Middle East Technical University Department Of Cyber Security

CSEC 507

APPLIED CRYPTOLOGY

Fall 2024-2025

First Assignment

Name Surname: Alkım Doğan
Student ID: 2521482

I will use the following table for determining the indices of the letters of English alphabet. This is required for Ceaser's cipher.

A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8
J	9
K	10
L	11
M	12
N	13
O	14
P	15
Q	16
R	17
S	18
T	19
U	20
V	21
W	22
X	23
Y	24
Z	25

1 Ceaser's Cipher

1) Finding the Cipher

First, let me write the indices of the letters of the word "CYBERSECURITY". It corresponds to 2-24-1-4-17-18-4-2-20-17-8-19-24. So, we will decrement these indices by k given in the questions below in a circular manner.

- Plaintext: **CYBERSECURITY**
- (a) $k = 1$ Let me decrease the indices 3-24-1-4-17-18-4-2-20-17-8-19-24 by 1. We get 1-23-0-3-16-17-3-1-19-16-7-18-23. These indices correspond to the letters B-X-A-D-Q-R-D-B-T-Q-H-S-Y respectively. So, the result is BXADQRDBTQHSY.
- (b) $k = 2$ Let me decrease the indices 3-24-1-4-17-18-4-2-20-17-8-19-24 by 2. We get 0-22-25-2-15-16-2-0-18-15-6-17-22. These indices correspond to the letters A-W-Z-C-P-Q-C-A-S-P-G-R-W respectively. So, the result is AWZCPQCASPGRW.
- (c) $k = 3$ Let me decrease the indices 3-24-1-4-17-18-4-2-20-17-8-19-24 by 3. We get 25-21-24-1-14-15-1-25-17-14-5-16-21. These indices correspond to the letters Z-V-Y-B-O-P-B-Z-R-O-F-Q-V respectively. So, the result is ZVYBOPBZROFQV.
- (d) $k = 25$ For this part, I will not increment but decrease by 1. Since the algorithm is circular manner decrementing by x is equal to increasing by $26 - x$. Therefore, I will increment each letter/indices of 2-24-1-4-17-18-4-2-20-17-8-19-24 by 1. After incrementing, we get the indices 3-25-2-5-18-19-5-3-21-18-9-20-25. These indices correspond to letters D-Z-C-F-S-T-F-D-V-S-J-U-Z respectively. So, the result is DZCFSTFDVSJUZ.

2) Finding the Plaintext

Part a

Here, the decrypted text is **NCCYVRQPELCGBYBTL**. First let me find the indices of the each letter respectively. My first thougth was the first 4 letters **NCCY**. This part should be something like **vowel-consonent-consonent-vowel** because of the structures of the languages (I assumed either English or Turkish). So, I will **NOT brute force** the solution but rather try to equate the first letter to the vowels from the alphabet respectively. Also, the indices of the letters are 13-2-2-24-21-17-16-15-4-11-2-6-1-24-1-19-11.

- Step 1) $k = 5$ Because I want to equate the first cipher letter N to I. So, the first 4 letters become 8-23-23-19 that correspond to IXXT. This seems to have no meaning so I do not think this is the correct way.
- Step 1) $k = 9$ Because I want to equate the first cipher letter N to E. So, the first 4 letters become 4-19-19-15 that correspond to ETTP. This seems to have no meaning so I do not think this is the correct way.
- Step 1) $k = 13$ Because I want to equate the first cipher letter N to E. So, the first 4 letters become 0-15-15-11 that correspond to APPL. I thought first the word **application** or something like that so I continued to decrypt. So, the remaining part becomes the indices 8-4-3-2-17-24-15-19-14-11-14-6-24. Those indices correspond to the letters I-E-D-C-R-P-T-O-L-O-G-Y. When we concatenate two parts we get **APPLIEDCRYPTOLOGY**. This seems to be result. **K=13**

Part b

Here, the decrypted text is **ECGUCTEKRJGT**. I will again start with observing the first 5 letters. Why 5? Because I thought the first 4-5 letters are enough to deteremine the result. Addittionally the second and fifth letters are the same, which makes observation easier. Let me first writing the indices of the cipher. Lettes correspond to the indices 4-2-6-20-2-19-4-10-17-9-6-19.

- Step 1) $k = 25$ decrementing (cipher to plain) by 1 each letter becomes 3-1 ... and the first the letters are DB. I thought no word could start with the letters **DB** So, I suddenly skip this step.
- Step 1) $k = 24$ decrementing (cipher to plain) by 2 each letter, the first 5 letters become 2-0-4-18-2 and this is C-A-E-S-E. I thought CAESERSCIPHER as soon as I saw. So, I continued to decrypt. Remaining part is the indices 17-2-8-15-7-6-17. These indices correspond to the letters R-C-I-P-H-E-R. I was correct, so the result is C-A-E-S-E-R-C-I-P-H-E-R, **CAESERCIPHER**. **K=24** (I accidentally start from reverse and realized that while checking the homework. But this way is much shorter by luck)

2 XOR

I will use the following table to convert **hexadecimal** to **bit** representation.

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

a)

$$FFFFFFFF_x \oplus 11111111_x$$

I will write step by step. I will first convert them to bitwise representations. XOR is equal to bitwise modulo 2, or not equal to operator.

$$= 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111 \oplus 0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001$$

$$= 1110\ 1110\ 1110\ 1110\ 1110\ 1110\ 1110\ 1110$$

$$= E\ \ E\ \ E\ \ E\ \ E\ \ E\ \ E\ \ E = EEEEEEEE_x$$

b)

$$11111111_x \oplus 22222222_x$$

I will write step by step. I will first convert them to bitwise representations. XOR is equal to bitwise modulo 2, or not equal to operator.

$$= 0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001 \oplus 0010\ 0010\ 0010\ 0010\ 0010\ 0010\ 0010\ 0010$$

$$= 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011$$

$$= 3\ 3\ 3\ 3\ 3\ 3\ 3\ 3 = 33333333_x$$

c)

$$ABCDE_x \oplus A87CA4_x$$

I will write step by step. I will first convert them to bitwise representations. XOR is equal to bitwise modulo 2, or not equal to operator. I will add 0000 as padding to the beginning of the left in order make them equal in terms of length.

$$= 0000\ 1010\ 1011\ 1100\ 1101\ 1110 \oplus 1010\ 1000\ 0111\ 1100\ 1010\ 0100$$

$$= 1010\ 0010\ 1100\ 0000\ 0111\ 1010$$

$$= A\ 2\ C\ 0\ 7\ A = A2C07A_x$$

d)

$$\overline{FFFFFFFF}_x \oplus 11111111_x$$

I will write step by step. I will first convert them to bitwise representations. XOR is equal to bitwise modulo 2, or not equal to operator. I will first compute the complement

$$= \overline{1111}\ \overline{1111}\ \overline{1111}\ \overline{1111}\ \overline{1111}\ \overline{1111}\ \overline{1111}\ \overline{1111} \oplus 0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001$$

$$= 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000 \oplus 0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001$$

$$0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001\ 0001$$

$$= 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ \ = 11111111_x$$

e)

$$\overline{FFFFFFFF}_x \oplus \overline{11111111}_x$$

I will write step by step. I will first convert them to bitwise representations. XOR is equal to bitwise modulo 2, or not equal to operator. I will first compute the complement.

$$= \overline{1111}\ \overline{1111}\ \overline{1111}\ \overline{1111}\ \overline{1111}\ \overline{1111}\ \overline{1111}\ \overline{1111} \oplus \overline{0001}\ \overline{0001}\ \overline{0001}\ \overline{0001}\ \overline{0001}\ \overline{0001}\ \overline{0001}\ \overline{0001}$$

$$= 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000 \oplus 1110\ 1110\ 1110\ 1110\ 1110\ 1110\ 1110\ 1110$$

$$= 1110\ 1110\ 1110\ 1110\ 1110\ 1110\ 1110\ 1110$$

$$= E\ \ E\ \ E\ \ E\ \ E\ \ E\ \ E\ \ E\ \ = EEEEEEEE_x$$

3 FREQUENCY ANALYSIS

For this part, I made use of Python programming language. Below is my python script that I made use of for my frequency analysis. Since it is just a script, I will show it in this document. Below file is **frequencyAnalysis.py**

```python
from constants import *
import requests
import string

TEXT = None

def fetch_text():
    global TEXT
    response = requests.get( URL )
    TEXT = response.text.replace('\n','').replace('\'','').lower()
    #print(TEXT)

def countLetters():
    global TEXT
    dict = {}

    for l in string.ascii_lowercase:

            occurences = TEXT.count(l)

            print(occurences)

            dict[l] = occurences

    # This below is for the Turkish part, I uncomment it while trying ↩
        English analysis
    for l in ['ş','ç','ö','ı','ğ','ü']:
            occurences = TEXT.count(l)

            dict[l] = occurences

    sorted_items = sorted_items = sorted(dict.items(), key=lambda item: item[1], reverse = True)

    result = ''

    total_len = len(TEXT) / 100

    with open('../info/TFA.txt', 'w') as f:

        for item in sorted_items:

            line = f'{item[0]} : {item[1]} : { item[1] / total_len }\n' # For debug
            #line = f'{item[0]}\n' # Order the letters descending order with frequency

            result += line

        f.write(result)



def main():
    fetch_text()
    countLetters()

if __name__ == '__main__':
    main()
```

I will explain this script briefly. I send request to the given URL and get the response with **request** library in python. The reason is I do not download the files for counting the letters. Instead, I get them from the response of the request I send to corresponding website. I first convert response into the string and get its length. Then, I find the number of occurrences of each letter in a response and save it into a dictionary. Finally, I save the results into a file with descending orders of letters with their number of occurences and ratios. From the code above, I hardcoded filepaths. For example, ' ../ info / TFA . txt ' is what I used for Turkish frequency analisys but I changed the names for the English Frequency Analysis.

```
URL = 'https://raw.githubusercontent.com/dwyl/english-words/refs/heads/master/words.txt'
URL = 'https://websites.umich.edu/~jlawler/wordlist'
URL = 'https://raw.githubusercontent.com/dwyl/english-words/master/words_alpha.txt'
URL = 'https://github.com/CanNuhlar/Turkce-Kelime-Listesi/blob/master/turkce_kelime_listesi.txt'
```

Those are the **URL's** that I sent request to. First 3 of them are in English and the last one is for Turkish. I just hardcoded these URL's in a file called **constants.py** and included them in other scripts.

Since I am explaining the general methodology here, this part is not specific to a Turkish/English analysis. Only difference between Turkish and English frequency analysis is their input data for my implementation and some edge cases for Turkish alphabet specific chars. All of the wordlists were more than 50,000 words. The longer the data, the more accurate the results will be. From this logic, I included 3 websites for English. I also think English will be more challenging since It is not my native language.

1) Turkish Frequency Analysis

I will share the results hardcoded way. I saved it into a file and now I am copy-pasting it into this latex document. I made use of the script I explained above. Here, the first column is letters, second is their number of occurrences and the third one is the ratio among all letters. This analysis corresponds to the link (click on link).

```
a : 91029 : 6.817706909529391
e : 71112 : 5.326003512621846
l : 60764 : 4.550979826765579
i : 58075 : 4.34958451450548
k : 50710 : 3.7979755614390514
r : 42809 : 3.2062223586993563
m : 42740 : 3.201054535513805
t : 40307 : 3.0188325962319826
ı : 34451 : 2.5802416893539095
n : 34413 : 2.577395641802447
s : 30469 : 2.2820058643558756
u : 22982 : 1.721259600729487
o : 21737 : 1.6280140954249784
d : 19385 : 1.4518587311870639
b : 18458 : 1.3824301501290084
y : 16944 : 1.2690376239996704
ş : 15359 : 1.1503274827083887
c : 13591 : 1.0179113755771672
z : 13494 : 1.0106464647221172
ü : 13008 : 0.9742470144586705
p : 11036 : 0.826552125735385
h : 9907 : 0.7419945550616581
g : 9630 : 0.7212483663312574
v : 9261 : 0.6936117466867887
ç : 9027 : 0.6760860854488329
f : 6963 : 0.5215007658114793
ğ : 5241 : 0.39252987413729185
ö : 4982 : 0.37313181319442623
q : 1413 : 0.10582803132150226
j : 1288 : 0.09646603279695323
w : 753 : 0.05639667911188337
x : 598 : 0.04478780094144257
```

2) English Frequency Analysis

will share the results hardcoded way. I saved it into a file and now I am copy-pasting it into this LaTex document. I will share 3 tables since there were 3 URL's as input. The reason behind is that I will make use of all of them for breaking ciphers. I will consider the average as the main one. The results are similar anyway. Here, the first column is letters, second is their number of occurrences and the third one is the ratio among all letters.

This analysis corresponds to the link (click on link).

```
e : 477141 : 10.863359064524538
i : 375395 : 8.546846060236259
a : 367425 : 8.365388227553131
n : 310546 : 7.070389474079647
o : 307800 : 7.007869623571759
s : 306059 : 6.968231218715884
r : 305957 : 6.965908922739262
t : 281227 : 6.402865986446443
l : 249256 : 5.67496280342106
c : 182326 : 4.151126825819833
u : 157477 : 3.585374544220955
d : 153205 : 3.488111324494189
p : 134734 : 3.0675708442570415
m : 126769 : 2.8862268496119827
h : 119027 : 2.7099600314648336
g : 105640 : 2.405170068336974
y : 83263 : 1.895699312759764
b : 81974 : 1.866351866545391
f : 54387 : 1.2382618752995362
v : 42894 : 0.9765937609924855
k : 38866 : 0.8848858375235217
w : 36767 : 0.8370966291418547
z : 17676 : 0.40244023218406244
x : 12373 : 0.2817036090073209
q : 8096 : 0.18432655124248526
j : 7801 : 0.17761010699637197
```

This analysis corresponds to the link (click on link).

```
e : 64242 : 9.651351282324784
a : 53801 : 8.082755056510628
i : 51656 : 7.760502503654449
n : 41440 : 6.225708993174856
r : 41418 : 6.222403838786588
o : 40683 : 6.111981635360344
t : 40144 : 6.031005352847766
s : 37603 : 5.649260021002753
l : 33167 : 4.982820708895522
c : 27046 : 4.063236617504999
u : 22820 : 3.4283465063766942
d : 20945 : 3.146657211921992
p : 18397 : 2.763860240044349
```

```
m : 18192 : 2.7330622105173017
h : 15246 : 2.2904719910700737
g : 13924 : 2.0918622591932117
y : 11822 : 1.7760697808231936
b : 11503 : 1.7281450421933002
f : 7775 : 1.1680716076721647
v : 5969 : 0.8967484792533956
k : 4577 : 0.6876223470502247
w : 4381 : 0.6581764261365599
x : 1828 : 0.27462828280703755
z : 1663 : 0.24983962489502376
q : 1167 : 0.17532341686860659
j : 1075 : 0.1615018621540292
```

This analysis corresponds to the link (click on link).

```
e : 376454 : 9.74058418044509
i : 313008 : 8.098946413513355
a : 295792 : 7.653489871012697
o : 251596 : 6.509937515508569
n : 251435 : 6.505771710249356
s : 250282 : 6.4759383346973545
r : 246142 : 6.368817628031885
t : 230895 : 5.974308107614394
l : 194915 : 5.04334119316425
c : 152979 : 3.9582653586900642
u : 131495 : 3.4023761649700286
p : 113663 : 2.9409808893036873
d : 113192 : 2.928793968327978
m : 105208 : 2.722211426777952
h : 92368 : 2.3899819887140317
g : 82627 : 2.137937833248249
y : 70580 : 1.8262269266784639
b : 63941 : 1.6544456775113014
f : 39238 : 1.015266253173839
v : 33075 : 0.8558012978165229
k : 26814 : 0.6938006349101209
w : 22407 : 0.5797714189017333
z : 14757 : 0.38183098267206134
x : 10493 : 0.2715018297199932
q : 5883 : 0.15222007664564186
j : 5456 : 0.1411716366103386
```

4 Cryptanalysis via Frequency Analysis

Again for this part, my method was generic for both languages. So, I will provide the code here since they are not specific to any languages. Below is the file **counter.py**. I used this script the count the each letter from the cipher texts.

```python
import string
from constants import *

def read_file():

    dict = {}

    with open(INPUT_FILE_NAME, 'r') as f:

        text = f.read()

        total_len = len(text) / 100

        for l in string.ascii_lowercase:

            occurences = text.count(l)

            line = f'{l} : {occurences} : { occurences / total_len }\n'

            dict[l] = occurences

        # Below is Turkish specific chars handling
        for l in ['ş','ç','ö','ı','ğ','ü']:
            occurences = text.count(l)

            line = f'{l} : {occurences} : { occurences / total_len }\n'

            dict[l] = occurences


    sorted_items = sorted_items = sorted(dict.items(), key=lambda item:↩
        item[1], reverse = True)

    result = ''

    with open(OUTPUT_FILE_NAME, 'w') as f:

        for item in sorted_items:

            #line = f'{item[0]} : {item[1]}\n' # For debug
            line = f'{item[0]}\n' # Order the letters descending order ↩
               with frequency

            result += line

        f.write(result)



def main():

    read_file()


if __name__ == '__main__':
    main()
```

I also want to mention another python script below. This script is the main one and used for replacing the chars in cipher. Script below is called **replacerOto.py**. I replace the letters in cipher text from an input letter file. I make replaced letters uppercase since replacing then twice should not result in an unwanted replacement. To go further, there are 3 input files. One of them is the letters that are replaced (this is the result file of **counter.py**), the second one is the letters I will replace (this is the result of frequency analysis with brute force changes) and the last one is the cipher text.

```
from constants import *

def read_file():

    with open(INPUT_FILE_NAME,'r') as f, open( '../result/↩
       turciphertextLetterCount.txt' ,'r') as letterCount, open('../↩
       info/TFA_REPLACE.txt', 'r') as replacedChars, open(↩
       REPLACER_OUTPUT_FILE,'w') as result:

        text = f.read()

        # f -> input file
        # letterCount -> frequency result file

        ascending_letters = letterCount.read().replace('\n','').replace↩
           (' ','')
        replaced_chars = replacedChars.read().replace('\n','').replace(↩
           ' ','')

        for i in range(0,29):
            # Turkish case
            if replaced_chars[i] == 'i':
                text = text.replace(ascending_letters[i], 'İ')
                continue

            text = text.replace(ascending_letters[i], replaced_chars[i↩
               ].upper())

        result.write(text)


```

```python

def main():
    read_file()

if __name__ == '__main__':
    main()
```

1) Breaking Turkish Cipher

Below is the letters in descending order in the given cipher text (i.e. n is the most common letter). So, each letter below must be replaced by result key to get the correct decrypted text.

Rank (common to rare)	Cipher Occurrences	Frequency Analysis Occurrences
1	e	a
2	m	e
3	ç	i
4	g	l
5	b	k
6	ö	r
7	c	m
8	l	t
9	y	ı
10	f	n
11	v	s
12	t	u
13	j	o
14	d	d
15	s	b
16	k	y
17	ş	ş
18	a	c
19	z	z
20	ğ	ü
21	r	p
22	ı	h
23	h	g
24	n	v
25	o	ç
26	u	f
27	p	ğ
28	ü	ö
29	i	j

My approach was that each letter from **middle column** was being replaced by each letter from **rightmost column**. For example, letter **e** from **middle column** was replaced by the letter **a** from the **rightmost column** and it goes like that from top to bottom. The output was the replaced version of the cipher text. However, Rightmost column was not the correct replacements even though It initially was the result of frequency analysis. So, I changed the order in a brute force method. As I changed the letters I looked at the output and searched for the meaningful words.

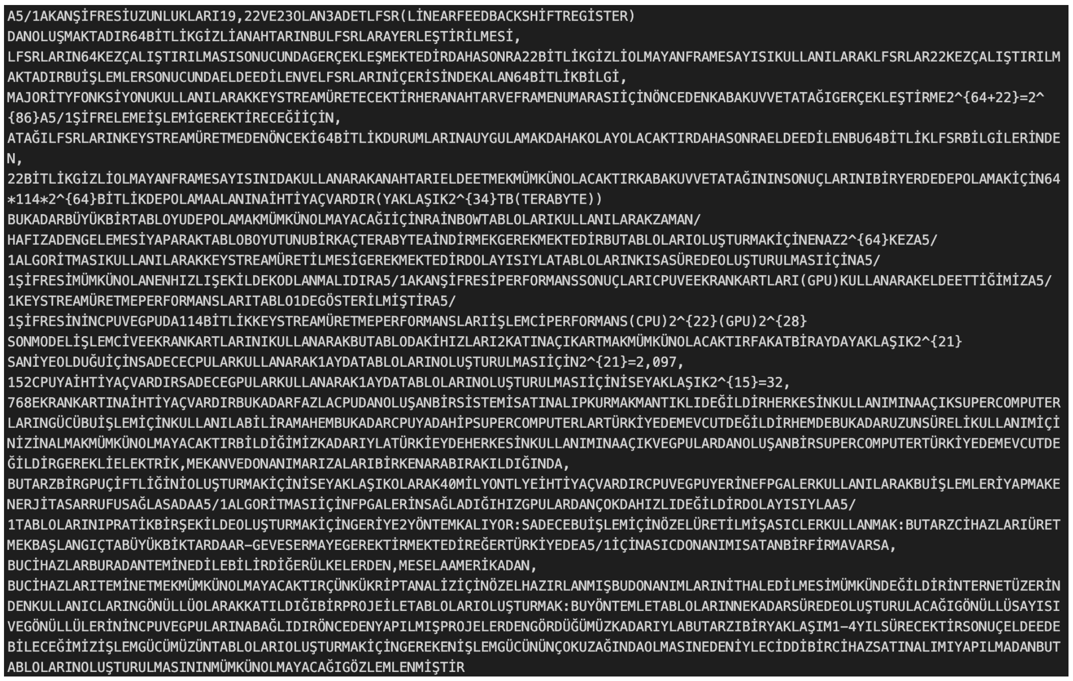
I first found **YAKLAŞIK2$\hat{\{}$34}TB(TERABYTE)**. Then, I noted down that the letters in the corresponding words are in the correct places. Then I found the word **TÜRKGYE**. It was obvious that It corresponded **TÜRKİYE**. I changed the places of the letters **i** and **g** to fix it. Afterwards, I observed **BAŞLAHGIÇTA** and **EKRANKARTINAIÖTIYAÇVAR** and fixed them. As I went through and founnd the words, the process became very easy and I got to observe the decrypted text and the key.

Cipher	Plain
e	a
m	e
ç	l
g	r
b	i
ö	n
c	k
l	ı
y	t
f	m
v	u
t	d
j	s
d	o
s	b
k	y
ş	ü
a	c
z	ş
ğ	p
r	g
ı	ç
h	z
n	f
o	h
u	ğ
p	v
ü	ö
i	j

I will arrange it alphabetically so that it can be easy to check. However, table above is also important because right column is the answer for the last part of this question. So, I will keep it to refer later on. I also changed the columns. Left one is the plain and the right one is the cipher.

Plain	Cipher
a	e
b	s
c	a
ç	ı
d	t
e	m
f	n
g	r
ğ	u
h	o
ı	l
i	b
j	i
k	c
l	ç
m	f
n	ö
o	d
ö	ü
p	ğ
r	g
s	j
ş	z
t	y
u	v
ü	ş
v	p
y	k
z	h

Above is the resulting key in an alphabetical manner. Left column is alphabet and the right one is the cipher key. Also, the resulting text is the following.



```
A5/1AKANŞİFRESİUZUNLUKLARI19,22VE23OLAN3ADETLFSR(LİNEARFEEDBACKSHİFTREGİSTER)
DANOLUŞMAKTADIR64BİTLİKGİZLİANAHTARINBULFSRLARAYERLEŞTİRİLMESİ,
LFSRLARIN64KEZÇALIŞTIRILMASISONUCUNDAGERÇEKLEŞMEKTEDİRDAHASONRA22BİTLİKGİZLİOLMAYANFRAMESAYISIKULLANILARAKLFSRLAR22KEZÇALIŞTIRILM
AKTADIRBUİŞLEMLERSONUCUNDAELDEEDİLENVELFSRLARINİÇERİSİNDEKALAN64BİTLİKBİLGİ,
MAJORİTYFONKSİYONUKULLANILARAKKEYSTREAMÜRETECEKTİRHERANAHTARVEFRAMENUMARASIİÇİNÖNCEDENKABAKUVVETATAĞIGERÇEKLEŞTİRME2^{64+22}=2^
{86}A5/1ŞİFRELEMEİŞLEMİGEREKTİRECEĞİİÇİN,
ATAĞILFSRLARINKEYSTREAMÜRETMEDENÖNCEKİ64BİTLİKDURUMLARINAUYGULAMAKDAHAKOLAYOLACAKTIRDAHASONRAELDEEDİLENBU64BİTLİKLFSRBİLGİLERİNDE
N,
22BİTLİKGİZLİOLMAYANFRAMESAYISINIDAKULLANARAKANAHTARIELDEETMEKMÜMKÜNOLACAKTIRKABAKUVVETATAĞININSONUÇLARINIBİRYERDEDEPOLAMAKİÇİN64
*114*2^{64}BİTLİKDEPOLAMAALANINAİHTİYAÇVARDIR(YAKLAŞIK2^{34}TB(TERABYTE))
BUKADARBÜYÜKBİRTABLOYUDEPOLAMAKMÜMKÜNOLMAYACAĞIİÇİNRAİNBOWTABLOLARIKULLANILARAKZAMAN/
HAFIZADENGELEMESİYAPARAKTABLOBOYUTUNUBİRKAÇTERABYTEAİNDİRMEKGEREKMEKTEDİRBUTABLOLARIOLUŞTURMAKİÇİNENAZ2^{64}KEZA5/
1ALGORİTMASIKULLANILARAKKEYSTREAMÜRETİLMESİGEREKMEKTEDİRDOLAYISIYLATABLOLARINKISASÜREDEOLUŞTURULMASIİÇİNA5/
1ŞİFRESİMÜMKÜNOLANENHIZLIŞEKİLDEKODLANMALIDIRA5/1AKANŞİFRESİPERFORMANSSONUÇLARICPUVEEKRANKARTLARI(GPU)KULLANARAKELDEETTİĞİMİZA5/
1KEYSTREAMÜRETMEPERFORMANSLARITABLO1DEGÖSTERİLMİŞTİRA5/
1ŞİFRESİNİNCPUVEGPUDA114BİTLİKKEYSTREAMÜRETMEPERFORMANSLARIİŞLEMCİPERFORMANS(CPU)2^{22}(GPU)2^{28}
SONMODELİŞLEMCİVEEKRANKARTLARINIKULLANARAKBUTABLODAKİHIZLARI2KATINAÇIKARTMAKMÜMKÜNOLACAKTIRFAKATBİRAYDAYAKLAŞIK2^{21}
SANİYEOLDUĞUİÇİNSADECECPULARKULLANARAK1AYDATABLOLARINOLUŞTURULMASIİÇİN2^{21}=2,097,
152CPUYAİHTİYAÇVARDIRSADECEGPULARKULLANARAK1AYDATABLOLARINOLUŞTURULMASIİÇİNİSEYAKLAŞIK2^{15}=32,
768EKRANKARTINAİHTİYAÇVARDIRBUKADARFAZLACPUDANOLUŞANBİRSİSTEMİSATINALIPKURMAKMANTIKLIDEĞİLDİRHERKESİNKULLANIMINAAÇIKSUPERCOMPUTER
LARINGÜCÜBUİŞLEMİÇİNKULLANILABİLİRAMAHEMBUKADARCPUYADAHİPSUPERCOMPUTERLARTÜRKİYEDEMEVCUTDEĞİLDİRHEMDEBUKADARUZUNSÜRELİKULLANIMİÇİ
NİZİNALMAKMÜMKÜNOLMAYACAKTIRBİLDİĞİMİZKADARIYLATÜRKİEYDEHERKESİNKULLANIMINAAÇIKVEGPULARDANOLUŞANBİRSUPERCOMPUTERTÜRKİYEDEMEVCUTDE
ĞİLDİRGEREKLİELEKTRİK,MEKANVEDONANIMARIZALARIBİRKENARABIRAKILDIĞINDA,
BUTARZBİRGPUÇİFTLİĞİNİOLUŞTURMAKİÇİNİSEYAKLAŞIKOLARAK40MİLYONTLYEİHTİYAÇVARDIRCPUVEGPUYERİNEFPGALERKULLANILARAKBUİŞLEMLERİYAPMAKE
NERJİTASARRUFUSAĞLASADAA5/1ALGORİTMASIİÇİNFPGALERİNSAĞLADIĞIHIZGPULARDANÇOKDAHIZLIDEĞİLDİRDOLAYISIYLAA5/
1TABLOLARINIPRATİKBİRŞEKİLDEOLUŞTURMAKİÇİNGERİYE2YÖNTEMKALIYOR:SADECEBUİŞLEMİÇİNÖZELÜRETİLMİŞASICLERKULLANMAK:BUTARZCİHAZLARIÜRET
MEKBAŞLANGIÇTABÜYÜKBİKTARDAAR-GEVESERMAYEGEREKTİRMEKTEDİREĞERTÜRKİYEDEA5/1İÇİNASICDONANIMISATANBİRFİRMAVARSA,
BUCİHAZLARBURADANTEMİNEDİLEBİLİRDİĞERÜLKELERDEN,MESELAAMERİKADAN,
BUCİHAZLARITEMİNETMEKMÜMKÜNOLMAYACAKTIRÇÜNKÜKRİPTANALİZİÇİNÖZELHAZIRLANMIŞBUDONANIMLARINİTHALEDİLMESİMÜMKÜNDEĞİLDİRİNTERNETÜZERİN
DENKULLANICLARINGÖNÜLLÜOLARAKKATILDIĞIBİRPROJEİLETABLOLARIOLUŞTURMAK:BUYÖNTEMLETABLOLARINNEKADARSÜREDEOLUŞTURULACAĞIGÖNÜLLÜSAYISI
VEGÖNÜLLÜLERİNİNCPUVEGPULARINABAĞLIDIRÖNCEDENYAPILMIŞPROJELERDENGÖRDÜĞÜMÜZKADARIYLABUTARZIBİRYAKLAŞIM1-4YILSÜRECEKTİRSONUÇELDEEDE
BİLECEĞİMİZİŞLEMGÜCÜMÜZÜNTABLOLARIOLUŞTURMAKİÇİNGEREKENİŞLEMGÜCÜNÜNÇOKUZAĞINDAOLMASINEDENİYLECİDDİBİRCİHAZSATINALIMIYAPILMADANBUT
ABLOLARINOLUŞTURULMASININMÜMKÜNOLMAYACAĞIGÖZLEMLENMİŞTİR
```

Figure 1: Turkish Plain text. I posted it as image because of encoding problems in LaTex

2) Breaking English Cipher

Below is the letters in descending order in the given cipher text (i.e. n is the most common letter). So, each letter below must be replaced by result key to get the correct decrypted text.

Rank (common to rare)	Cipher Occurrences	Frequency Analysis Occurrences
1	n	e
2	c	i
3	y	a
4	o	n
5	z	o
6	h	s
7	e	r
8	b	t
9	i	l
10	d	c
11	x	u
12	t	d
13	v	p
14	r	m
15	s	h
16	p	g
17	k	y
18	l	b
19	a	f
20	j	v
21	u	k
22	f	w
23	q	z
24	g	x
25	m	q
26	w	j

My approach was that each letter from **middle column** was being replaced by each letter from **rightmost column**. For example, letter **n** from **middle column** was replaced by the letter **e** from the **rightmost column** and it goes like that from top to bottom. The output was the replaced version of the cipher text. However, Rightmost column was not the correct replacements even though It initially was the result of frequency analysis. So, I changed the order in a brute force method. As I changed the letters I looked at the output and searched for the meaningful words such as the, are, we and so on.

I first found the appendix **TION**. I initally thought there might be words like encryption, decryption and so on. So, I assumed the letters **T-I-O-N** are in the correct places. Then I found **HOMERN** that seemed to be **MODERN**. So, I changed the places of the letters **M** and **D**. The next word was **LOMPGTERS** that was obviously **COMPUTERS**. Finally, I observed **CRGPTOLOYG** that is **CRYPTOLOGY**. So, after steps similar to those, I found the key and decrypted text.

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

I will arrange it alphabetically so that it can be easy to check. However, table above is also important because right column is the answer for the last part of this question. So, I will keep it to refer later on. I also changed the columns. Left one is the plain and the right one is the cipher.

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

Above is the resulting key in an alphabetical manner. Left column is alphabet and the right one is the cipher key.

Also, the resulting text is the following.

```
UNTILMODERNTIMESCRYPTOGRAPHYREFERREDA↩
    LMOSTEXCLUSIVELYTOENCRYPTIONWHICHISTHEPROCESSOFCONVERTINGORDINA↩
    RYINFORMATION(CALLEDPLAINTEXT)INTOUNINTELLIGIBLETEXT(CA↩
    LLEDCIPHERTEXT)↩
    DECRYPTIONISTHEREVERSEINOTHERWORDSMOVINGFROMTHEUNINTELLIGIBLECIPHERTEXTB↩
    ACKTOPLAINTEXTACIPHER(ORCYPHER)ISAPAIROFALGORITHMSTHATCREA↩
    TETHEENCRYPTIONANDTHEREVERSINGDECRYPTIONTHEDETAILEDOPERATIONOFA↩
    CIPHERISCONTROLLEDBOTHBYTHEALGORITHMANDINEACHINSTANCEBYAKEYTHISISA↩
    SECRET(IDEALLYKNOWNONLYTOTHECOMMUNICANTS)USUALLYASHORTSTRINGOFCHARA↩
    CTERSWHICHISNEEDEDTODECRYPTTHECIPHERTEXTFORMALLYA↩
    CRYPTOSYSTEMISTHEORDEREDLISTOFELEMENTSOFFINITEPOSSIBLEPLA↩
    INTEXTSFINITEPOSSIBLECYPHERTEXTSFINITEPOSSIBLEKEYSANDTHEENCRYPTIONA↩
    NDDECRYPTIONALGORITHMSWHICHCORRESPONDTOEACHKEYKEYSAREIMPORTA↩
    NTBOTHFORMALLYANDINACTUALPRACTICEASCIPHERSWITHOUTVARIABLEKEYSCA↩
    NBETRIVIALLYBROKENWITHONLYTHEKNOWLEDGEOFTHECIPHERUSEDANDA↩
    RETHEREFOREUSELESS(OREVENCOUNTER-PRODUCTIVE)FORMOSTPURPOSESHISTORICA↩
    LLYCIPHERSWEREOFTENUSEDDIRECTLYFORENCRYPTIONORDECRYPTIONWITHOUTA↩
    DDITIONALPROCEDURESSUCHASAUTHENTICATIONORINTEGRITYCHECKSTHEREA↩
    RETWOKINDSOFCRYPTOSYSTEMS:SYMMETRICANDA↩
    SYMMETRICINSYMMETRICSYSTEMSTHESAMEKEY(THESECRETKEY)ISUSEDTOENCRYPTA↩
    NDDECRYPTAMESSAGEDATAMANIPULATIONINSYMMETRICSYSTEMSISFASTERTHANA↩
    SYMMETRICSYSTEMSASTHEYGENERALLYUSESHORTERKEYLENGTHSA↩
    SYMMETRICSYSTEMSUSEPUBLICKEYTOENCRYPTAMESSAGEANDAPRIVA↩
    TEKEYTODECRYPTITUSEOFASYMMETRICSYSTEMSENHANCESTHESECURITYOFCOMMUNICA↩
    TION

INCOLLOQUIALUSETHETERMCODEISOFTENUSEDTOMEANANYMETHODOFENCRYPTIONORCONCE↩
    ALMENTOFMEANINGHOWEVERINCRYPTOGRAPHYCODEHASAMORESPECIFICMEANINGITMEA↩
    NSTHEREPLACEMENTOFAUNITOFPLAINTEXT(IEAMEANINGFULWORDORPHRASE)WITHA↩
    CODEWORD(FOREXAMPLEWALLABYREPLACESATTACKATDAWN)CODESA↩
    RENOLONGERUSEDINSERIOUSCRYPTOGRA P H Y EXCEPTINCIDENT A↩
    LLYFORSUCHTHINGSASUNITDESIGNATIONS(EGBRONCOFLIGHTOROPERATIONOVERLORD↩
    )   SINCEPROPERLYCHOSENCIPHERS   AREBOTHMOREPRACTICALANDMORESECURETHA↩
    NEVENTHEBESTCODESANDALSOAREBETTERADAPTEDTOCOMPUTERS

CRYPTANALYSISISTHETERMUSEDFORTHESTUDYOFMETHODSFOROBTAININGTHEMEA↩
    NINGOFENCRYPTEDINFORMATIONWITHOUTACCESSTOTHEKEYNORMA↩
    LLYREQUIREDTODOSO;IEITISTHESTUDYOFHOWTOCRACKENCRYPTIONA↩
    LGORITHMSORTHEIRIMPLEMENTATIONS

SOMEUSETHETERMSCRYPTOGRAPHYANDCRYPTOLOGYINTERCHANGEA↩
    BLYINENGLISHWHILEOTHERS(INCLUDINGUSMILITARYPRACTICEGENERALLY)↩
    USECRYPTOGRAPHYTOREFERSPECIFICALLYTOTHEUSEANDPRACTICEOFCRYPTOGRA↩
    PHICTECHNIQUESANDCRYPTOLOGYTOREFERTOTHECOMBINEDSTUDYOFCRYPTOGRAPHYA↩
    NDCRYPTANALYSISENGLISHISMOREFLEXIBLETHANSEVERALOTHERLANGUA↩
    GESINWHICHCRYPTOLOGY(DONEBYCRYPTOLOGISTS)ISALWA↩
    YSUSEDINTHESECONDSENSEABOVERFC2828ADVISESTHATSTEGANOGRA↩
    PHYISSOMETIMESINCLUDEDINCRYPTOLOGY

THESTUDYOFCHARACTERISTICSOFLANGUAGESTHATHAVESOMEAPPLICATIONINCRYPTOGRA↩
    PHYORCRYPTOLOGY(EGFREQUENCYDATALETTERCOMBINATIONSUNIVERSALPA↩
    TTERNSETC)ISCALLEDCRYPTOLINGUISTICS
```

3) Key and Frequency Analysis Result Match

The perfect match did not occur as expected even though there were some matched letters. I will name 5 reasons for this situation.

1. The encrypted texts are just within a specific context of a language. However, analyzed context are more generic. There should be context based changes in terms of the number of occurrences of the letters. In our case, the contexts of the ciphers were related to **Cryptology**.

2. The analyzed context length and the length of the encrypted text is also one of the main reasons. The longer the encrypted texts, the more generic they will be.

3. Decrypted texts contain words from other languages. Even the word **Cryptology** is not **English** derived. The origin is Latin. Also, **Linear FeedBack Shift Register** is not a **Turkish** clause as well. Therefore, these words are exceptions that will change the balance of the letter occurrence ratio.

4. This is not our case but the designer could have added random sequences of letters to make analysis impossible.

5. The languages also differ from time to time and area to area. So, not only context but the time and area is also important in terms of accent. The newspaper from 1960's are quite different from todays'. This is also the case for **Turkey** and **Azerbaijan** languages even though both are Turkic originated languages.