OverTheWire Krypton Series

Author: Jinay Shah

Tools:

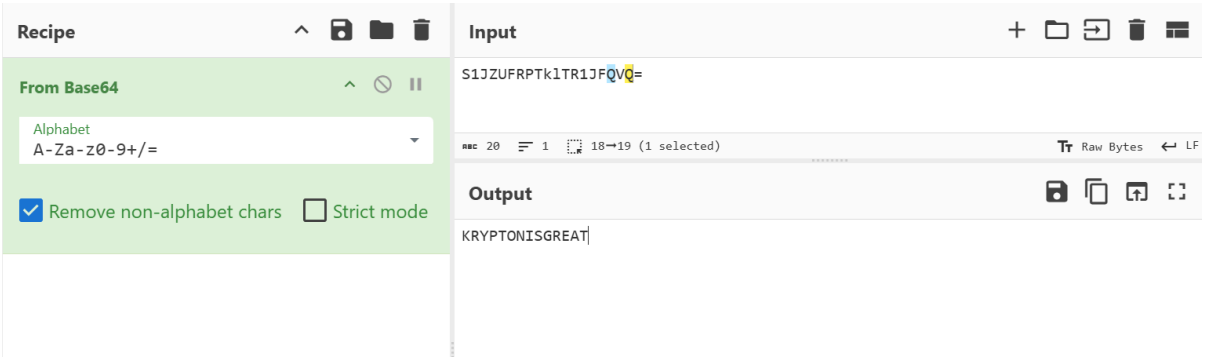
- CyberChef
- Kali Linux
- Wikipedia
- 101computing.net/frequency-analysis [frequency analysis]
- quipqiup.com [cryptography solver]
- dcode.fr/vigenere-cipher [Vigenère Cipher/Deciphering]

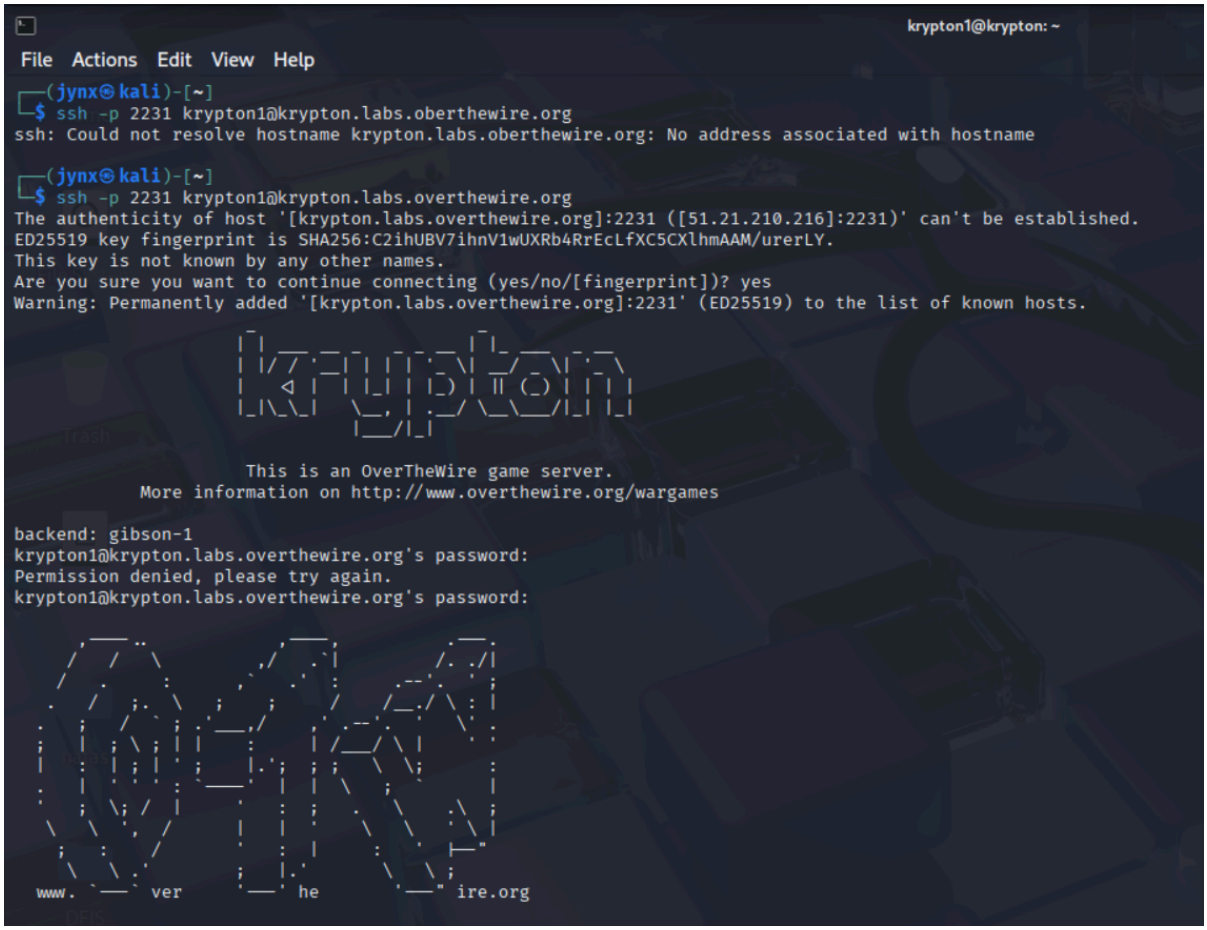
▼ Level 0

Level Info:

> Welcome to Krypton! The first level is easy. The following string encodes the password using Base64:
> `S1JZUFRPTkITR1JFQVQ=`
> Use this password to log in to krypton.labs.overthewire.org with username krypton1 using SSH on port 2231. You can find the files for other levels in /krypton/

→ This one seems to be easy, a simple base64 decode should give us our answer.





And it worked, we logged in. Lets move to the next real task now then...

Password:

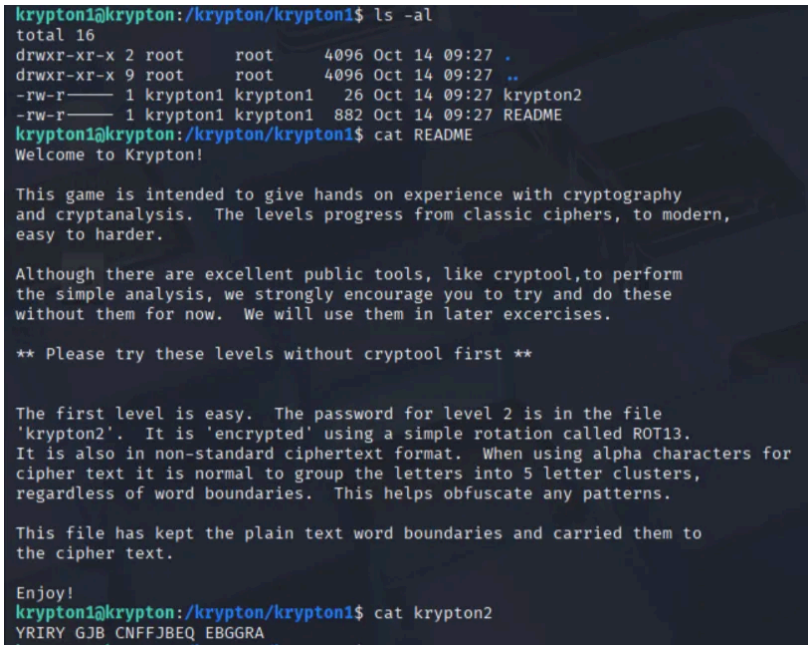
KRYPTONISGREAT

▼ Level 1

Level Info:

> The password for level 2 is in the file 'krypton2'. It is 'encrypted' using a simple rotation. It is also in non-standard ciphertext format. When using alpha characters for cipher text it is normal to group the letters into 5 letter clusters, regardless of word boundaries. This helps obfuscate any patterns. This file has kept the plain text word boundaries and carried them to the cipher text. Enjoy!

→ Let's begin with reading the `README` file here:



→ It is already made very clear that the encoding method used is ROT13, we can simple use any online ROT13 decoder to reveal the password. Here's a little intro about what ROT13 is:

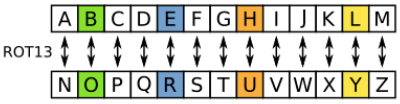
ROT13

31 languages

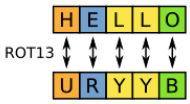
Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

ROT13 is a simple letter substitution cipher that replaces a letter with the 13th letter after it in the Latin alphabet.

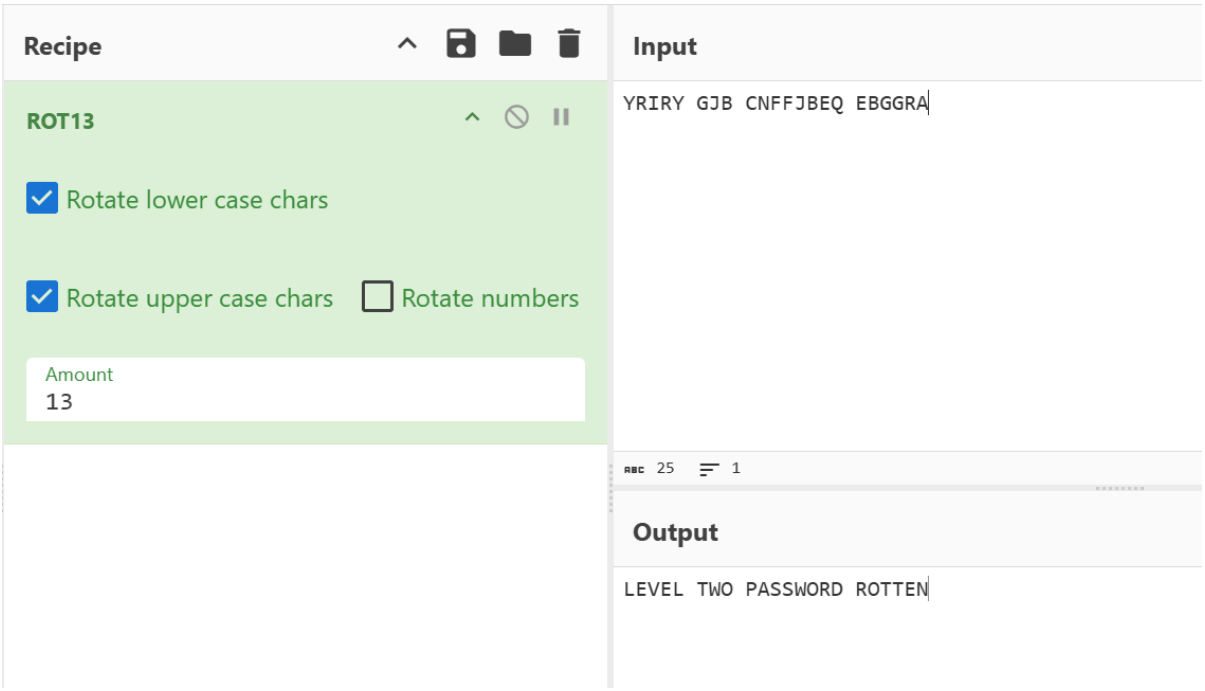


ROT13 is a special case of the Caesar cipher which was developed in ancient Rome, used by Julius Caesar in the 1st century BC.[1] An early entry on the Timeline of cryptography.

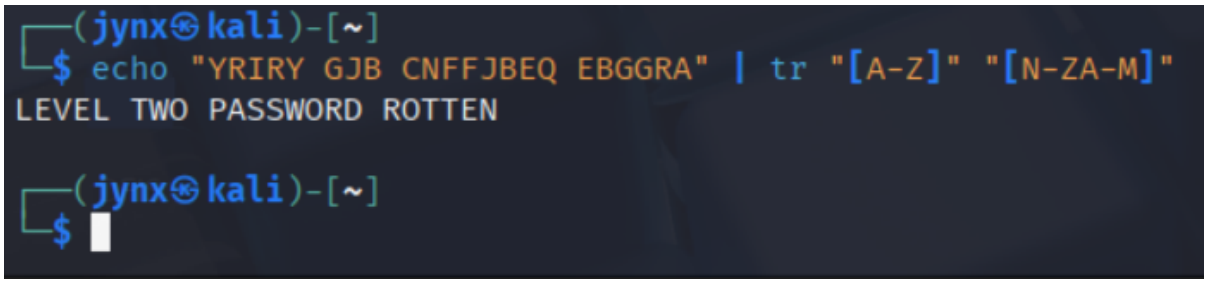


ROT13 can be referred by "Rotate13", "rotate by 13 places", hyphenated "ROT-13" or sometimes by its autonym "EBG13".

Let's now decode:



If you understand the simple logic on how we are only substituting the values for the 13th alphabet after it we can actually decode this in terminal itself with simple logic:



The `tr` command translates characters from the first set to the corresponding characters in the second set:

- **First set:** `[A-Z]` - All uppercase letters A through Z
- **Second set:** `[N-ZA-M]` - Letters N through Z, followed by A through M

The mapping:

- A → N (position 1 → position 14)
- B → O (position 2 → position 15)
- ...
- M → Z (position 13 → position 26)
- N → A (position 14 → position 1)
- O → B (position 15 → position 2)
- ...
- Z → M (position 26 → position 13)

Each letter is shifted by 13 positions, wrapping around at the end of the alphabet. This is exactly ROT13!

Important note: This command only handles uppercase letters. To handle both upper and lowercase, you'd need:

```
tr "[A-Za-z]" "[N-ZA-Mn-za-m]"
```

Since ROT13 is its own inverse (applying it twice gives you the original), this same command both **encodes** and **decodes** ROT13 text.

Password:

ROTTEN

Let's now go to the next level;

▼ Level 2

Level Info:

> ROT13 is a simple substitution cipher.
>
> Substitution ciphers are a simple replacement algorithm. In this example of a substitution cipher, we will explore a 'monoalphebetic' cipher. Monoalphebetic means, literally, "one alphabet" and you will see why.
>
> This level contains an old form of cipher called a 'Caesar Cipher'. A Caesar cipher shifts the alphabet by a set number. For example:
>
> `plain: a b c d e f g h i j k ...`
> `cipher: G H I J K L M N O P Q ...`
>
> In this example, the letter 'a' in plaintext is replaced by a 'G' in the ciphertext so, for example, the plaintext 'bad' becomes 'HGJ' in ciphertext.
>
> The password for level 3 is in the file krypton3. It is in 5 letter group ciphertext. It is encrypted with a Caesar Cipher. Without any further information, this cipher text may be difficult to break. You do not have direct access to the key, however you do have access to a program that will encrypt anything you wish to give it using the key. If you think logically, this is completely easy.
>
> One shot can solve it!
>
> Have fun.
>
> Additional Information:
>
> The `encrypt` binary will look for the keyfile in your current working directory. Therefore, it might be best to create a working direcory in /tmp and in there a link to the keyfile. As the `encrypt` binary runs setuid `krypton3`, you also need to give `krypton3` access to your working directory.
>
> Here is an example:
>
> ```
> krypton2@melinda:~$ mktemp -d
> /tmp/tmp.Wf2OnCpCDQ
> ```

```
krypton2@melinda:~$ cd /tmp/tmp.Wf2OnCpCDQ
krypton2@melinda:/tmp/tmp.Wf2OnCpCDQ$ ln -s /krypton/krypton2/keyfile.dat
krypton2@melinda:/tmp/tmp.Wf2OnCpCDQ$ ls
keyfile.dat
krypton2@melinda:/tmp/tmp.Wf2OnCpCDQ$ chmod 777 .
krypton2@melinda:/tmp/tmp.Wf2OnCpCDQ$ /krypton/krypton2/encrypt /etc/issue
krypton2@melinda:/tmp/tmp.Wf2OnCpCDQ$ ls
ciphertext  keyfile.dat
```

Lets just first look up, to know whether we understand what a Caesar cipher is:

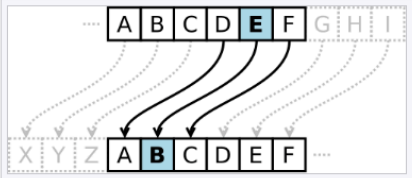
Caesar cipher

56 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

In cryptography, a **Caesar cipher**, also known as **Caesar's cipher**, the **shift cipher**, **Caesar's code**, or **Caesar shift**, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, `D` would be replaced by `A`, `E` would become `B`, and so on.[1] The method is named after Julius Caesar, who used it in his private correspondence.



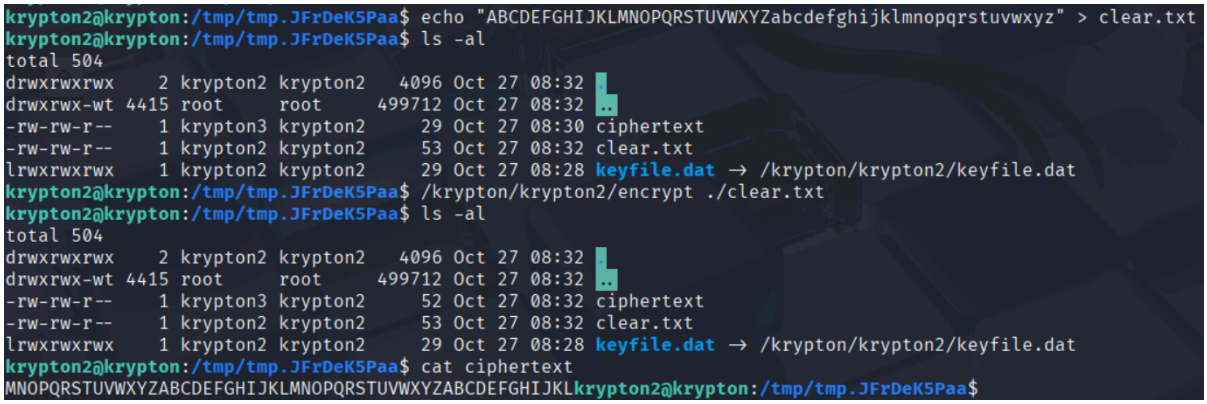
The action of a Caesar cipher is to replace each plaintext letter with a different one a fixed number of places down the alphabet. The cipher illustrated here uses a left shift of 3, so that (for example) each occurrence of `E` in the plaintext becomes `B` in the ciphertext.

The encryption step performed by a Caesar cipher is often incorporated as part of more complex schemes, such as the Vigenère cipher, and still has modern application in the ROT13 system. As with all single-alphabet substitution ciphers, the Caesar cipher is easily broken and in modern practice offers essentially no communications security.

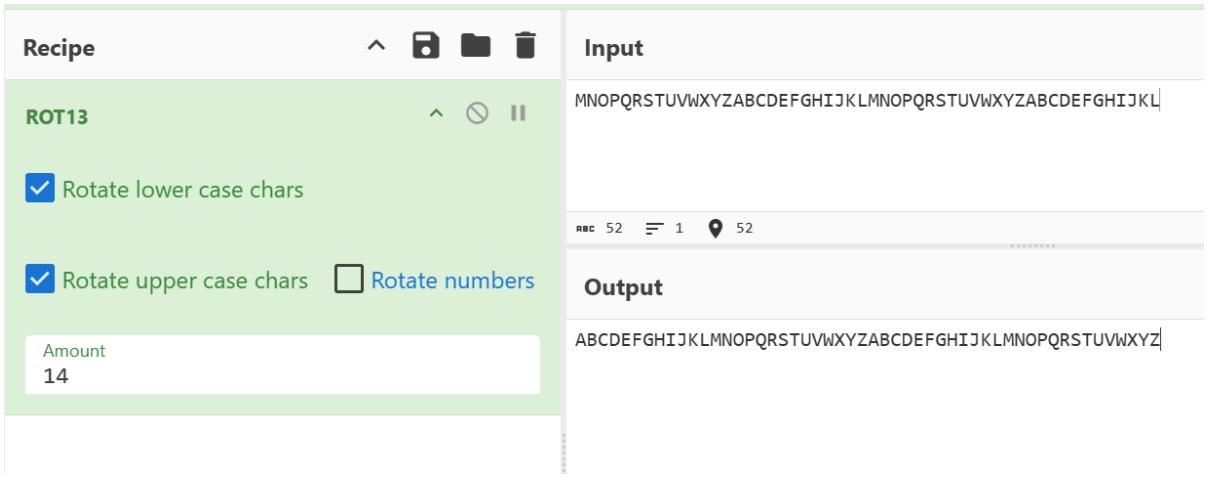
Okay next, lets look at our `krypton3` file:

```
krypton2@krypton:/krypton/krypton2$ ls
encrypt  keyfile.dat  krypton3  README
krypton2@krypton:/krypton/krypton2$ cat krypton3
OMQEMDUEQMEK
krypton2@krypton:/krypton/krypton2$
```

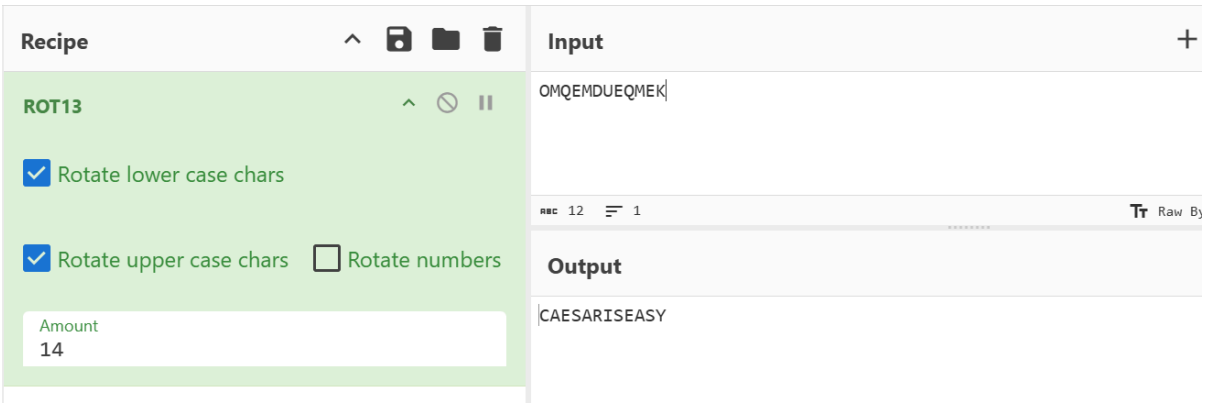
It look kind of a repetitive patter of sorts, okay... we have encrypt program that would behave similarly, and we know that a fixed patter beforehand is always followed here, so lets try encrypting ‘ABC’ with encrypt binary file:



And we nailed it, it doesn't rotate 13 times as inROT13 instead its just 12 times or ROT12, I don't know if that’s a legit thing lol, but okay now we can simply decode this using cyber chef, also notice how it automatically capitalize the lowercase alphabets as well to decode we simply can:



Now lets decode the actual cipher text:



And we got the password :)

Password:

```
CAESARISEASY
```

Let's move to the next level now that this is done and dusted;

▼ Level 3

Level Info:

> Well done. You've moved past an easy substitution cipher.
>
> The main weakness of a simple substitution cipher is repeated use of a simple key. In the previous exercise you were able to introduce arbitrary plaintext to expose the key. In this example, the cipher mechanism is not available to you, the attacker.
>
> However, you have been lucky. You have intercepted more than one message. The password to the next level is found in the file 'krypton4'. You have also found 3 other files. (found1, found2, found3)
>
> You know the following important details:
>
> - The message plaintexts are in American English (*** very important) - They were produced from the same key (*** even better!)
>
> Enjoy.

Let's begin with reading the `README` file first:

```
krypton3@krypton:/krypton/krypton3$ cat README
Well done.  You've moved past an easy substitution cipher.

Hopefully you just encrypted the alphabet a plaintext
to fully expose the key in one swoop.

The main weakness of a simple substitution cipher is
repeated use of a simple key.  In the previous exercise
you were able to introduce arbitrary plaintext to expose
the key.  In this example, the cipher mechanism is not
available to you, the attacker.

However, you have been lucky.  You have intercepted more
than one message.  The password to the next level is found
in the file 'krypton4'.  You have also found 3 other files.
(found1, found2, found3)

You know the following important details:

- The message plaintexts are in English (*** very important)
- They were produced from the same key (*** even better!)


Enjoy.
```

Lets no read all the other files and encrypted values we have as well as the `HINTS` .

```
krypton3@krypton:/krypton/krypton3$ cat krypton4
KSVVW BGSJD SVSIS VXBMN YQUUK BNWCU ANMJS krypton3@krypton:/krypton/krypton3$
krypton3@krypton:/krypton/krypton3$ cat found1
CGZNL YJBEN QYDLQ ZQSUQ NZCYD SNQVU BFGBK GQUQZ QSUQN UZCYD SNJDS UDCXJ ZCYDS NZQSU QNUZB WSBNZ QSUQN UDCXJ CUBGS BXJDS UCTYV SUJQG WTBUJ KCWSV LFGBK GSGZN LYJCB GJSZD
 GCHMS UCJCU QJLYS BXUMA UJCJM JCBGZ CYDSN CGKDC ZDSQZ DVSJJ SNCGJ DSYVQ CGJSO JCUNS YVQZS WALQV SJJSN UBTSX COSWG MTASN BXYBU CJCBG UWBKG JDSQV YDQAS JXBNS OQTYV SKCJ
D QUDCX JBXQK BMVWA SNSYV QZSWA LWAKB MVWAS ZBTSS QGWUB BGJDS TSJDB WCUGQ TSWQX JSNRM VCMUZ QSUQN KDBMU SWCJJ BZBTT MGCZQ JSKCJ DDCUE SGSNQ VUJDS SGZNL YJCBG UJSYY SNX
BN TSWAL QZQSU QNZCY DSNCU BXJSG CGZBN YBNQJ SWQUY QNJBX TBNSZ BTYVS OUZDS TSUUM ZDQUJ DSICE SGNSZ CYDSN QGWUJ CVVDQ UTBWS NGQYY VCZQJ CBGCG JDSNB JULUJ STQUK CJDQV VU
CGE VSQVY DQASJ UMAUJ CJMJC BGZCY DSNUJ DSZQS UQNZC YDSNC USQUC VLANB FSGQG WCGYN QZJCZ SBXXS NUSUU SGJCQ VVLGB ZBTTM GCZQJ CBGUS ZMNCJ LUDQF SUYSQ NSYNB WMZSW TBUJB X
DCUF GBKGK BNFAS JKSSG QGWDC USQNV LYVQL UKSNS TQCGV LZBTS WCSUQ GWDCU JBNCS UESGN SUDSN QCUSW JBJDS YSQFB XUBYD CUJCZ QJCBG QGWQN JCUJN LALJD SSGWB XJDSU COJSS GJDZS
GJMNL GSQJD SKNBJ STQCG VLJNQ ESWCS UMGJC VQABM JCGZV MWCGE DQTVS JFCGE VSQNQ GWTQZ ASJDZ BGUCW SNSWU BTSBX JDSXC GSUJS OQTYV SUCGJ DSSGE VCUDV QGEMQ ESCGD CUVQU JYDQU
 SDSKN BJSJN QECZB TSWCS UQVUB FGBKG QUNBT QGZSU QGWZB VVQAB NQJSW KCJDB JDSNY VQLKN CEDJU TQGLB XDCUY VQLUK SNSYM AVCUD SWCGS WCJCB GUBXI QNLCG EHMQV CJLQG WQZZM NQZL
W MNCGE DCUVC XSJCT SQGWC GJKBB XDCUX BNTSN JDSQJ NCZQV ZBVVS QEMSU YMAVC UDSWJ DSXCN UJXBV CBQZB VVSZJ SWSWC JCBGB XDCUW NQTQJ CZKBN FUJDQ JCGZV MWSWQ VVAMJ JKBBX JDS
YV QLUGB KNSZB EGCUS WQUUD QFSUY SQNSU krypton3@krypton:/krypton/krypton3$ cat found2
QVJDB MEDGB QJJSG WQGZS NSZBN WUXBN JDSYS NCBWU MNICI STBUJ ACBEN QYDSN UQENS SJDQJ UDQFS UYSQN SKQUS WMZQJ SWQJJ DSFCG EUGSK UZDBB VCGUJ NQJXB NWQXN SSUZD BBVZD QNJSN
 SWCGQ ABMJQ HMQNJ SNBXQ TCVSX NBTDC UDBTS ENQTT QNUZD BBVUI QNCSW CGHMQ VCJLW MNCGE JDSSV CPQAS JDQGS NQAMJ JDSZM NNCZM VMTKQ UWCZJ QJSWA LVQKJ DNBME DBMJS GEVQG WQGW
J DSUZD BBVKB MVWDQ ISYNB ICWSW QGCGJ SGUCI SSWMZ QJCBG CGVQJ CGENQ TTQNQ GWJDS ZVQUU CZUQJ JDSQE SBXUD QFSUY SQNST QNNCS WJDSL SQNBV WQGGS DQJDQ KQLJD SZBGU CUJBN LZB
MN JBXJD SWCBZ SUSBX KBNZS UJSNC UUMSW QTQNN CQESV CZSGZ SBGGB ISTAS NJKBB XDQJD QKQLU GSCED ABMNU YBUJS WABGW UJDSG SOJWQ LQUUM NSJLJ DQJJD SNSKS NSGBC TYSWC TSGJU JB
JDS TQNNC QESJD SZBMY VSTQL DQISQ NNQGE SWJDS ZSNST BGLCG UBTSD QUJSU CGZSJ DSKBN ZSUJS NZDQG ZSVVB NQVVB KSWJD STQNN CQESA QGGUJ BASNS QWBGZ SCGUJ SQWBX JDSMU MQVJD N
SSJC TSUQG GSUYN SEGQG ZLZBM VWDQI SASSG JDSNS QUBGX BNJDC UUCOT BGJDU QXJSN JDSTQ NNCQE SUDSE QISAC NJDJB QWQME DJSNU MUQGG QKDBK QUAQY JCUSW BGTQL JKCGU UBGDQ TGSJQ
GWWQM EDJSN RMWCJ DXBVV BKSWQ VTBUJ JKBLS QNUVQ JSNQG WKSNS AQYJC USWBG XSANM QNLDQ TGSJW CSWBX MGFGB KGZQM USUQJ JDSQE SBXQG WKQUA MNCSW BGQME MUJQX JSNJD SACNJ DBXJD
 SJKCG UJDSN SQNSX SKDCU JBNCZ QVJNQ ZSUBX UDQFS UYSQN SMGJC VDSCU TSGJC BGSWQ UYQNJ BXJDS VBGWB GJDSQ JNSUZ SGSCG ASZQM USBXJ DCUEQ YUZDB VQNUN SXSNJ BJDSL SQNUA SJKS
S GQGWQ UUDQF SUYSQ NSUVB UJLSQ NUACB ENQYD SNUQJ JSTYJ CGEJB QZZBM GJXBN JDCUY SNCBW DQISN SYBNJ SWTQG LQYBZ NLYDQ VUJBN CSUGC ZDBVQ UNBKS UDQFS UYSQN SUXCN UJACB ENQ
YD SNNSZ BMGJS WQUJN QJXBN WVSES GWJDQ JUDQF SUYSQ NSXVS WJDSJ BKGXB NVBGW BGJBS UZQYS YNBUS ZMJCB GXBNW SSNYB QZDCG EQGBJ DSNSC EDJSS GJDZS GJMNL UJBNL DQUUD QFSUY SQ
NSU JQNJC GEDCU JDSQJ NCZQV ZQNSS NTCGW CGEJD SDBNU SUBXJ DSQJN SYQJN BGUCG VBGWB GRBDG QMANS LNSYB NJSWJ DQJUD QFSUY SQNSD QWASS GQZBM GJNLU ZDBBV TQUJS NUBTS JKSGJ C
SJDZ SGJMN LUZDB VQNUD QISUM EESUJ SWJDQ JUDQF SUYSQ NSTQL DQISA SSGST YVBLS WQUQU ZDBBV TQUJS NALQV SQQGW SNDBE DJBGB XVQGZ QUDCN SQZQJ DBVCZ VQGWB KGSNK DBGQT SWQZS
NJQCG KCVVC QTUDQ FSUDQ XJSCG DCUKC VVGBS ICWSG ZSUMA UJQGJ CQJSU UMZDU JBNCS UBJDS NJDQG DSQNU QLZBV VSZJS WQXJS NDCUW SQJDkrypton3@krypton:/krypton/krypton3$
krypton3@krypton:/krypton/krypton3$
krypton3@krypton:/krypton/krypton3$ cat found3
DSNSM YBGVS ENQGW QNBUS KCJDQ ENQIS QGWUJ QJSVL QCNQG WANBM EDJTS JDSAS SJVSX NBTQE VQUUZ QUSCG KDCZD CJKQU SGZVB USWCJ KQUQA SQMJC XMVUZ QNQAQ SMUQG WQJJD QJJCT SMGFG
 BKGJB GQJMN QVCUJ UBXZB MNUSQ ENSQJ YNCPS CGQUZ CSGJC XCZYB CGJBX ICSKJ DSNSK SNSJK BNBMG WAVQZ FUYBJ UGSQN BGSSO JNSTC JLBXJ DSAQZ FQGWQ VBGEB GSGSQ NJDSB JDSNJ DSUZ
Q VSUKS NSSOZ SSWCG EVLDQ NWQGW EVBUU LKCJD QVVJD SQYYS QNQGZ SBXAM NGCUD SWEBV WJDSK SCEDJ BXJDS CGUSZ JKQUI SNLNS TQNFQ AVSQG WJQFC GEQVV JDCGE UCGJB ZBGUC WSNQJ CBG
CZ BMVWD QNWVL AVQTS RMYCJ SNXBN DCUBY CGCBG NSUYS ZJCGE CJkrypton3@krypton:/krypton/krypton3$
krypton3@krypton:/krypton/krypton3$
krypton3@krypton:/krypton/krypton3$ cat HINT1
Some letters are more prevalent in English than others.
krypton3@krypton:/krypton/krypton3$ cat HINT2
"Frequency Analysis" is your friend.
```

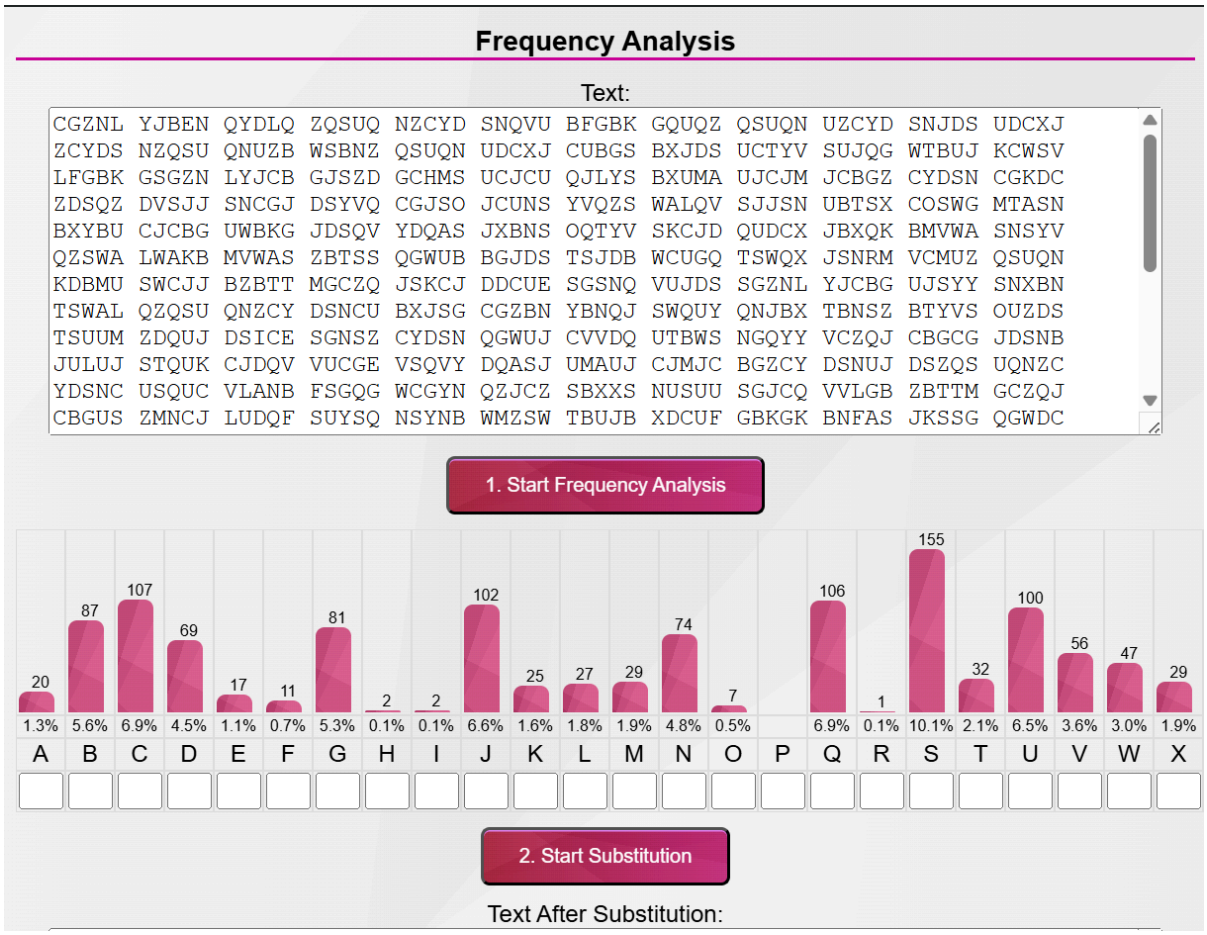
It seems the words are very common and used repetitively, the level also suggest one important fact:

> The message plaintexts are in American English (*** very important)

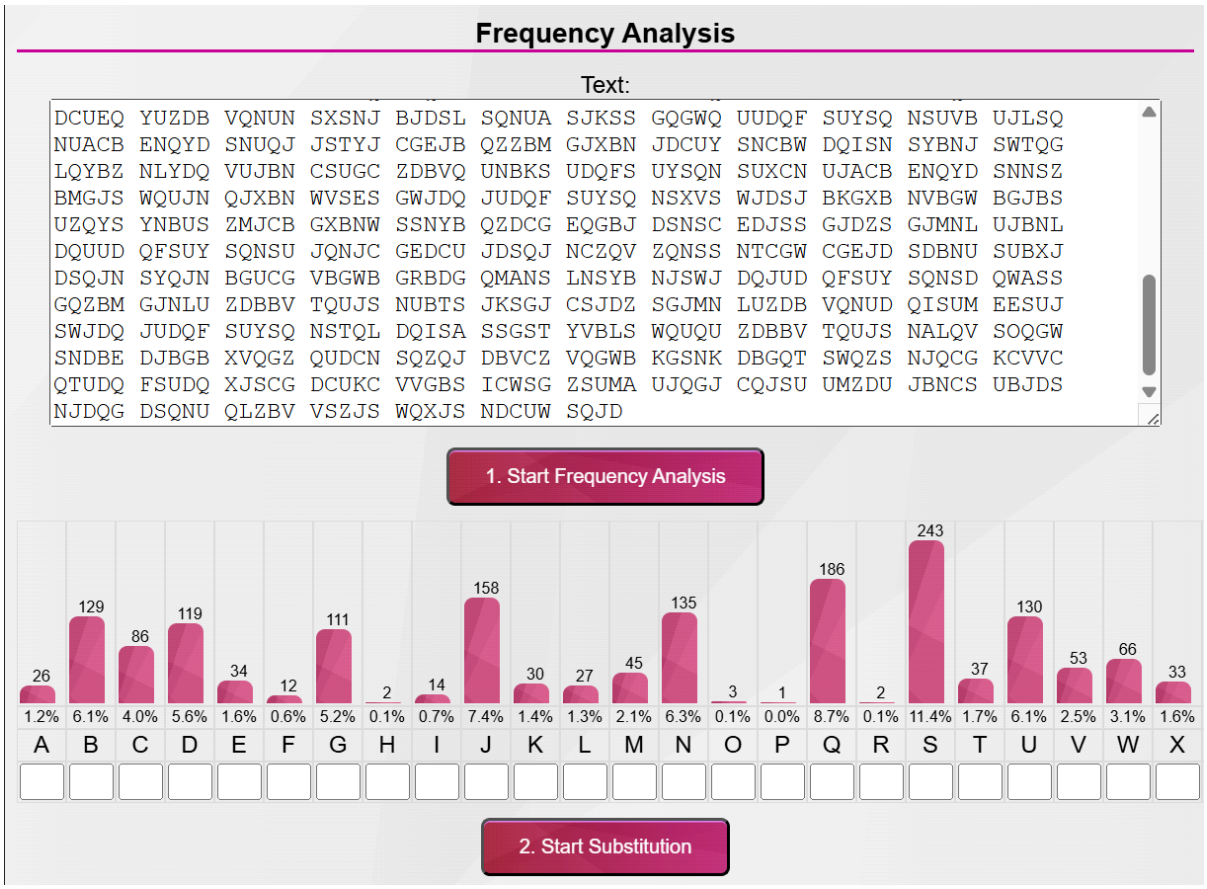
lets use some online tool to determine the frequency analysis, what is frequency analysis though:

In cryptography, frequency analysis is the study of the **frequency of letters** or groups of letters in a ciphertext. The method is used as an aid to breaking **substitution ciphers** (e.g. `mono-alphabetic substitution cipher`, `Caesar shift cipher`, `Vatsyayana cipher`).

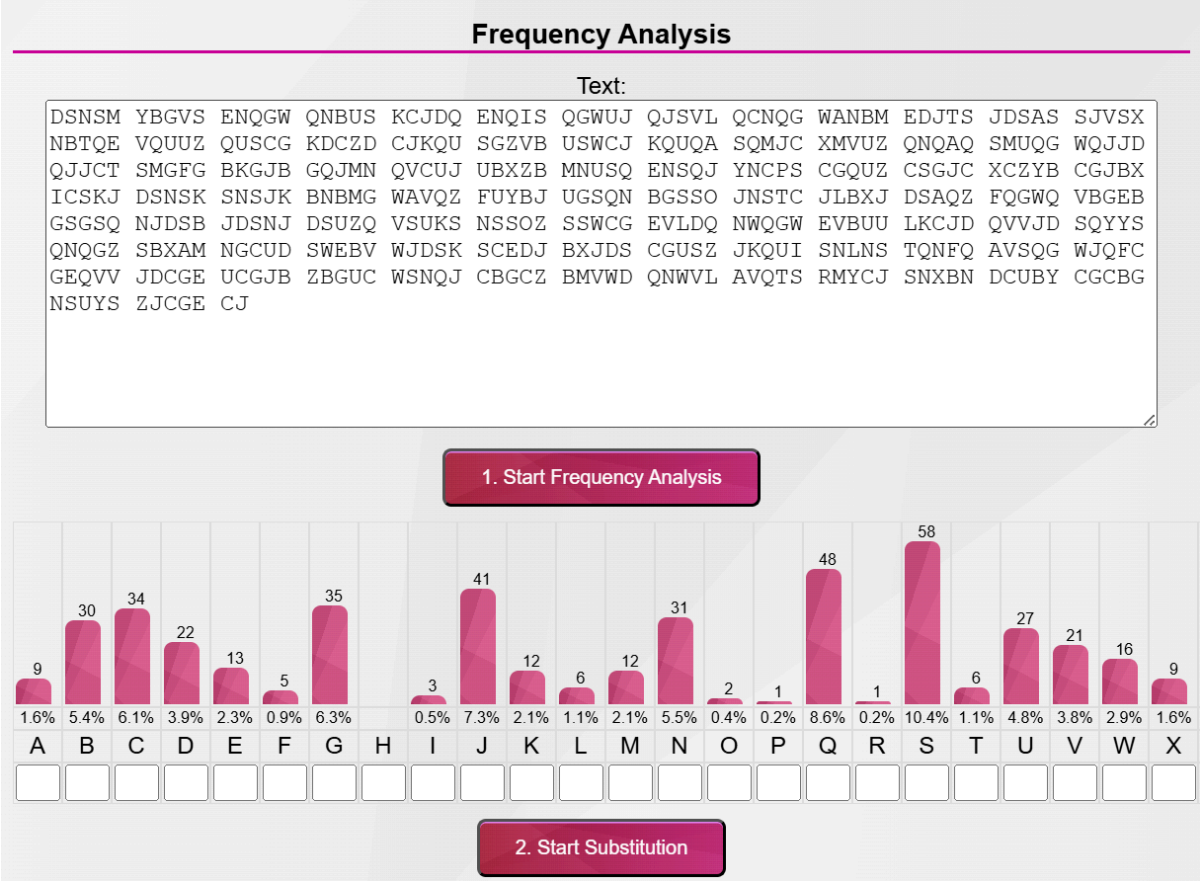
Frequency analysis consists of **counting the occurrence of each letter** in a text. Frequency analysis is based on the fact that, in any given piece of text, certain letters and combinations of letters occur with varying frequencies. For instance, given a section of English language, letters **E, T, A and O** are the most common, while letters Z, Q and X are not as frequently used.
`found1` **file:**


Frequency Analysis
Text:
CGZNL YJBEN QYDLQ ZQSUQ NZCYD SNQVU BFGBK GQUQZ QSUQN UZCYD SNJDS UDCXJ
ZCYDS NZQSU QNUZB WSBNZ QSUQN UDCXJ CUBGS BXJDS UCTYV SUJQG WTBUJ KCWSV
LFGBK GSGZN LYJCB GJSZD GCHMS UCJCU QJLYS BXUMA UJCJM JCBGZ CYDSN CGKDC
ZDSQZ DVSJJ SNCGJ DSYVQ CGJSO JCUNS YVQZS WALQV SJJSN UBTSX COSWG MTASN
BXYBU CJCBG UWBKG JDSQV YDQAS JXBNS OQTYV SKCJD QUDCX JBXQK BMVWA SNSYV
QZSWA LWAKB MVWAS ZBTSS QGWUB BGJDS TSJDB WCUGQ TSWQX JSNRM VCMUZ QSUQN
KDBMU SWCJJ BZBTT MGCZQ JSKCJ DDCUE SGSNQ VUJDS SGZNL YJCBG UJSYY SNXBN
TSWAL QZQSU QNZCY DSNCU BXJSG CGZBN YBNQJ SWQUY QNJBX TBNSZ BTYVS OUZDS
TSUUM ZDQUJ DSICE SGNSZ CYDSN QGWUJ CVVDQ UTBWS NGQYY VCZQJ CBGCG JDSNB
JULUJ STQUK CJDQV VUCGE VSQVY DQASJ UMAUJ CJMJC BGZCY DSNUJ DSZQS UQNZC
YDSNC USQUC VLANB FSGQG WCGYN QZJCZ SBXXS NUSUU SGJCQ VVLGB ZBTTM GCZQJ
CBGUS ZMNCJ LUDQF SUYSQ NSYNB WMZSW TBUJB XDCUF GBKGK BNFAS JKSSG QGWDC
1. Start Frequency Analysis
20 87 107 69 17 11 81 2 2 102 25 27 29 74 7 106 1 155 32 100 56 47 29
1.3% 5.6% 6.9% 4.5% 1.1% 0.7% 5.3% 0.1% 0.1% 6.6% 1.6% 1.8% 1.9% 4.8% 0.5% 6.9% 0.1% 10.1% 2.1% 6.5% 3.6% 3.0% 1.9%
A B C D E F G H I J K L M N O P Q R S T U V W X
2. Start Substitution
Text After Substitution:

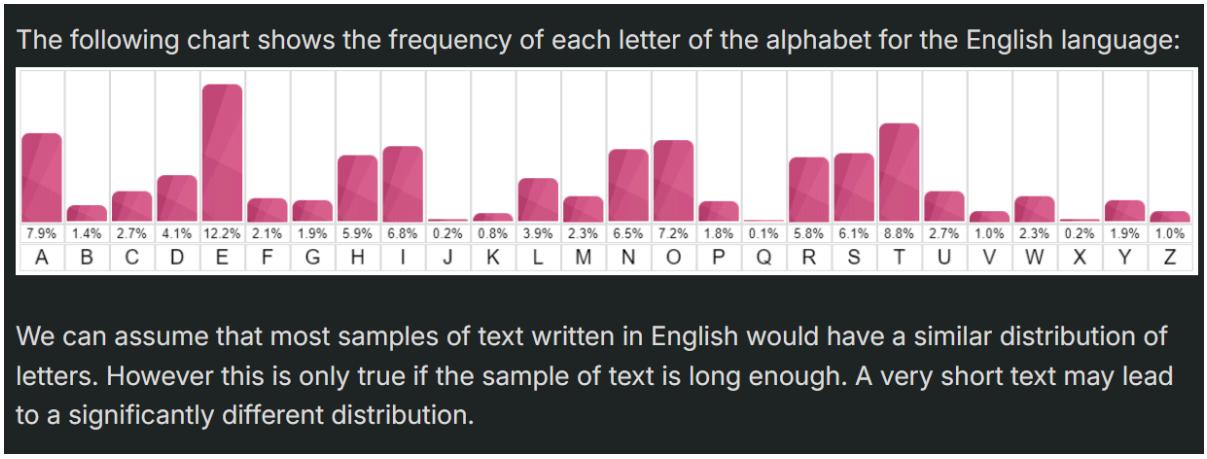
found2 file:


Frequency Analysis
Text:
DCUEQ YUZDB VQNUN SXSNJ BJDSL SQNUA SJKSS GQGWQ UUDQF SUYSQ NSUVB UJLSQ
NUACB ENQYD SNUQJ JSTYJ CGEJB QZZBM GJXBN JDCUY SNCBW DQISN SYBNJ SWTQG
LQYBZ NLYDQ VUJBN CSUGC ZDBVQ UNBKS UDQFS UYSQN SUXCN UJACB ENQYD SNNSZ
BMGJS WQUJN QJXBN WVSES GWJDQ JUDQF SUYSQ NSXVS WJDSJ BKGXB NVBGW BGJBS
UZQYS YNBUS ZMJCB GXBNW SSNYB QZDCG EQGBJ DSNSC EDJSS GJDZS GJMNL UJBNL
DQUUD QFSUY SQNSU JQNJC GEDCU JDSQJ NCZQV ZQNSS NTCGW CGEJD SDBNU SUBXJ
DSQJN SYQJN BGUCG VBGWB GRBDG QMANS LNSYB NJSWJ DQJUD QFSUY SQNSD QWASS
GQZBM GJNLU ZDBBV TQUJS NUBTS JKSGJ CSJDZ SGJMN LUZDB VQNUD QISUM EESUJ
SWJDQ JUDQF SUYSQ NSTQL DQISA SSGST YVBLS WQUQU ZDBBV TQUJS NALQV SOQGW
SNDBE DJBGB XVQGZ QUDCN SQZQJ DBVCZ VQGWB KGSNK DBGQT SWQZS NJQCG KCVVC
QTUDQ FSUDQ XJSCG DCUKC VVGBS ICWSG ZSUMA UJQGJ CQJSU UMZDU JBNCS UBJDS
NJDQG DSQNU QLZBV VSZJS WQXJS NDCUW SQJD
1. Start Frequency Analysis
26 129 86 119 34 12 111 2 14 158 30 27 45 135 3 1 186 2 243 37 130 53 66 33
1.2% 6.1% 4.0% 5.6% 1.6% 0.6% 5.2% 0.1% 0.7% 7.4% 1.4% 1.3% 2.1% 6.3% 0.1% 0.0% 8.7% 0.1% 11.4% 1.7% 6.1% 2.5% 3.1% 1.6%
A B C D E F G H I J K L M N O P Q R S T U V W X
2. Start Substitution

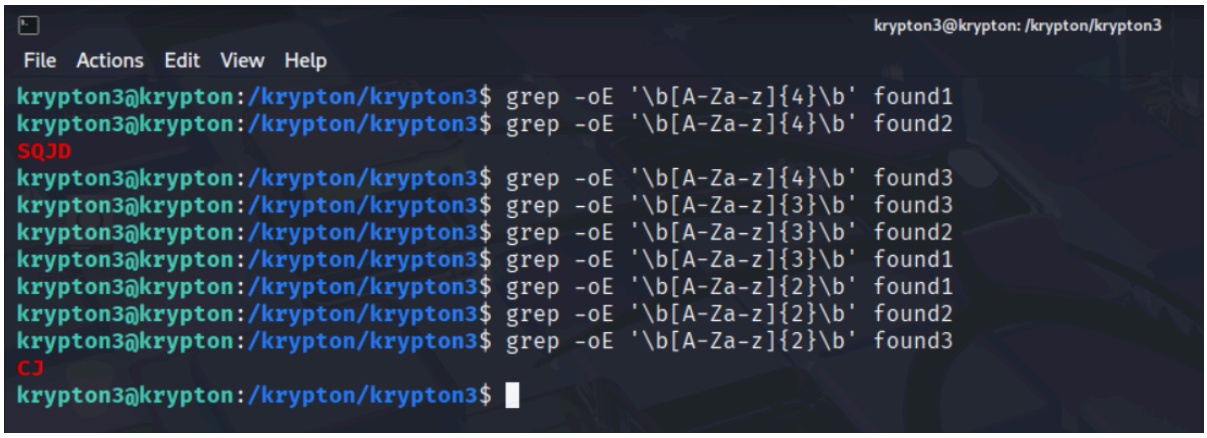
found3 **file:**



Common letter frequencies:



Time for smart guessing, CJ is a two lettered work with a high probability of being the actual word '`is/am/of/in/to`', we can also try and find the word 'THE', 3 lettered since E is the most common word and generally all the other words are 4-5 letters:



cant find '`THE`' for a 3 lettered word substitute, but `SQJD` could be '`THIS`'.
Okay this is me documenting after hours of failed trial and error, permutations and combinations and everything else you can think of under the sky, I admit my shortcomings here lol, but I did find a website that actually did the work as intended:

Files [found1, found2, found3]:

CGZNL YJBEN QYDLQ ZQSUQ NZCYD SNQVU BFGBK GQUQZ QSUQN UZ CYD SNJDS UDCXJ ZCYDS NZQSU QNUZB WSBNZ QSUQN UDCXJ CUBG S BXJDS UCTYV SUJQG WTBUJ KCWSV LFGBK GSGZN LYJCB GJSZD G CHMS UCJCU QJLYS BXUMA UJCJM JCBGZ CYDSN CGKDC ZDSQZ DVS JJ SNCGJ DSYVQ CGJSO JCUNS YVQZS WALQV SJJSN UBTSX COSWG MTASN BXYBU CJCBG UWBKG JDSQV YDQAS JXBNS OQTYV SKCJD QU DCX JBXQK BMVWA SNSYV QZSWA LWAKB MVWAS ZBTSS QGWUB BGJ DS TSJDB WCUGQ TSWQX JSNRM VCMUZ QSUQN KDBMU SWCJJ BZBT T MGCZQ JSKCJ DDCUE SGSNQ VUJDS SGZNL YJCBG UJSYY SNXBN T SWAL QZQSU QNZCY DSNCU BXJSG CGZBN YBNQJ SWQUY QNJBX TB NSZ BTYVS OUZDS TSUUM ZDQUJ DSICE SGNSZ CYDSN QGWUJ CVVD Q UTBWS NGQYY VCZQJ CBGCG JDSNB JULUJ STQUK CJDQV VUCGE V SQVY DQASJ UMAUJ CJMJC BGZCY DSNUJ DSZQS UQNZC YDSNC USQ UC VLANB FSGQG WCGYN QZJCZ SBXXS NUSUU SGJCQ VVLGB ZBTTM GCZQJ CBGUS ZMNCJ LUDQF SUYSQ NSYNB WMZSW TBUJB XDCUF G BKGK BNFAS JKSSG QGWDC USQNV LYVQL UKSNS TQCGV LZBTS WCS UQ GWDCU JBNCS UESGN SUDSN QCUSW JBJDS YSQFB XUBYD CUJC Z QJCBG QGWQN JCUJN LALJD SSGWB XJDSU COJSS GJDZS GJMNL G

SOJD SKNBJ STQCG VLJNQ ESWCS UMGJC VQABM JCGZV MWCGE DQ
TVS JFCGE VSQNQ GWTQZ ASJDZ BGUCW SNSWU BTSBX JDSXC GSUJ
S OQTYV SUCGJ DSSGE VCUDV QGEMQ ESCGD CUVQU JYDQU SDSKN
BJSJN QECZB TSWCS UQVUB FGBKG QUNBT QGZSU QGWZB VVQAB NQ
JSW KCJDB JDSNY VQLKN CEDJU TQGLB XDCUY VQLUK SNSYM AVCU
D SWCGS WCJCB GUBXI QNLCG EHMQV CJLQG WQZZM NQZLW MNCG
E DCUVC XSJCT SQGWC GJKBB XDCUX BNTSN JDSQJ NCZQV ZBVVS Q
EMSU YMAVC UDSWJ DSXCN UJXBV CBQZB VVSZJ SWSWC JCBGB XD
CUW NQTQJ CZKBN FUJDQ JCGZV MWSWQ VVAMJ JKBBX JDSYV QLU
GB KNSZB EGCUS WQUUD QFSUY SQNSU QVJDB MEDGB QJJSG WQGZ
S NSZBN WUXBN JDSYS NCBWU MNICI STBUJ ACBEN QYDSN UQENS S
JDQJ UDQFS UYSQN SKQUS WMZQJ SWQJJ DSFCG EUGSK UZDBB VCG
UJ NQJXB NWQXN SSUZD BBVZD QNJSN SWCGQ ABMJQ HMQNJ SNBX
Q TCVSX NBTDC UDBTS ENQTT QNUZD BBVUI QNCSW CGHMQ VCJLW
MNCGE JDSSV CPQAS JDQGS NQAMJ JDSZM NNCZM VMTKQ UWCZJ Q
JSWA LVQKJ DNBME DBMJS GEVQG WQGWJ DSUZD BBVKB MVWDQ IS
YNB ICWSW QGCGJ SGUCI SSWMZ QJCBG CGVQJ CGENQ TTQNQ GWJ
DS ZVQUU CZUQJ JDSQE SBXUD QFSUY SQNST QNNCS WJDSL SQNBV
WQGGS DQJDQ KQLJD SZBGU CUJBN LZBMN JBXJD SWCBZ SUSBX KB
NZS UJSNC UUMSW QTQNN CQESV CZSGZ SBGGB ISTAS NJKBB XDQJD
QKQLU GSCED ABMNU YBUJS WABGW UJDSG SOJWQ LQUUM NSJLJ D
QJJD SNSKS NSGBC TYSWC TSGJU JBJDS TQNNC QESJD SZBMY VSTQ
L DQISQ NNQGE SWJDS ZSNST BGLCG UBTSD QUJSU CGZSJ DSKBN ZS
UJS NZDQG ZSVVB NQVVB KSWJD STQNN CQESA QGGUJ BASNS QWB
GZ SCGUJ SQWBX JDSMU MQVJD NSSJC TSUQG GSUYN SEGQG ZLZBM
VWDQI SASSG JDSNS QUBGX BNJDC UUCOT BGJDU QXJSN JDSTQ NN
CQE SUDSE QISAC NJDJB QWQME DJSNU MUQGG QKDBK QUAQY JCUS
W BGTQL JKCGU UBGDQ TGSJQ GWWQM EDJSN RMWCJ DXBVV BKSW
Q VTBUJ JKBLS QNUVQ JSNQG WKSNS AQYJC USWBG XSANM QNLDQ
TGSJW CSWBX MGFGB KGZQM USUQJ JDSQE SBXQG WKQUA MNCSW
BGQME MUJQX JSNJD SACNJ DBXJD SJKCG UJDSN SQNSX SKDCU JB
NCZ QVJNQ ZSUBX UDQFS UYSQN SMGJC VDSCU TSGJC BGSWQ UYQ
NJ BXJDS VBGWB GJDSQ JNSUZ SGSCG ASZQM USBXJ DCUEQ YUZDB
VQNUN SXSNJ BJDSL SQNUA SJKSS GQGWQ UUDQF SUYSQ NSUVB UJ
LSQ NUACB ENQYD SNUQJ JSTYJ CGEJB QZZBM GJXBN JDCUY SNCB
W DQISN SYBNJ SWTQG LQYBZ NLYDQ VUJBN CSUGC ZDBVQ UNBKS U

```
DQFS UYSQN SUXCN UJACB ENQYD SNNSZ BMGJS WQUJN QJXBN WV
SES GWJDQ JUDQF SUYSQ NSXVS WJDSJ BKGXB NVBGW BGJBS UZQY
S YNBUS ZMJCB GXBNW SSNYB QZDCG EQGBJ DSNSC EDJSS GJDZS G
JMNL UJBNL DQUUD QFSUY SQNSU JQNJC GEDCU JDSQJ NCZQV ZQN
SS NTCGW CGEJD SDBNU SUBXJ DSQJN SYQJN BGUCG VBGWB GRBD
G QMANS LNSYB NJSWJ DQJUD QFSUY SQNSD QWASS GQZBM GJNLU
ZDBBV TQUJS NUBTS JKSGJ CSJDZ SGJMN LUZDB VQNUD QISUM EES
UJ SWJDQ JUDQF SUYSQ NSTQL DQISA SSGST YVBLS WQUQU ZDBBV
TQUJS NALQV SOQGW SNDBE DJBGB XVQGZ QUDCN SQZQJ DBVCZ VQ
GWB KGSNK DBGQT SWQZS NJQCG KCVVC QTUDQ FSUDQ XJSCG DCU
KC VVGBS ICWSG ZSUMA UJQGJ CQJSU UMZDU JBNCS UBJDS NJDQG
DSQNU QLZBV VSZJS WQXJS NDCUW SQJD DSNSM YBGVS ENQGW QN
BUS KCJDQ ENQIS QGWUJ QJSVL QCNQG WANBM EDJTS JDSAS SJVSX
NBTQE VQUUZ QUSCG KDCZD CJKQU SGZVB USWCJ KQUQA SQMJC X
MVUZ QNQAQ SMUQG WQJJD QJJCT SMGFG BKGJB GQJMN QVCUJ UB
XZB MNUSQ ENSQJ YNCPS CGQUZ CSGJC XCZYB CGJBX ICSKJ DSNSK
SNSJK BNBMG WAVQZ FUYBJ UGSQN BGSSO JNSTC JLBXJ DSAQZ FQG
WQ VBGEB GSGSQ NJDSB JDSNJ DSUZQ VSUKS NSSOZ SSWCG EVLDQ
NWQGW EVBUU LKCJD QVVJD SQYYS QNQGZ SBXAM NGCUD SWEBV
WJDSK SCEDJ BXJDS CGUSZ JKQUI SNLNS TQNFQ AVSQG WJQFC GEQ
VV JDCGE UCGJB ZBGUC WSNQJ CBGCZ BMVWD QNWVL AVQTS RMYC
J SNXBN DCUBY CGCBG NSUYS ZJCGE
```

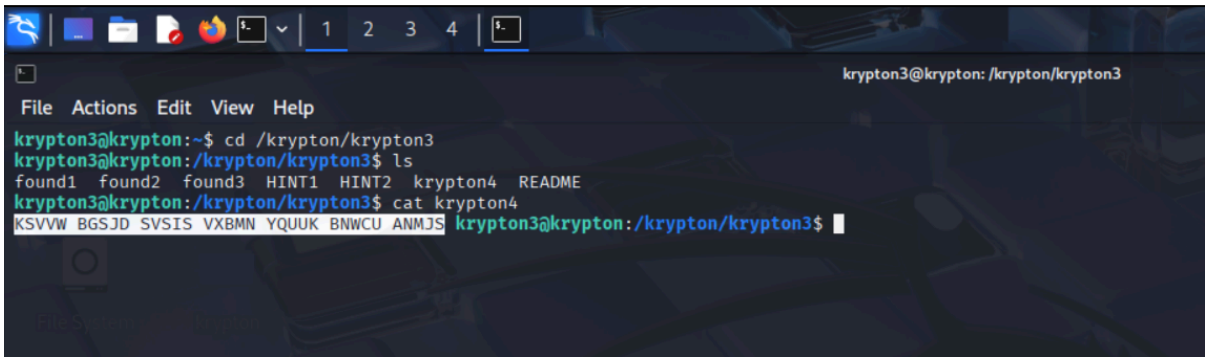
Analysis using `quipqiup`:

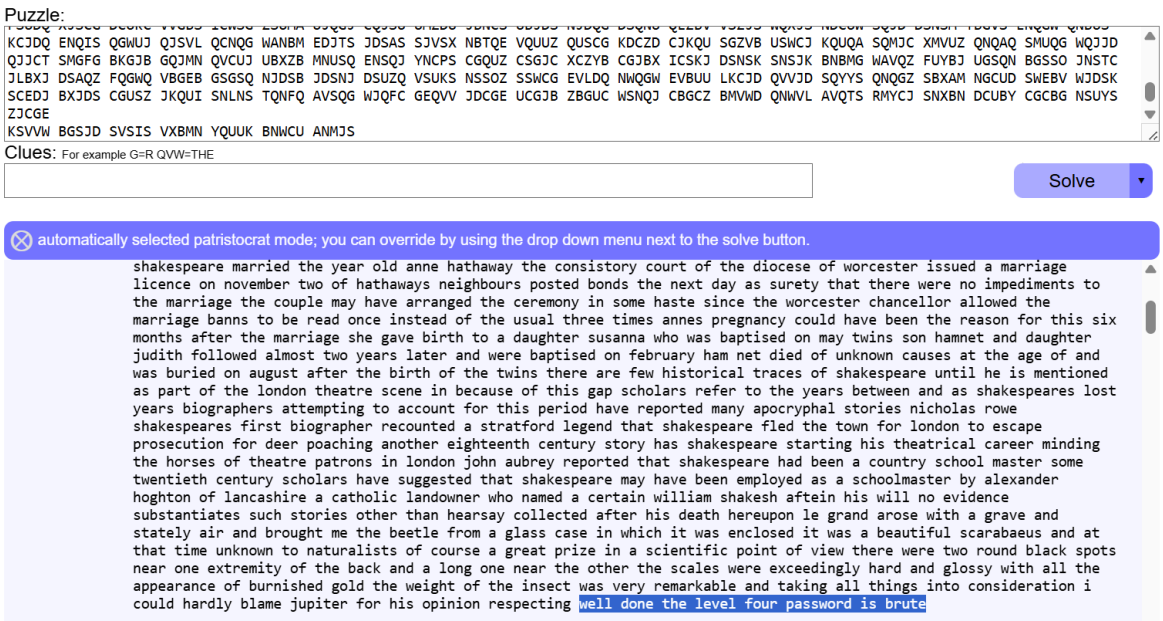
```
in cryptography a caesar cipher also known as a caesars cipher the shift ci
pher caesars code or caesar shift is one of the simplest and most widely k
nown encryption techniques it is a type of substitution cipher in which eac
h letter in the plain text is replaced by a letter some fixed number of positio
ns down the alphabet for example with a shift of a would be replaced by db
would become e and soon the method is named after julius caesar who us
ed it to communicate with his generals the encryption step performed by a
caesar cipher is often incorporated as part of more complex schemes such
as the vigenre cipher and still has modern application in the rot system as
with all single alphabet substitution ciphers the caesar cipher is easily brok
en and in practice offers essentially no communication security shakespea
```

re produced most of his known work between and his early plays were mai nly comedies and histories genre she raised to the peak of sophistication a nd artistry by the end of the sixteenth century next he wrote mainly tragedi es until about including hamlet king lear and macbeth considered some of t he finest examples in the english language in his last phase he wrote tragic omedies also known as romances and collaborated with other playwrights many of his plays were published in editions of varying quality and accurac y during his lifetime and in two of his former theatrical colleagues publishe d the first folio a collected edition of his dramatic works that included all bu t two of the plays now recognised as shakespeares although no attendanc e records for the period survive most biographers agree that shakespeare was educated at the kings new school in stratford a free school chartered i n about a quarter of a mile from his home grammar schools varied in qualit y during the elizabethan era but the curriculum was dictated by law throug hout england and the school would have provided an intensive education i n latin grammar and the classics at the age of shakespeare married the yea r old anne hathaway the consistory court of the diocese of worcester issue d a marriage licence on november two of hathaways neighbours posted bo nds the next day as surety that there were no impediments to the marriage the couple may have arranged the ceremony in some haste since the worc ester chancellor allowed the marriage banns to be read once instead of the usual three times annes pregnancy could have been the reason for this six months after the marriage she gave birth to a daughter susanna who was b aptised on may twins son hamnet and daughter judith followed almost two years later and were baptised on february ham net died of unknown cause s at the age of and was buried on august after the birth of the twins there a re few historical traces of shakespeare until he is mentioned as part of the l ondon theatre scene in because of this gap scholars refer to the years bet ween and as shakespeares lost years biographers attempting to account fo r this period have reported many apocryphal stories nicholas rowe shakesp eares first biographer recounted a stratford legend that shakespeare fled t he town for london to escape prosecution for deer poaching another eight eenth century story has shakespeare starting his theatrical career minding the horses of theatre patrons in london john aubrey reported that shakespe are had been a country school master some twentieth century scholars ha ve suggested that shakespeare may have been employed as a schoolmast

er by alexander hoghton of lancashire a catholic landowner who named a c
ertain william shakesh aftein his will no evidence substantiates such stories
other than hearsay collected after his death hereupon le grand arose with a
grave and stately air and brought me the beetle from a glass case in which
it was enclosed it was a beautiful scarabaeus and at that time unknown to
naturalists of course a great prize in a scientific point of view there were t
wo round black spots near one extremity of the back and a long one near t
he other the scales were exceedingly hard and glossy with all the appeara
nce of burnished gold the weight of the insect was very remarkable and ta
king all things into consideration i could hardly blame jupiter for his opinion
respecting

And finally its solved, lets now decode the actual password for krypton4:





> 💡 Some might consider using tools as plain cheating, and I'm honestly also leaning on their side of opinion, but I did try for my own self and did all I could, only as a last resort I used a tool not designed by me, I'm a newbie and trying my hands on something I'm not comfortable with, hopefully Ill learn a thing or two.

Password:

```
BRUTE
```

▼ Level 4

Level Info:

> Good job!
>
> You more than likely used frequency analysis and some common sense to solve that one.
>
> So far we have worked with simple substitution ciphers. They have also been 'monoalphabetic', meaning using a fixed key, and giving a one to one mapping of plaintext (P) to ciphertext (C).
> Another type of substitution cipher is referred to as 'polyalphabetic', where one character of P may map to many, or all, possible ciphertext characters.
>
> An example of a polyalphabetic cipher is called a Vigen`?`re Cipher. It works like this:
>
> If we use the key(K) 'GOLD', and P = PROCEED MEETING AS AGREED, then "add"
> P to K, we get C. When adding, if we exceed 25, then we roll to 0 (modulo 26).
>
> P P R O C E E D M E E T I N G A S A G R E E D
> K G O L D G O L D G O L D G O L D G O L D G O
>
> becomes:

> P 15 17 14 2 4 4 3 12 4 4 19 8 13 6 0 18 0 6 17 4 4 3
> K 6 14 11 3 6 14 11 3 6 14 11 3 6 14 11 3 6 14 11 3 6 14
> C 21 5 25 5 10 18 14 15 10 18 4 11 19 20 11 21 6 20 2 8 10 17
>
> So, we get a ciphertext of:
>
> VFZFK SOPKS ELTUL VGUCH KR
>
> This level is a Vigen`re Cipher. You have intercepted two longer, english language messages. You also have a key piece of information. You know the
> key length!
>
> For this exercise, the key length is 6. The password to level five is in the usual
> place, encrypted with the 6 letter key.
>
> Have fun!

HINT File:

```
krypton4@krypton:/krypton/krypton4$ cat HINT
Frequency analysis will still work, but you need to analyse it
by "keylength". Analysis of cipher text at position 1, 7, 13, etc
should reveal the 1st letter of the key, in this case. Treat this as
6 different mono-alphabetic ciphers...
```

Files [found1, found2]:

```
YYICS JIZIB AGYYX RIEWV IXAFN JOOVQ QVHDL CRKLB SSLYX RIQYI IOX
QT WXRIC RVVKP BHZXI YLYZP DLCDI IKGFJ UXRIP TFQGL CWVXR IEZR
V NMYSF JDLCL RXOWJ NMINX FNJSP JGHVV ERJTT OOHRM VMBWN J
TXKG JJJXY TSYKL OQZFT OSRFN JKBIY YSSHE LIKLO RFJGS VMRJC C
YTCS VHDLC LRXOJ MWFYB JPNVR NWUMZ GRVMF UPOEB XKSDL CBZ
GU IBBZX MLMKK LOACX KECOC IUSBS RMPXR IPJZW XSPTR HKRQB VV
OHR MVKEE PIZEX SDYYI QERJJ RYSLJ VZOVU NJLOW RTXSD LYYNE IL
MBK LORYW VAOXM KZRNL CWZRA YGWVH DLCLZ VVXFF KASPJ GVIK
W WWVTV MCIKL OQYSW SBAFJ EWRII SFACC MZRVO MLYYI MSSSK VI
SDY YIGML PZICW FJNMV PDNEH ISSFE HWEIJ PSEEJ QYIBW JFMIC TC
```

WYE ZWLTK WKMBY YICGY WVGBS UKFVG IKJRR DSBJJ XBSWM VVYLR MRXSW BNWJO VCSKW KMBYY IQYYW UMKRM KKLOK YYVWX SMSVL KWCAV VNIQY ISIIB MVVLI DTIIC SGSRX EVYQC CDLMZ XLDWF JNSEP B RROO WJFMI CSDDF YKWQM VLKWM KKLOV CXKFE XRFBI MEPJW SBW FJ ZWGMA PVHKR BKZIB GCFEH WEWSF XKPJT NCYYR TUICX PTPLO VI JVT DSRMV AOWRB YIBIR MVWER QJKWK RBDFY MELSF XPEGQ KSPML IYIBX FJPXR ELPVH RMKFE HLEBJ YMWKM TUFII YSUXE VLJUX YAYWU XRIUJ JXGEJ PZRQS TJIJS IJIJS PWMKK KBEQX USDXC IYIBI YSUXR IPJN M DLBFZ WSIQF EHLYR YVVMY NXUSB SRMPW DMJQN SBIRM VTBIR YP WSP IIIIC WQMVL KHNZK SXMLY YIZEJ FTILY RSFAD SFJIW EVNWZ WO WFJ WSERB NKAKW LTCSX KCWXV OILGL XZYPJ NLSXC YYIBM ZGFRK V MZEH DSRTJ ROGIM RHKPQ TCSCX GYJKB ICSTS VSPFE HGEQF JARMR JRWNS PTKLI WBWVW CXFJV QOVYQ UGSXW BRWCS MSCIP XDFIF OL GSU ECXFJ PENZY STINX FJXVY YLISI MEKJI SEKFJ IEXHF NCPSI PKFVD LCWVA OVCSF JKVKX ESBLM ZJICM LYYMC GMZEX BCMKK LOACX KEX HR MVKBS SSUAK WSSKM VPCIZ RDLCF WXOVL TFRDL CXLRC LMSVL Y XGSK LOMPK RGOWD TIXRI PJNIB ILTKV OIQYF SPJCW KLOQQ MRHOW MYYED FCKFV ORGLY XNSPT KLIEL IKSDS YSUXR IJNFR GIPJK MBIBF EH VEW IFAXY NTEXR IEWRW CELIW IVPYX CIOTU NKLDL CBFSN QYSRR NX FJJ GKVCH ISGOC JGMXK UFKGR YYIIA CWVSL PGLVH DSAFD TYYRY Y EDRG LYXER BJIEV EPLVX BICNE XRIDT IICXD TIXRI PJNIB ILTYS EWCXE I KVRM VXBIC RRHOE ETFHD LGHBG YZCWZ RQXMU ISDIA YKLOQ DWFQ D LCIVA KRBYY IDMLB FSNQY STLYT NJUEQ VCFKT SPCTW AYSBB ZXR LG XRBOE LIUSB SRMPF EMJYR WZPCS UMNJG WVXRE RBRVW IBMVV KRBRR HOLCW WIOPJ JJWVS LJCCC LCFEH DSRTR XOXFJ CECXM KKLO M PGIIK HYSUR YAQMV HSHLT KOXSU BYEDX FJPAY YJIUS PSPGI IKODF JXSJW TLASW FXRMN XFJCM YRGBZ PVKMN EXYXF JWSBI QYRRN OG QCE NICWW SBCMZ PSEGY SISKW RNKFI XFJWM BIQNE GOCMZ IXKWR JJEBI QTGIM YJNRV DLYYP SETPJ WIBGM TBINJ MTUEX HRMVR ISSBZ P VLYA VEFIP DXSYH ZWVEU JYXKH YRRUC IKWCI FRDFC LXINX FJKMX A MTUQ KRGXY SEPBH VVDEG SCCGI CUZJI SSPZP VIBFG SYVBJ VVKRB YYIXQ WORAC AMZCH BYQYR KKMLG LXDLC QZSXA CSKEG EWNEX YXF JW SBIQY RRNJM ZEHRM QTNRC YNUVV KRBSF SXICA VVURC BNLKX G YNEC JMWYI NMBSK QORRN FRSXY SUXRI QHRVO GPTNJ YYLIR XBICK LPVSD SLXCE LIWMV PCIUS BSRMP WLEQP VXGMR MKLOQ QTKLK XQM VA YYJIE SDFCM LRQVW KFVKP MSXXS QCXYI DLMZX LDXFN JAKWT JI

CUM LIRRN XFTLK RXDZC SPXFJ JGKVC HISGF SYJLO PYZXL OHFJR VD
MJD RXDLC FNOGE PINEI MLBYM MLRMV TYSPH IIKXS WVTSG IJUYZ XF
JEY DWFNJ TKHBJ ULKRB XNIBI QTTPE QQDRR NXFJE YDWUJ IICSQ RR
PVX FFKLO HPTGT OHYQD SCXYX DEXCY XYIZY RNEXR IZFJO OXZZK XR
IQH RVOGP TNHSH LTKQS RBMFA VSLLZ XDSMP YMWXM KZPVX FJSEC
OCYWS BMRJE ELPCI YMWXM PVIZE UFPJB SKYYI PMPJR WRIDJ RVOHY
XGEBO KNXLD KCYZR DSFNJ WDVYB RRNFS WELSQ SUJSR IIJGX KKMT
U HSWRF EGOEU FPJBS KYYIP PYRVW KRBTE PIGYR VROEP YFGYZ CWU
SB SRMPA SXFII CVIYA VWGLC SJLOP YDUSG RRTJP OINYY ICIIJ GXRIP
AVVIW LZXEX HUFIQ KRBXY ICPCU KWYYL ICCER RNCQY VLNEK GLCSZ
XGEQI RCVME MKXRI ENIPL ERMVH RIPKR GOMLF CMDXJ JIMZT JNEKL
VMTBE XHQTF RKJRJ IXRIW FCPCX YWKIN XMBRV NXFJV QOVYQ UGSX
W YYMCA YXKSL IYSVZ ORRKL PNEWK FVDLC YIEFI JJIWD LCDYE NLY
WU PIFCJ EAKPI NEKKR FTLVG LCSKL OCQFN FOJMW VXRIK FXVOE RIZ
XM LRMRX MVMXJ INXFJ ISKHY SUHSZ GIVHD LCKFV OWRFJ JKVYX KL
OCA TLPNW CJFRO MRMVV CMBJZ XGEQF MIBCU NUINM RHYEX HUMV
R DLCDT VOTRZ GXYXF JVHQI YSUPY SIJUM XXMNK XRIWH FYVHQ JV
MDA YXRPC STJIC NICUR RNXFJ IIGIP JDEXC ZNXNK KEJUV YGIXR XDLC
G FXDSK YYICM BJJAO VCXFW DICUK LKXLT EIYJR MVQMS SQUGV MK
GUS GRYSU JYVYR FQORR NKWOI KJUXR ERYYI SVHTL VXIWR LWDIL IN
LKX QMRPV ACIFE COCIU SBSRM PHOWN FZVSR EQPMR ETJEX DLCKR
MXXCX KMNIY XRMNX FJKMX AMTUQ KRYSU XRIJN FRCLM TBLSW QM
RKQ CKFEI KRBQF SUIBY YSEKF YWYVF SYKLO WAFII MVMBJ ESHUJ TE
XRM YWPIX FFKMC GCWKE SRLJZ XRIPH RRGIA QZQLH MBEMX XMYYM
CKPJR XNMRH YXRIP JWSBI GKNIM ELSFX TYKUF ZOVGY NIWYQ YJXYT
UMVVO ACFII SXFNE OSGMZ CHTYK UFZOV GYJES HRMVG YAYWU PIPG
T EEPXC WDIKW SWZRQ XFJUM CXYST IMEPJ WYVPW NELSW KNEHD L
CSNI KVCFC PBMEM KEXWU JIINX FJJGK VCHIS GJMWP SEGYS TEBVW
ZJEVP MAVVY RWTLV LEAPF ROERF KMWIU JCPSP JYICS XQFZH DLCQZ
SXAFT NMVPE TWMBW RNNMV PBJTP KVCIK LOWAF IIMVM BWSBM DD
FYP SSSUX RERDF YMSSQ URYXH ZDTYZ CWKLO KSQWH YVMYY CGSS
Q UFOOG QCINS PYYID MLBFS NQYSS ENPWI VRDIB TEXRI PTTOC FCQF
A LYRNW MKQMS PSEVZ FTOSX UNCPX SRRRX DIPXF QEGFK FVDLC KR
PVA MZCHX SRMLV DQCFK EVP

`Krypton5` encrypted file:

HCIKV RJOX

Okay, now that the *'givens'* are wrapped up lets begin with the decryption. First let's just know how this encryption i.e. polyalphabetic cipher is different from monoalphabetic cipher:

AI generated Response:

1. Monoalphabetic Cipher:
Uses a single fixed substitution alphabet throughout the entire message
Each letter is always replaced by the same substitute letter
Example: If 'A' maps to 'Q', then every 'A' in the message becomes 'Q'
Common example: Caesar cipher (shift by 3: A→D, B→E, C→F, etc.)

2. Polyalphabetic Cipher:
Uses multiple substitution alphabets that change during encryption
The same letter can be encrypted differently depending on its position
Example: If 'A' appears twice, it might become 'Q' the first time and 'Z' the second time
Common example: Vigenère cipher (uses a keyword to determine which al phabet to use)

Why this matters for security:
Monoalphabetic ciphers are vulnerable to frequency analysis. In English, 'E' is the most common letter, so attackers can look for the most frequent s ymbol in the ciphertext and guess it represents 'E'. This pattern holds throu ghout the message.
Polyalphabetic ciphers are much harder to crack because they obscure the se frequency patterns. Since the same letter encrypts differently each tim e, simple frequency analysis doesn't work as well.
Think of it this way: monoalphabetic is like wearing the same disguise all d ay, while polyalphabetic is like changing disguises multiple times—much h arder to recognize!

Okay, that makes sense, but how does it actually work then?
→ let’s illustrate this with an example:

Message: J Y N X
Keyword: K E Y S (repeat "KEYS" to match length of larger strings KEYSKE YS....)

Now encrypt each letter [Process]:

J + K:
J = 9 (position in alphabet, starting from 0)
K = 10 (shift amount)
(9 + 10) mod 26 = 19 = T (modulus 26 to reiterate the alphabetical order)

Y + E:
Y = 24
E = 4
(24 + 4) mod 26 = 28 mod 26 = 2 = C

N + Y:
N = 13
Y = 24
(13 + 24) mod 26 = 37 mod 26 = 11 = L

X + S:
X = 23
S = 19
(23 + 19) mod 26 = 42 mod 26 = 16 = P

Result: JYNX → TCLP

Okay now we do understand how the encryption actually works, so we are halfway to solving something interesting here, remember half of the solution is knowing what you intending to solve in the first place, so I’m not exaggerating here, when I say we are halfway to the actual solution finding.
→ Now, the thing is we **can not** do frequency analysis- it just isn’t going to

work, since that's the whole idea behind using polyalphabetic encryption over monoalphabetic encryption.
What we can actually think about it in a way is, since we are known to the fact that the key length is 6 every 6th key should be the same, right? YES. then, what we can try to do here is, if we individually take every 6th letter in the big group of `found` files we have, it again tends to become monoalphabetic as well, I mean at least in a way if you think about it not literally.

So we will need a python in fact more than one python script to help us here; let's start with first extracting letters and then move ahead:

Python Script to extract letters:

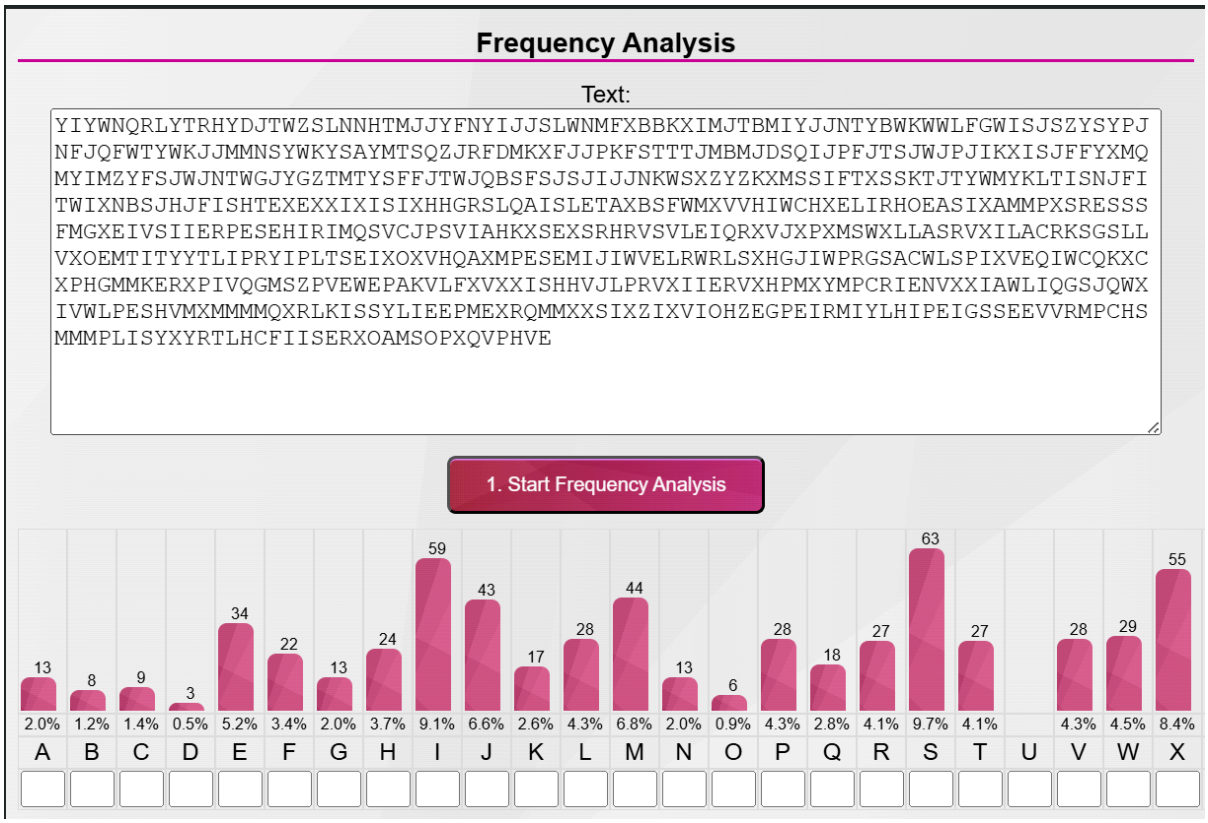
```
found1and2 = "YYICSJIZIBAGYYXRIEWVIXAFNJOOVQQVHDLCRKLBSSLY
XRIQYIIOXQTWXRICRVVKPBHZXIYLYZPDLCDIIKGFJUXRIPTFQGLCWVXRI
EZRVNMYSFJDLCLRXOWJNMINXFNJSPJGHVVERJTTOOHRMVMBWNJTX
KGJJJXYTSYKLOQZFTOSRFNJKBIYYSSHELIKLORFJGSVMRJCCYTCSVH
DLCLRXOJMWFYBJPNVRNWUMZGRVMFUPOEBXKSDLCBZGUIBBZXMLM
KKLOACXKECOCIUSBSRMPXRIPJZWXSPTRHKRQBVVOHRMVKEEPIZEXS
DYYIQERJJRYSLJVZOVUNJLOWRTXSDLYYNEILMBKLORYWVAOXMKZRN
LCWZRAYGWVHDLCLZVVXFFKASPJGVIKWWWVTVMCIKLOQYSWSBAFJE
WRIISFACCMZRVOMLYYIMSSSKVISDYYIGMLPZICWFJNMVPDNEHISSFEH
WEIJPSEEJQYIBWJFMICTCWYEZWLTKWKMBYYICGYWVGBSUKFVGIKJR
RDSBJJXBSWMVVYLRMRXSWBNWJOVCSKWKMBYYIQYYWUMKRMKKL
OKYYVWXSMSVLKWCAVVNIQYISIIBMVVLIDTIICSGSRXEVYQCCDLMZXL
DWFJNSEPBRROOWJFMICSDDFYKWQMVLKWMKKLOVCXKFEXRFBIMEPJ
WSBWFJZWGMAPVHKRBKZIBGCFEHWEWSFXKPJTNCYYRTUICXPTPLOV
IJVTDSRMVAOWRBYIBIRMVWERQJKWKRBDFYMELSFXPEGQKSPMLIYIBX
FJPXRELPVHRMKFEHLEBJYMWKMTUFIIYSUXEVLJUXYAYWUXRIUJJXGE
JPZRQSTJIJSIJIJSPWMKKKBEQXUSDXCIYIBIYSUXRIPJNMDLBFZWSIQFE
HLYRYVVMYNXUSBSRMPWDMJQNSBIRMVTBIRYPWSPIIIICWQMVLKHNZ
KSXMLYYIZEJFTILYRSFADSFJIWEVNWZWOWFJWSERBNKAKWLTCSXKC
WXVOILGLXZYPJNLSXCYYIBMZGFRKVMZEHDSRTJROGIMRHKPQTCSCX
GYJKBICSTSVSPFEHGEQFJARMRJRWNSPTKLIWBWVWCXFJVQOVYQUG
SXWBRWCSMSCIPXDFIFOLGSUECXFJPENZYSTINXFJXVYYLISIMEKJISEK
FJIEXHFNCPSIPKFVDLCWVAOVCSFJKVKXESBLMZJICMLYYMCGMZEXBC
MKKLOACXKEXHRMVKBSSSUAKWSSKMVPCIZRDLCFWXOVLTFRDLCXLR
```

CLMSVLYXGSKLOMPKRGOWDTIXRIPJNIBILTKVOIQYFSPJCWKLOQQMRH
OWMYYEDFCKFVORGLYXNSPTKLIELIKSDSYSUXRIJNFRGIPJKMBIBFEHV
EWIFAXYNTEXRIEWRWCELIWIVPYXCIOTUNKLDLCBFSNQYSRRNXFJJGK
VCHISGOCJGMXKUFKGRYYIIACWVSLPGLVHDSAFDTYYRYYEDRGLYXER
BJIEVEPLVXBICNEXRIDTIICXDTIXRIPJNIBILTYSEWCXEIKVRMVXBICRRHO
EETFHDLGHBGYZCWZRQXMUISDIAYKLOQDWFQDLCIVAKRBYYIDMLBFS
NQYSTLYTNJUEQVCFKTSPCTWAYSBBZXRLGXRBOELIUSBSRMPFEMJYR
WZPCSUMNJGWVXRERBRVWIBMVVKRBRRHOLCWWIOPJJJWVSLJCCCL
CFEHDSRTRXOXFJCECXMKKLOMPGIIKHYSURYAQMVHSHLTKOXSUBYED
XFJPAYYJIUSPSPGIIKODFJXSJWTLASWFXRMNXFJCMYRGBZPVKMNEXY
XFJWSBIQYRRNOGQCENICWWSBCMZPSEGYSISKWRNKFIXFJWMBIQNE
GOCMZIXKWRJJEBIQTGIMYJNRVDLYYPSETPJWIBGMTBINJMTUEXHRM
VRISSBZPVLYAVEFIPDXSYHZWVEUJYXKHYRRUCIKWCIFRDFCLXINXFJK
MXAMTUQKRGXYSEPBHVVDEGSCCGICUZJISSPZPVIBFGSYVBJVVKRBY
YIXQWORACAMZCHBYQYRKKMLGLXDLCQZSXACSKEGEWNEXYXFJWSBI
QYRRNJMZEHRMQTNRCYNUVVKRBSFSXICAVVURCBNLKXGYNECJMWYI
NMBSKQORRNFRSXYSUXRIQHRVOGPTNJYYLIRXBICKLPVSDSLXCELIWM
VPCIUSBSRMPWLEQPVXGMRMKLOQQTKLKXQMVAYYJIESDFCMLRQVW
KFVKPMSXXSQCXYIDLMZXLDXFNJAKWTJICUMLIRRNXFTLKRXDZCSPXF
JJGKVCHISGFSYJLOPYZXLOHFJRVDMJDRXDLCFNOGEPINEIMLBYMMLR
MVTYSPHIIKXSWVTSGIJUYZXFJEYDWFNJTKHBJULKRBXNIBIQTTPEQQD
RRNXFJEYDWUJIICSQRRPVXFFKLOHPTGTOHYQDSCXYXDEXCYXYIZYRN
EXRIZFJOOXZZKXRIQHRVOGPTNHSHLTKQSRBMFAVSLLZXDSMPYMWXM
KZPVXFJSECOCYWSBMRJEELPCIYMWXMPVIZEUFPJBSKYYIPMPJRWRID
JRVOHYXGEBOKNXLDKCYZRDSFNJWDVYBRRNFSWELSQSUJSRIIJGXKK
MTUHSWRFEGOEUFPJBSKYYIPPYRVWKRBTEPIGYRVROEPYFGYZCWUSB
SRMPASXFIICVIYAVWGLCSJLOPYDUSGRRTJPOINYYICIIJGXRIPAVVIWLZ
XEXHUFIQKRBXYICPCUKWYYLICCERRNCQYVLNEKGLCSZXGEQIRCVME
MKXRIENIPLERMVHRIPKRGOMLFCMDXJJIMZTJNEKLVMTBEXHQTFRKJR
JIXRIWFCPCXYWKINXMBRVNXFJVQOVYQUGSXWYYMCAYXKSLIYSVZOR
RKLPNEWKFVDLCYIEFIJJIWDLCDYENLYWUPIFCJEAKPINEKKRFTLVGLCS
KLOCQFNFOJMWVXRIKFXVOERIZXMLRMRXMVMXJINXFJISKHYSUHSZGI
VHDLCKFVOWRFJJKVYXKLOCATLPNWCJFROMRMVVCMBJZXGEQFMIB
CUNUINMRHYEXHUMVRDLCDTVOTRZGXYXFJVHQIYSUPYSIJUMXXMNK
XRIWHFYVHQJVMDAYXRPCSTJICNICURRNXFJIIGIPJDEXCZNXNKKEJUV
YGIXRXDLCGFXDSKYYICMBJJAOVCXFWDICUKLKXLTEIYJRMVQMSSQUG

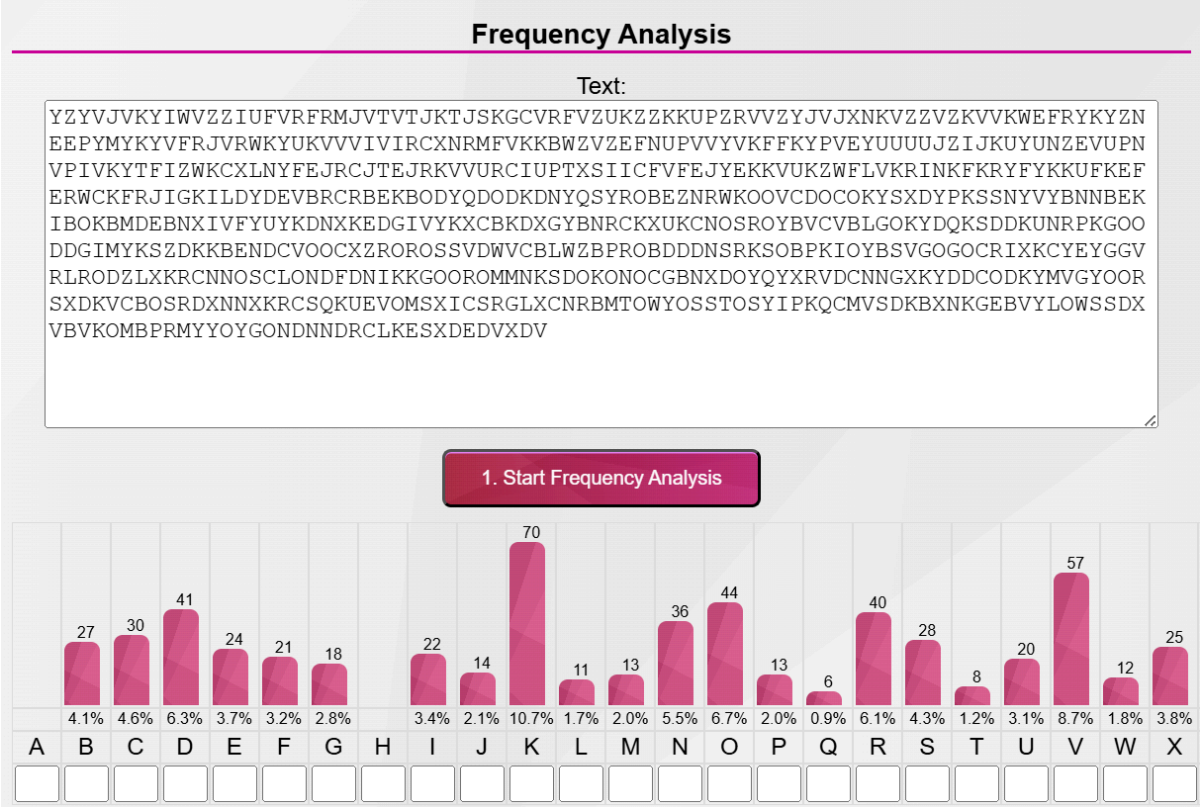
```
VMKGUSGRYSUJYVYRFQORRNKWOIKJUXRERYYISVHTLVXIWRLWDILINL
KXQMRPVACIFECOCIUSBSRMPHOWNFZVSREQPMRETJEXDLCKRMXXCX
KMNIYXRMNXFJKMXAMTUQKRYSUXRIJNFRCLMTBLSWQMRKQCKFEIKR
BQFSUIBYYSEKFYWYVFSYKLOWAFIIMVMBJESHUJTEXRMYWPIXFFKMC
GCWKESRLJZXRIPHRRGIAQZQLHMBEMXXMYYMCKPJRXNMRHYXRIPJW
SBIGKNIMELSFXTYKUFZOVGYNIWYQYJXYTUMVVOACFIISXFNEOSGMZC
HTYKUFZOVGYJESHRMVGYAYWUPIPGTEEPXCWDIKWSWZRQXFJUMCXY
STIMEPJWYVPWNELSWKNEHDLCSNIKVCFCPBMEMKEXWUJIINXFJJGKV
CHISGJMWPSEGYSTEBVWZJEVPMAVVYRWTLVLEAPFROERFKMWIUJCP
SPJYICSXQFZHDLCQZSXAFTNMVPETWMBWRNNMVPBJTPKVCIKLOWAFI
IMVMBWSBMDDFYPSSSUXRERDFYMSSQURYXHZDTYZCWKLOKSQWHY
VMYYCGSSQUFOOGQCINSPYYIDMLBFSNQYSSENPWIVRDIBTEXRIPTTOC
FCQFALYRNWMKQMSPSEVZFTOSXUNCPXSRRRXDIPXFQEGFKFVDLCKRP
VAMZCHXSRMLVDQCFKEVP"

result = found1and2[0::6] #1st position
print(f"Indexing at '1':{result}")

result = found1and2[1::6] #2nd position
print(f"Indexing at '2':{result}")

result = found1and2[2::6] #3rd position
print(f"Indexing at '3':{result}")

result = found1and2[3::6] #4th position
print(f"Indexing at '4':{result}")

result = found1and2[4::6] #5th position
print(f"Indexing at '5':{result}")

result = found1and2[5::6] #6th position
print(f"Indexing at '6':{result}")
```

Now that we have extracted outputs for each position, we can do frequency analysis:

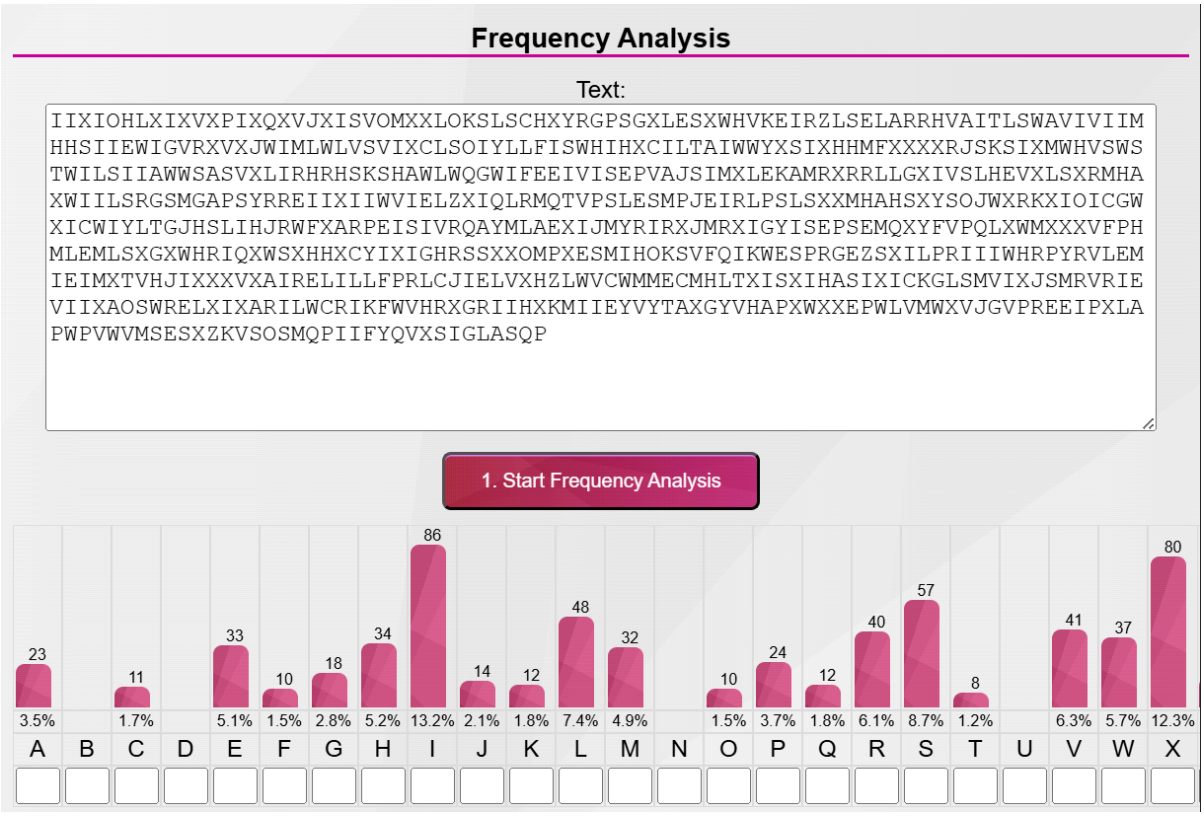
Position 1:



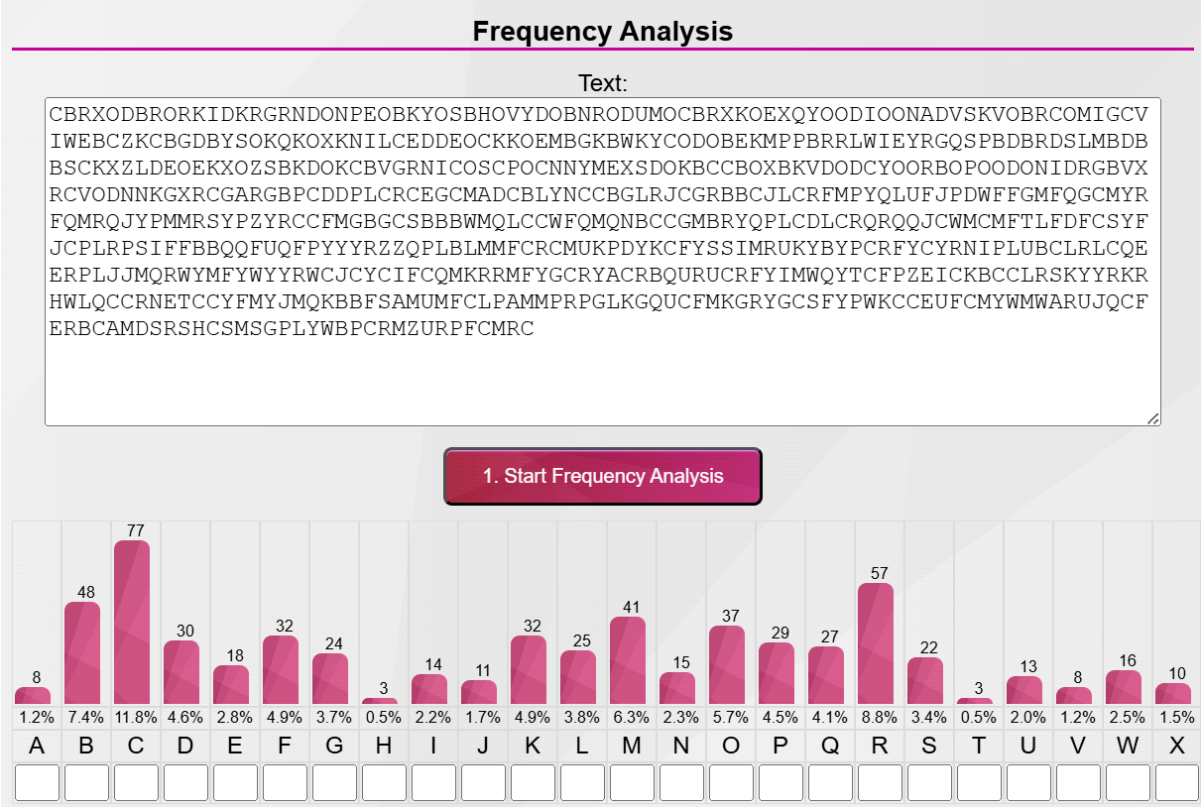
Position 2:


Frequency Analysis
Text:
YZYVJVKYIWVZZIUFVRFRMJVTVTJKTJSKGCVRFVZUKZZKKUPZRVVZYJVJXNKVZZVZKVVKWEFRYKYZN
EEPYMYKYVFRJVRWKYUKVVVIVIRCXNRMFVKKBWZVZEFNUPVVYVKFFKYPVEYUUUUJZIJKUYUNZEVUPN
VPIVKYTFIZWKCXLNYFEJRCJTEJRKVVURCIUPTXSIICFVFEJYEKKVUKZWFLVKRINKFKRYFYKKUFKEF
ERWCKFRJIGKILDYDEVBRCRBEKBODYQDODKDNYQSYROBEZNRWKOOVCDOCOKYSXDYPKSSNYVYBNNBEK
IBOKBMDEBNXIVFYUYKDNXKEDGIVYKXCBKDXGYBNRCKXUKCNOSROYBVCVBLGOKYDQKSDDKUNRPKGOO
DDGIMYKSZDKKBENDCVOOCXZROROSSVDWVCBLWZBPROBDDDNSRKSOBPKIOYBSVGOGOCRIXKCYEYGGV
RLRODZLXKRCNNOSCLONDFDNIKKGOOROMMNKSDOKONOCGBNXDOYQYXRVDCNNGXKYDDCODKYMVGYOOR
SXDKVCBOSRDXNNXKRCSQKUEVOMSXICSRGLXCNRBMTOWYOSSTOSYIPKQCMVSDKBXNKGEBVYLOWSSDX
VBVKOMBPRMYYOYGONDNNDRCLKESXDEDVXDV
1. Start Frequency Analysis
27 30 41 24 21 18 22 14 70 11 13 36 44 13 6 40 28 8 20 57 12 25
4.1% 4.6% 6.3% 3.7% 3.2% 2.8% 3.4% 2.1% 10.7% 1.7% 2.0% 5.5% 6.7% 2.0% 0.9% 6.1% 4.3% 1.2% 3.1% 8.7% 1.8% 3.8%
A B C D E F G H I J K L M N O P Q R S T U V W X

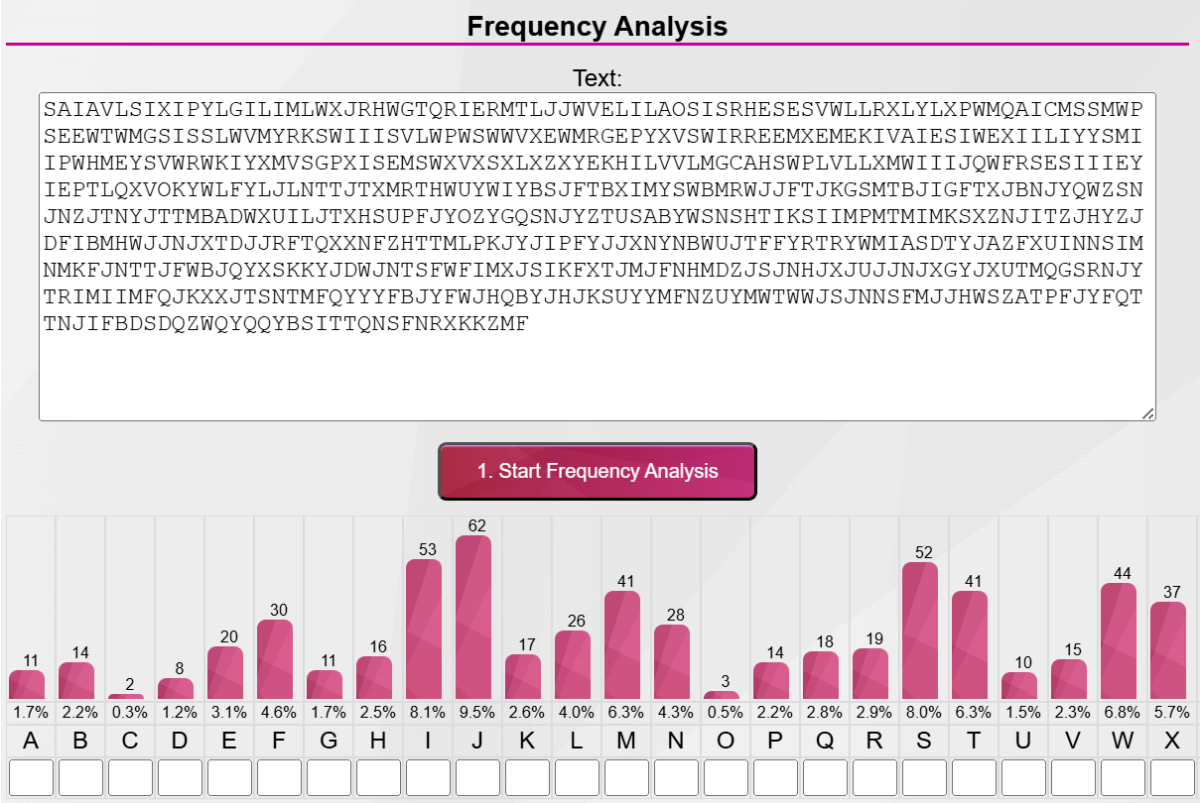
Position 3:


Frequency Analysis
Text:
IIXIOHLXIXVXPIXQXVJXISVOMXXLOKSLSCHXYRGPSGXLESXWHVKEIRZLSELARRHVAITLSWAVIVIIM
HHSIIEWIGVRXVXJWIMLWLVSVIXCLSOIYLLFISWHIHXCILTAIWWYXSIXHHMFXXXXRJSKSIXMWHVSWS
TWILSIIAWWSASVXLIRHRHSKSHAWLWQGWIFEEIVISEPVAJSIMXLEKAMRXRRLLGXIVSLHEVXLSXRMHA
XWIILSRGSMGAPSYRREIIXIIWVIELZXIQLRMQTVPSLESMPJEIRLPSLSXXMHAHSXYSOJWXRKXIOICGW
XICWIYLTGJHSLIHJRWFXARPEISIVRQAYMLAEXIJMYRIRXJMRXIGYISEPSEMQXYFVPQLXWMXXXVFPH
MLEMLSXGXWHRIQXWSXHHXCYIXIGHRSSXXOMPXESMIHOKSVFQIKWESPRGEZSXILPRIIIWHRPYRVLEM
IEIMXTVHJIXXXVXAIRELILLFPRLCJIELVXHZLWVCWMMECMHLTXISXIHASIXICKGLSMVIXJSMRVRIE
VIIXAOSWRELXIXARILWCRIKFWVHRXGRIIHXKMIIEYVYTAXGYVHAPXWXXEPWLVMWXVJGVPREEIPXLA
PWPVWVMSESXZKVSOSMQPIIFYQVXSIGLASQP
1. Start Frequency Analysis
23
11
33
10
18
34
86
14
12
48
32
10
24
12
40
57
8
41
37
80
3.5%
1.7%
5.1%
1.5%
2.8%
5.2%
13.2%
2.1%
1.8%
7.4%
4.9%
1.5%
3.7%
1.8%
6.1%
8.7%
1.2%
6.3%
5.7%
12.3%
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X

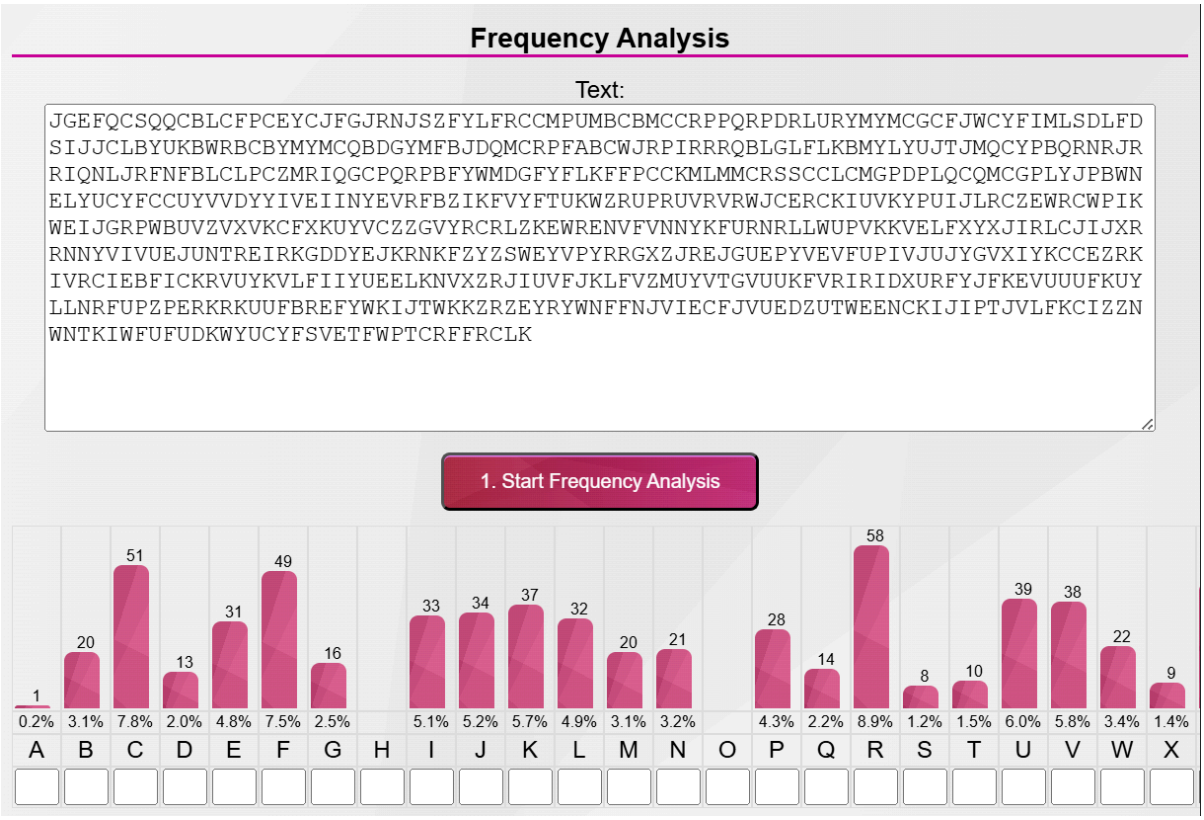
Position 4:


Frequency Analysis
Text:
CBRXODBRORKIDKRGRNDONPEOBKYOSBHOVYDOBNRODUMOCBRXKOEXQYOODIOONADVSKVOBRCOMIGCV
IWEBCZKCBGDBYSOKQKOXKNILCEDDEOCKKOEMBGKBWKYCODOBEKMPPBRRLWIEYRGQSPBDBRDSLMBDB
BSCKXZLDEOEKXOZSBKDOKCBVGRNICOSCPOCNNYMEXSDOKBCCBOXBKVDODCYOORBOPOODONIDRGBVX
RCVODNNKGXRCGARGBPCDDPLCRCEGCMADCBLYNCCBGLRJCGRBBCJLCRFMPYQLUFJPDWFFGMFQGCMYR
FQMRQJYPMMRSYPZYRCCFMGBGCSBBBWMQLCCWFQMQNBCCGMBRYQPLCDLCRQRQQJCWMCMFTLFDFCSYF
JCPLRPSIFFBBQQFUQFPYYYRZZQPLBLMMFCRCMUKPDYKCFYSSIMRUKYBYPCRFYCYRNIPLUBCLRLCQE
ERPLJJMQRWYMFYWYYRWCJCYCIFCQMKRRMFYGCRYACRBQURUCRFYIMWQYTCFPZEICKBCCLRSKYYRKR
HWLQCCRNETCCYFMYJMQKBBFSAMUMFCLPAMMPRPGLKGQUCFMKGRYGCSFYPWKCCEUFCMYWMWARUJQCF
ERBCAMDSRSHCSMSGPLYWBPCRMZURPFCMRC
1. Start Frequency Analysis
8 48 77 30 18 32 24 3 14 11 32 25 41 15 37 29 27 57 22 3 13 8 16 10
1.2% 7.4% 11.8% 4.6% 2.8% 4.9% 3.7% 0.5% 2.2% 1.7% 4.9% 3.8% 6.3% 2.3% 5.7% 4.5% 4.1% 8.8% 3.4% 0.5% 2.0% 1.2% 2.5% 1.5%
A B C D E F G H I J K L M N O P Q R S T U V W X

Position 5:


Frequency Analysis
Text:
SAIAVLSIXIPYLGILIMLWXJRHWGTQRIERMTLJJWVELILAOSISRHESESVWLLRXLYLXPWMQAICMSSMWP
SEEWTWMGSISSLWVMYRKSWIIISVLWPWSWWVXEWMRGEPYXVSWIRREEMXEMEKIVAIESIWEXIILIYYSMI
IPWHMEYSVWRWKIYXMVSGPXISEMSWXVXSXLXZXYEKHILVVLMGCAHSWPLVLLXMWIIIJQWFRSESIIIEY
IEPTLQXVOKYWLFYLJLNTTJTXMRTHWUYWIYBSJFTBXIMYSWBMRWJJFTJKGSMTBJIGFTXJBNJYQWZSN
JNZJTNYJTTMBADWXUILJTXHSUPFJYOZYGQSNJYZTUSABYWSNSHTIKSIIMPMTMIMKSXZNJITZJHYZJ
DFIBMHWJJNJXTDJJRFTQXXNFZHTTMLPKJYJIPFYJJXNYNBWUJTFFYRTRYWMIASDTYJAZFXUINNSIM
NMKFJNTTJFWBJQYXSKKYJDWJNTSFWFIMXJSIKFXTJMJFNHMDZJSJNHJXJUJJNJXGYJXUTMQGSRNJY
TRIMIIMFQJKXXJTSNTMFQYYYFBJYFWJHQBYJHJKSUYYMFNZUYMWTWWJSJNNSFMJJHWSZATPFJYFQT
TNJIFBDSDQZWQYQQYBSITTQNSFNRXKKZMF
1. Start Frequency Analysis
11 14 2 8 20 30 11 16 53 62 17 26 41 28 3 14 18 19 52 41 10 15 44 37
1.7% 2.2% 0.3% 1.2% 3.1% 4.6% 1.7% 2.5% 8.1% 9.5% 2.6% 4.0% 6.3% 4.3% 0.5% 2.2% 2.8% 2.9% 8.0% 6.3% 1.5% 2.3% 6.8% 5.7%
A B C D E F G H I J K L M N O P Q R S T U V W X

Position 6:



We know letter ‘E’ is the most common in American English, lets deduce each position for ‘E’ here then, here’s the logic I followed with a lot and a ton lot of head ,messing permutations and combinations to find the ‘KEY’, here’s the logic behind it:

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
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Position 1:
Encrypted Letter: S/I/X - 18/8/23 [Highest Probabilities]

KEY Letter: O/E/T - 14/4/19 [Either 'O,E,T'] // Let's keep them aside we wil l have to try permutations and combinations for them
---
Plain Text: E - 04

Position 2:
Encrypted Letter: K/V - 10/21 [Highest Probabilities]

KEY Letter: G/R - 06/17 [Either 'G/R'] // Let's keep them aside we will have to try permutations and combinations for them
---
Plain Text: E - 04

Position 3:
Encrypted Letter: I/X - 08/23 [Highest Probabilities]

KEY Letter: E/T - 04/19 [Either 'E/T'] // Let's keep them aside we will h ave to try permutations and combinations for them
---
Plain Text: E - 04

Position 4:
Encrypted Letter: C - 02

KEY Letter: Y - 24 // Clear winner it seems

Plain Text: E - 04

Position 5:
Encrypted Letter: J/I/S - 09/08/18 [Highest Probabilities]

KEY Letter: F/E/O - 05/04/14 [Either 'F,E,O'] // Let's keep them aside we will have to try permutations and combinations for them
---
Plain Text: E - 04

Position 6:
Encrypted Letter: R/F/C - 17/05/02 [Highest Probabilities]

KEY Letter: N/B/Y - 13/01/24 [Either 'N,B,Y'] // Let's keep them aside we will have to try permutations and combinations for them
---
Plain Text: E - 04

Finally, with a lot of time effort and honestly annoyance at a point, here's the 'KEY':

J V I O I C
- - - - - -
F R E K E Y [The much hard earned 'KEY']
- - - - - -
E E E E E E

Now all that's left is to reverse the encryption, with a python script/any tool you can get for free:

Python Script:

```
def vigenere_decrypt(ciphertext, key):
    plaintext = ""
    key = key.upper()
    key_index = 0

    for char in ciphertext:
        if char.isalpha():
            is_upper = char.isupper()
            char = char.upper()
            # Get the shift value from the key
            shift = ord(key[key_index % len(key)]) - ord('A')
            # Decrypt by shifting backwards
            decrypted_char = chr((ord(char) - ord('A') - shift) % 26 + ord('A'))
            plaintext += decrypted_char
            key_index += 1
        else:
            plaintext += char

    return plaintext

print(vigenere_decrypt('HCIKV RJOX','FREKEY'))
```

Password:

```
CLEARTEXT
```

▼ Level 5

Level Info:

> Frequency analysis can break a known key length as well. Lets try one last polyalphabetic cipher, but this time the key length is unknown.
>
> Enjoy.

Files [`found1` , `found2` and `found3`]:

SXULW GNXIO WRZJG OFLCM RHEFZ ALGSP DXBLM PWIQT XJGLA RIYRI
BLPPC HMXMG CTZDL CLKRU YMYSJ TWUTX ZCMRH EFZAL OTMNL BL
ULV MCQMG CTZDL CPTBI AVPML NVRJN SSXWT XJGLA RIQPE FUGVP
PGRLG OMDKW RSIFK TZYRM QHNXD UOWQT XJGLA RIQAV VTZVP LMA
IV ZPHCX FPAVT MLBSD OIFVT PBACS EQKOL BCRSM AMULP SPPYF CX
OKH LZXUO GNLID ZVRAL DOACC INREN YMLRH VXXJD XMSIN BXUGI U
PVRG ESQSG YKQOK LMXRS IBZAL BAYJM AYAVB XRSIC KKPYH ULWFU
YHBPG VIGNX WBIQP RGVXY SSBEL NZLVW IMQMG YGVSW GPWGG NA
RSP TXVKL PXWGD XRJHU SXQMI VTZYO GCTZR JYVBK MZHBX YVBIT
TPVTM OOWSA IERTA SZCOI TXXLY JAZQC GKPCS LZRYE MOOVC HIEKT
RSREH MGNTS KVEPN NCTUN EOFIR TPPDL YAPNO GMKGC ZRGNX ARV
MY IBLXU QPYYH GNXYO ACCIN QBUQA GELNR TYQIH LANTW HAYCP R
JOMO KJYTV SGVLY RRSIG NKVXI MQJEG GJOML MSGNV VERRC MRYB
A GEQNP RGKLB XFLRP XRZDE JESGN XSYVB DSSZA LCXYE ICXXZ OVT
PW BLEVK ZCDEA JYPCL CDXUG MARML RWVTZ LXIPL PJKKL CIREP RJ
YVB ITPVV ZPHCX FPCRG KVPSS CPBXW VXIRS SHYTU NWCGI ANNUN
VCOEA JLLFI LECSO OLCTG CMGAT SBITP PNZBV XWUPV RIHUM IBPHG
UXUQP YYHNZ MOKXD LZBAK LNTCC MBJTZ KXRSM FSKZC SSELP UM
ARE BCIPK GAVCY EXNOG LNLCC JVBXH XHRHI AZBLD LZWIF YXKLM PE
LQG RVPAF ZQNVK VZLCE MPVKP FERPM AZALV MDPKH GKKCL YOLRX
TSNIB ELRYN IVMKP ECVXH BELNI OETUX SSYGV TZARE RLVEG GNOQC
YXFCX YOQYO ISUKA RIQHE YRHDS REFTB LEVXH MYEAJ PLCXK TRFZX
YOZCY XUKVV MOJLR RMAVC XFLHO KXUVE GOSAR RHBSS YHQUS LXS
DJ INXLH PXCCV NVIPX KMFXV ZLTOW QLKRY TZDLC DTVXB ACSDE LV
YOL BCWPE ERTZD TYDXF AILBR YEYEG ESIHC QMPOX UDMLZ VVMBU
KPGEC EGIWO HMFXG NXPBW KPVRS XZCEE PWVTM OOIYC XURRV BH
CCS SKOLX XQSEQ RTAOP WNSZK MVDLC PRTRB ZRGPZ AAGGK ZIMAP
RLKVW EAZRT XXZCS DMVVZ BZRWS MNRIM ZSRYX IEOVH GLGNL FZK
HX KCESE KEHDI FLZRV KVFIB XSEKB TZSPE EAZMV DLCSY ZGGYK GCE
LN TTUIG MXQHT BJKXG ZRFEX ABIAP MIKWA RVMFK UGGFY JRSIP NBJ
UI LDSSZ ALMSA VPNTX IBSMO GLCYX UKFHS PEZXF AVJOW QQYYR RA
YHM GIEOG ARIAZ YEYXV PXFPJ BXXUY SLELR NXHNH PLARX TADLC C
SLGE NOSPR IUUML VSNPR RJMOO GMLGU JHVBE QSMFI NZDSK HEFN
X KSHGE AVZAZ YQCQP BAKPC LMQGR XXTYR WQSEG FHSPH ZYETX F

PVMX PBTWV XMLHM AZXYG EQLRN IAPOZ CXIAZ MVMSL RVNZN SKXC
L RNJOL XXSCS HYMYK ZCWPR XNWYR ZJXUG MASQC ELRXX DKWMY
PLUGL KHTPR GAKVE WRCEI KESOV JPJGH XJYRE CEGAE HDIBQ SEZAL
DAMZX UKKZR EBMIR TLLDH MHRNZ MOOMP CIFVX JDMTP VBGWZ SH
COI FZBUK XGZRF ZALWM JOIJE BUCMB PSSZA LMSYN LJOMO SXQOE
ZVTUN HGCXL YMYKA GEWQO LHQIC LFYKL TOPJL RQOMZ YFQNY EO
MFG EQCEG NXYVM IPEYG KNOVB ZKXKG UOPKC PBXKF DLCAE FYXUQ
IPDLN QBUQL GXWRR YVEXM QMGOG JREGY WBLLA BEULX NTZSO SD
DLN MZFGV YATRX YSKTN TRTNT AKRBX YQJRS OKQHE FXTAR IPWMX
KTSKV EPVFU KAYJB ZKGNX YOAGW POKTW KGIPX GUVHV EGDXB SHY
BS UOVNC XYIIQ DMEOY ARIUP EGNXY RSJOW NTWAR IUTRQ YXACX M
WIEG USOJY TVGNX ASHCH MYRLL BZCAV RZMFX MAPPL GMHLS SEXJ
U BUDLC LJGKK UYSLD MEHXK CMPTW UGESX SRRSG UULNX GWPAO
ZODFS EMJGG AKFCO VBUFH XHYME EHXYK RBELR TUYOE IQEFZ LPBC
C DWVXM OKXUL CFOKP PCMFT YKTZO WFZAP UGJYV BRIAZ ELWEL D
ZNRB ZOELO LBZPH DIPES PUGJY VBAYY RHMPK CYXYK FHXWZ ZSGYB
UMSLN SEJRV EAGWP SOGKK JGYIF KTJYE JQMEK LPBJC EGUHT YLIPE
SPUGJ YVBDX VXTIY YRELR XXUYA DZVPU GJYVB ELRIH UMSPO FRJVO
KQZPV OKBUQ EJHEL YTZCM EYIQZ HHZEQ DIAMX YLCRS IZGBS KRBAE
FYXUQ IPDFL ZALWE GWFRO GNKPU LCFNX HFMJJ AEGIW OHSAJ EUF
OO EBESS UHADL CCSBS AHNXF PSQJB UDIPP WGLHY DLCPW GGUSS
WFXIA ZHMDL CCSLG ENOSP RIGNT AKPRS SHMAI EXMYI XOGKY JKLRJ
GLZOI LESTU BUDSG EEYRD PXHQL RQBTY SIRTI FUYTO RALQR UNAYJ
GEGBT LLAYC YXYET UYXFP VQXTD OVYYH GCHWY VRPVF GGKCI TPV
NR FHSHQ LRQZA LVELO PNJRD OVCLP YRHPD IPTRT HRHMG GOIAZ TA
FEP TSHYI VSRRD SSZAL BSYOF RZPLO RRSIP UGJYV BLRQZ ALMSD QIR
XH VWAFP RNMXU DPCXE AUYZS BRJJB XFHVP WOVRY LLNML LFEUP
UCYGE SSIEV DLCDT EKMAI ACWPJ UKULY RGIEE PLVPI PTGCB ARPYC K
RYJB KVCNY SLLHX HJLVT KYSKT QESGN XWYGI PXFVT ZCIBL PBTZV X
LGDA NEMVR MQMVR GDMKW RFIPJS EJXYV CYYHZ KMOYH GNEYN XS
YSI PHJOM OKLYY HBTXH MLIYI RGGKK PMFHJ GMJRX GNOVT ZHCSL Z
VBAL ZOVKZ RHTWL BLGDJ YGIWO HULMF ZVVKX YDXUU NNRMR AMG
ZX KSXQR VNBBA IELOP BTZLF MRJET GBUCX RSIYK OPDCY YHRBT UO
WAP RPKHM DLCMV VYDMS VCSIU GWHQS MOPRM TUNAY DEYOM AVI
TL MAUYP DJMCL VYUYY ALDXB IDPXK QQMGZ XKCPC PONTW JVSQP
EAJPL BIMQE SOGLD IVEYE KAPCW FZIFG GKLYA VPRYM VYXFZ YTNIS

KMLHI EKMYS QFPAB XXHXS BOPVZ MSOWJ PIXIK PCTDW EKKGD SKQP
X GOGNF IPJGY ULLDS FTWUK TKGLG NLJOZ PDMQE SOKIY OWSXI QCT
ZW EBPSS NTPBF SEAUO VOVSM VIQLT YWSPP EFZAV EKFTX JKKLC TS
YJE UFMSP YXIAZ LVPWG WOBXZ SKWQS MFRBU ORRSS HMAUY XMQE
S OGLXI QDMAG VJYVB LRPKP PDLFT WFZHJ UMLRW JGLHC AFTXR GL
ARI RZTFU YARIU LZRYM OKXZC SXKNW YRRSI AKBNR FMFVV TZIOE AS
SEZ ALCTC NOFUY ZKMJE LNZZS SRRPH VTMOO WSYPV MAAPE PLXFK
THPEA PLNHB AEEJW CFAIW BIQDI QGGKA YGPXR JPHCW RTPYR BNRX
C OYCAG KOVRS IDATP XXUTK OETWK MPZJZ UBZDF PTKUZ XFOWR SE
GOM TEWRS EIKVV CXRSI VXHDX IPTRL KTYCK MYIOE LVWIN LMAYM V
NVGW PGUMO OGMXT BYXKK RBCIF KKCOH CITEK LZSSL ZJGKE SCSLD
FNTDO OLYOE UKTSD LWNSY UNYSR FTWPN XLUWY YHUOL MKGCE LB
AZO VMLPH OUKLP IUEVN IXZYJ YYBVK MFLYR AIENT WCXFP GBTYP NI
LEM NRUHM LCWSE IELBO QTRGK ESCSL DFNTD DOVCA VVTVP ZEJWC
BIVBZ MCOAV ZAARI ALVRY HMYXF PVCKH WVIYY HCKKO KTQDI PUGK
R ELOGN XXZVM IPWRI HUNLY YHPRH ARIQN SZKXH CMJJS SLTUN SLN
SZ VELDM LRLVY KLCIK MPNTV LDSYX EACAV GEQDM GZBUQ JMCLV YI
VBX PLMGS KSYVP JHEUI WOHMQ JGULS OINEL RGKYS ZYWSS NBZLV
CLOSG LABSS DIQNB TKRBS IFGBK DSRSI QXTDO VYDLR SHCOH FTWPN
TPBXM TXVCB ZREAN SZSHK KXGZR CXXWK VCOJB XTFYY LRPNJ RDR
SK LCPUF LRIPP EGGGF DMKPX BJTFC LCXEL GLRPS PXVWG KCSWJ ZV
EEH YCLCX ELUGS IEQVJ BXTNO RRWIZ GGMBS KEIYR LVXWZ LRXVE LK
WCE SYKMT OOLZA LKLZS VRPPY YHUCF YYOVT EVXHM YWVXR LCCC
D WVXPL RETPS SZXUD MKPWG NXOYR MFVGU XUDIP EEVTR VEVEP R
GRXT ORGYX UKBYD VYGIY RBUQF YNOJG KKCEL OJBXP HBHQM IGCBE
DPMYH BTTUN TYCMF YBYKZ YDXQK TSYJR CEIKE SSRED MEOGA OPJD
S AGGKM SKAEA ELOYY QPCRY PLKVC BYVZX HPVCY GUNHB CIYDA RR
EHC ELPRT RBZRS LPCRY LPBRM EQHIA PXXFP LNHBA YJQFG UZKHF IJ
WMA MRVEV QPPSO MOSRI DMETH AYJJL XREXH BWGEM FLBMD ICYC
R GKZCM LNIJK LPXGC TGNSX SKWRQ VBSYY KRAP

Level 6 - pass [krypton6]:

BELOS Z

Now the, since we do not know the key length which usually is the case more often than not, we try to observe the repeated sequences that are a certain amount of characters apart- then the factors of that difference can tell the story for us.
So for example, if they are 15 characters apart, then 5 and 3 and then we add them to the total of the amount of spaces apart for all the factors. It will make more sense when we try to do the same;

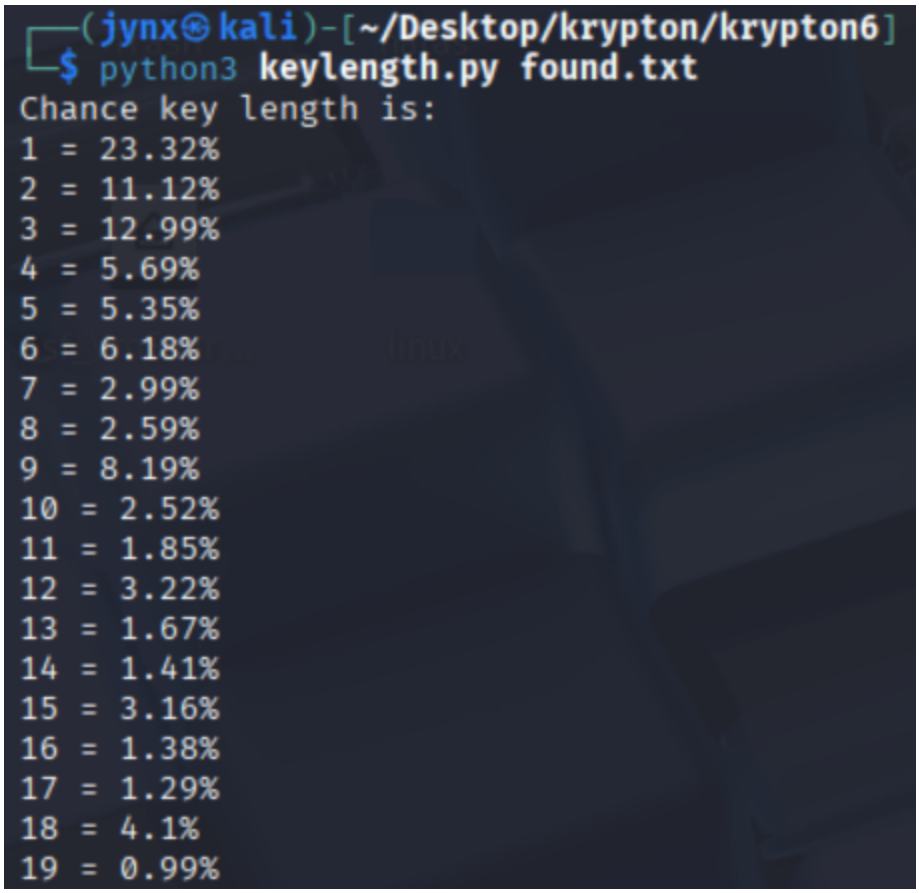
I found a wonderful script that helps us achieve this:

```
import sys

if __name__=='__main__':

    rep_seq = []
    rep_seq_spaces = []
    tallys = [0]*20

    if len(sys.argv) != 2:
        print("Usage: python3 keyLength.py filename")
        exit(0)
    else:

        try:
            with open(sys.argv[1]) as fh:
                lines = fh.readlines()
        except:
            print("No file named '" + sys.argv[1] + "'")
            exit(0)

        for line in lines:
            line=line.replace(" ", "")
            line=line.replace("\n", "")
            for length in range(3,5+1):
                for i in range(len(line)-length+1):
                    q = line[i:i+length]
```

```
            for j in range(i+1,len(line)-length+1):
                r = line[j:j+length]
                if q == r:
                    rep_seq.append(q)
                    rep_seq_spaces.append(j-i)

    for n in rep_seq_spaces:
        for i in range(1,20):
            if n % i == 0:
                tallys[i] = tallys[i] + 1

    s = sum(tallys)
    if s == 0:
        print("No repeating sequences")
    else :
        print("Chance key length is: ")
        for i in range(1,len(tallys)):
            print(str(i) + " = " + str(round((tallys[i]/s)* 100,2)) + "%")
```

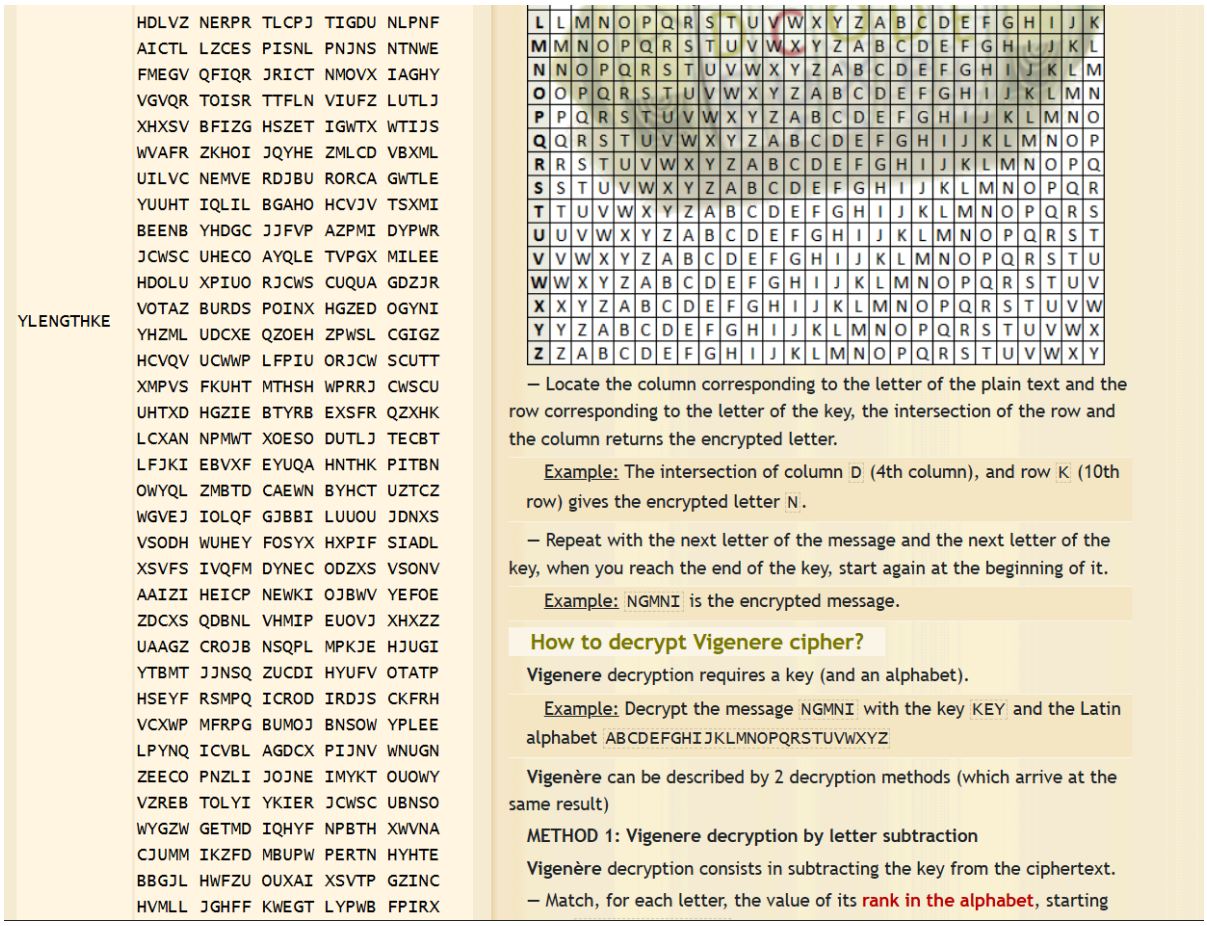
Credits: https://github.com/m-rosinsky/Krypton_Scripts/blob/master/keyLength.py

And we get:



9 has a good tangible possibility a whooping `8.19%` as compare to others 1, 2 and 3 that generally always have a greater probability but more often than not is never the key length. Will stick with 9 for now.

I can go through the previous method of manually decoding now that we have the key length, but this time around just to save some time and effort I will be using an online Vigenère decoder, sorry not sorry lol;



```
YLENGTHKE
```

```
HDLVZ NERPR TLCPJ TIGDU NLPNF
AICTL LZCES PISNL PNJNS NTNWE
FMEGV QFIQR JRICT NMOVX IAGHY
VGVQR TOISR TTFLN VIUFZ LUTLJ
XHXSV BFIZG HSZET IGWTX WTIJS
WVAFR ZKHOI JQYHE ZMLCD VBXML
UILVC NEMVE RDJBU RORCA GWTLE
YUUHT IQLIL BGAHO HCVJV TSXMI
BEENB YHDGC JJFVP AZPMI DYPWR
JCWSC UHECO AYQLE TVPGX MILEE
HDOLU XPIUO RJCWS CUQUA GDZJR
VOTAZ BURDS POINX HGZED OGYNI
YHZML UDCXE QZOEH ZPWSL CGIGZ
HCVQV UCWWP LFPIU ORJCW SCUTT
XMPVS FKUHT MTHSH WPRRJ CWSCU
UHTXD HGZIE BTYRB EXSFR QZXHK
LCXAN NPMWT XOESO DUTLJ TECBT
LFJKI EBVXF EYUQA HNTHK PITBN
OWYQL ZMBTD CAEWN BYHCT UZTCZ
WGVEJ IOLQF GJBBI LUUOU JDNXS
VSODH WUHEY FOSYX HXPIF SIADL
XSVFS IVQFM DYNEC ODZXS VSONV
AAIZI HEICP NEWKI OJBWV YEFOE
ZDCXS QDBNL VHMIP EUOVJ XHXZZ
UAAGZ CROJB NSQPL MPKJE HJUGI
YTBMT JJNSQ ZUCDI HYUFV OTATP
HSEYF RSMPQ ICROD IRDJS CKFRH
VCXWP MFRPG BUMOJ BNSOW YPLEE
LPYNQ ICVBL AGDCX PIJNV WNUGN
ZEECO PNZLI JOJNE IMYKT OUOWY
VZREB TOLYI YKIER JCWSC UBNSO
WYGZW GETMD IQHYF NPBTH XWVNA
CJUMM IKZFD MBUPW PERTN HYHTE
BBGJL HWFZU OUXAI XSVTP GZINC
HVMLL JGHFF KWEGT LYPWB FPIRX
```

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

— Locate the column corresponding to the letter of the plain text and the row corresponding to the letter of the key, the intersection of the row and the column returns the encrypted letter.

Example: The intersection of column `D` (4th column), and row `K` (10th row) gives the encrypted letter `N`.

— Repeat with the next letter of the message and the next letter of the key, when you reach the end of the key, start again at the beginning of it.

Example: `NGMNI` is the encrypted message.

How to decrypt Vigenere cipher?

Vigenere decryption requires a key (and an alphabet).

Example: Decrypt the message `NGMNI` with the key `KEY` and the Latin alphabet `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Vigenère can be described by 2 decryption methods (which arrive at the same result)

METHOD 1: Vigenere decryption by letter subtraction

Vigenère decryption consists in subtracting the key from the ciphertext.

— Match, for each letter, the value of its **rank in the alphabet**, starting

Although this isn't the key itself: `YLENGTHKE` **I think it must be** `KEYLENGTH` instead, lets presume that is what it is, and try to decode our next password:

And there we go, finally found the password.

Password:

```
RANDOM
```

▼ Level 6

And now for our final stage, Level Info:

> Hopefully by now its obvious that encryption using repeating keys is a bad idea. Frequency analysis can destroy repeating/fixed key substitution crypto.
>
> A feature of good crypto is random ciphertext. A good cipher must not reveal any clues about the plaintext. Since natural language plaintext (in this case, English) contains patterns, it is left up

to the encryption key or the encryption algorithm to add the 'randomness'.

Modern ciphers are similar to older plain substitution ciphers, but improve the 'random' nature of the key.

An example of an older cipher using a complex, random, large key is a vigniere using a key of the same size of the plaintext. For example, imagine you and your confident have agreed on a key using the book 'A Tale of Two Cities' as your key, in 256 byte blocks.

The cipher works as such:

Each plaintext message is broken into 256 byte blocks. For each block of plaintext, a corresponding 256 byte block from the book is used as the key, starting from the first chapter, and progressing. No part of the book is ever re-used as key. The use of a key of the same length as the plaintext, and only using it once is called a "One Time Pad".

Look in the krypton6/onetime directory. You will find a file called 'plain1', a 256
byte block. You will also see a file 'key1', the first 256 bytes of 'A Tale of Two Cities'. The file 'cipher1' is the cipher text of plain1. As you can see (and try) it is very difficult to break the cipher without the key knowledge.

(NOTE - it is possible though. Using plain language as a one time pad key has a weakness. As a secondary challenge, open README in that directory)

If the encryption is truly random letters, and only used once, then it is impossible to break. A truly random "One Time Pad" key cannot be broken. Consider intercepting a ciphertext message of 1000 bytes. One could brute force for the key, but due to the random key nature, you would produce every single valid 1000 letter plaintext as well. Who is to know which is the real plaintext?!?

Choosing keys that are the same size as the plaintext is impractical. Therefore, other methods must be used to obscure ciphertext against frequency analysis in a simple substitution cipher. The

impracticality of an 'infinite' key means that the randomness, or entropy, of the encryption is introduced via the method.

We have seen the method of 'substitution'. Even in modern crypto, substitution is a valid technique. Another technique is 'transposition', or swapping of bytes.

Modern ciphers break into two types; symmetric and asymmetric.

Symmetric ciphers come in two flavours: block and stream.

Until now, we have been playing with classical ciphers, approximating 'block' ciphers. A block cipher is done in fixed size blocks (suprise!). For example, in the previous paragraphs we discussed breaking text and keys
into 256 byte blocks, and working on those blocks. Block ciphers use a fixed key to perform substituion and transposition ciphers on each block discretely.

Its time to employ a stream cipher. A stream cipher attempts to create an on-the-fly 'random' keystream to encrypt the incoming plaintext one byte at a time. Typically, the 'random' key byte is xor'd with the plaintext to produce the ciphertext. If the random keystream can be replicated at the recieving end, then a further xor will produce the plaintext once again.

From this example forward, we will be working with bytes, not ASCII text, so a hex editor/dumper like hexdump is a necessity. Now is the right time to start to learn to use tools like cryptool.

In this example, the keyfile is in your directory, however it is
not readable by you. The binary 'encrypt6' is also available.
It will read the keyfile and encrypt any message you desire, using the key AND a 'random' number. You get to perform a 'known ciphertext' attack by introducing plaintext of your choice. The challenge here is not simple, but the 'random' number generator is weak.

As stated, it is now that we suggest you begin to use public tools, like cryptool,
to help in your analysis. You will most likely need a hint to get going.
See 'HINT1' if you need a kicktstart.

> If you have further difficulty, there is a hint in 'HINT2'.
>
> The password for level 7 (krypton7) is encrypted with 'encrypt6'.
>
> Good Luck!

File [`plain1`]:

```
SINGOGODDESSTHEANGEROFACHILLESSONOFPELEUSTHATBROUGHTC
OUNTLESSILLSUPONTHEACHAEANSMANYABRAVESOULDIDITSENDHURR
YINGDOWNTOHADESANDMANYAHERODIDITYIELDAPREYTODOGSANDV
ULTURESFORSOWERETHECOUNSELSOFJOVEFULFILLEDFROMTHEDAYO
NWHICHTHESONOFATREUSKINGOFMENANDGREATACHILL
```

File [`key1`]:

```
ITWASTHEBESTOFTIMESITWASTHEWORSTOFTIMESITWASTHEAGEOFWI
SDOMITWASTHEAGEOFFOOLISHNESSITWASTHEEPOCHOFBELIEFITWAS
THEEPOCHOFINCREDULITYITWASTHESEASONOFLIGHTITWASTHESEAS
ONOFDARKNESSITWASTHESPRINGOFHOPEITWSTHEWINTEROFDESPAIR
WEHADEVERYTHINGBEFOREUSWEHADNOTHINGBEF
```

File [`cipher1`]:

```
ABJGGZVHEIKLHMXIZKWZHBAUAPPHSJKHBTYXQPWCLPHSMIVOAKVYY
WMQHXMLOIDEZYPURHMJOQSIWHAWESVRWBJTCIWDINKWIJXDMRIPN
NRQBUKHDKPACMIQGJEQXXIGWIAARGWPHAXYASYRFAZKFMWWKGKT
UHNYLLIESXIOICBAWJMMDEUHBRKTCABLXTCSUYTYELDXKJNWZMLVR
FBSFLHQTDXOEVSISWYMYMHYLMSUFJGWJEUDJESTAIPNJPQ
```

→ As mentioned in level info, these are the files- an illustration of how the cipher is working.

HINT File(s):

`HINT 1`:

```
The 'random' generator has a limited number of bits, and is periodic.
Entropy analysis and a good look at the bytes in a hex editor will help.
```

> There is a pattern!

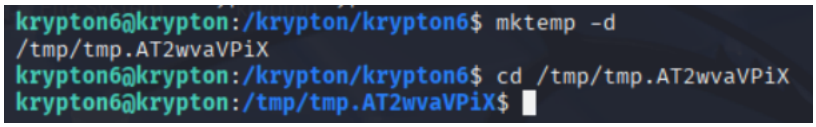
`HINT 2`:

> 8 bit LFSR

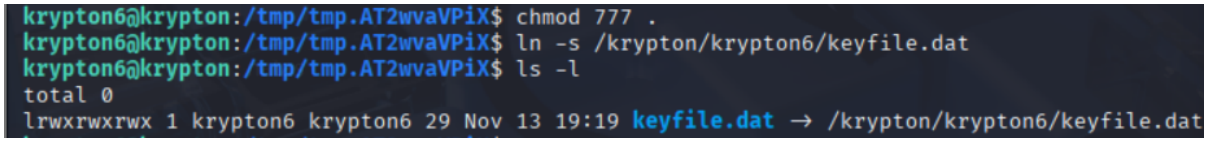
Level 7 - pass [`krypton7`]:

> PNUKLYLWRQKGKBE

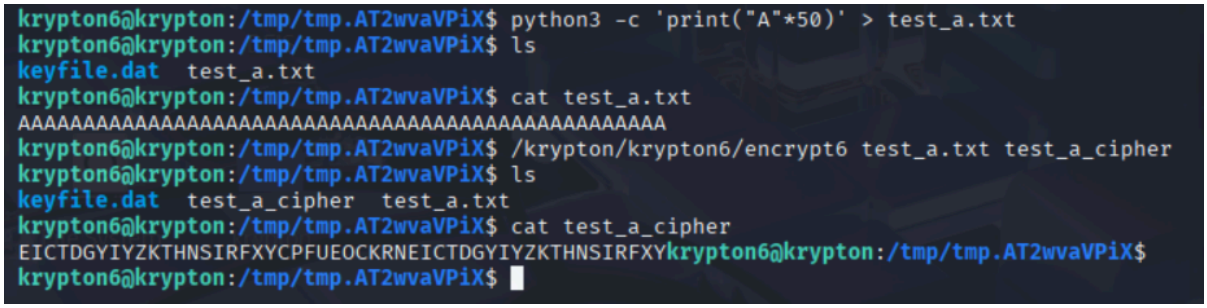
Let's begin with understanding how the encryption begins, first we should create a temporary directory to perform our operations:



Now we will create a soft link to the `keyfile.dat` file and execute to see how it behaves, we will also make our temp directory executable;

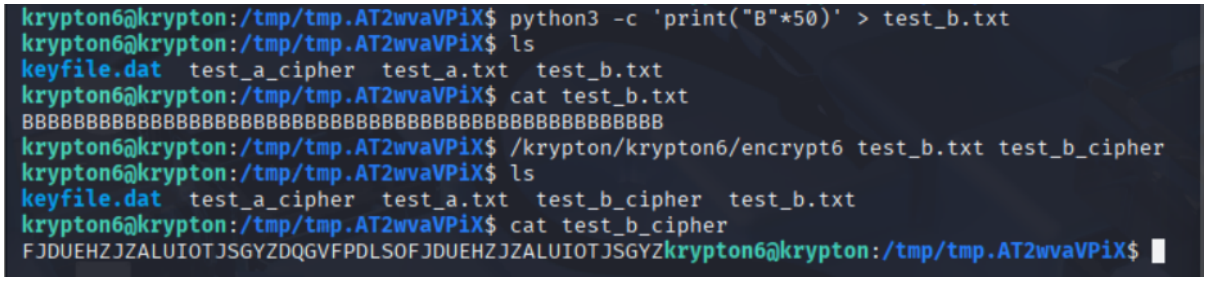


We will now try to find how it behaves with "A" for everytime:



you can notice how for every A it gets different output(s):

Now, lets try doing the same for "B" :



You can notice each character moved only 1 character ahead for each sequential character once.

Now, we can straight up try to decode this like:

> Level 7 Pass: P N U K L Y L W R Q K G K B E
> CIPHER [key]: E I C T D G Y I Y Z K T H N S

Example:

Let's decrypt the first character manually:

Cipher	Key	Calculation	Result
P (80)	E (69)	(26 + 80 - 69) % 26 = 11 → 11 + 65 = 76	L

So the first decrypted character is **L**.

A python script to do this can be:

```
cipher_text = 'PNUKLYLWRQKGKBE'
key_text = 'EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGY'
decrypt_text = ''
for i in range(len(cipher_text)):
    decrypt_text += chr((26 + ord(cipher_text[i]) - ord(key_text[i])) % 26 + o
rd('A'))

print(decrypt_text)
```

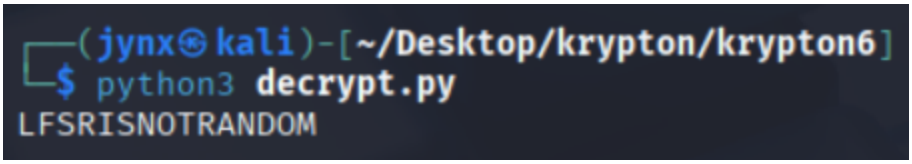
This script implements **Vigenère cipher decryption**, which is a classical encryption method based on Caesar shifts — but each letter is shifted by a value derived from a *key letter* instead of a fixed number.

Step	Explanation
`ord(cipher_text[i])`	Converts cipher letter to its ASCII code (e.g., `'A' → 65`, `'B' → 66`)
`ord(key_text[i])`	Converts the key letter to its ASCII code
`ord(cipher_text[i]) - ord(key_text[i])`	Finds how far cipher letter is shifted compared to key letter
`+ 26`	Ensures result is non-negative before modulo
`% 26`	Wraps around alphabet range (A–Z → 0–25)
`+ ord('A')`	Shifts back into printable ASCII range (65–90 for uppercase)
`chr(...)`	Converts that number back into a letter

Each resulting letter is appended to `decrypt_text`.

After processing all 14 characters, you get the full decrypted message stored in `decrypt_text`, which is printed at the end.

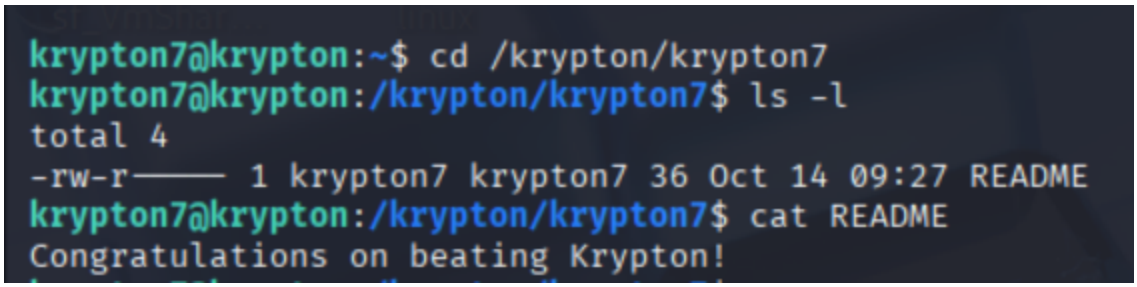
And now when we execute the script:



Password:

```
LFSRISNOTRANDOM
```

▼ Level 7


krypton7@krypton:~$ cd /krypton/krypton7
krypton7@krypton:/krypton/krypton7$ ls -l
total 4
-rw-r----- 1 krypton7 krypton7 36 Oct 14 09:27 README
krypton7@krypton:/krypton/krypton7$ cat README
Congratulations on beating Krypton!