ELEC6032 Assignment

Dionisio Perez-Mavrogenis

May 12, 2014

1 Part 1

The assigned elliptic curve E curve over the finite field Z_{17} is

$$y^2 = x^3 + 4x + 3$$

together with the point at infinity, O.

1.1 Find the order of your unique group

Table 1: Field $E(Z_{17})$

$(1, \pm 5)$	$(1, \pm 12)$	$(2, \pm 6)$	$(2, \pm 11)$	$(3, \pm 5)$	$(3, \pm 12)$
$(4, \pm 7)$	$(4, \pm 10)$	$(7, 0)$	$(11, \pm 1)$	$(11, \pm 16)$	$(13, \pm 5)$
$(13, \pm 12)$	$(14, \pm 7)$	$(14, \pm 10)$	$(15, \pm 2)$	$(15, \pm 15)$	$(16, \pm 7)$
$(16, \pm 10)$	O				

By the script given in Listing 3 the cardinality of $E(Z_{17})$ is 38, the elements given in Table 1. Although one would expect that points should occur in pairs due to the symmetrical properties of elliptic curves, e.g. (15,2) and (15,-2), points with negative coordinates are not in Z_{17}, but there exist equivalent coordinates satisfying the equation, e.g. (15,15) instead of (15,-2).

1.2 Is your group cyclic? Explain your answer

Every elliptic curve over a field is an Abelian group under addition, with O acting as the identity element. Every elliptic curve is a cyclic group or the product of two cyclic groups (i.e. has at most two generators), unless it is or prime order in which case it is cyclic [8] [3].

1.3 Explain how elliptic curves can be used to construct secure ciphers

The following follows closely [7].

Two parties publicly agree on a prime number p, an elliptic curve E over Z/pZ (which is the field Z_p of characteristic p) and a point $P \in E(Z_p)$. Then sender A chooses a secret element a and sends aP to B. B also computes his secret element b and sends bP to A. They can both compute abP, which is their secret key. For an adversary to compute n from nP (or m) it would require solving the discrete logarithm problem for elliptic curves over a field.

The motivation to use elliptic curves is that the discrete logarithm problem is considered (mathematically) harder to solve for an elliptic curve over a field than it is for a finite field. Furthermore they are able to provide the same level of security as traditional PKI systems, with smaller key lengths (making them more attractive for using on smaller devices such as phones).

1.4 Encrypt the following message using an appropriate elliptic curve cipher

"A little knowledge is a dangerous thing"

2 Part 2

I believe this message is enciphered with a Vigenre cipher and the original text is:

`the message starts here recently hardware trojans have attracted the attention of governments and researchers one of the main concerns is that integrated circuits in military or critical infrastructure applications could be maliciously manipulated during the manufacturing process which often takes place abroad however since there have been no reported hardware trojans in practice yet little is known about how such a trojan would look like and how hard to implement one in practice one example is dopant trojan this can be used to compromise the security of a meaningful real world target while avoiding detection by functional testing as well as trojan detection mechanisms such trojans can be used to establish a hidden side channel in an otherwise side channel resistant design this trojan does not change the logic value of any gate but instead changes only the power profile of two gates an evaluator who is not aware of the trojan cannot attack the trojan design using common side channel attacks the owner of the trojan however can use his knowledge of the trojan power model to establish a hidden side channel that reliably leaks out secret keys this is the end of the message`

This message looks like structured text and hence it might be encrypted with some mono-alphabetic substitution method. First I run it through a Caesar cracking tool with no meaningful results. Then I run it through [2], which yielded a meaningful message for key `xayxayxayxayxay`. However, by observing the key that [2] derived one can see that it repeats itself and hence ,due to the nature of the cipher, the key could simply be `xay`. Furthermore Script 4 reports that the highest index of coincidence is found for a key length of 18, which is a multiple of 3.

In order to crack a Vigenre cipher, and due to this text's length which allows for meaningful statistical analysis, one can either employ the Kasisky test or the index of coincidence text to determine they key length. Once the key length has been determined one can treat the parts that correspond to each key-letter as simple mono-alphabetic substitution cipher and employ statistical methods.

Kasinsky Test The Kasinski test is founded on the notion that texts which appear similar on the cipher-text will be the same plain-text that coincidentally was encrypted with the same key. The cryptanalyst looks for identical ciphertext bits that are at least three characters long and records the distance between those($\mathbf{\Delta} = \Delta_1, \Delta_2, \ldots, \Delta_n$), as the distance is likely to be a multiple of the key-length (i.e. Length = gcd($\mathbf{\Delta}$)).

Index of Coincidence The IoC method exploits the fact that letters in a language are not uniformly distributed in words and hence some letters will be more frequent than others. If the letter frequency was equal, then the probability of selecting the same random letter twice (in English) would be $\sum_{i=a}^{i=z}(1/26)^2 = 0.038$. Given the letter frequencies in English this probability is $\sum_{i=a}^{i=z} f_i^2 = 0.067$, f_i is the frequency of letter i. One can use this fact to test whether the plain-text was enciphered with a *single-alphabet* cipher (e.g. Caesar) or not ,as the cipher-text's IoC would approximate that of the plain-text, i.e. $0.067/0.038 = 1.73$. Once key-lengths that approach plain-text IoC are found they can be factored and a more systematic search can reveal likely key lengths.

3 Part 3

Here we knew that the second stage is an XOR cipher and the first two characters of the output, Hg. We get the key for the first two bytes by doing `Q" XOR Hg = 0x1945`(or 25 69 in decimal). A naive assumption worth investigating is to try this key for every two bytes of text, done by the script shown in Listing 1. This produces the intermediate text shown bellow :

```
Hgt ltsbuba pm dumt uy hgt lpyh wqpmpxbr sbr tdxyuct lvyhtqv pm hgtl sdd, xbobpib hp
tctb hgt aqtshtyh lubry. Yxqtdv sbvpbt igp htddy vpx hgtv gsct hgt sbyitq uy npouba,
lsr pq yulwdv luyhsotb. Hgtqt sqt lsbv hgubay hgsh lsot dumt ipqhg gpdruba pb hp sbr
yscpxquba. Exh dumt uy xbwqtrukhsedt sbr it sqt pmhtb lvyhtquty tctb hp pxqytdcty.
It hgubo yxkktyy, gswwubtyy, gtdwuba phgtqy, pq yxqwsyyuba pxqytdcty iudd lsot dumt
ipqhg ducuba, exh it ksb sdisvy et iqpba pq mqxyhqshtr ev tctbhy. Hguy uy s qsbrpl
wgqsyt. Wgudpypwgtqy gsct s dph hp ysv sepxh hgt csdxt pm sdd hgtyt hgubay, sbr s
duhhdt dtyy hp ysv sepxh pbt pm hgt lpyh csdxsedt hgubay pm sdd: dpct. Yp it ksb
et kdtsq tbpxag sepxh igsh uh ltsby mpq dumt hp gsct ltsbuba sbr csdxt, exh igtb it
wxh rpib pxq wgudpypwgv eppoy sbr skhxsddv ath pb iuhg ducuba, ltsbuba sbr csdxt ksb
et tdxyuct. Ducuba itdd uy lpqt sqh hgsb ykutbkt pq wgudpypwgv. Hgtqtmpqt, hgt pbdv
ytbyt it ksb lsot pm hgt urts hgsh dumt gsy ltsbuba uy hgsh hgtqt sqt yplt qtsypby hp
duct qshgtq hgsb hp rut, sbr hgpyt qtsypby sqt hp et mpxbr ub hgt ducuba pm dumt uhytdm.
```

This text is sufficiently large to meaningful provide statistical information about its contents. More importantly, it looks like structured text, something which suggests that the first encryption stage might be a mono-alphabetic substitution. By using [1] to get frequency overviews and perform substitutions, we get :

```
the meaning of life is the most profound and elusive mystery of them all, unknown to
even the greatest minds. surely anyone who tells you they have the answer is joking,
mad or simply mistaken. there are many things that make life worth holding on to and
savouring. but life is unpredicTable and we are often mysteries even to ourselves.
we think success, happiness, helping others, or surpassing ourselves will make life
worth living, but we can always be wrong or frustrated by events. this is a random
phrase. philosophers have a lot to say about the value of all these things, and a
little less to say about one of the most valuable things of all: love. so we can
be clear enough about what it means for life to have meaning and value, but when we
put down our philosophy books and actually get on with living, meaning and value can
be elusive. living well is more art than science or philosophy. therefore, the only
sense we can make of the idea that life has meaning is that there are some reasons to
live rather than to die, and those reasons are to be found in the living of life itself.
```

In order to understand the substitutions I looked for 1-letter words, which is most likely the article `a`. Having done that 2 and 3-letter words start appearing and analysing those is easier, as well as observing effects on longer words. Furthermore, the three-grams `SRB` , `SDD` and `NGT` are frequently found, and if we assume that `S` is really `a`, then `SRB` is `and` and `SDD` is `all`(that is the most likely scenario for these two words). Making these substitutions helped reveal more common words.

The next step that helped a lot was letter frequency, exploiTable due to the texts length. The frequencies produced by [1] are give in Table 2. Given the text's length it would be a fair to assume that some of the first letters are vowels, and hence substituting `T` for `e` gave further insight on the text. Now words with odd repetitions could be exploited, like `TCTB(eCeB)`, which contains an averagely common letter with another rare one. Furthermore, it would be safe to assume that `C` and `B` are both consonants. By trying words like `ever, even, eyes` revealed more information and the rest of the text was deciphered in a similar fashion.

Table 2: Statistical information about the second-stage decrypted text. The top row statistics are for the English language, while the bottom refer to the text.

E	T	A	O	I	N	S	H	R	D	L	C	U	M	W	F	G	Y	P	B	V	K	J	X	Q	Z
12.7	9.1	8.2	7.5	7.0	6.7	3.3	6.1	6.0	4.3	4.0	2.8	2.8	2.4	2.4	2.2	2.0	2.0	1.9	1.5	1.0	0.8	0.15	0.15	0.1	0.07
T	S	H	P	B	Y	U	D	G	Q	X	C	L	M	A	I	R	V	W	E	K	O	N	F	J	Z
113	73	72	68	67	65	56	52	49	41	28	21	21	21	20	19	19	17	15	14	11	8	1	0	0	0
12.9	8.38	8.26	7.80	7.69	7.46	6.42	5.97	5.62	4.70	3.21	2.41	2.41	2.41	2.29	2.18	2.18	1.95	1.72	1.60	1.26	0.91	0.11	0	0	0
e	a	t	o	n	s	i	l	h	r	u	v	m	f	g	w	d	y	p	b	c	k	j			

4 Part 4

In order to solve this part I used some online tools, as well as scripts that I wrote myself. In particular, I used [6] to perform frequency analysis and interactive letter substitutions, [5] to learn how the puzzles work and [1] to search for words matching a given pattern.

Appendix B gives snapshots from the tool with the deciphered headlines and the letter frequencies.

4.1 Headline 1

Solution I believe this headline is the text: `ntsb urges new ways to combat rising runway incidents`.

One thing to note on this headline is that the first and last words ,`YNTS` and `CYOCMBYNT`, contain the same trigraph, `YNT`. Furthermore `CYOCMBYNT` contains two repeated letters and its ending must make sense in another word. The latter reasoning regarding word endings turned out to be misleading, as NTSB is an acronym for National Transportation Safety Board.

After going through a couple of words produced with the pattern `---------/abcadebfg` by [1], the word `incident` seemed to make more sense in the context of a headline and how the substitutions affected other words. Substituting `inciMBnts` to the sentence, `HCTCYA` turned into `HisinA` and hence `A` is most likely `g`. After performing that substitution and `B` for `e` it appeared that `K` was most likely `w`. Now words like `QHges new wJVs` are appearing, suggesting text along the lines of `urges new ways` (`new` was a big give-away) and helping conclude that `R` is `o` after performing the substitutions so far.

After these previously mentioned steps, the rest of the text is fairly simple to deduce, contains no mistakes and makes sense in the context of a newspaper headline.

4.2 Headline 2

Result I believe that this headline is the text: `dutch authorities closing in on human smuggling ring`.

To decipher this headline I firstly attempted to investigate the repetition of the letter `B` in `OAXBBJNHB` and the 9-letter words matching this pattern are shown in Table 3. Furthermore `OAXBBJNHB` ends in `HB`, as does `WNHB` and `YJVONHB`.

Additionally the two two-letter words `NH` and `VH` are in the set shown in Table 4, both end with the letter `H` and should make sense syntactically and meaning-wise. The letter `H` in these words eliminates all words with `g, l` in their 8^{th} position from Table 3. Furthermore the word list in Table 4 is further filtered by word likeliness, words whose second letter is not in any other word and by the possible 8^{th} position letters of Table 3 words. This leaves a reduced list, shown in Table 4.

After using words ending in `s` and after using `crossties`, it seemed unlikely that `B` was `s` or that the word started with a `c`. Substituting `OAXBBJNHB` for the word `smuggling` some patterns in the words appeared, starting with the word `authorities` and from there on it was a case of the text making sense.

Table 3: 9-letter words with repeated 4th and 5th letters, ending on that letter as well.

blessings	blottiest	brattiest	brittlest	brummagem	chasseurs	chattiest
coassumes	crossings	crosslets	crossties	crossways	draggling	dressings
glassines	glissades	glossinas	gnattiest	grottiest	knottiest	plottiest
plussages	pressings	reassigns	scheelite	smuggling	threesome	trussings
unseeable						

Table 4: 2-letter words and reduced list.

ad	am	an	as	at	be	by	de	do	el	en
go	he	id	if	in	is	it	ma	me	my	no
nu	of	oh	on	or	pi	so	to	up	us	we

an	on	in	as	is	us	at
it	be	he	me	we	by	my
do	go	so	to	no	if	of

4.3 Headline 3

I believe this headline is the text : `dollar,euro outpace yen in choppy trading`.
How this headline was found follows later.

4.4 Headline 4

I believe this headline is the text : `ripken looking for quick return from disabled list`.
How this headline was found follows later.

4.5 Headline 5

I believe this headline is the text : `chrldless employees see discrimination in family friendly policies`.

The starting point for this headline was the ending of the word `TBDSPKTTY`, the word `YTT` and the ending of the word `HQUSESTYY`. The word `YTT` contains two very frequent letters, and one is a repetition. The letter `T` could be a vowel. The word `see` seemed like a very good candidate. Substituting `see` for `YTT` made the word `TBDSPKTTY` seem like `employees`, and after substitution, the first words started appearing.

The next stage of decryption was the word `JCBRSKJURTWESK`. The tool [1] produces a list of words matching this pattern and after trial and error the most likely word is `discrimination`.

After substitution of `discrimination`, the next word that seemed promising is `DPSRHRTY`, which most likely seems the be the word `policies`.

After performing all these substitutions, it is a matter of substituting the remaining letters and deciphering the message. The resulting message contains an error (one would expect ch*i*ldless rather than ch*r*ldless), but one would expect more errors to be in the sentence if the mapping was somehow forced to make the resulting plain-text.

4.6 Discussion

According to [5], the first step to solving the puzzle is to build the mixed alphabet. Following the process described on the website (weaving of substitution chains of headlines 2 and 5 here, as I was unable to weave 2 and 3), the mixed alphabet produced was `girumbnwxachqlsykfjopdetvz`. To verify that this is correct, the mapping generated by sliding the alphabet against itself should decipher the solved headlines and produce meaningful text for the unsolved ones, which happens. The shift in the alphabet that solves each headline is $9, 5, 21, 7, 1$ and by stacking the resulting shifted-putative alphabets we can find the setting word, which I found to be `toady`. One can then reorder the alphabet so that `toady` is in the first column of the stacked alphabets and hence the

Table 5: The keyword square.

S	Y	C	O	P	H	A	N	T	B
D	E	F	G	I	J	K	L	M	Q
R	U	V	W	X	Z				

shifted alphabet becomes `sykfjopdetvzgirumbnwxachql`. Inputing that into the tool [4] yields the key `sycophant` and keyword square presented in Table 5, indicating that the hat is of length 10.

References

[1] Robert Giordano. Word pattern finder v1.3. `http://www.design215.com/toolbox/wordpattern.php`, 2012.

[2] Groundspeak Inc. Vigenre cipher codebreaker. `http://www.mygeocachingprofile.com/codebreaker.vigenerecipher.aspx`, 2014.

[3] G. A. Jones. Ecc mathematics notes. `https://secure.ecs.soton.ac.uk/notes/elec6032/gaj/6032EllCurves.pdf`, 2011.

[4] Bill Mason. Headline key solver using web workers. `ttp://bionsgadgets.appspot.com/ww_forms/hl_key_web_worker.html`, 2014.

[5] Jude Patterson. The headline puzzle : Solving. `https://sites.google.com/site/theheadlinepuzzle/help/resources`, 2014.

[6] Daniel Rodriguez-Clark. Frequency analysis : Breaking the code. `http://crypto.interactive-maths.com/frequency-analysis-breaking-the-code.html`, 2013.

[7] William Stein. *Elementary Number Theory: Primes, Congruences, and Secrets*. 2011.

[8] Nick Sullivan. A (relatively easy to understand) primer on elliptic curve cryptography. `http://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography`, 2013.

A Source code

Note : The scripts in this section were produced by me in order to help me understand some concepts or aid my investigation.

```
file = open('secret.hex','r').readline()

key = [ord(file[0]) ^ ord('H'), ord(file[1]) ^ ord('g')]
print "Key is:", key, hex(key[0]), hex(key[1]) #decimal, then hex

string = [chr(ord(byte)^key[i%2]) for i, byte in enumerate(file)]
print "".join(string)
```

Listing 1: Code that performed the XOR operation with the key for question 3.

```
palph = 'girumbnwxachqlsykfjopdetvz' *2
txt = 'ciphertext_here'.lower().replace(' ','')

for i in range(0,26):
        cur_all = palph[i:i+26]
        txt2 = txt
        for j,l in enumerate(palph[0:26]) :
                txt2 = txt2.replace(cur_all[j],l.upper())
        print i,txt2
```

Listing 2: Code that perform shifting of the putative alphabet.

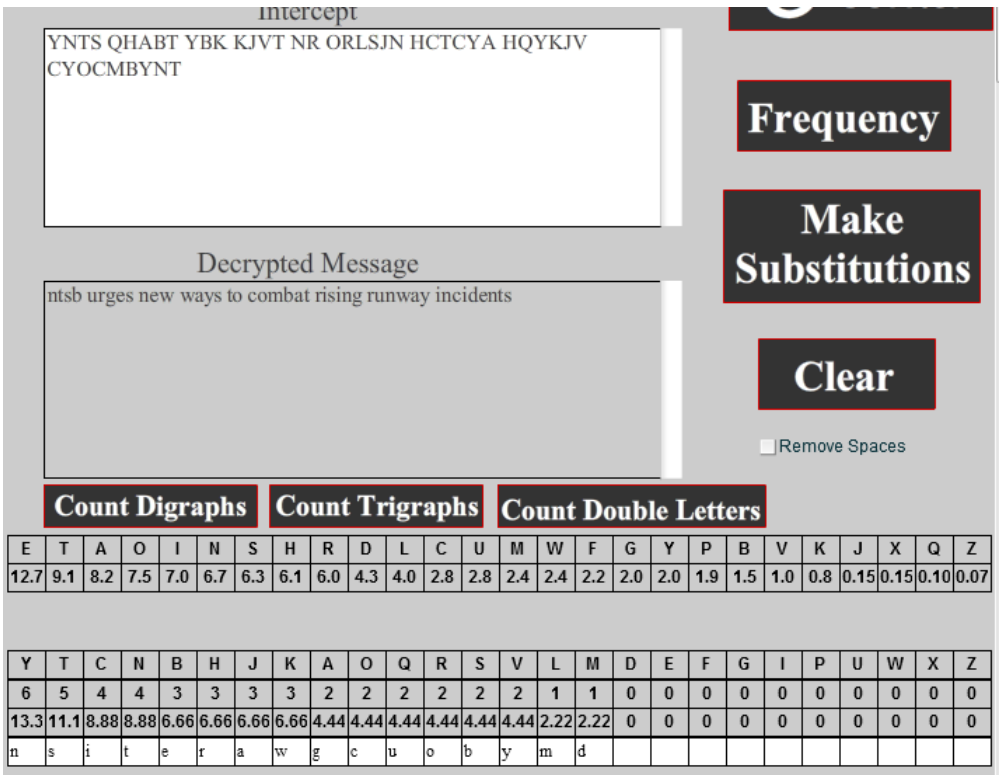
```
from math import sqrt

field = 17

def E(x):
        return ((x ** 3) + (4*x) + 3) % field

y_s = [ (i**2) % field for i in range(0,17)]
x_s = [ E(x) for x in range(0,17)]
results = []

for i_x,x in enumerate(x_s):
        for i_y,y in enumerate(y_s):
                if x == y :
                        results.append((i_x,i_y))
                        if i_y == 0 :
                                results.append((i_x,-i_y))

print results
```

Listing 3: Code that iterates though the field Z_{17} to check for solutions to elliptic curve E with equation $y^2{=}x^3 + 4x + 3$.

```
from __future__ import division

text = 'input_ciphertext_here'

def ioc(freqs) :
        enumerator  = 0
        denominator = 0
        for letter in freqs:
                enumerator += freqs[letter]*(freqs[letter] -1 )
                denominator += freqs[letter]

        return enumerator/(denominator*(denominator-1))

def build_dict(t, length):
        fr = dict()
        for i,l in enumerate(t) :
                index = i * length
                if index >= len(t) : #check out of bounds
                        return fr

                letter = t[index]
                if letter == ' ' :
                        pass
                elif letter in fr :
                        fr[letter] += 1
                else:
                        fr[letter] = 1
        return fr

fp = open ('res.txt','w')
fp.write('"Key length", IoC \n')
highest = 0
highest_index = []
for i in range(1,20):
        fp.write("KEY length {:2d}\n".format(i))
        for j in range(0,i):
                t = text[j:] #shift text one position to the right
                list = build_dict(t,i)
                index_oc = ioc(list)
                if index_oc > highest :
                        highest = index_oc
                        highest_index = [i,j]
                fp.write("{:2d} : {}\n".format(j,index_oc/0.0385))

fp.write("Highest value {}, index : {}".format(highest, highest_index))
fp.close()
```

Listing 4: Code that performs Index of Coincidence analysis.

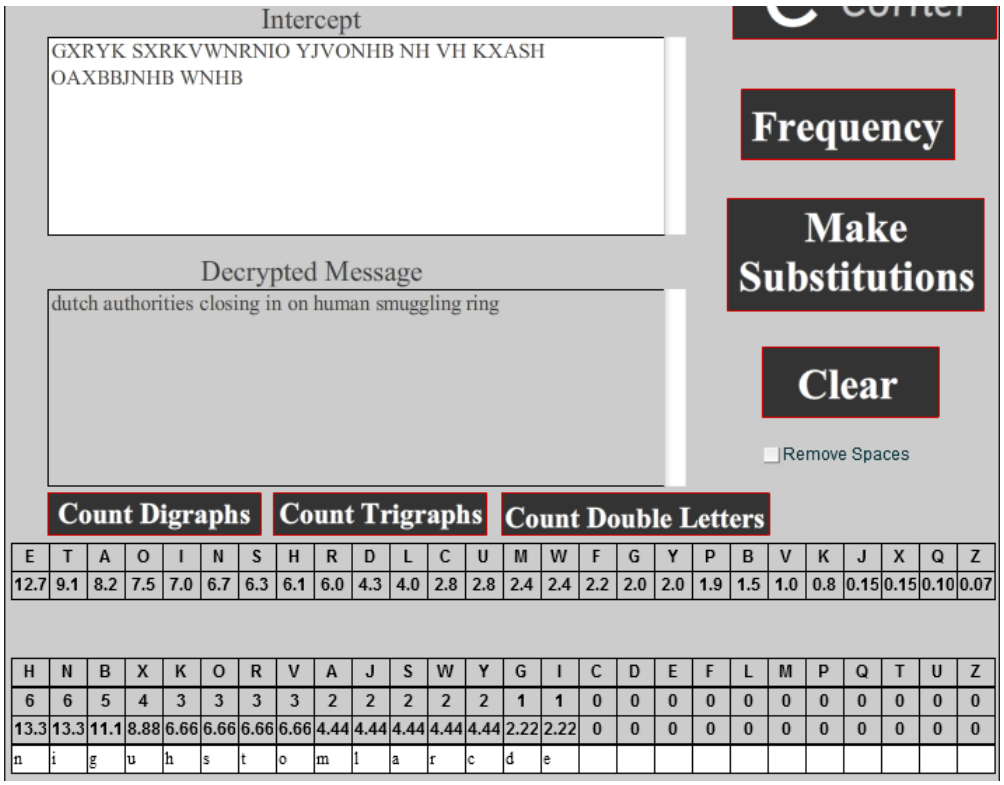
B Headline Puzzle Substitutions



E	T	A	O	I	N	S	H	R	D	L	C	U	M	W	F	G	Y	P	B	V	K	J	X	Q	Z
12.7	9.1	8.2	7.5	7.0	6.7	6.3	6.1	6.0	4.3	4.0	2.8	2.8	2.4	2.4	2.2	2.0	2.0	1.9	1.5	1.0	0.8	0.15	0.15	0.10	0.07

Y	T	C	N	B	H	J	K	A	O	Q	R	S	V	L	M	D	E	F	G	I	P	U	W	X	Z
6	5	4	4	3	3	3	3	2	2	2	2	2	2	1	1	0	0	0	0	0	0	0	0	0	0
13.3	11.1	8.88	8.88	6.66	6.66	6.66	6.66	4.44	4.44	4.44	4.44	4.44	4.44	2.22	2.22	0	0	0	0	0	0	0	0	0	0
n	s	i	t	e	r	a	w	g	c	u	o	b	y	m	d										

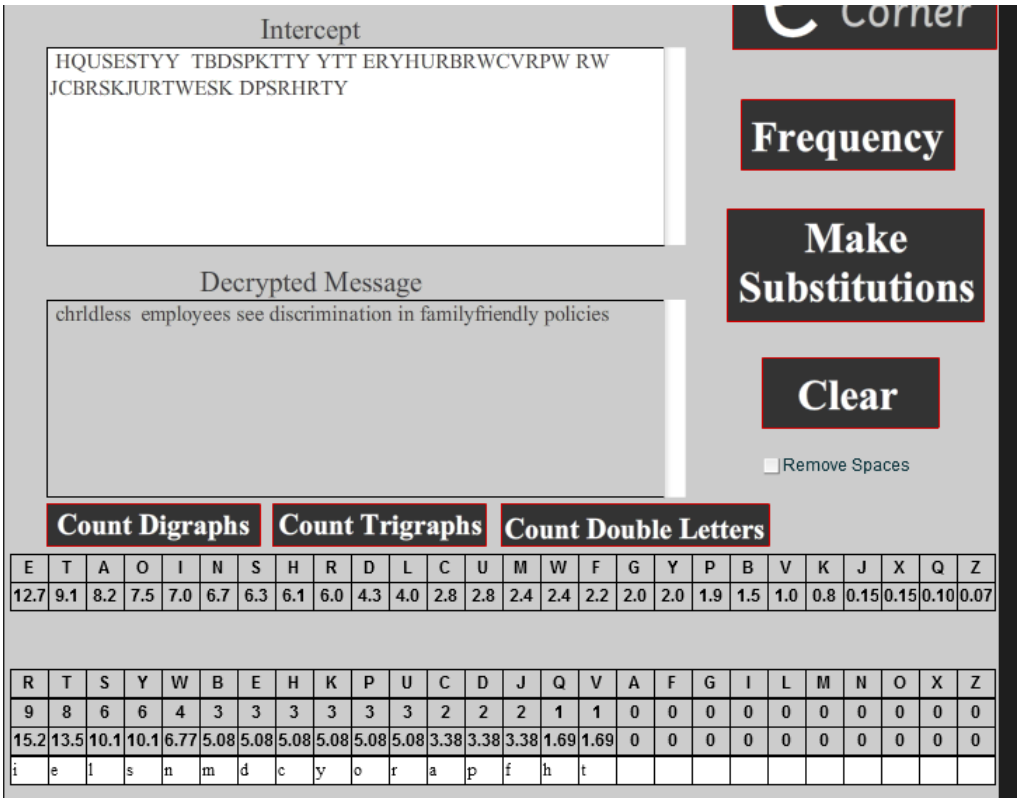
Figure 1: Letter frequency analysis and mapping for headline 1.



E	T	A	O	I	N	S	H	R	D	L	C	U	M	W	F	G	Y	P	B	V	K	J	X	Q	Z
12.7	9.1	8.2	7.5	7.0	6.7	6.3	6.1	6.0	4.3	4.0	2.8	2.8	2.4	2.4	2.2	2.0	2.0	1.9	1.5	1.0	0.8	0.15	0.15	0.10	0.07

H	N	B	X	K	O	R	V	A	J	S	W	Y	G	I	C	D	E	F	L	M	P	Q	T	U	Z
6	6	5	4	3	3	3	3	2	2	2	2	2	1	1	0	0	0	0	0	0	0	0	0	0	0
13.3	13.3	11.1	8.88	6.66	6.66	6.66	6.66	4.44	4.44	4.44	4.44	4.44	2.22	2.22	0	0	0	0	0	0	0	0	0	0	0
n	i	g	u	h	s	t	o	m	l	a	r	c	d	e											

Figure 2: Letter frequency analysis and mapping for headline 2.



E	T	A	O	I	N	S	H	R	D	L	C	U	M	W	F	G	Y	P	B	V	K	J	X	Q	Z
12.7	9.1	8.2	7.5	7.0	6.7	6.3	6.1	6.0	4.3	4.0	2.8	2.8	2.4	2.4	2.2	2.0	2.0	1.9	1.5	1.0	0.8	0.15	0.15	0.10	0.07

R	T	S	Y	W	B	E	H	K	P	U	C	D	J	Q	V	A	F	G	I	L	M	N	O	X	Z
9	8	6	6	4	3	3	3	3	3	3	2	2	2	1	1	0	0	0	0	0	0	0	0	0	0
15.2	13.5	10.1	10.1	6.77	5.08	5.08	5.08	5.08	5.08	5.08	3.38	3.38	3.38	1.69	1.69	0	0	0	0	0	0	0	0	0	0
i	e	l	s	n	m	d	c	y	o	r	a	p	f	h	t										

Figure 3: Letter frequency analysis and mapping for headline 5.