

# Raport pentru lucrarea 5: Cifrarea cu cheie secretă

Autor: Birlutiu Claudiu-Andrei, gr 30643

## Sarcina 1: Analiza frecvenței împotriva unui cifru de substituție monoalfabetic

- În continuare am urmat tuorialul de criptare:
  - in prima faza am creat un script bash care ia cifrul și îl transforma în plain text (fără semene de punctuație și litere mici) – fișierul se numește **modify\_script.sh** și i s-a dat permisiuni de execuție; se observa ca din textul article.txt s-a generat plaintext.txt care reprezintă textul fără ceme de punctuație și lowercase

```

$ modify_script.sh
LOS > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina1_birlutiu > $ modify_script.sh
1 #!/bin/bash
2
3 tr [:upper:] [:lower:] < article.txt > lowercase.txt &&
4 tr -cd '[a-z][\n]':[:space:] < lowercase.txt > plaintext.txt;
5
LOS > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina1_birlutiu >
1 Object detection performance, as measur
2 The best-performing methods are complex
3 In this paper, we propose a simple and
4 Our approach combines two key insights:
5 We also compare R-CNN to OverFeat, a re
LOS > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina1_birlutiu > plain
1 object detection performance as measured on t
2 the bestperforming methods are complex ensemb
3 in this paper we propose a simple and scalabl
4 our approach combines two key insights one c
5 we also compare rcnn to overfeat a recently p
  
```

- am creat un script python care genereaza o cheie aleatoare de mapare a literelor alfabetului la o alta litera și am făcut inlocuirile pe plaintext.txt => cipher.txt

```

Run Terminal Help
generate_key.py
LOS > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina1_birlutiu > generate_key.py
1 #!/usr/bin/env python3
2
3 import random
4 #lista cu elliterele afabetului englez
5 s = "abcdefghijklmnopqrstuvwxyz"
6 #generarea unei liste de litere random de lungimea sirului s
7 # => amestecarea aleatoare a literelor alfabetului
8 list = random.sample(s, len(s))
9
10 #o permutare a alafabetului englez
11 print(''.join(list))
12
13
LOS > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina1_birlutiu >
1 object detection performance as measured or
2 the bestperforming methods are complex ense
3 in this paper we propose a simple and scal
4 our approach combines two key insights one
5 we also compare rcnn to overfeat a recentl
LOS > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina1_birlutiu > cipher.txt
1 kpycge bcecegvka zcqtqajuagc ui jcuighcb ka emc guakavgun zuigu
2 emc priezctqajvad jcemkbi uqg gkjzncr caicjnc ioiecji emue eo
3 va emvi zuzcq fc zqzkzic u ivjznc uab igunupnc bcecegvka undkqy
4 khq uzzakugm gkjpvaci efk xco vaivdmei kac gua uzzno mvdnguzuc
5 fc unik gkjzuc qgaag ek klcqtcue u qcgcaeno zqzkzicb invbvadvfe
  
```

- Am trecut la descifrarea textului criptat din lucrare. Scopul a fost sa analizez frecventa literelor sau grupurilor de litere astfel încât sa găsesc cheia de criptare a mesajului
- am modificat programul python de analiza a textului sa scrie într-un fisier frecventele
- am căutat pe internet care sunt cele mai frecvente litere din limba engleza

The screenshot shows a VS Code editor with a Python script named `analiza_frecventei_birlu.py` and its output in the terminal. The script reads a ciphertext file, counts the frequency of letters and n-grams, and writes the results to a file named `frequencies.txt`.

**LETTER FREQUENCY**

- The most commonly used letters of the English language are e, t, a, i, o, n, s, h, and r.
- The letters that are most commonly found at the beginning of words are t, a, o, d, and w.
- The letters that are most commonly found at the end of words are e, s, d, and t.

**Script Output (frequencies.txt):**

```

16 2: 99
17 l: 90
18 g: 83
19 b: 83
20 r: 82
21 e: 76
22 d: 59
23 -----
24 2-gram (top 20):
25 yt: 115
26 tn: 89
27 mu: 74
28 nh: 58
29 vh: 57
30 hn: 57
31 vu: 56
32 nq: 53
33 -----

```

**Terminal Output:**

```

[04/05/23]seed@VM:~/.../analiza_frecventei_birlu
[04/05/23]seed@VM:~/.../analiza_frecventei_birlu
[04/05/23]seed@VM:~/.../analiza_frecventei_birlu

```

Rank <sup>[1]</sup>	Trigram	Frequency <sup>[3]</sup> (Different source)
1	the	1.81%
2	and	0.73%
3	tha	0.33%
4	ent	0.42%
5	ing	0.72%
6	ion	0.42%
7	tio	0.31%
8	for	0.34%
9	nde	
10	has	
11	nce	
12	edt	
13	tis	
14	oft	0.22%
15	sth	0.21%
16	men	

## Securitatea sistemelor și a aplicațiilor

```
freq.py  cipher.txt x
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina1_birlutiu > analiza_frecventei_birlutiu > cipher.txt
1 ytn xqavhq yzhu xu qzupvd lmat qnccq vgxyz hmrty ytmq ixur qyhvurn
2 vlvhpq yhme ytn gvrnbn bniq imsn v uxuvruvnmv yxx
3
4 ytn vlvhpq hvan lvq gxsnupnp gd ytn pncmqn xb tvhfn lnuqynmu vy myq xzyqny
5 vup ytn veevhuu mceixqmxu xb tmq bmic axcevd vy ytn nup vup my lvq qtenp gd
6 ytn ncnhrnuan xb cnxxx ymcq ze givasrllu eximymaq vhcavupd vaymfmcq vup
7 v uvymxuvl axufnhqymxu vq ghmb vup cvp vq v bfnh phnvc vgxyz ltnytnh ytnhn
8 xzrty yx gn v ehqmpnuu lmbhnd ytn qnvqxu pmpuy ozqy qnnc nkyhv ixur my lvq
9 nkyhv ixur gnavzqn ytn xqavhq lnhn cxfp yx ytn bmqy lnnsup mu cvhat yx
10 vfxmp axubimaymur lmyt ytn aixqmur anhnxcud xb ytn lmuynh xidcemaq ytvusq
11 ednxuratvur
12
13 xun qmr jznqymxu qzhxzupmur ytmq dnvhq vavpncd vlvhpq mq txl xh mb ytn

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[04/08/23]seed@W:~/.../analiza_frecventei_birlutiu$ tr 'ytn' 'THE' < cipher.txt > out1.txt
```

- în urma analizei a reiesit faptul ca n apare de cele mai multe ori (488); yt - apare de asemenea de cel mai multe ori și ytn - apare de asemenea de cele mai multe ori => vom inlocui astfel **y** cu **t**, **n** cu **e** și **t** cu **h**
- **and** este de asemenea o triagrama desul de întâlnită în textul englezesc, iar la noi pe a doua poziție a triagramelor regasim vup => încercam astfel sa inlocuim v cu a, u cu n și p cu d
  - de asemenea **ing** este o trigramă frecventă -> iar la noi **mur** (a 3-a cea mai frecventă) s-ar potrivi cu ing având în vedere supozitia anterioară (
  - v = a, u = n, p = d, m = i, r = g

```
Run Terminal Help
freq.py  cipher.txt x
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina1_birlutiu > analiza_frecventei_birlutiu > cipher.txt
1 ytn xqavhq yzhu xu qzupvd lmat qnccq vgxyz hmrty ytmq ixur qyhvurn
2 vlvhpq yhme ytn gvrnbn bniq imsn v uxuvruvnmv yxx
3
4 ytn vlvhpq hvan lvq gxsnupnp gd ytn pncmqn xb tvhfn lnuqynmu vy myq xzyqny
5 vup ytn veevhuu mceixqmxu xb tmq bmic axcevd vy ytn nup vup my lvq qtenp gd
6 ytn ncnhrnuan xb cnxxx ymcq ze givasrllu eximymaq vhcavupd vaymfmcq vup
7 v uvymxuvl axufnhqymxu vq ghmb vup cvp vq v bfnh phnvc vgxyz ltnytnh ytnhn
8 xzrty yx gn v ehqmpnuu lmbhnd ytn qnvqxu pmpuy ozqy qnnc nkyhv ixur my lvq
9 nkyhv ixur gnavzqn ytn xqavhq lnhn cxfp yx ytn bmqy lnnsup mu cvhat yx
10 vfxmp axubimaymur lmyt ytn aixqmur anhnxcud xb ytn lmuynh xidcemaq ytvusq
11 ednxuratvur
12
13 xun qmr jznqymxu qzhxzupmur ytmq dnvhq vavpncd vlvhpq mq txl xh mb ytn
14 anhnxcud lail vpphnq cnxxx ngenamvld vbyhn ytn rxipnu rixgnq lmat gnacn
15 v ozgmivuy axemurxzy evhyd bnh ymcq ze ytn cxfncnu qenvhtnvpnp gd
16 exlnhbzi lliidlxsd lxenu ltx tniemp hvamq cmiinxuq xb pxilvhq yx bmrty qnkzvi
17 tvhvqgcnuv vhxzup ytn axzuyhd
18

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[04/08/23]seed@W:~/.../analiza_frecventei_birlutiu$ tr 'ytnvupmr' 'THEANDIG' < cipher.txt > out1.txt
```

- avem:
  - ytn = the
  - vup = and
  - mur = ing
  - ynh = te\* => ter [0] **h = r**
  - xzy = \*\*t => o\*t => out [2] **z = u**
  - mxu = i\*n => [1] **x = o**
  - gnq = \*e\* => \*es [4] bes => **g = b**
  - ytv = tha
  - nqy = e\*t => [3] **q = s**

- o vii = a\*\* => [7] **i = l** (NATIONa**i**)
- o bxh = \*or = [5] for **b=f**
- o lvq = \*as = [6] was **l = w**
- o nuy = ent
- o vyn = ate
- o uvy = nat
- o lmu = win
- o nvh = ear
- o cmu = \*in [8] => min **c=m** (DEclqE = Dm**c**lSE = DEMISE)
- o tmq = his
- o vhp => ard

=>

- a b c d e f g h i j k l m n o p q r s t u v w x y z
- v g \* p n b r t m i c u x h q y z l o

- încercam cu valorile descoperite iar apoi vom identifica și pe celelalte lipsa
- abdefghilmnorstuw (ABDEFGHILMNORSTUWX) = vgnbrtmicuxhqyzlo

```

Run Terminal Help
freq.py  cipher.txt x
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina1_birlutiu > analiza_frecventei_birlutiu > cipher.txt
1 ytn xqavhq yzhu xu qzupvd lmat qnnc vxzy hmrty vbynh ytmq ixur qyhvurn
2 vlvhpq yhme ytn gvrnnh bnniq lmsn v uxuvrnvnmv yxx
3
4 ytn vlvhpq hvan lvq gxsnupnp gd ytn pncmqn xb tvhnd lmuqymu vy myq xzyqny
5 vup ytn veehnuy nceixqmxu xb tmq bmic axcevd vy ytn nup vup my lvq qtenp gd
6 ytn ncnhrnuan xb cnxxx ymcq ze givasrlu eximymag vhcavupd vaymfmc vup
7 v uvmxuvi axufnhqymxu vq ghmb vup cvp vq v bfnh phnc vxzy ltnytn ytnhn
8 xzrty yx gn v ehqmpny lumbnd ytn qnvqxu pmpuy ozqy qnnc nkyhv ixur my lvq
9 nkyhv ixur gnayzn ytn xqavhq lnhn cxfnp yx ytn bmqy lnnspu mu cvhat yx
10 vfxmp axubimaymur lmyt ytn aixqur anhnxcud xb ytn lmuynh xidcemaq ytvusq
11 ednxuravur
12
13 xun gmr jznqymxu qzhxzipmur ytmq dnvhq vavpncd vlvhpq mq txl xh mb ytn
14 anhnxcud lmi vpphqq cnxxx nqenamviid vbynh ytn rxiptu rixgq lmat gnaycn
15 v ozgmivuy axcmurxy evyhd bxh ymcq ze ytn cxfncuy qenvhtvppn gd
16 exlnhbzi txidlxp lxcnu ltx tnienp hvnmq cmiimxuq xb pxlihvq yx bmrty qnkzvi
17 tvhvqgcny vhxzup ytn axzuyhd
18

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[04/08/23] seed@WM:~/.../analiza_frecventei_birlutiu$ tr 'ytnvupmr' 'THEANDIG' < cipher.txt > out1.txt
[04/08/23] seed@WM:~/.../analiza_frecventei_birlutiu$ tr 'ytnvupmr' 'THEANDIG' < cipher.txt > out1.txt
[04/08/23] seed@WM:~/.../analiza_frecventei_birlutiu$ tr 'vgnbrtmicuxhqyzlo' 'ABDEFGHILMNORSTUWX' < cipher.txt > out1.txt
[04/08/23] seed@WM:~/.../analiza_frecventei_birlutiu$

```

- identificam restul corespondetelor din cuvinte: OSaARS => **a = c**; TRle => **e = p**; LIsE => **s = k**; Bd => **d = y**; HARfEY => **f = v**; SEkIST => **k = x**; jUESTION => **j = q**; EkTRA => **k = x**; (**modificare**); (**X**UST=JUST) => **o = j**; (eRIwE = PRIwE) => **w = z**

## REZULTAT:

ab**c**defghij**k**lmnop**q**rstu**v**w**x**y**z**(AB**C**DEFGHI**J**KLMNOP**Q**RSTUV**W**XYZ)=  
vg**a**pnb**r**tm**o**sicux**e**jhqyz**f**k**d**w

# Securitatea sistemelor și a aplicațiilor

```
Run Terminal Help
┌ freq.py ── cipher.txt X ── ... ── frequencies.txt ── out1.txt U ── out2.txt U X ── HX  [?] [?] ...
└─ Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina1_birlutiu > analiza_frecventei_birlutiu > ── cipher.txt
1  ytn xqavhq yzhu xu qzupvd ltnat qnncq vgxy hmrty vbynh ytmq ixur qyhvurn
2  vlvhpq yhme ytn gvrnrh bnniq imsn v uxuvrnvumvu yxx
3
4  ytn vlvhpq hvan lvq gxssnupn gd ytn pncqn xb tvhfn lnuqynmu vy myq xzyqny
5  vup ytn veevhuu mceixqmxu xb tmq bmic axcevd vy ytn nup vup my lvq qtenp gd
6  ytn ncnhrnuan xb cnyxx ymcnq ze givasrxlu exlmyaq vhcavupd vaymfaqc vup
7  v uyvmxuvi axufnhqymxu vq ghmbn vup cvp vq v bfnh phnvc vgxy ltnytnh ytnhn
8  xzrty yx gn v ehngmpny lmbhnd ytn qnvqxu pmpuy ozqy qnnc nkyhv ixur my lvq
9  nkyhv ixur gnazqn ytn xqavhq lnhn cxfnp yx ytn bmqy lnnsnup mu cvhat yx
10 vfxmp axubimaymur lmyt ytn aixqmur anhnxcud xb ytn lmuynh xidcemaq ytvusq
11 ednxuratvur
12
13 xun qmr jznqymxu qzhxzupmur ytmq dnvhq vavpncd vlvhpq mq txl xh mb ytn
14 anhnxcud lmi vphnqg cnyxx ngenamviid vbynh ytn rxipnu rixgnq ltnat gnavecn
15 v ozgmivuy axcmrxy evhyd bxh ymcnq ze ytn cxfnucuy qenvhtnypn gd
16 exlnhbzi txiidxp lxcnu ltx tnienp hvmaq cmilmxuq xb pxlivhq yx bmrty qnkzvi
17 tvhqqcny vhxzup ytn axzuyhd
18
19 qmruvimur ytmh qzeexy rxipnu rixgnq vynnupnq qlvytnp ytnqnfng mu givas
20 qexhynp iveni emuq vup qxzipn xbb vgxy qnkmaq exlnh mcgviuqan bhxc ytn hnp
21 avheny vup ytn qvrn xu ytn vmh n lvq avilnp xzy vgxy evd munjzmyd vbynh
22 myq bxhcnh vuatqh avvy qvpinh jzmy xuan qtn invhnp ytyv qtn lvq cvsmur bvh
23 inqg ytvu v cvin axtxqy vup pzhmur ytn anhnxcud uyvimm ehycvu yxys v gizuy
24 vup qvymqbdmur pmr vy ytn viicvin hxqynh xb uxcmuvynp pmhnayxhq txl axzip
25 ytyv gn yxeenp
26
27 vq my yzhuq xzy vy invqy mu ynhcq xb ytn xqavhq my ehxvgid lxuy gn
28
29 lxcnu mufxifnp mu ymcnq ze qvmp ytyv viytxzrt ytn rixgnq qmrumbnnp ytn
30 mummymvmfng ivzuat ytnd unfnh muynupn my yx gn ozqy vu vlvhpq qnvqxu
31 avcevmru xh xun ytyv gnavecn vqgxamvynp xuid lmyt hnpavheny vaymxuq muqynvp
32 v qexsnqlxcvu qvmp ytn rhxez mq lxhsmur gntmup aixqnp pxxhq vup tvq qmuau
33 vcqagqp cmilmxu bxh myq invri pnbnuqn bzup ltnat vbynh ytn rixgnq lvq
34 bixxnpn lmyt ytxzqvupq xb pxuvmxuq xb xh inqg bhxc enxein mu qxcn
35 axzuyhmq
36
37
38 ux avii vx lnhv qivas rxluq lnuv xzy mu vpfvuan xb ytn xqavhq ytxzrt ytn

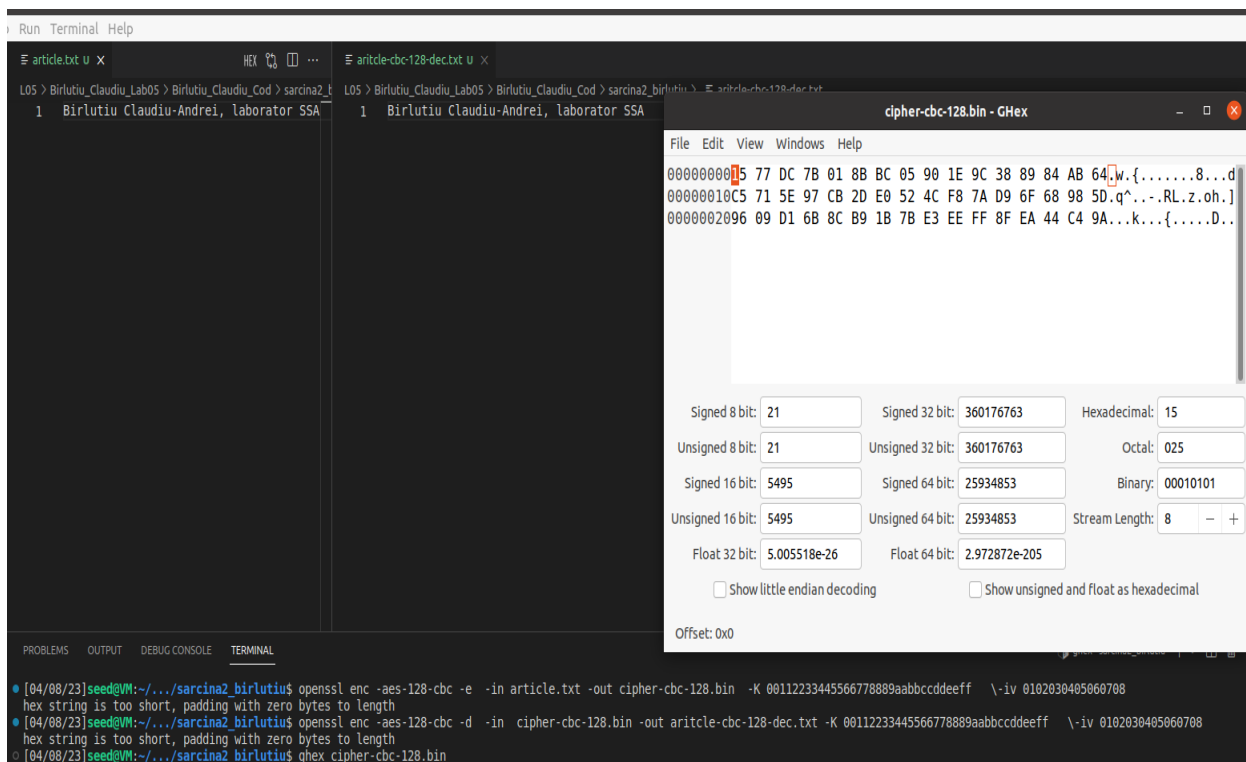
┌─ Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina1_birlutiu > analiza_frecventei_birlutiu > ── out2.txt
1  THE OSCARS TURN ON SUNDAY WHICH SEEMS ABOUT RIGHT AFTER THIS LONG STRANGE
2  AWARDS TRIP THE BAGGER FEELS LIKE A NONAGENARIAN TOO
3
4  THE AWARDS RACE WAS BOOKENDED BY THE DEMISE OF HARVEY WEINSTEIN AT ITS OUTSET
5  AND THE APPARENT IMPLSION OF HIS FILM COMPANY AT THE END AND IT WAS SHAPED BY
6  THE EMERGENCE OF METOO TIMES UP BLACKGOWN POLITICS ARMCANDY ACTIVISM AND
7  A NATIONAL CONVERSATION AS BRIEF AND MAD AS A FEVER DREAM ABOUT WHETHER THERE
8  OUGHT TO BE A PRESIDENT WINFREY THE SEASON DIDNT JUST SEEM EXTRA LONG IT WAS
9  EXTRA LONG BECAUSE THE OSCARS WERE MOVED TO THE FIRST WEEKEND IN MARCH TO
10 AVOID CONFLICTING WITH THE CLOSING CEREMONY OF THE WINTER OLYMPICS THANKS
11 PYEONGCHANG
12
13 ONE BIG QUESTION SURROUNDING THIS YEARS ACADEMY AWARDS IS HOW OR IF THE
14 CEREMONY WILL ADDRESS METOO ESPECIALLY AFTER THE GOLDEN GLOBES WHICH BECAME
15 A JUBILANT COMINGOUT PARTY FOR TIMES UP THE MOVEMENT SPEARHEADED BY
16 POWERFUL HOLLYWOOD WOMEN WHO HELPED RAISE MILLIONS OF DOLLARS TO FIGHT SEXUAL
17 HARASSMENT AROUND THE COUNTRY
18
19 SIGNALING THEIR SUPPORT GOLDEN GLOBES ATTENDEES SWATHED THEMSELVES IN BLACK
20 SPORTED LAPEL PINS AND SOUNDED OFF ABOUT SEXIST POWER IMBALANCES FROM THE RED
21 CARPET AND THE STAGE ON THE AIR E WAS CALLED OUT ABOUT PAY INEQUITY AFTER
22 ITS FORMER ANCHOR CATT SADLER QUIT ONCE SHE LEARNED THAT SHE WAS MAKING FAR
23 LESS THAN A MALE COHOST AND DURING THE CEREMONY NATALIE PORTMAN TOOK A BLUNT
24 AND SATISFYING DIG AT THE ALLMALE ROSTER OF NOMINATED DIRECTORS HOW COULD
25 THAT BE TOPPED
26
27 AS IT TURNS OUT AT LEAST IN TERMS OF THE OSCARS IT PROBABLY WONT BE
28
29 WOMEN INVOLVED IN TIMES UP SAID THAT ALTHOUGH THE GLOBES SIGNIFIED THE
30 INITIATIVES LAUNCH THEY NEVER INTENDED IT TO BE JUST AN AWARDS SEASON
31 CAMPAIGN OR ONE THAT BECAME ASSOCIATED ONLY WITH REDCARPET ACTIONS INSTEAD
32 A SPOKESWOMAN SAID THE GROUP IS WORKING BEHIND CLOSED DOORS AND HAS SINCE
33 AMASSED MILLION FOR ITS LEGAL DEFENSE FUND WHICH AFTER THE GLOBES WAS
34 FLOODED WITH THOUSANDS OF DONATIONS OF OR LESS FROM PEOPLE IN SOME
35 COUNTRIES
36
37
38 NO CALL TO WEAR BLACK GOWNS WENT OUT IN ADVANCE OF THE OSCARS THOUGH THE
```

bash - analiza\_frecventei\_birlutiu

[04/08/23]seed@VM:~/.../analiza\_frecventei\_birlutius tr 'vgapnbrtmusicxejhqyzflkdw' 'ABCDEFGHijklMNOPQRSTUVWXYZ' < cipher.txt > out2.txt

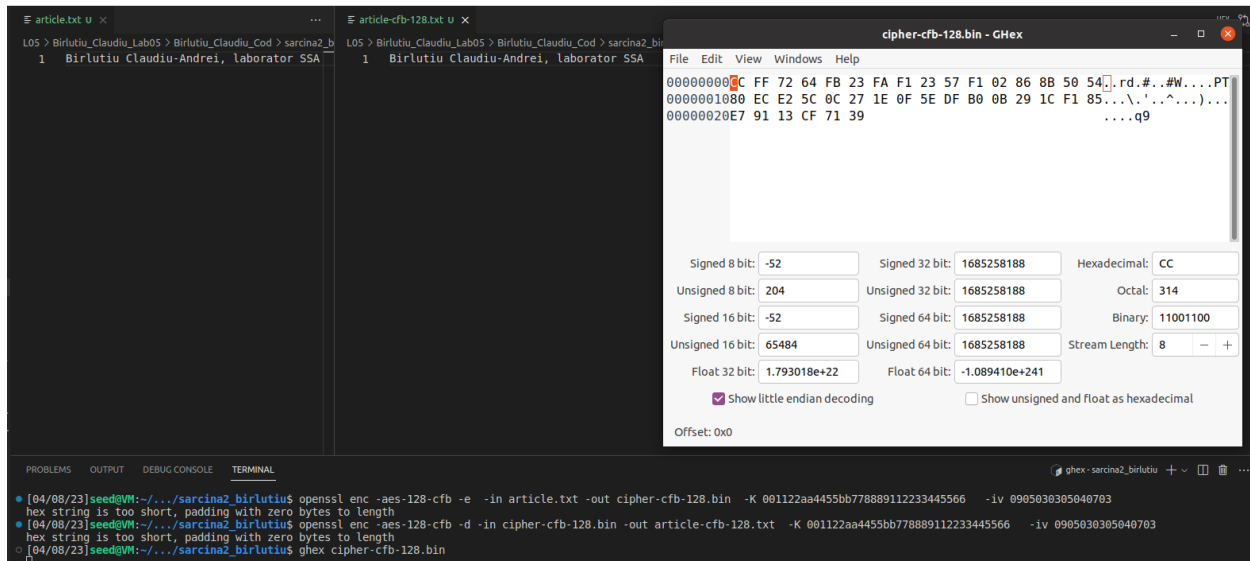
## Sarcina 2: Criptarea cu diferite cifruri și în diverse moduri

- am încercat următoarele comenzi de cifrare a textului article.txt
  - openssl enc -aes-128-cbc -e -in article.txt -out cipher-cbc-128.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708**
  - comanda cripteaza fisierul "article.txt" folosind cifrul AES cu o cheie de 128 de biti si modul de operare CBC, generand fisierul criptat "cipher.bin".
  - parametrul "-K" specifica cheia de criptare, in format hexazecimal. In acest caz, cheia specificata este "00112233445566778889aabbccddeeff".
  - parametrul "-iv" specifica vectorul de initializare (IV), in format hexazecimal. IV-ul este un parametru necesar pentru modurile de operare CBC si OFB. In acest caz, IV-ul specificat este "0102030405060708"



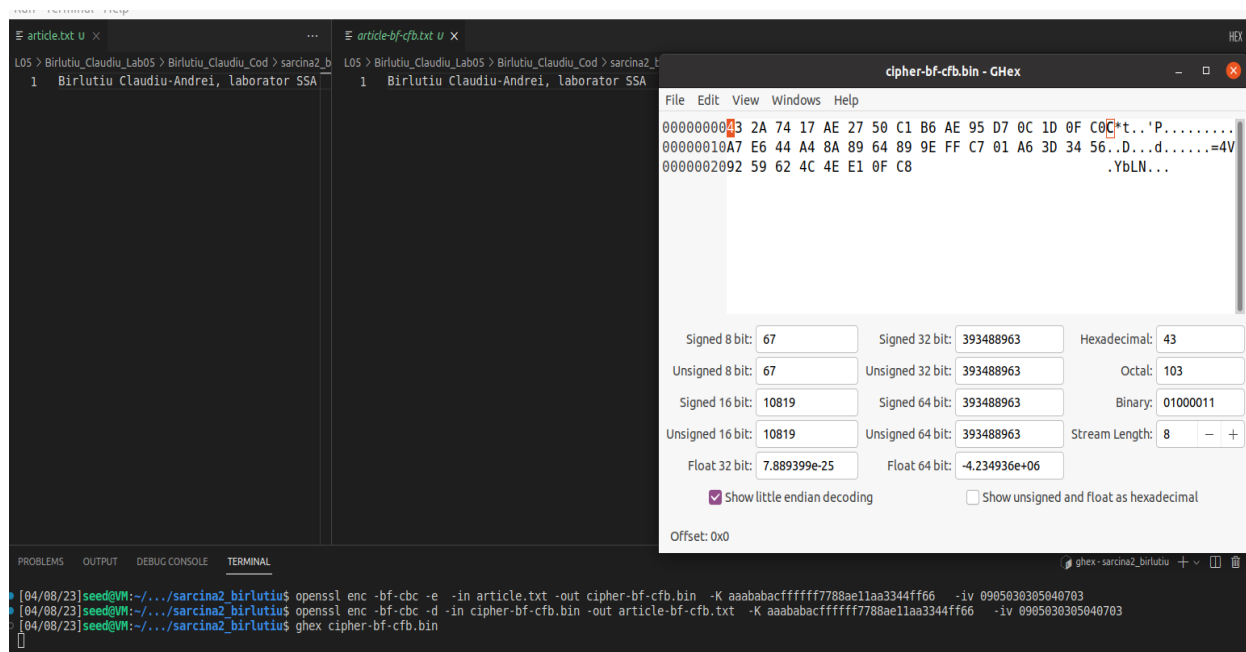
- am decriptat de asemea fișierul: **openssl enc -aes-128-cbc -e -in cipher-cbc-128.bin -out aritcle-cbc-128.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708**

- openssl enc **-aes-128-cfb** -e -in article.txt -out cipher-cfb-128.bin -K 001122aa4455bb778889112233445566 -iv 0905030305040703
- openssl enc **-aes-128-cfb** -d -in cipher-cfb-128.bin -out article-cfb-128.txt -K 001122aa4455bb778889112233445566 -iv 0905030305040703



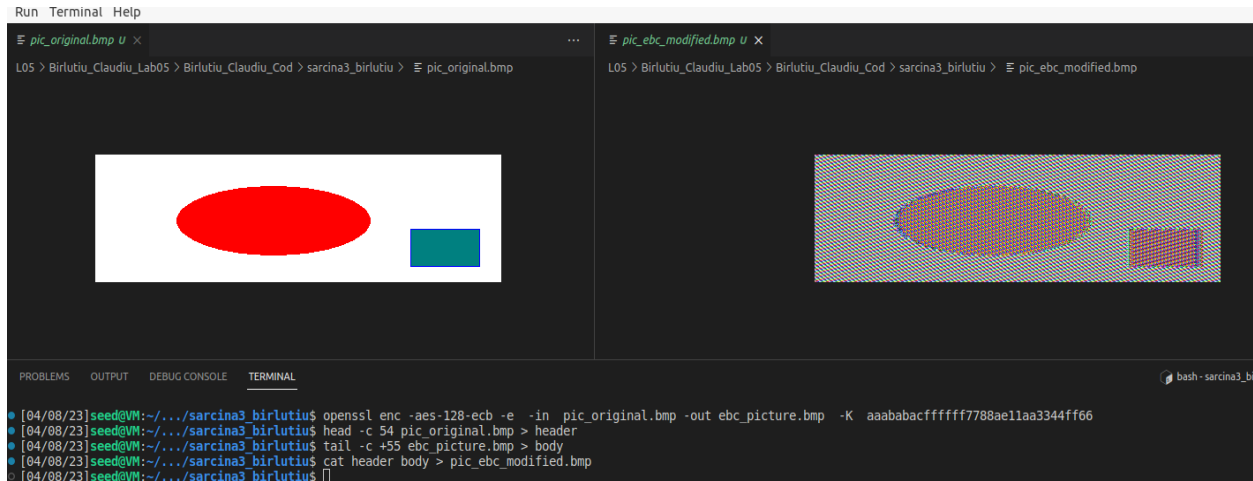
- openssl enc **-bf-cbc** -e -in article.txt -out cipher-bf-cfb.bin -K aaababacffffff7788ae11aa3344ff66 -iv 0905030305040703
- openssl enc **-bf-cbc** -d -in cipher-bf-cfb.bin -out article-bf-cfb.txt -K aaababacffffff7788ae11aa3344ff66 -iv 0905030305040703



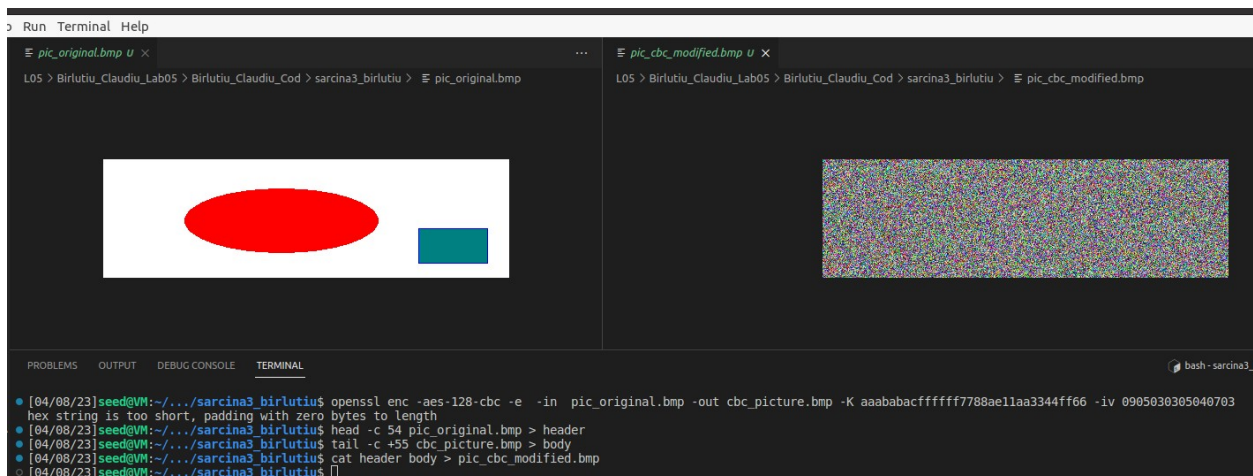


## Sarcina 3: Modul de criptare -- ECB vs. CBC

- In continuare vom encodifica o imagine cu extensia .bmp cu cele 2 modalitati de criptare ECB so CBC si vom modifica headerul imaginii criptate pentru a putea fi vizulizata cu ajutorul comenzilor afisate in alborator;
- din docmunenatre am gasit ca:
  - modul ECB:** funcționează prin criptarea fiecărui bloc de date BMP separat, utilizând aceeași cheie de criptare pentru fiecare bloc astfel ca blocurile de date identice vor fi criptate în același mod, ceea ce poate face ca datele criptate să fie vulnerabile
  - modul CBC:** funcționează prin criptarea fiecărui bloc BMP utilizând cheia de criptare, dar combină rezultatele criptării cu blocul BMP anterior prin aplicarea unei operații de XOR →datele criptate să fie mai sigure, deoarece blocurile identice de date vor fi criptate în moduri diferite
- openssl enc **-aes-128-ecb** -e -in pic\_original.bmp -out ebc\_picture.bmp -K aaababacffffff7788ae11aa3344ff66
  - Se vor rula urmatoarele comenzi pentru ca fisierul ebc\_picture.bmp sa fie tratat ca si un fisier bmp.
    - head -c 54 pic\_original.bmp > header
    - tail -c +55 ebc\_picture.bmp > body
    - cat header body > pic\_ebc\_modified.bmp



- se observa cum blocurile de date identice se cripteaza la fel si se pt distinge formele obiectelor
- openssl enc **-aes-128-cbc** -e -in pic\_original.bmp -out cbc\_picture.bmp -K aaababacffffff7788ae11aa3344ff66 -iv 0905030305040703
  - Se vor rula urmatoarele comenzi pentru ca fisierul ebc\_picture.bmp sa fie tratat ca si un fisier bmp.
    - head -c 54 pic\_original.bmp > header
    - tail -c +55 cbc\_picture.bmp > body
    - cat header body > pic\_cbc\_modified.bmp

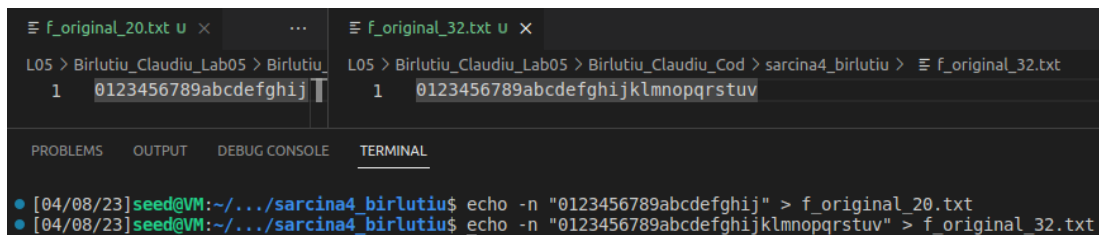


- in cazul encodarii cu CBC se observa ca nu se disting formele din imaginea initiala, deci gradul de securitate e mai ridicat, cee ce demonstreaza ca CBC reduce vulnerabilitatea descifrarii mesajelor

## ***Sarcina 4 : Caractere de completare pentru textul în clar***

### **4.1. Care sunt caracterele de completare în cifrarea AES atunci când lungimea textului în clar este 20 octeți și 32 octeți**

- Am ales ca și studiu de caz modul **CBC** pentru criptare, care este un mod care introduce caractere complementare și vom observa acest lucru prin executarea pașilor ce s-au menționat în lucrarea de laborator și vom folosi **bleess** editor pentru a observa caracterele introduse în plus
- cream un fișier cu 20 de caractere și unul cu 32 de caractere (octeți)

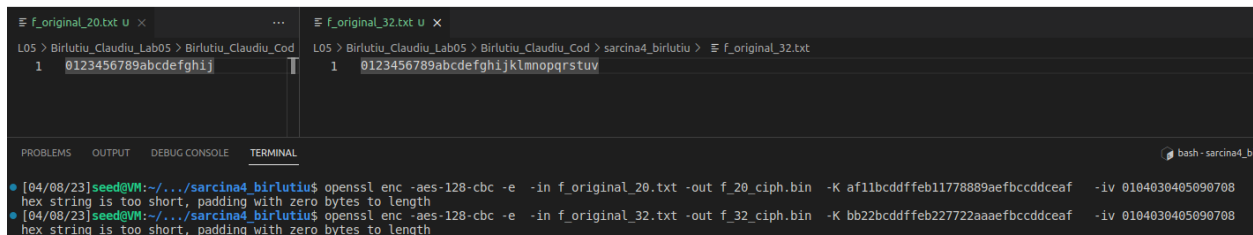


```

f_original_20.txt  f_original_32.txt
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_...  L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina4_birlutiu > f_original_32.txt
1 0123456789abcdefghij 1 0123456789abcdefghijklmnopqrstuv
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[04/08/23] seed@VM:~/.../sarcina4_birlutiu$ echo -n "0123456789abcdefghij" > f_original_20.txt
[04/08/23] seed@VM:~/.../sarcina4_birlutiu$ echo -n "0123456789abcdefghijklmnopqrstuv" > f_original_32.txt

```

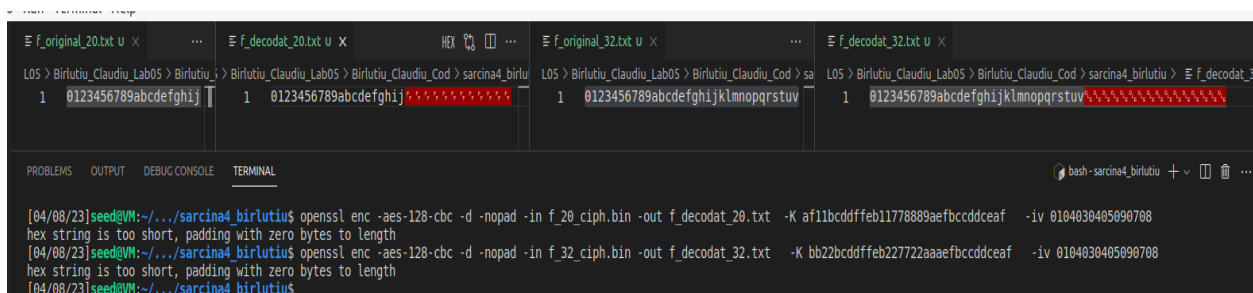
- vom encodifica cele 2 fișiere cu modul **cbc aes** pe 128 de biti:
  - `openssl enc -aes-128-cbc -e -in f_original_20.txt -out f_20_ciph.bin -K af11bcdfffeb11778889aefbccddceaf -iv 0104030405090708`
  - `openssl enc -aes-128-cbc -e -in f_original_32.txt -out f_32_ciph.bin -K bb22bcdfffeb227722aaaefbccddceaf -iv 0104030405090708`



```
f_original_20.txt u x ... f_original_32.txt u x
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina4_birlutiu > f_original_20.txt
1 0123456789abcdefghij
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina4_birlutiu > f_original_32.txt
1 0123456789abcdefghijklmnopqrstuv

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[04/08/23]seed@VM:~/.../sarcina4_birlutiu$ openssl enc -aes-128-cbc -e -in f_original_20.txt -out f_20_ciph.bin -K af11bcdffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../sarcina4_birlutiu$ openssl enc -aes-128-cbc -e -in f_original_32.txt -out f_32_ciph.bin -K bb22bcdffeb227722aaafbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
```

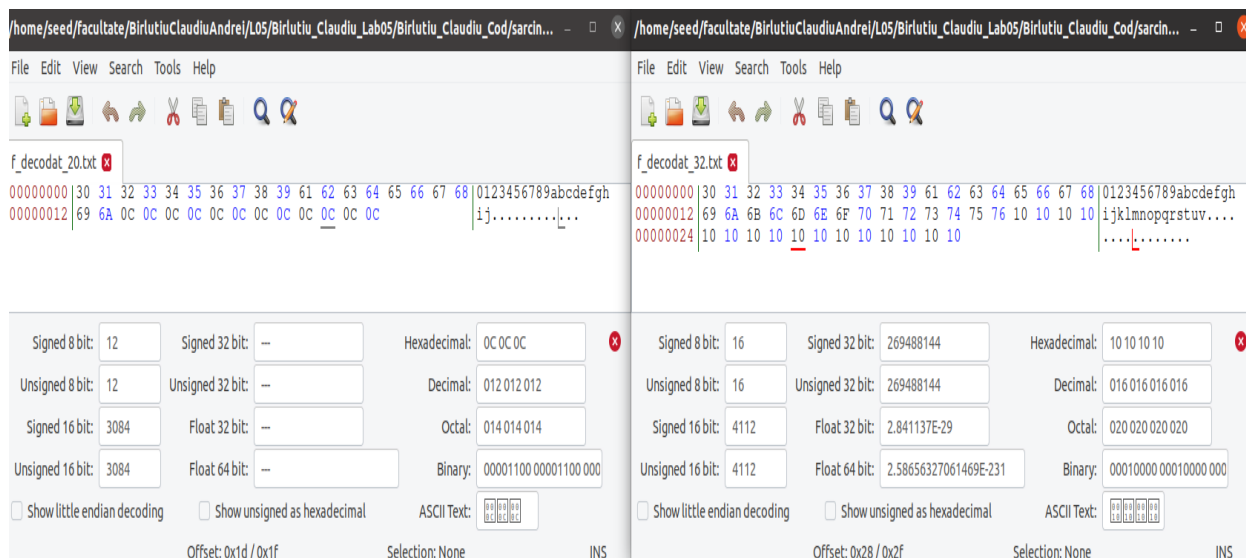
- vom decoda cele 2 fisier binare cu optiunea **nopad** astfel încât caracterele adaugate în plus se vor vedea în fișerele noi decodate
  - openssl enc -aes-128-cbc -d -nopad -in f\_20\_ciph.bin -out f\_decodat\_20.txt -K af11bcdffeb11778889aefbccddceaf -iv 0104030405090708
  - openssl enc -aes-128-cbc -d -nopad -in f\_32\_ciph.bin -out f\_decodat\_32.txt -K bb22bcdffeb227722aaafbccddceaf -iv 0104030405090708
- în imaginea de mai jos se observa cum au fost adaugate caractere complementare la final; acestea nu sunt afisabile, dar putem sa le vizualizam cu un editor de text cum ar fi **bless**



```
f_original_20.txt u x ... f_decodat_20.txt u x HEX ... f_original_32.txt u x ... f_decodat_32.txt u x
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina4_birlutiu > f_decodat_20.txt
1 0123456789abcdefghijxxxxxxxxxxxx
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina4_birlutiu > f_decodat_32.txt
1 0123456789abcdefghijklmnopqrstuvxxxxxxxxxxxxxxxxxxxxxxxxxxxx

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[04/08/23]seed@VM:~/.../sarcina4_birlutiu$ openssl enc -aes-128-cbc -d -nopad -in f_20_ciph.bin -out f_decodat_20.txt -K af11bcdffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../sarcina4_birlutiu$ openssl enc -aes-128-cbc -d -nopad -in f_32_ciph.bin -out f_decodat_32.txt -K bb22bcdffeb227722aaafbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../sarcina4_birlutiu$
```

- am examinat prin intermediul tool-ului **bless** ce caractere s-au adaugat în plus și am observat ce în cadrul fișierului cu 20 de caractere s-a adaugat octetul 0x0C(form feed) de 12 ori => 32 de octeti, iar pentru cel de 32 de caractere s-a pus 0x10 (linefeed) de 16 ori => 48 de octeti; astfel numărul de octeti a devenit **multiplu** de 8



## 4.1. Considerați modurile ECB, CBC, CFB, și OFB de cifrare a unui fișier. Care au caractere de completare și care nu au?

- Din documentare am aflat ca modurile **ECB** și **CBC** folosesc o dimensiune fixă a blocului de date pentru criptare și de aceea vor utiliza caractere de completare. Datele trebuie să fie în mod explicit completate cu caractere suplimentare, astfel încât să poată fi împărțite în blocuri egale.
- În cazul modurilor **CFB** și **OFB** permit o dimensiune variabilă a blocului de date și astfel nu este necesară completarea datelor -> aceste moduri de criptare crijtează o secvență de biți și generează un flux de ieșire pentru a fi combinat cu datele de intrare în loc să cripteze blocuri de date întregi
  - EBC**
    - `openssl enc -aes-128-ecb -e -in f_original_20.txt -out f_20_ciph_ecb.bin -K af11bcddffeb11778889aefbccddceaf`

- `openssl enc -aes-128-ecb -d -nopad -in f_20_ciph_ecb.bin -out f_decodat_20_ecb.txt -K af11bcddffeb11778889aefbccddceaf`
- **CBC:**
  - `openssl enc -aes-128-cbc -e -in f_original_20.txt -out f_20_ciph_cbc.bin -K af11bcddffeb11778889aefbccddceaf -iv 0104030405090708`
  - `openssl enc -aes-128-cbc -d -nopad -in f_20_ciph_cbc.bin -out f_decodat_20_cbc.txt -K af11bcddffeb11778889aefbccddceaf -iv 0104030405090708`
- **OFB**
  - `openssl enc -aes-128-ofb -e -in f_original_20.txt -out f_20_ciph_ofb.bin -K af11bcddffeb11778889aefbccddceaf -iv 0104030405090708`
  - `openssl enc -aes-128-ofb -d -nopad -in f_20_ciph_ofb.bin -out f_decodat_20_ofb.txt -K af11bcddffeb11778889aefbccddceaf -iv 0104030405090708`
- **CFB:**
  - `openssl enc -aes-128-cfb -e -in f_original_20.txt -out f_20_ciph_cfb.bin -K af11bcddffeb11778889aefbccddceaf -iv 0104030405090708`
  - `openssl enc -aes-128-cfb -d -nopad -in f_20_ciph_cfb.bin -out f_decodat_20_cfb.txt -K af11bcddffeb11778889aefbccddceaf -iv 0104030405090708`
- se poate observa n următoarea imagine care dintre moduri aduaga caractere complementare

## Securitatea sistemelor și a aplicațiilor

o Run Terminal Help

```
f_original_20.txt U X ... f_decodat_20_cbc.txt U X ... f_decodat_20_ecb.txt U X ... f_decodat_20_cfb.txt U X ... f_decodat_20_ofb.txt U X ...
LO5 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Codiu_Cod > sarcina4_birlutiu > 2 > f_decoda LO5 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina4_birlutiu > 2 > f_decodat_20_cfb.txt LO5 > Birlutiu_Claudiu_Lab05 > Birlutiu_C
1 0123456789abcdefg hij 1 0123456789abcdefg hij 1 0123456789abcdefg hij 1 0123456789abcdefg hij 1 0123456789abcdefg hij

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL bash -2 + v ...

[04/08/23]seed@VM:~/.../2$ openssl enc -aes-128-ecb -e -in f_original_20.txt -out f_20_ciph_ecb.bin -K af11bcd0ffeb11778889aefbccddceaf
[04/08/23]seed@VM:~/.../2$ openssl enc -aes-128-ecb -d -nopad -in f_20_ciph_ecb.bin -out f_decodat_20_ecb.txt -K af11bcd0ffeb11778889aefbccddceaf
[04/08/23]seed@VM:~/.../2$ openssl enc -aes-128-cbc -e -in f_original_20.txt -out f_20_ciph_cbc.bin -K af11bcd0ffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../2$ openssl enc -aes-128-cbc -d -nopad -in f_20_ciph_cbc.bin -out f_decodat_20_cbc.txt -K af11bcd0ffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../2$ openssl enc -aes-128-ofb -e -in f_original_20.txt -out f_20_ciph_ofb.bin -K af11bcd0ffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../2$ openssl enc -aes-128-ofb -d -nopad -in f_20_ciph_ofb.bin -out f_decodat_20_ofb.txt -K af11bcd0ffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../2$ openssl enc -aes-128-cfb -e -in f_original_20.txt -out f_20_ciph_cfb.bin -K af11bcd0ffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../2$ openssl enc -aes-128-cfb -d -nopad -in f_20_ciph_cfb.bin -out f_decodat_20_cfb.txt -K af11bcd0ffeb11778889aefbccddceaf -iv 0104030405090708
Extra arguments given.
enc: Use -help for summary.
[04/08/23]seed@VM:~/.../2$ openssl enc -aes-128-cfb -d -nopad -in f_20_ciph_cfb.bin -out f_decodat_20_cfb.txt -K af11bcd0ffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../2$
```

## Sarcina 5: Propagarea erorilor -- Text cifrat alterat

### 5.1. Câtă informație puteți recupera din descifrarea fișierului alterat, dacă modul de cifrare a fost ECB, CBC, CFB, respectiv OFB?

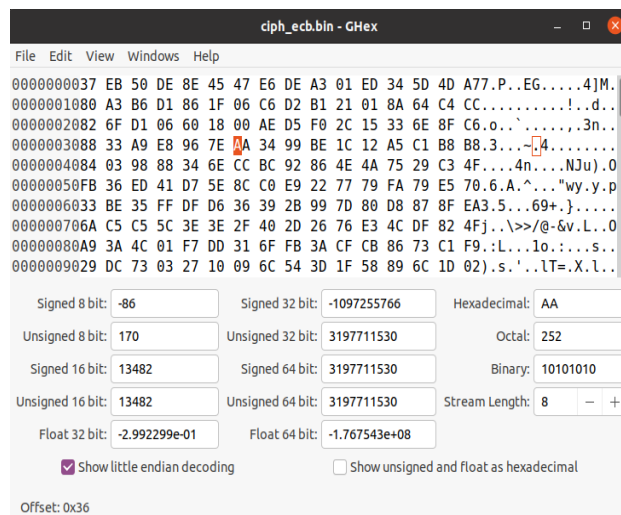
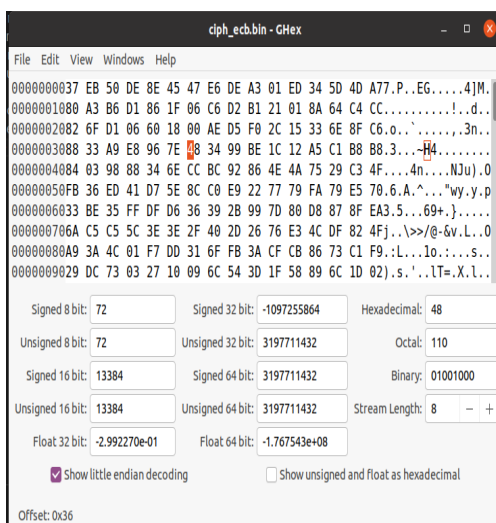
- În cazul ECB o parte desctul de semnificativa.

### 5.2. Răspunsul dat înainte de efectuarea sarcinii

- Ma gândesc, ca doar în cadrul ECB putem recupera mare parte din informație deoarece codificare se face la nivelul fiecarui bloc și nu deinde de alte blocuri cum este în cazul celorlaltor modele de criptare. În cazul ECB acest lucru poate face ca modificările într-un bloc să nu afecteze blocurile ulterioare, ceea ce poate permite recuperarea unor părți semnificative ale informației originale din fișierul alterat. De asemenea cred ca și în cadrul OFB putem recupera date

### 5.3. Răspunsul după execuția acestei sarcini.

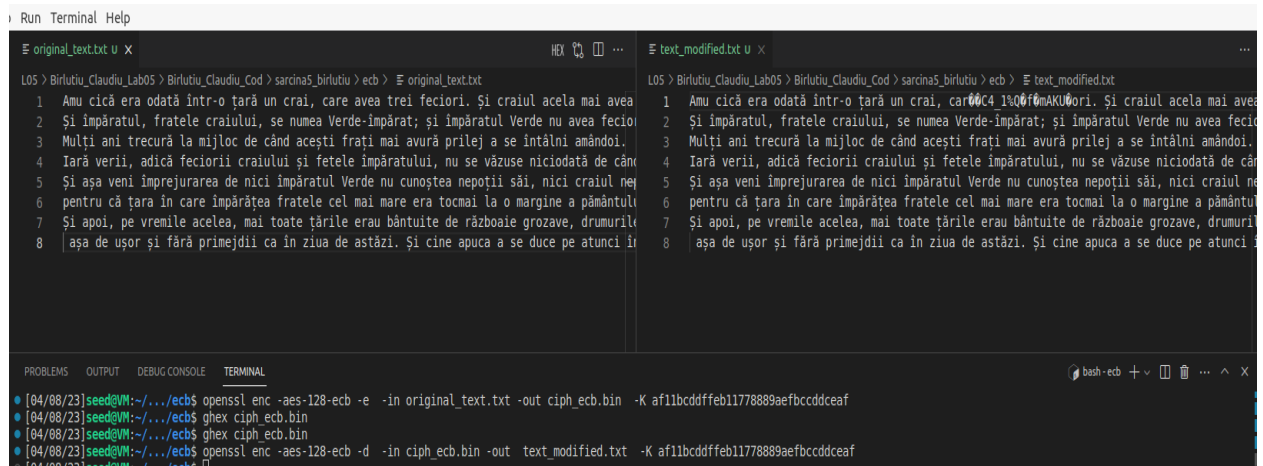
- Voi face testul pe o imagine: aceasta are mai mult de 1000 de octeti si vom observa mai bine modificarile la nivelul blocurilor
- 55 in hexa e 0x37 => trebuie moidifcat octetul de la adresa 0x36 si vom folosi tool-ul **Ghex**
- ECB:**
  - openssl enc -aes-128-ecb -e -in original\_text.txt -out ciph\_ecb.bin -K af11bccddffeb11778889aefbccddceaf
  - modificam un byte-ul de la adresa 0x36 din **0x48** în **0xAA**





## Securitatea sistemelor și a aplicațiilor

- decodificam: openssl enc -aes-128-ecb -d -in ciph\_ecb.bin -out text\_modified.txt -K af11bccddffeb11778889aefbccddceaf



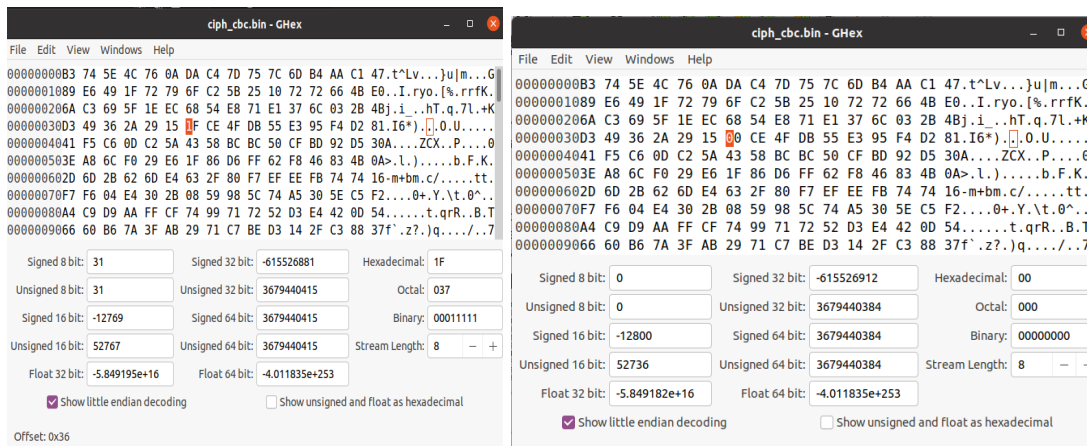
```
Run Terminal Help
original_text.txt x
1 Amu cica era odata intr-o tara un crai, care avea trei feciori. Si craiul acela mai avea
2 Si imparatul, fratele craiului, se numea Verde-imparat; si imparatul Verde nu avea feci
3 Multi ani trecura la mijloc de cand acesti frati mai avura prilej a se intalni amandoi.
4 Iara verii, adica feciorii craiului si fetele imparatului, nu se vazuse niciodata de car
5 Si asa veni imprejurarea de nici imparatul Verde nu cunoastea nepotii sai, nici craiul ne
6 pentru ca tara in care imparatea fratele cel mai mare era tocmai la o margine a pamantulu
7 Si apoi, pe vremile acelea, mai toate tarile erau bantuite de razboaie grozave, drumuril
8 asa de usor si fara primejdii ca in ziua de astazi. Si cine apuca a se duce pe atunci in

text_modified.txt x
1 Amu cica era odata intr-o tara un crai, care avea trei feciori. Si craiul acela mai avea
2 Si imparatul, fratele craiului, se numea Verde-imparat; si imparatul Verde nu avea feci
3 Multi ani trecura la mijloc de cand acesti frati mai avura prilej a se intalni amandoi.
4 Iara verii, adica feciorii craiului si fetele imparatului, nu se vazuse niciodata de car
5 Si asa veni imprejurarea de nici imparatul Verde nu cunoastea nepotii sai, nici craiul ne
6 pentru ca tara in care imparatea fratele cel mai mare era tocmai la o margine a pamantulu
7 Si apoi, pe vremile acelea, mai toate tarile erau bantuite de razboaie grozave, drumuril
8 asa de usor si fara primejdii ca in ziua de astazi. Si cine apuca a se duce pe atunci in
```

- vom încerca descifrarea mesajului sa vedem cât de corupt e mesajul și am observat ca textul decodata nu suferă modificari majore, doar într-un singur loc foarte puține caractere au fost pierdute

### • CBC:

- openssl enc -aes-128-cbc -e -in original\_text.txt -out ciph\_cbc.bin -K af11bccddffeb11778889aefbccddceaf -iv 0104030405090708
- modificam byte-ul de la adresa 0x36 din 0x1F în 0x00



The left screenshot shows the GHex editor with the file 'ciph\_cbc.bin'. The data is displayed in hexadecimal and ASCII. At offset 0x36, the byte is 0x1F. The right screenshot shows the same file after modification. At offset 0x36, the byte has been changed to 0x00. Below the hex view, there are conversion fields for Signed, Unsigned, Signed 32-bit, Unsigned 32-bit, Hexadecimal, Octal, Signed 16-bit, Unsigned 16-bit, Signed 64-bit, Unsigned 64-bit, Binary, Stream Length, Float 32-bit, and Float 64-bit. The 'Show little endian decoding' checkbox is checked.

- decodema princ comanda: openssl enc -aes-128-cbc -d -in ciph\_cbc.bin -out text\_decodat.txt -K af11bccddffeb11778889aefbccddceaf -iv 0104030405090708

## Securitatea sistemelor și a aplicațiilor

- observam ca numărul de caractere pierdute e mai mare



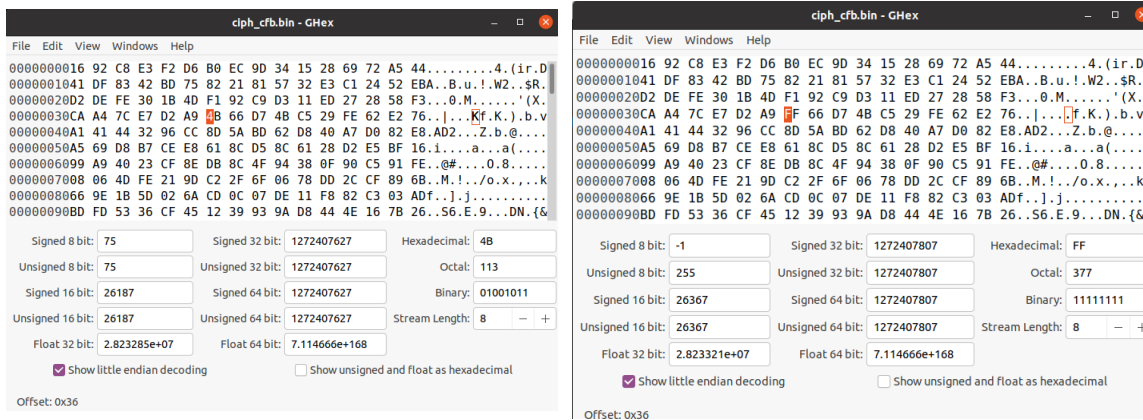
```
original_text.txt x
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarchina5_birlutiu > cbc > E original_text.txt
1 Amu, cica era odata intr-o tara un crai, care avea trei feciori. Si craiul acela mai avea
2 Si imparatul, fratele craiului, se numea Verde-imparat; si imparatul Verde nu avea feci
3 Multi ani trecura la mijloc de cand acesti frati mai avura prilej a se intalni amandoi.
4 Taru verii, adica feciorii craiului si fetele imparatului, nu se vazuse niciodata de c
5 Si asa veni imprejurarea de nici imparatul Verde nu cunoastea nepotii sai, nici craiul ne
6 pentru ca tara in care imparatea fratele cel mai mare era tocmai la o margine a pamantulu
7 Si apoi, pe vremile acelea, mai toate tarile erau bantuie de razboaie grozave, drumuri
8 asa de usor si fara primejdii ca in ziua de astazi. Si cine apuca a se duce pe atunci in

text_decodat.txt M x
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarchina5_birlutiu > cbc > E text_decodat.txt
1 Amu, cica era odata intr-o tara un crai, care avea trei feciori. Si craiul acela mai avea
2 Si imparatul, fratele craiului, se numea Verde-imparat; si imparatul Verde nu avea feci
3 Multi ani trecura la mijloc de cand acesti frati mai avura prilej a se intalni amandoi.
4 Taru verii, adica feciorii craiului si fetele imparatului, nu se vazuse niciodata de c
5 Si asa veni imprejurarea de nici imparatul Verde nu cunoastea nepotii sai, nici craiul ne
6 pentru ca tara in care imparatea fratele cel mai mare era tocmai la o margine a pamantulu
7 Si apoi, pe vremile acelea, mai toate tarile erau bantuie de razboaie grozave, drumuri
8 asa de usor si fara primejdii ca in ziua de astazi. Si cine apuca a se duce pe atunci in

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[04/08/23]seed@VM:~/.../cbc$ openssl enc -aes-128-ecb -d -in ciph_ecb.bin -out pic_modified.bmp -K af11bccddffeb11778889aefbccddceaf
Can't open ciph_ecb.bin for reading: No such file or directory
140697844172096:error:02001002:system library:open:No such file or directory:crypto/bio/bss_file.c:69: (Open('ciph_ecb.bin','rb'))
140697844172096:error:20060080:BIO routines:BIO_new_file:No such file:crypto/bio/bss_file.c:76:
[04/08/23]seed@VM:~/.../cbc$ openssl enc -aes-128-ecb -e -in original_text.txt -out ciph_ecb.bin -K af11bccddffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../cbc$ ghex ciph_ecb.bin
[04/08/23]seed@VM:~/.../cbc$ openssl enc -aes-128-ecb -d -in ciph_ecb.bin -out text_decodat.txt -K af11bccddffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../cbc$
```

### • CFB:

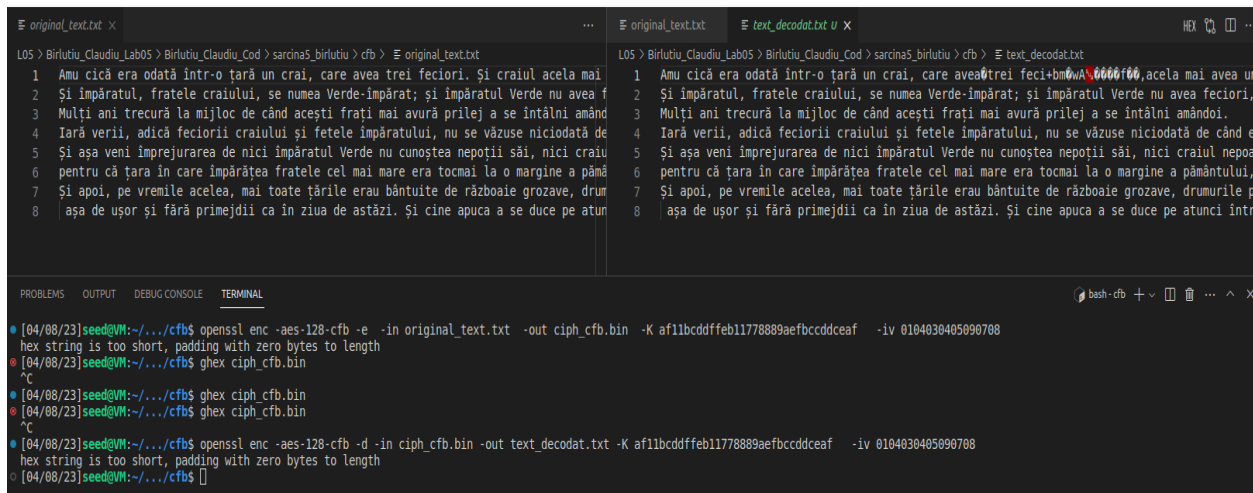
- openssl enc -aes-128-cfb -e -in original\_text.txt -out ciph\_cfb.bin -K af11bccddffeb11778889aefbccddceaf -iv 0104030405090708
- modificam byte-ul de la adresa 0x36 din 0x4B în 0xFF



The left screenshot shows the GHex hex editor with the file 'ciph\_cfb.bin' open. The hex data is displayed in a grid. At offset 0x36, the byte 0x4B is highlighted. The right screenshot shows the same file after modification. The byte at offset 0x36 has been changed to 0xFF. The hex editor interface includes fields for signed/unsigned 8, 16, 32, and 64 bit values, as well as float values, and checkboxes for 'Show little endian decoding' and 'Show unsigned and float as hexadecimal'.

- decodema princ comanda: openssl enc -aes-128-cfb -d -in ciph\_cfb.bin -out text\_decodat.txt -K af11bccddffeb11778889aefbccddceaf -iv 0104030405090708
- obsservam și în czaul acesta sunt mai multe caractere pierdute

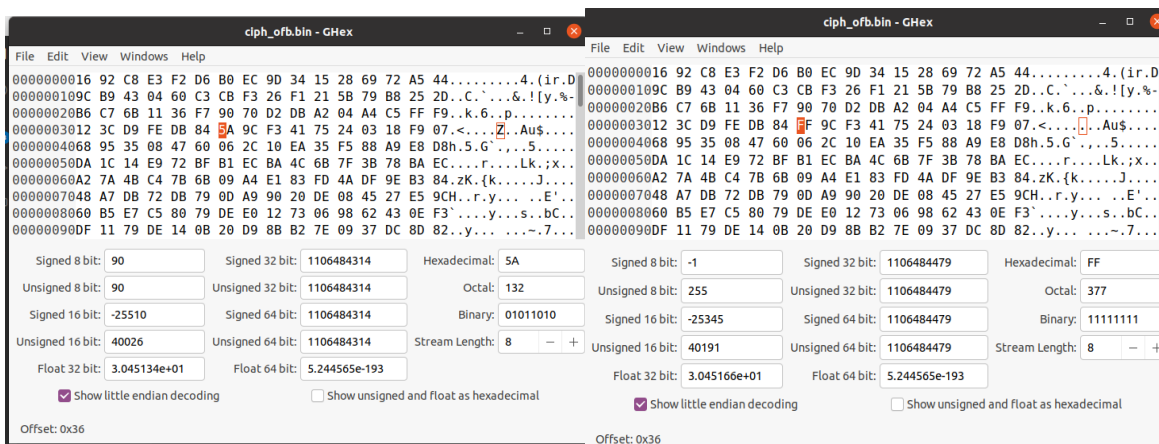
## Securitatea sistemelor și a aplicațiilor



```
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina5_birlutiu > cfb > original_text.txt
1 Amu cica era odata intr-o tara un crai, care avea trei feciori. Si craiul acela mai
2 Si imparatul, fratele craiului, se numea Verde-imparat; si imparatul Verde nu avea f
3 Multi ani trecura la mijloc de cand acesti frati mai avura prilej a se intalni amand
4 Iara verii, adica feciorii craiului si fetele imparatului, nu se vazuse niciodata de
5 Si asa veni imprejurarea de nici imparatul Verde nu cunoastea nepotii sai, nici crai
6 pentru ca tara in care imparatea fratele cel mai mare era tocmai la o margine a pama
7 Si apoi, pe vremile acelea, mai toate tarile erau bantuite de razboaie grozave, drum
8 asa de usor si fara primejdii ca in ziua de astazi. Si cine apuca a se duce pe atun

L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina5_birlutiu > cfb > text_decodat.txt
1 Amu cica era odata intr-o tara un crai, care avea trei feciori. Si craiul acela mai
2 Si imparatul, fratele craiului, se numea Verde-imparat; si imparatul Verde nu avea feciori,
3 Multi ani trecura la mijloc de cand acesti frati mai avura prilej a se intalni amandoi.
4 Iara verii, adica feciorii craiului si fetele imparatului, nu se vazuse niciodata de cand e
5 Si asa veni imprejurarea de nici imparatul Verde nu cunoastea nepotii sai, nici craiul nepoa
6 pentru ca tara in care imparatea fratele cel mai mare era tocmai la o margine a pamantului,
7 Si apoi, pe vremile acelea, mai toate tarile erau bantuite de razboaie grozave, drumurile p
8 asa de usor si fara primejdii ca in ziua de astazi. Si cine apuca a se duce pe atunci intr
```

- **OFB:**
  - openssl enc -aes-128-ofb -e -in original\_text.txt -out ciph\_ofb.bin -K af11bccddffeb11778889aefbccddceaf -iv 0104030405090708
  - modificam byte-ul de la adresa 0x36 din **0x5A** în **0xFF**



- decodema prin comanda: openssl enc -aes-128-ofb -d -in ciph\_ofb.bin -out text\_decodat.txt -K af11bccddffeb11778889aefbccddceaf -iv 0104030405090708
- observam în ca incadrul modelului OFB avem cea mai mica deterioare

```

Run Terminal Help
original_text.txt u x
Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina5_birlutiu > ofb > original_text.txt
1 Amu cica era odata intr-o tara un crai, care avea trei feciori. Si craiul acela mai avea
2 Si imparatul, fratele craiului, se numea Verde-imparat; si imparatul Verde nu avea fec
3 Multi ani trecura la mijloc de cand acesti frati mai avura prilej a se intalni amandoi.
4 Iara verii, adica feciorii craiului si fetele imparatului, nu se vazuse niciodata de c
5 Si asa veni imprejurarea de nici imparatul Verde nu cunoastea nepotii sai, nici craiul ne
6 pentru ca tara in care imparata fratele cel mai mare era tocmai la o margine a pamantulu
7 Si apoi, pe vremile acelea, mai toate tarile erau bantuite de razboaie grozave, drumur
8 asa de usor si fara primejdii ca in ziua de astazi. Si cine apuca a se duce pe atunci in

text_decodat.txt u x
Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina5_birlutiu > ofb > text_decodat.txt
1 Amu cica era odata intr-o tara un crai, care avea trei feciori. Si craiul acela mai av
2 Si imparatul, fratele craiului, se numea Verde-imparat; si imparatul Verde nu avea fec
3 Multi ani trecura la mijloc de cand acesti frati mai avura prilej a se intalni amandoi
4 Iara verii, adica feciorii craiului si fetele imparatului, nu se vazuse niciodata de c
5 Si asa veni imprejurarea de nici imparatul Verde nu cunoastea nepotii sai, nici craiul
6 pentru ca tara in care imparata fratele cel mai mare era tocmai la o margine a pamant
7 Si apoi, pe vremile acelea, mai toate tarile erau bantuite de razboaie grozave, drumur
8 asa de usor si fara primejdii ca in ziua de astazi. Si cine apuca a se duce pe atunci in

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[04/08/23]seed@VM:~/.../ofb$ openssl enc -aes-128-ofb -e -in original_text.txt -out ciph_ofb.bin -K afl1bcddffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../ofb$ ghex ciph_ofb.bin
[04/08/23]seed@VM:~/.../ofb$ openssl enc -aes-128-ofb -d -in ciph_ofb.bin -out text_decodat.txt -K afl1bcddffeb11778889aefbccddceaf -iv 0104030405090708
hex string is too short, padding with zero bytes to length
[04/08/23]seed@VM:~/.../ofb$

```

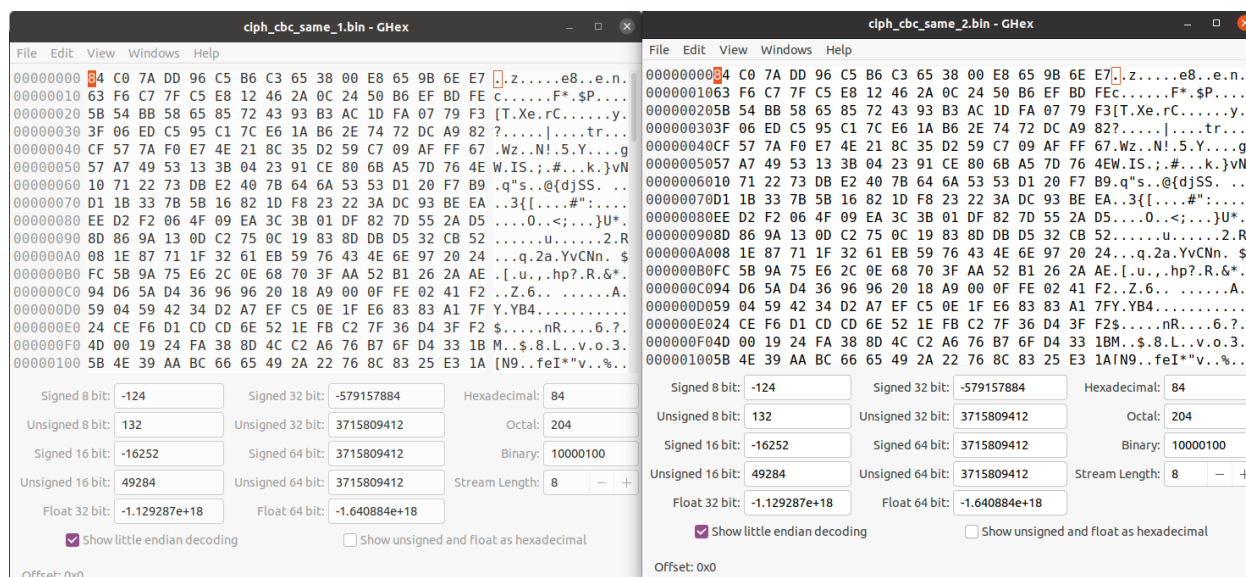
## 5.4. Explicați de ce.

- In cadrul **ECB** putem recupera mare parte din informație deoarece codificare se face la nivelul fiecarui bloc și nu deinde de alte blocuri cum este în cazul celorlaltor modele de criptare. În cazul ECB acest lucru poate face ca modificările într-un bloc să nu afecteze blocurile ulterioare, ceea ce poate permite recuperarea unor părți semnificative ale informației originale din fișierul alterat.
- Pentru **CBC** blocurile de date sunt cifrate în funcție de blocurile precedente și este utilizată o valoare de inițializare pentru primul bloc -> interdependență între blocuri (modificările la un bloc afectează blocurile următoare). Mai precis, fiecare bloc de date este combinat cu blocul anterior înainte de a fi criptat prin intermediul unei operații XOR.
- Pentru **CFB** este textul clar este cifrat în blocuri, iar rezultatul cifrat este apoi utilizat pentru a cifra următorul bloc -> aceeași situație ca la CBC, este greu de recuperat datele.
- Pentru **OFB** va permite recuperarea semnificativa de date deoarece un bloc initial este cifrat, iar apoi utilizat pentru cifrarea urmatorului bloc fără utilizare textului clar. Biții de ieșire generați sunt independenți de blocurile de date de intrare și de biții de ieșire anteriori, ceea ce face ca OFB să fie mai rezistent la unele tipuri de atacuri criptografice.

## Sarcina 6. Vectorul inițial (IV) și erori comune

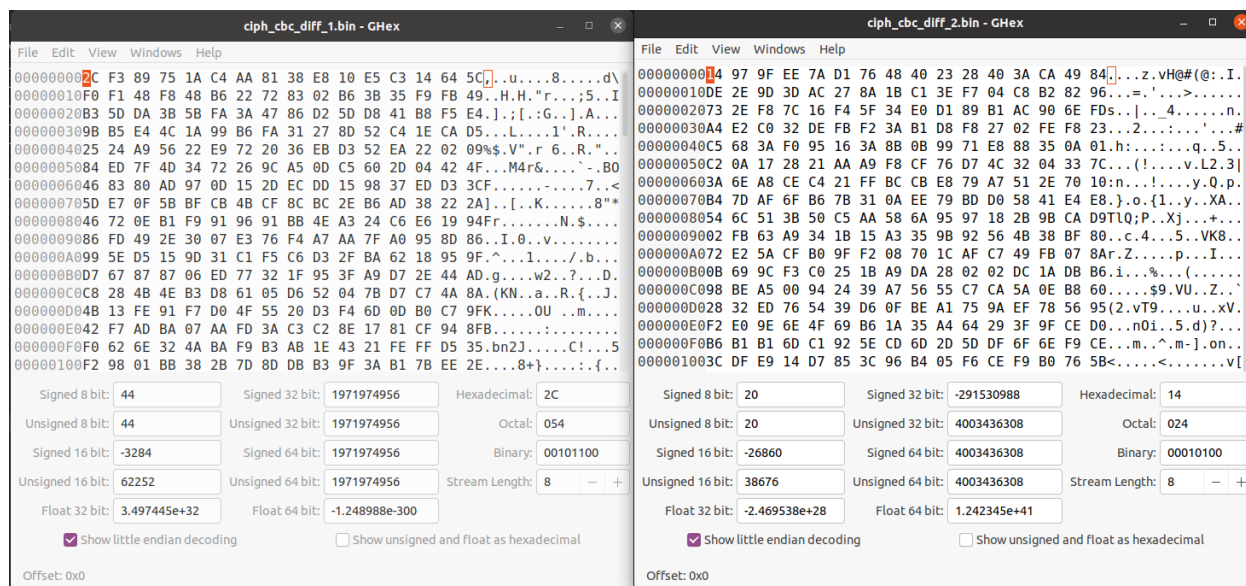
### 6.1 Sarcina 6.1. Unicitatea IV

- La recomandările din laborator vom executa în prima fază pe textul original\_text.txt ce reprezintă o parte din povestea lui Harap Alb, un algoritm de criptare AES modul CBC în 2 cazuri:
  - cifram textul cu doi IV la fel (vom executa comanda de 2 ori):
    - `openssl enc -aes-128-cbc -e -in original_text.txt -out ciph_cbc_smae_x.bin -K af11bccddffeb11778889aefbccddceaf -iv aaafafafbb090708` - unde x va fi 1 și apoi 2



- observăm că cele 2 fișiere binare de codificare sunt la fel
  - cifram textul cu doi IV diferiți (vom executa comanda de 2 ori):
    - `openssl enc -aes-128-cbc -e -in original_text.txt -out ciph_cbc_diff_x.bin -K af11bccddffeb11778889aefbccddceaf -iv 0b0bafafbb0907aa /0c0cafafbb0907ff` - unde x va fi 1 și apoi 2

## Securitatea sistemelor și a aplicațiilor



- observam ca cele 2 fisiere cu mesajul codificat difera chiar dacă au aceasi cheie
- această **unicitate** a IV-ului este importantă deoarece, în caz contrar, ar putea exista mai multe blocuri de date criptate care să folosească aceeași valoare de IV și aceeași cheie de criptare, ceea ce ar putea conduce la dezvăluirea informațiilor secrete din mesajul original -> un atacator ar putea să deducă blocurile de date originale din textul criptat prin analizarea valorilor repetate ale IV.



## 6.2 Sarcina 6.2. Eroare comună: folosirea aceluiași IV

- Având în vedere faptul că cele 2 cifre au fost encodificate folosit același IV - urile înseamnă că **Plain text 1 XOR Plain text 2** va fi egal cu **Cipher text 1 XOR Cipher text 2** - deoarece având aceeași cheie de criptare și același vector de inițializare =>
- $PT2 = CT1 \text{ xor } CT2 \text{ xor } PT1$
- ne vom folosi de programul `sample_code.py` pentru a face operațiile de xor pe caractere

```

L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina6_birlutiu > sarcina6.2_birlutiu > sample_code.py > ...
1  #!/usr/bin/python3
2
3  # XOR two bytearrays
4  def xor(first, second):
5      return bytearray(x^y for x,y in zip(first, second))
6
7  PT1 = "This is a known message!"           #decararea mesajului cunoscut
8  CT1 = "a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159" #cifru în hexa pentru mesajul cunoscut
9  CT2 = "bf73bcd3509299d566c35b5d450337e1bb175f903fafc159" #cifru în hexa pentru mesajul necunoscut
10
11 # Convert ascii string to bytearray
12 P1 = bytes(PT1, 'utf-8')                   #obținerea de bytes din șirul de caractere
13
14 # Convert hex string to bytearray
15 C1 = bytearray.fromhex(CT1)                #obținerea șirului de bytes din valorile hexa
16 C2 = bytearray.fromhex(CT2)
17
18 P2 = xor(P1, xor(C1, C2))                  #aplicarea formulei de determinare a textului necunoscut
19 PT2 = P2.decode('utf-8')                   #decodarea șirului de bytes la string utf-8
20 print(PT2)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

• [04/08/23]seed@VM:~/.../sarcina6.2_birlutiu$ sample_code.py
Order: Launch a missile!
○ [04/08/23]seed@VM:~/.../sarcina6.2_birlutiu$

```

### 6.3 Sarcina 6.3. Eroare comună: folosirea unui IV predictibil

- În prima faza am porint containerul docker cu server-ul de oracle:

```
[04/08/23]seed@VM:~/.../BirlutiuClaudiuAndrei$ cd L05/Birlutiu_Claudiu_Lab05/Birlutiu_Cla
udiu_Cod/
[04/08/23]seed@VM:~/.../Birlutiu_Claudiu_Cod$ dcup
Creating oracle-10.9.0.80 ... done
Attaching to oracle-10.9.0.80
oracle-10.9.0.80 | Server listening on 3000 for known_iv
[]
```

- ca și premisa avem faptul ca Eve știe ca Bob a trims un mesaj Yes or No (nu știe care dintre), da pe lângă aceasta cunoaște cifrul mesajului și IV ul folosit. Pe lângă acestea Eve mai știe și Iv-ul pe care îl va folosi Bob pentru criptarea urmatorului mesaj
- sarcina pe care am auto a fost sa construiesc un mesaj P2 pe care o să-i cerem lui Bob sa îl cifreze și sa dea textul cifrat -> aceasta ocazie am folosit pentru a determina dacă conținutul real al mesajului lui Bob este **Yes** sau **No**
- am făcut logica de calcul în fișierul **sample\_code.py** unde am luat ca plat text P2 ca fiind codul hexa pentru „Yes”
  - am făcut xor între codul acesta hexa (Yes) și primul IV afisat de server (Bob) iar apoi am făcut XOR cu IV-2; valorile se observa în imaginea de mai jos
  - am trimis rezultatul obținut în hexa lui Bob pentru a-l cifra și observam ca cele 2 coduri sunt identice CT-1 și CT-2 (la început) ceea ce demonstrează faptul ca mesajul trimis de Bob a fost yes
  - OBSERVATIE: a trebuit să se facă padding la stringul Yes pentru a avea un numar de octeti multiplu de 16. S-a ADAUGTA **0X0D**



[illegible]

- mai sus se regaseste ss-ul cu scriptul scris

## Sarcina 7: Programarea folosind biblioteca criptografică

- În prima faza am analizat fisierele pe care le aveam în **encryption\_oracle** unde aveam **evp-encrypt.hpp** care conține cele funcții de **encrypt** și **decrypt** cu AES. Am modificat **known\_iv.cpp** astfel încât să îmi citească fiecare cuvânt din **word.txt** și să facă padding cu **#** până la 16 caractere. Am luat aceste cuvinte și le-am pus într-un sir de bytes ce reprezintă cheia de encryptare. Pentru plain text deoarece are 21 de caractere (octeti) am făcut padding până la 32 de octeti pentru a fi multiplu de 16 numărul. => am adăugat astfel **11** octeti de **0x0B** conform regulii PKS.
- Compilarea programului am făcut-o cu **make**
- Pentru fiecare cuvânt luat ca și cheie am apelat funcția de encryptare și am verificat dacă **cipher\_text**-ul dat e conținut în cel rezultat. (am creat o funcție de verificare; iar pentru acea cheie am afișat informațiile relevante

```

G: known_iv.cpp u x
L05 > Birlutiu_Claudiu_Lab05 > Birlutiu_Claudiu_Cod > sarcina7_birlutiu > G: known_iv.cpp > ...

46     }
47
48     //read words from the file
49     std::ifstream inputFile("words.txt");
50     std::string line;
51     // Read each line of the input file
52     while (std::getline(inputFile, line)) {
53         // Iterate over each word in the line
54         std::istringstream iss(line);
55         std::string word;
56         while (iss >> word) {
57             // Add "#" characters to the word until it has a length of 16 characters
58             while (word.length() < 16) {
59                 word += "#";
60             }
61             ///CREARE CHEIE///////////
62             for(unsigned int i=0; i< KEY_SIZE; i++) {
63                 key[i] = word[i];
64             }
65             //OBTIENE CIFRU REZULTAT
66             Bytes ctext1 = aes_encrypt(key.data(), iv.data(), ptext1);
67             //DACA S-A GASIT CHEIA O AFISAM
68             if(contains_cipher_dat(cipher_dat, ctext1)){
69                 // print essential information
70                 std::cout << "Ciphertext rezultat: " << hexlify(ctext1) << endl
71                     << "Ciphertext dat: " << hexlify(cipher_dat) << endl
72                     << "Word key: " << word << endl
73                     << "Plain text: " << hexlify(ptext1) << endl
74                     << "The IV used      : " << hexlify(iv) << endl

```

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
[04/11/23]seed@VM:~/.../sarcina7_birlutiu$ make
mkdir -p build
g++ -O2 -Wall -std=c++11 known_iv.cpp -o ./build/known_iv -lcrypto
[04/11/23]seed@VM:~/.../sarcina7_birlutiu$ cd build/
[04/11/23]seed@VM:~/.../build$ known_iv
Ciphertext rezultat: 764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2f6541f2fd90ff829c3dc96c0cba748b5
Ciphertext dat: 764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2
Word key: Syracuse#####
Plain text: 54686973206973206120746f70207365637265742e0b0b0b0b0b0b0b0b0b0b
The IV used      : aabbccddeeff00998877665544332211
The Key used     : 537972616375736523232323232323232323
[04/11/23]seed@VM:~/.../build$

```

IMPORTANT: CODUL SURSA SE AFLA IN Birlutiu\_Claudiu\_Cod/sarcina7\_birlutiu

```

17 bool contains_cipher_dat(Bytes cipher_dat, Bytes rezultat){
18     if(cipher_dat.size() > rezultat.size()) return false;
19     for(size_t i=0; i<cipher_dat.size(); i++){
20         if(cipher_dat[i] != rezultat[i])
21             return false;
22     }
23     return true;
24 }
25
26 int main(int argc, char const *argv[]){
27     // key, iv1, iv2
28     array<Byte, KEY_SIZE> key;
29     array<Byte, BLOCK_SIZE> iv;
30     ///////////////////////////////////////////////////CREARE IV////////////////////////////////////
31     Bytes iv_bytes= {0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff, 0x00, 0x99, 0x88,
32                     0x77, 0x66, 0x55, 0x44, 0x33, 0x22, 0x11};
33     for (unsigned int i = 0; i < BLOCK_SIZE; i++) {
34         iv[i] = static_cast<Byte>(iv_bytes[i]);
35     }
36     Bytes cipher_dat = {0x76,0x4a,0xa2,0x6b,0x55,0xa4,0xda,0x65,0x4d,0xf6,0xb1,0x9e,0x4b, 0xce,0x00,
37                        0xf4, 0xed, 0x05, 0xe0, 0x93, 0x46, 0xfb, 0x0e,0x76, 0x25, 0x83, 0xcb, 0x7d, 0xa2, 0xac,0x93, 0xa2};
38
39     // plaintext define array of bytes
40     std::string inputString = "This is a top secret.";
41     Bytes ptext1(inputString.begin(), inputString.end());
42     //facem padding cu 11 valori de 0x0B pentru a fi multiplu de 16
43     for(int i=0; i<11; i++){
44         ptext1.emplace_back(0x0B);
45     }

```

```

46
47     //read words from the file
48     std::ifstream inputFile("words.txt");
49     std::string line;
50     // Read each line of the input file
51     while (std::getline(inputFile, line)) {
52         // Iterate over each word in the line
53         std::istringstream iss(line);
54         std::string word;
55         while (iss >> word) {
56             // Add "#" characters to the word until it has a length of 16 characters
57             while (word.length() < 16) {
58                 word += "#";
59             }
60             ///////////////////////////////////////////////////CREARE CHEIE////////////////////////////////////
61             for(unsigned int i=0; i< KEY_SIZE; i++ ){
62                 key[i] = word[i];
63             }
64             //OBTIERE CIFRU REZULTAT
65             Bytes ctext1 = aes_encrypt(key.data(), iv.data(), ptext1);
66             //DACA S-A GASIT CHEIA O AFISAM
67             if(contains_cipher_dat(cipher_dat, ctext1)){
68                 // print essential information
69                 std::cout << "Ciphertex rezultat: " << hexlify(ctext1) << endl
70                             << "Ciphertex dat: " << hexlify(cipher_dat) << endl
71                             << "Word key: " << word << endl
72                             << "Plain text: " << hexlify(ptext1) << endl
73                             << "The IV used      : " << hexlify(iv) << endl
74                             << "The Key used      : " << hexlify(key) << endl;
75             }
76         }
77     }
78     return 0;

```

```
#include <array>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <algorithm>
#include <vector>
#include <openssl/rand.h>
#include "evp-encrypt.hpp"
#include "utils.hpp"

using namespace std;

/**
 * Aceasta functie verifica daca cifrul dat e continut in cel rezultat
 */
bool contains_cipher_dat(Bytes cipher_dat, Bytes rezultat){
    if(cipher_dat.size() > rezultat.size()) return false;
    for(size_t i=0; i<cipher_dat.size(); i++){
        if(cipher_dat[i] != rezultat[i])
            return false;
    }
    return true;
}

int main(int argc, char const *argv[]){
    // key, iv1, iv2
    array<Byte, KEY_SIZE> key;
    array<Byte, BLOCK_SIZE> iv;
    ///////////////////////////////////
    Bytes iv_bytes= {0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff, 0x00, 0x99, 0x88,
                    0x77, 0x66, 0x55, 0x44, 0x33, 0x22, 0x11};
    for (unsigned int i = 0; i < BLOCK_SIZE; i++) {
        iv[i] = static_cast<Byte>(iv_bytes[i]);
    }
    Bytes cipher_dat =
{0x76,0x4a,0xa2,0x6b,0x55,0xa4,0xda,0x65,0x4d,0xf6,0xb1,0x9e,0x4b,
0xce,0x00,
    0xf4, 0xed, 0x05, 0xe0, 0x93, 0x46, 0xfb, 0x0e,0x76, 0x25, 0x83, 0xcb, 0x7d,
0xa2, 0xac,0x93, 0xa2};

    // plaintext define array of bytes
    std::string inputString = "This is a top secret.";
```

```

Bytes ptext1(inputString.begin(), inputString.end());
//facem padding cu 11 valori de 0x0B pentru a fi multiplu de 16
for(int i=0; i<11; i++){
    ptext1.emplace_back(0x0B);
}

//read words from the file
std::ifstream inputFile("words.txt");
std::string line;
// Read each line of the input file
while (std::getline(inputFile, line)) {
    // Iterate over each word in the line
    std::istringstream iss(line);
    std::string word;
    while (iss >> word) {
        // Add "#" characters to the word until it has a length of 16 characters
        while (word.length() < 16) {
            word += "#";
        }
        ///CREARE CHEIE//////////
        for(unsigned int i=0; i< KEY_SIZE; i++ ){
            key[i] = word[i];
        }
        //OBTIERE CIFRU REZULTAT
        Bytes ctext1 = aes_encrypt(key.data(), iv.data(), ptext1);
        //DACA S_A GASIT CHEIA O AFISAM
        if(contains_cipher_dat(cipher_dat, ctext1)){
            // print essential information
            std::cout << "Ciphertex rezultat: " << hexlify(ctext1) << endl
                << "Ciphertex dat: " << hexlify(cipher_dat) << endl
                << "Word key: " << word << endl
                << "Plain text: " << hexlify(ptext1) << endl
                << "The IV used   : " << hexlify(iv) << endl
                << "The Key used   : " << hexlify(key) << endl;
        }
    }
}
return 0;
}

```

