-rw-r-- 1 seed seed   16 Mar 11 01:16 enc_test.txt
-rw-rw-r-- 1 seed seed    2 Mar 11 01:07 test.txt
[03/11/22]seed@VM:~/.../Task4$
```

padded -> 16 bytes. IV needed

CFB:

openssl enc -e -aes-256-cfb -in test.txt -out enc_test.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708

```
[03/11/22]seed@VM:~/.../Task4$ openssl enc -e -aes-256-cfb -in test.txt -out enc_test.txt -K  00112233445566778889aabbccddeeff -
iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/11/22]seed@VM:~/.../Task4$ ls -la
total 16
drwxrwxr-x 2 seed seed 4096 Mar 11 01:10 .
drwxrwxr-x 6 seed seed 4096 Mar 11 01:06 ..
-rw-rw-r-- 1 seed seed    2 Mar 11 01:19 enc_test.txt
-rw-rw-r-- 1 seed seed    2 Mar 11 01:07 test.txt
[03/11/22]seed@VM:~/.../Task4$
```

padded, IV needed, but size remains the same

OFB:

openssl enc -e -aes-256-ofb -in test.txt -out enc_test.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708

```
[03/11/22]seed@VM:~/.../Task4$ openssl enc -e -aes-256-ofb -in test.txt -out enc_test.txt -K  00112233445566778889aabbccddeeff -
iv 0102030405060708
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/11/22]seed@VM:~/.../Task4$ ls -la
total 16
drwxrwxr-x 2 seed seed 4096 Mar 11 01:10 .
drwxrwxr-x 6 seed seed 4096 Mar 11 01:06 ..
-rw-rw-r-- 1 seed seed    2 Mar 11 01:21 enc_test.txt
-rw-rw-r-- 1 seed seed    2 Mar 11 01:07 test.txt
[03/11/22]seed@VM:~/.../Task4$
```

padded, IV used, size remains

Every mode says that they are using padding. However, with OFB and CFB size remained the same. OFB and CFB:s are not actually resizing/filling the encrypted file by padding, because they make block ciphers to stream ciphers.

2)

Creating 5, 10 and 16 bytes plain text files:

```
[03/11/22]seed@VM:~/.../Task4$ echo -n "12345" > f5.txt
[03/11/22]seed@VM:~/.../Task4$ echo -n "1234567890" > f10.txt
[03/11/22]seed@VM:~/.../Task4$ echo -n "1234567890123456" > f16.txt
[03/11/22]seed@VM:~/.../Task4$ ls -la
total 28
drwxrwxr-x 2 seed seed 4096 Mar 11 01:37 .
drwxrwxr-x 6 seed seed 4096 Mar 11 01:06 ..
-rw-rw-r-- 1 seed seed    2 Mar 11 01:21 enc_test.txt
-rw-rw-r-- 1 seed seed   10 Mar 11 01:37 f10.txt
-rw-rw-r-- 1 seed seed   16 Mar 11 01:37 f16.txt
-rw-rw-r-- 1 seed seed    5 Mar 11 01:37 f5.txt
-rw-rw-r-- 1 seed seed    2 Mar 11 01:07 test.txt
```

encrypting:

```
[03/11/22]seed@VM:~/.../Task4$ openssl enc -aes-128-cbc -e -in f5.txt -out enc_f5.txt
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
[03/11/22]seed@VM:~/.../Task4$ openssl enc -aes-128-cbc -e -in f10.txt -out enc_f10.txt -k dees
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
[03/11/22]seed@VM:~/.../Task4$ openssl enc -aes-128-cbc -e -in f16.txt -out enc_f16.txt -k dees
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
[03/11/22]seed@VM:~/.../Task4$ ls -la
total 40
drwxrwxr-x 2 seed seed 4096 Mar 11 01:42 .
drwxrwxr-x 6 seed seed 4096 Mar 11 01:06 ..
-rw-rw-r-- 1 seed seed   32 Mar 11 01:42 enc_f10.txt
-rw-rw-r-- 1 seed seed   48 Mar 11 01:42 enc_f16.txt
-rw-rw-r-- 1 seed seed   32 Mar 11 01:41 enc_f5.txt
-rw-rw-r-- 1 seed seed    2 Mar 11 01:21 enc_test.txt
-rw-rw-r-- 1 seed seed   10 Mar 11 01:37 f10.txt
-rw-rw-r-- 1 seed seed   16 Mar 11 01:37 f16.txt
-rw-rw-r-- 1 seed seed    5 Mar 11 01:37 f5.txt
-rw-rw-r-- 1 seed seed    2 Mar 11 01:07 test.txt
[03/11/22]seed@VM:~/   /Task4$
```

Size increases 5 and 10 bytes -> 32 bytes and 16 bytes -> 48 bytes

Decoding:

openssl enc -aes-128-cbc -d -nopad -in enc_f5.txt -out de_f5.txt -k dees

```
[03/11/22]seed@VM:~/.../Task4$ openssl enc -aes-128-cbc -d -nopad -in enc_f5.txt -out de_f5.txt -k dees
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
[03/11/22]seed@VM:~/.../Task4$ openssl enc -aes-128-cbc -d -nopad -in enc_f10.txt -out de_f10.txt -k dees
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
[03/11/22]seed@VM:~/.../Task4$ openssl enc -aes-128-cbc -d -nopad -in enc_f16.txt -out de_f16.txt -k dees
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
[03/11/22]seed@VM:~/.../Task4$ ls -la
total 52
drwxrwxr-x 2 seed seed 4096 Mar 11 01:48 .
drwxrwxr-x 6 seed seed 4096 Mar 11 01:06 ..
-rw-rw-r-- 1 seed seed   16 Mar 11 01:48 de_f10.txt
-rw-rw-r-- 1 seed seed   32 Mar 11 01:48 de_f16.txt
-rw-rw-r-- 1 seed seed   16 Mar 11 01:48 de_f5.txt
-rw-rw-r-- 1 seed seed   32 Mar 11 01:42 enc_f10.txt
-rw-rw-r-- 1 seed seed   48 Mar 11 01:42 enc_f16.txt
-rw-rw-r-- 1 seed seed   32 Mar 11 01:41 enc_f5.txt
-rw-rw-r-- 1 seed seed    2 Mar 11 01:21 enc_test.txt
-rw-rw-r-- 1 seed seed   10 Mar 11 01:37 f10.txt
-rw-rw-r-- 1 seed seed   16 Mar 11 01:37 f16.txt
-rw-rw-r-- 1 seed seed    5 Mar 11 01:37 f5.txt
-rw-rw-r-- 1 seed seed    2 Mar 11 01:07 test.txt
[03/11/22]seed@VM:~/.../Task4$
```

Trying first just “cat”, then xxd:

```
[03/11/22]seed@VM:~/.../Task4$ cat de_f5.txt
12345
     [03/11/22]seed@VM:~/.../Task4$ xxd de_f5.txt
00000000: 3132 3334 350b 0b0b 0b0b 0b0b 0b0b 0b0b  12345...........
[03/11/22]seed@VM:~/.../Task4$ xxd de_f10.txt
00000000: 3132 3334 3536 3738 3930 0606 0606 0606  1234567890......
[03/11/22]seed@VM:~/.../Task4$ xxd de_f16.txt
00000000: 3132 3334 3536 3738 3930 3132 3334 3536  1234567890123456
00000010: 1010 1010 1010 1010 1010 1010 1010 1010  ................
[03/11/22]seed@VM:~/.../Task4$
```

6. Task 5: Error Propagation – Corrupted Cipher Text

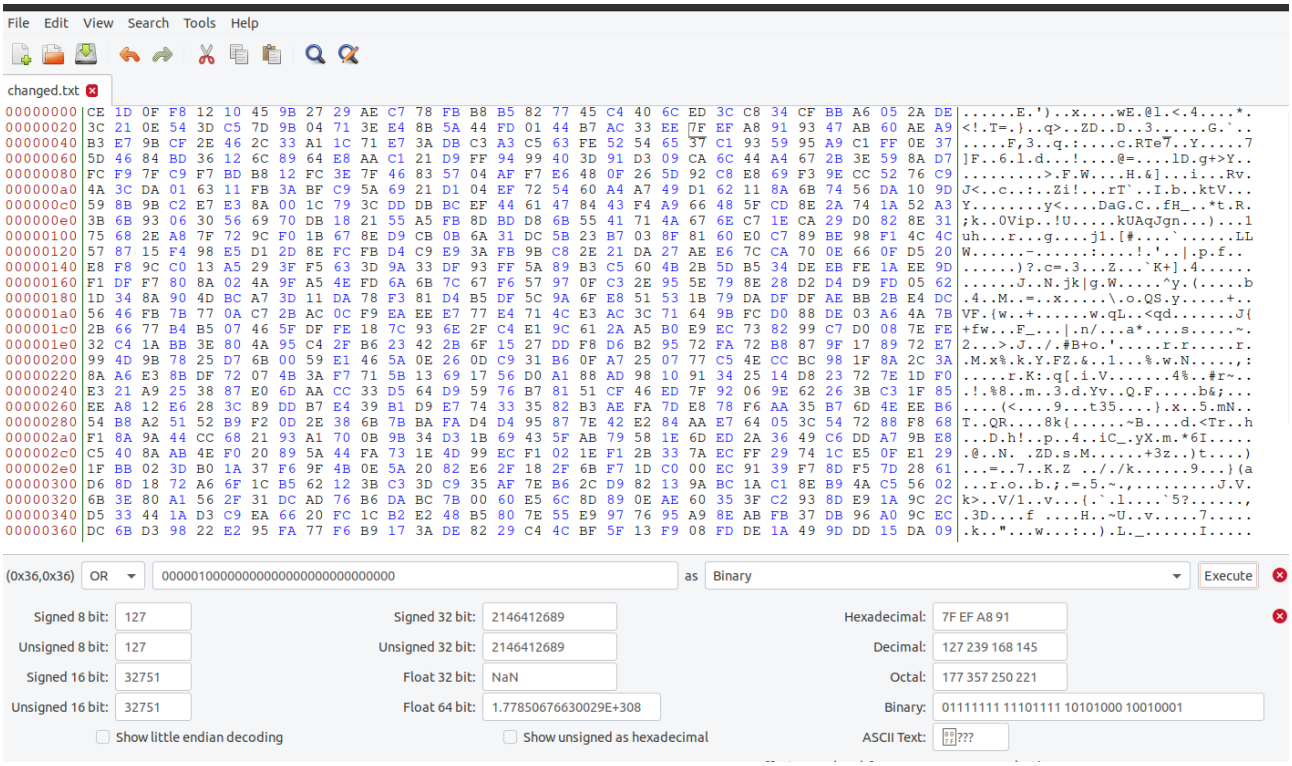
In this task, we tried to understand the error propagation property of various encryption modes [1].

Encrypt:

openssl enc -e -aes-128-ecb -in text.txt -out enc_ecb_text.txt -K 00112233445566778889aabbccddeeff

```
[03/11/22]seed@VM:~/.../Task5$ openssl enc -e -aes-128-cbc -in text.txt -out enc_cbc_text.txt -K  00112233445566778889aabbccddeeff -iv 007
hex string is too short, padding with zero bytes to length
[03/11/22]seed@VM:~/.../Task5$ openssl enc -e -aes-128-ecb -in text.txt -out enc_ecb_text.txt -K  00112233445566778889aabbccddeeff
[03/11/22]seed@VM:~/.../Task5$ openssl enc -e -aes-128-cfb -in text.txt -out enc_cfb_text.txt -K  00112233445566778889aabbccddeeff -iv 007
hex string is too short, padding with zero bytes to length
[03/11/22]seed@VM:~/.../Task5$ openssl enc -e -aes-128-ofb -in text.txt -out enc_ofb_text.txt -K  00112233445566778889aabbccddeeff -iv 007
hex string is too short, padding with zero bytes to length
[03/11/22]seed@VM:~/.../Task5$ ls -la
total 28
drwxrwxr-x 2 seed seed 4096 Mar 11 05:52 .
drwxrwxr-x 7 seed seed 4096 Mar 11 05:11 ..
-rw-rw-r-- 1 seed seed 1216 Mar 11 05:51 enc_cbc_text.txt
-rw-rw-r-- 1 seed seed 1208 Mar 11 05:52 enc_cfb_text.txt
-rw-rw-r-- 1 seed seed 1216 Mar 11 05:51 enc_ecb_text.txt
-rw-rw-r-- 1 seed seed 1208 Mar 11 05:52 enc_ofb_text.txt
-rw-rw-r-- 1 seed seed 1208 Mar 11 05:45 text.txt
[03/11/22]seed@VM:~/.../Task5$
```

CBC:



```
[03/11/22]seed@VM:~/.../Task5$ openssl enc -d -aes-128-cbc -in changed.txt -out de_cbc_text.t
xt -K  00112233445566778889aabbccddeeff -iv 007
hex string is too short, padding with zero bytes to length
[03/11/22]seed@VM:~/.../Task5$ cat de_cbc_text.txt
Haihtuvi nuoruus niinkuin vierivä virta.
Langat▯X▯4lq▯:▯▯▯+▯^ elon oultainen pirta.
Turhaan, oi turhaa tartun ma hetkehen kiini,
riemua ei suo rattoisa seura, ei viini.

Häipyvät taakse tahtoni ylpeät päivät.
Henkeni hurmat ammoin jo jälkehen jäivät.
Notkosta nousin. Taasko on painua tieni?
Toivoni ainoo: tuskaton tuokio pieni.

Tiedän ma: rauha mulle on mullassa suotu.
Etsijän tielle ei lepo lempeä luotu,
pohjoinen puhuu, myrskyhyn aurinko vaipuu,
jää punajuova: kauneuden voimaton kaipuu.

Upposi mereen unteni kukkivat kunnaat.
Mies olen köyhä: kalliit on laulujen lunnaat.
Kaikkeni annoin, hetken ma heilua jaksoin,
haavehen kullat  mieleni murheella maksoin.

Uupunut olen, ah, sydänjuurihin saakka!
Liikaako lienee pantukin paatinen taakka?
Tai olen niitä, joilla on tahto, ei voima?
Voittoni tyhjä, työn tulos tuntoni soima.

Siis oli suotta kestetyt, vaikeat vaivat,
katkotut kahleet, poltetut, rakkahat laivat?
Nytkö ma kaaduin, kun oli kaikkeni tarpeen?
Jähmetyn jääksi, kun meni haavani arpeen.

Toivoton taisto taivaan valtoja vastaan!
Kaikuvi kannel; lohduta laulu ei lastaan.
```

CFB:

```
[03/11/22]seed@VM:~/.../Task5$ openssl enc -d -aes-128-cfb -in or_enc_cfb_text.txt -out de_cfb_
text.txt -K  00112233445566778889aabbccddeeff -iv 007
hex string is too short, padding with zero bytes to length
[03/11/22]seed@VM:~/.../Task5$ cat de_cfb_text.txt
Haihtuvi nuoruus niinkuin vierivä virta.
Langat jo haRmaat lyö▯gicb▯▯f▯U▯▯Z&pirta.
Turhaan, oi turhaa tartun ma hetkehen kiini,
riemua ei suo rattoisa seura, ei viini.

Häipyvät taakse tahtoni ylpeät päivät.
Henkeni hurmat ammoin jo jälkehen jäivät.
Notkosta nousin. Taasko on painua tieni?
Toivoni ainoo: tuskaton tuokio pieni.

Tiedän ma: rauha mulle on mullassa suotu.
Etsijän tielle ei lepo lempeä luotu,
pohjoinen puhuu, myrskyhyn aurinko vaipuu,
jää punajuova: kauneuden voimaton kaipuu.

Upposi mereen unteni kukkivat kunnaat.
Mies olen köyhä: kalliit on laulujen lunnaat.
Kaikkeni annoin, hetken ma heilua jaksoin,
haavehen kullat  mieleni murheella maksoin.

Uupunut olen, ah, sydänjuurihin saakka!
Liikaako lienee pantukin paatinen taakka?
Tai olen niitä, joilla on tahto, ei voima?
Voittoni tyhjä, työn tulos tuntoni soima.

Siis oli suotta kestetyt, vaikeat vaivat,
katkotut kahleet, poltetut, rakkahat laivat?
Nytkö ma kaaduin, kun oli kaikkeni tarpeen?
Jähmetyn jääksi, kun meni haavani arpeen.

Toivoton taisto taivaan valtoja vastaan!
Kaikuvi kannel; lohduta laulu ei lastaan.
Hallatar haastaa, soi sävel sortuvin siivin.
Rotkoni rauhaan kuin peto kuoleva hiivin.
```

54. (+-1) Byte mixed off by one bit. This ruines either the whole package, as in ECB or a signle character as in OFB.

ECB

```
[03/11/22]seed@VM:~/.../Task5$ openssl enc -d -aes-128-ecb -in or_enc_ecb_text.txt -out de_ecb_text
.txt -K  00112233445566778889aabbccddeeff
[03/11/22]seed@VM:~/.../Task5$ cat de_ecb_text.txt
Haihtuvi nuoruus niinkuin vierivä virta.
?V.U4MH elon kultainen pirta.
Turhaan, oi turhaa tartun ma hetkehen kiini,
riemua ei suo rattoisa seura, ei viini.

Häipyvät taakse tahtoni ylpeät päivät.
Henkeni hurmat ammoin jo jälkehen jäivät.
Notkosta nousin. Taasko on painua tieni?
Toivoni ainoo: tuskaton tuokio pieni.

Tiedän ma: rauha mulle on mullassa suotu.
Etsijän tielle ei lepo lempeä luotu,
pohjoinen puhuu, myrskyhyn aurinko vaipuu,
jää punajuova: kauneuden voimaton kaipuu.

Upposi mereen unteni kukkivat kunnaat.
Mies olen köyhä: kalliit on laulujen lunnaat.
Kaikkeni annoin, hetken ma heilua jaksoin,
haavehen kullat  mieleni murheella maksoin.

Uupunut olen, ah, sydänjuurihin saakka!
Liikaako lienee pantukin paatinen taakka?
Tai olen niitä, joilla on tahto, ei voima?
Voittoni tyhjä, työn tulos tuntoni soima.

Siis oli suotta kestetyt, vaikeat vaivat,
katkotut kahleet, poltetut, rakkahat laivat?
```

OFB

```
[03/11/22]seed@VM:~/.../Task5$ openssl enc -d -aes-128-ofb -in or_enc_ofb_text.txt -out de_ofb_text
.txt -K  00112233445566778889aabbccddeeff -iv 007
hex string is too short, padding with zero bytes to length
[03/11/22]seed@VM:~/.../Task5$ cat de_ofb_text.txt
Haihtuvi nuoruus niinkuin vierivä virta.
Langat jo hasmaat lyö elon kultainen pirta.
Turhaan, oi turhaa tartun ma hetkehen kiini,
riemua ei suo rattoisa seura, ei viini.

Häipyvät taakse tahtoni ylpeät päivät.
Henkeni hurmat ammoin jo jälkehen jäivät.
Notkosta nousin. Taasko on painua tieni?
Toivoni ainoo: tuskaton tuokio pieni.

Tiedän ma: rauha mulle on mullassa suotu.
Etsijän tielle ei lepo lempeä luotu,
pohjoinen puhuu, myrskyhyn aurinko vaipuu,
jää punajuova: kauneuden voimaton kaipuu.

Upposi mereen unteni kukkivat kunnaat.
Mies olen köyhä: kalliit on laulujen lunnaat.
Kaikkeni annoin, hetken ma heilua jaksoin,
haavehen kullat  mieleni murheella maksoin.

Uupunut olen, ah, sydänjuurihin saakka!
Liikaako lienee pantukin paatinen taakka?
Tai olen niitä, joilla on tahto, ei voima?
```

First intuition would be that either the one byte / character is changed as it is changed inside encrypted file. Or then those blocks are linked together in a such way that when the first one is lost, so will be everything else too from that point on.

Happens that one character error is tolerated well in OFB, but even if one block is corrupted, it does not mean, other blocks will be corrupted too.

7. Task 6: Initial Vector (IV) and Common Mistakes

7.1 Task 6.1. IV Experiment

Testing same and different IV:s:

echo Testing same and different IV:s > test.txt

same

openssl enc -e -aes-256-cfb -in test.txt -out enc_test.txt -K 00112233445566778889aabbccddeeff -iv 007

again

openssl enc -e -aes-256-cfb -in test.txt -out enc_test2.txt -K 00112233445566778889aabbccddeeff -iv 007

different iv

openssl enc -e -aes-256-cfb -in test.txt -out enc_test3.txt -K 00112233445566778889aabbccddeeff -iv 008

```
[03/15/22]seed@VM:~$ cd Labsetup/
[03/15/22]seed@VM:~/Labsetup$ ls
docker-compose.yml  encryption_oracle  Files  Task3  Task4  Task5
[03/15/22]seed@VM:~/Labsetup$ mkdir Task6
[03/15/22]seed@VM:~/Labsetup$ cd Task6
[03/15/22]seed@VM:~/.../Task6$ ls
[03/15/22]seed@VM:~/.../Task6$ echo Testing same and different IV:s > test.txt
[03/15/22]seed@VM:~/.../Task6$ ls
test.txt
[03/15/22]seed@VM:~/.../Task6$ cat test.txt
Testing same and different IV:s
[03/15/22]seed@VM:~/.../Task6$ openssl enc -e -aes-256-cfb -in test.txt -out enc_test.txt -K  00112233445566778889aabbccddeeff -iv 007
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/15/22]seed@VM:~/.../Task6$ openssl enc -e -aes-256-cfb -in test.txt -out enc_test2.txt -K  00112233445566778889aabbccddeeff -iv 007
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/15/22]seed@VM:~/.../Task6$ openssl enc -e -aes-256-cfb -in test.txt -out enc_test3.txt -K  00112233445566778889aabbccddeeff -iv 008
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
[03/15/22]seed@VM:~/.../Task6$
```

Differences by diff and looking inside the files:

```
[03/15/22]seed@VM:~/.../Task6$ diff -yt enc_test1.txt enc_test2.txt
diff: enc_test1.txt: No such file or directory
[03/15/22]seed@VM:~/.../Task6$ diff -yt enc_test.txt enc_test2.txt
P'C@@q@u@@l@c@_:@^!@@W@I2@Ik`                                   P'C@@q@u@@l@c@_:@^!@@W@I2@Ik`[03/15/22]seed@VM:~/.../Task6$
 diff -yt enc_test.txt enc_test3.txt
P'C@@q@u@@l@c@_:@^!@@W@I2@Ik`                                  \  @
                                                                  @X@\LtY@%@@@6
                                                             >  d]n:Ul@O@@[03/15/22]seed@VM:~/.../Task6$

[03/15/22]seed@VM:~/.../Task6$ cat enc_test.txt
P'C@@q@u@@l@c@_:@^!@@W@I2@Ik`[03/15/22]seed@VM:~/.../Task6$
[03/15/22]seed@VM:~/.../Task6$ cat enc_test2.txt
P'C@@q@u@@l@c@_:@^!@@W@I2@Ik`[03/15/22]seed@VM:~/.../Task6$
[03/15/22]seed@VM:~/.../Task6$ cat enc_test3.txt
@
 @X@\LtY@%@@@6
d]n:Ul@O@@[03/15/22]seed@VM:~/.../Task6$
```

Same iv produces same encryption for the same text if key remains the same

7.2 Task 6.2. Common Mistake: Use the Same IV

Plaintext (P1): This is a known message!

Ciphertext (C1): a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159

Plaintext (P2): (unknown to you)

Ciphertext (C2): bf73bcd3509299d566c35b5d450337e1bb175f903fafc159

Finding a key = C1 $\oplus$ P1

Finding P2 = C2 $\oplus$ key

P2 = Launch a missile!

```python
#!/usr/bin/python3

# XOR two bytearrays
def xor(first, second):
   return bytearray(x^y for x,y in zip(first, second))

MSG   = "This is a known message!"
HEX_1 = "a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159"
HEX_2 = "bf73bcd3509299d566c35b5d450337e1bb175f903fafc159"

# Convert ascii string to bytearray
D1 = bytes(MSG, 'utf-8')

# Convert hex string to bytearray
D2 = bytearray.fromhex(HEX_1)
D3 = bytearray.fromhex(HEX_2)

r1 = xor(D1, D2)
r2 = xor(D2, D3)
r3 = xor(D2, D2)

key = xor(D2, D1)
plain = xor(D3, key)
print(r1.hex())
print(r2.hex())
print(r3.hex())
print(plain.decode('utf8'))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
python.python-2022.2.1924087327/pythonFiles/lib/python/debugpy/launcher 58153 -- /
f001d8b622a8b99907b6353e2d2356c1d67e2ce356c3a478
1b1a0d165253536c0055050d07570f00000c0000080b0000
00000000000000000000000000000000000000000000000000
Order: Launch a missile!
raulivirtanen@c6a7c158 ~ %
```

7.3 Task 6.3. Common Mistake: Use a Predictable IV

What I tried to to (several different ways) is to feed to the oracle answer A XOR AIV NEXTIV for answer A correspondent to Bob's cipher. This for both, "Yes" and "No", answers. I manage to get 0d or 0e padded answers, cipher is different for each pad left out, but remains identical to leftout "peers" ciphers. This is repeatable even with changing IV:s and thus different fed "plaintext". None is the same than Bob's cipher, which is full 16 bytes and messes the given cipher for another block.

```
Next IV        : 7b6ed5890a8ed1499e8510ab4ac086cf
Your plaintext : 596573
Your ciphertext: 6b0109604c917af6bcf6e6aa55f31166

Next IV        : 877832e60a8ed1499e8510ab4ac086cf
Your plaintext : 55c2876b01
Your ciphertext: 159dd32abc75cbba8902b276b682190b

Next IV        : 93c2d4010b8ed1499e8510ab4ac086cf
Your plaintext : 4178618c000d0d0d0d0d0d0d0d0d0d0d
Your ciphertext: 832af7ea2afd815a0d4250a89b2c759edc238315f1bbbbf3d023f435946f6fb
4

Next IV        : 172aca580b8ed1499e8510ab4ac086cf
Your plaintext : c5907fd500
Your ciphertext: 159dd32abc75cbba8902b276b682190b

Next IV        : 3d1863990b8ed1499e8510ab4ac086cf
Your plaintext :
```

Finally I manage to get identical ciphertexts from YES

This is different than the one we are looking for.

Manually tried to run encryption with given values:

openssl enc -e -aes-128-cbc -in “6218282e0f0d0d0d0d0d0d0d0d0d0d0d” -out testi.txt -K fdff4d25548b6953b188ed6e1ee7597f -iv 27f8fb6c9fd33cab013179a5bc61bb1f

Imported Crypto.Cipher AES library to build ciphers and investigate them inside python -code, but DL of the submission came too soon for that.

```python
#!/usr/bin/python3

# XOR two bytearrays
from bdb import BdbQuit
from binascii import unhexlify

from Crypto.Cipher import AES


def xor(first, second):
```

```
  return bytearray(x^y for x,y in zip(first, second))


BOBCIPHER = "b81f9e67c45558f5bdb499c6af8bef6d"  #16
BOBIV =     "1c85a04f9dd33cab013179a5bc61bb1f"

BOBYES = "5965730d0d0d0d0d0d0d0d0d0d0d0d0d0d"
BOBYES2 = "596573"
BOBNO = "4E6F0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e"
BOBNO2 = "4E6F"


BIV = bytearray.fromhex(BOBIV)
BC = bytearray.fromhex(BOBCIPHER)
BY = bytearray.fromhex(BOBYES)
BN = bytearray.fromhex(BOBNO)
BYSH = bytearray.fromhex(BOBYES2)
BNSH = bytearray.fromhex(BOBNO2)

keydplain = xor(BC, BIV)
bobkeyyes = xor(keydplain, BY)
bobkeyno = xor(keydplain, BN)



bob_to_block = xor(BY, BIV)
bob_from_block = BC
bob_key_if_yes = xor(bob_to_block, bob_from_block)
bob_to_block = xor(BN, BIV)
bob_key_if_no = xor(bob_to_block, bob_from_block)
print(bob_key_if_yes.hex())
biv = unhexlify(BOBIV)
bkey = unhexlify(bob_key_if_yes.hex())
byes = unhexlify(BY.hex())
bob_cipher = AES.new(bkey, AES.MODE_CBC,biv)
#print("Bob again? " + bob_cipher. + "  ")
print(bob_cipher.encrypt(BY.hex()).hex())

print("Bob to block (=A XOR AIV): " + bob_to_block.hex())
print("Bob key to yes/" + bob_key_if_yes.hex())
print("Bob key if no: " + bob_key_if_no.hex())
```

```
print("Bob's cipher/Bob from block: " + bob_from_block.hex())
print();

while True:
   answer = input("What word to test (yes/no)? ")
   iv_value = input("Next IV? ")
   NEXTIV = bytearray.fromhex(iv_value)


   if answer == "yes":

      print("Answer 'Yes', here " + BY.hex() + " and Bob's IV " + BIV.hex() + " are XORred: " +
xor(BY,BIV).hex() )
      print(bob_to_block.hex() + "  " + xor(bob_to_block, NEXTIV).hex())
      print ("This XORred with NEXT IV " + NEXTIV.hex() + " is: " + xor(xor(BY, BIV),NEXTIV).hex())
      print("NEXTIV XOR MYGUESS YES: " + xor(NEXTIV, BYSH).hex())
      cipher = input("Enter Cipher: ")
      CIP = bytearray.fromhex(cipher)
      cipher_xored = xor(CIP,bob_key_if_yes)
      print(CIP.hex() + " cipher XORred with key_yes ie. what goes to my block" +  cipher_xored.hex())
      plain = xor(cipher_xored, BIV)
      print("Xored with IV " + plain.hex())

   elif answer == "no":
      print ("No will be coded as " + xor(xor(BN, BIV),NEXTIV).hex())
      cipher = input("Enter Cipher: ")
      CIP = bytearray.fromhex(cipher)
      cipher_xored = xor(CIP,bob_key_if_no)
      print(CIP.hex() + " cipher XORred with key_no ie. what goes to my block " +  cipher_xored.hex())
      plain = xor(cipher_xored, BIV)
      print("Xored with IV " + plain.hex())
```

7 Conclusion

This practice lead us to play with different kinds of encryption modes and their attributes on – literally – bit by bit. Every exercise was a new one at this level, really seeing what encryption looks like.

It would be interesting to know better, how for example Python could be used to solve these puzzles. It seems kind of a tough area but contains handy tools to possess.

We found it easier to understand the theory behind the subject, when completing the tasks in practice via trial and error.

8 List of reference

[1]https://moodle.tuni.fi/pluginfile.php/2310955/mod_resource/content/1/Crypto_Encryption.pdf

[2] https://www.nytimes.com/2018/03/01/movies/oscars-sunday-what-to-expect.html

[3] https://www.cryptosys.net/pki/manpki/pki_paddingschemes.html