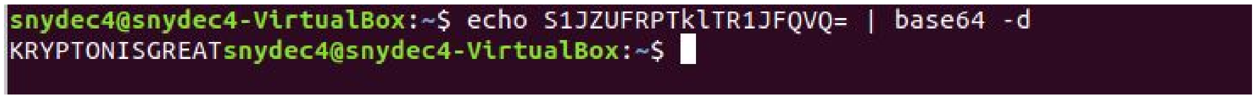
Krypton Guide:

Overview:
The goal is to document my thought process while working through each level while also being able to possibly guide others who may need a hint or some help on these challenges, just like I had done with the Bandit and Leviathan games. This wargame only has 7 levels, and the levels are designed to teach encryption/decryption. While writing this guide, I am going to assume that those who are reading it have a general sense of how to navigate the linux terminal / command line, so I won't go in depth on how to do every little step such as change directories, list contents of a directory, log into the game, etc. It is also suggested that if you are an absolute beginner you don't try krypton without doing the Bandit game first.

Important note: For some reason, the machine that we ssh into is set up so that when you ssh into krypton, you are not in the correct krypton directory to do the work, at least that is how it was for me. When I ssh into the machine the directory I start in is /home/krypton/krypton# which is not the correct directory to find the files they refer to in the level goals. Where you should be is /krypton/krypton# so just know that before the start of each level you must move to the correct folder depending on your krypton level.

Krypton 0 → 1:
The first level tells us that the string: S1JZUFRPTklTR1JFQVQ= encodes the password for krypton1. So I just opened up the terminal on my linux machine and did a quick base64 decode to acquire the password:



```
snydec4@snydec4-VirtualBox:~$ echo S1JZUFRPTklTR1JFQVQ= | base64 -d
KRYPTONISGREATsnydec4@snydec4-VirtualBox:~$ 
```

Krypton 1 → 2:
The level goal states: "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 ciphertext. Enjoy!"

Don't forget to go to the correct krypton directory if not there already

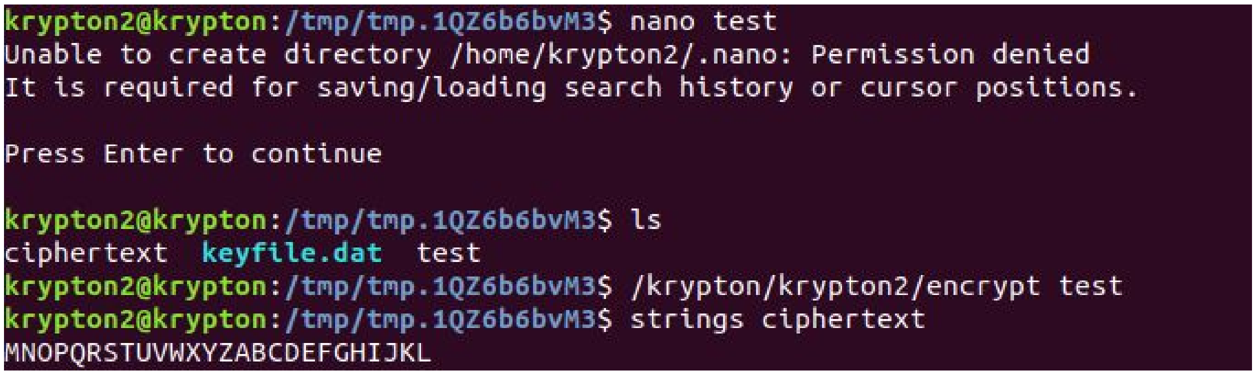
.

In the directory there is a `README` ascii file, and a `krypton2` ascii file. Printing the file krypton2 shows a string of jumbled words, and since we know the string is encrypted with a rotation, the first type of encryption that popped into my head was rot13. So how I solved the decryption was I went to rot13.com and pasted the string into the box, and hit enter, and it decrypted the string!

<u>Krypton 2 → 3:</u>

The level goal for this level is quite large, so I won't paste it word for word, but I can paste some of the important parts: "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."

So on the website, they layout some instructions for creating your temporary directory to work in because when you use the encrypt binary, it looks for the key file in the current working directory as described in the additional information section. So for the first step I just followed the steps exactly as they have in the example. From all this I was able to gather that the binary takes the input file, and encrypts it according to a certain type of encryption, then writes it to a file in the working directory called ciphertext. I needed a way to figure out how the cipher works, so how I figured it out was by using the nano text editor to make an ascii text file which just had all of the letters of the alphabet listed sequentially. Once I knew how the encryption binary encrypted the strings, I could decrypt it. I passed my test file into the binary:



```
krypton2@krypton:/tmp/tmp.1QZ6b6bvM3$ nano test
Unable to create directory /home/krypton2/.nano: Permission denied
It is required for saving/loading search history or cursor positions.

Press Enter to continue

krypton2@krypton:/tmp/tmp.1QZ6b6bvM3$ ls
ciphertext  keyfile.dat  test
krypton2@krypton:/tmp/tmp.1QZ6b6bvM3$ /krypton/krypton2/encrypt test
krypton2@krypton:/tmp/tmp.1QZ6b6bvM3$ strings ciphertext
MNOPQRSTUVWXYZABCDEFGHIJKL
```

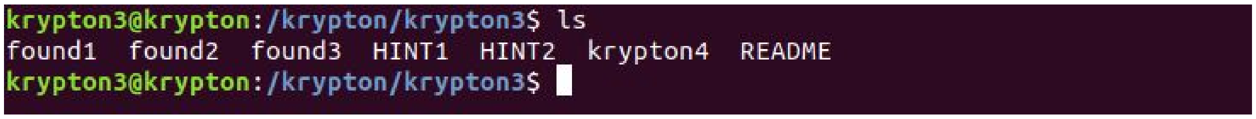
We can see here that the first letter of the alphabet got changed from 'A' to 'M' by way of a rotation. We know that if you were to rotate the alphabet 14 time to the right, we would get this type of alphabet, therefore we can determine that the type of encryption was Rot14. We can now go to https://rot13.com and plug in our original string, select rot14 as our decryption type, then get our password!

Krypton 3 → 4:

The level goal states: “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)

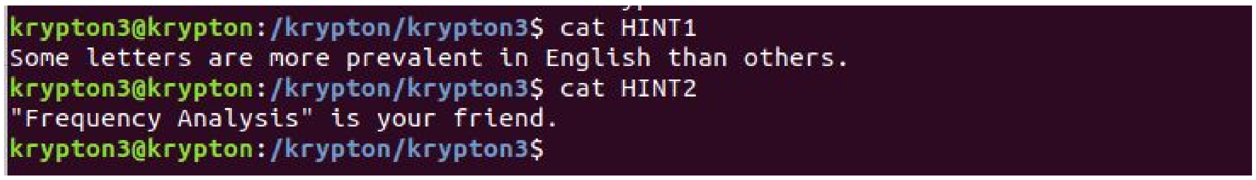
You know the following important details:
The message plaintexts are in English (*** very important) - They were produced from the same key (*** even better!). Enjoy.”

So let's just start by taking a look at what we have in the krypton3 directory:



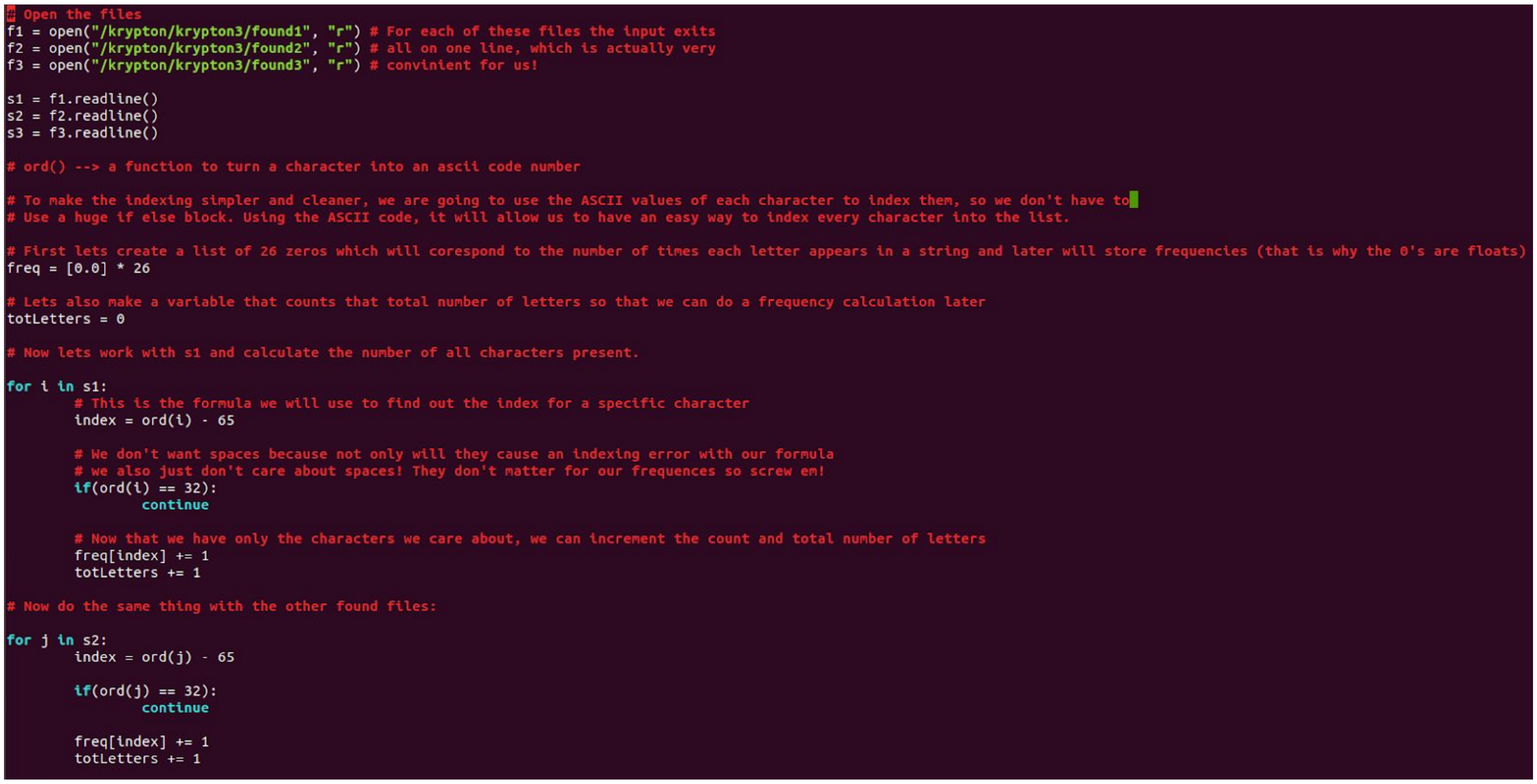
```
krypton3@krypton:/krypton/krypton3$ ls
found1  found2  found3  HINT1  HINT2  krypton4  README
krypton3@krypton:/krypton/krypton3$ 
```

In addition to the found files, there are also some hint files:

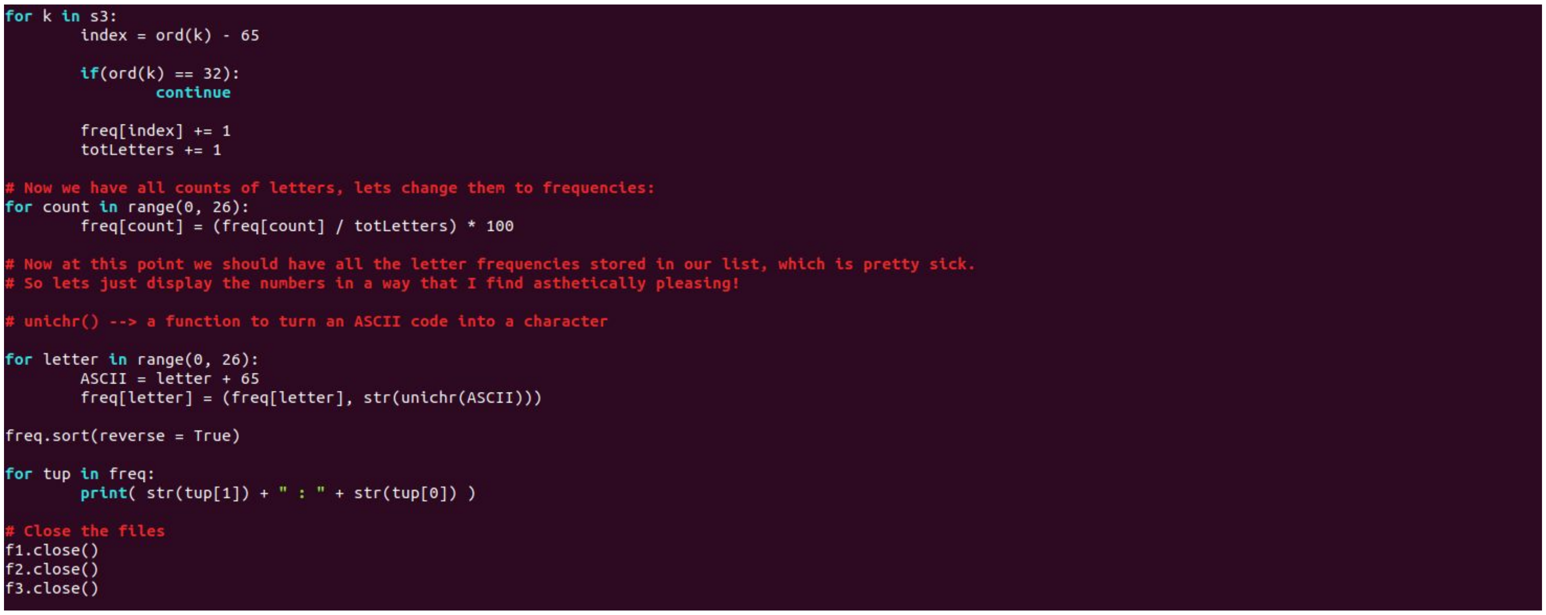


```
krypton3@krypton:/krypton/krypton3$ cat HINT1
Some letters are more prevalent in English than others.
krypton3@krypton:/krypton/krypton3$ cat HINT2
"Frequency Analysis" is your friend.
krypton3@krypton:/krypton/krypton3$
```

So this gives us a big hint, “Frequency analysis is the study of the distribution of the letters in a text. Analysis of frequencies help decrypting substitution-based ciphers using the fact that some letters apparitions are varying in a given language : in english, letters E, T or A are common while Z or Q are rare” (dcode.fr). So we are going to have to do some Frequency analysis to see if we can find out the cipher. Obviously I don’t want to go through and count every letter in every file and develop a frequency plot myself, that would suck. So my first idea currently is to create a python script that runs through all the found files, and compares the frequency of each letter in the file. It took me a little while to write, so I will paste the code below and the comments in the code hopefully will explain exactly what I had done:

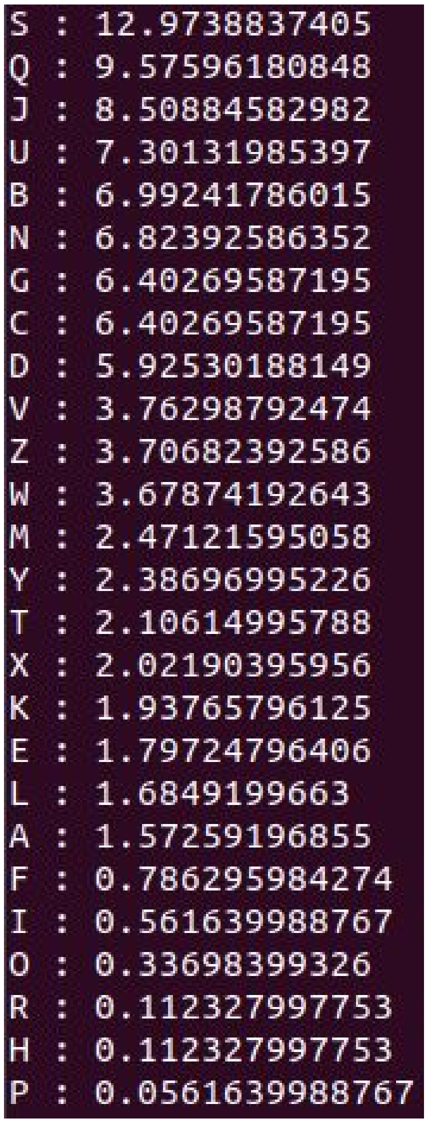


```
# Open the files
f1 = open("/krypton/krypton3/found1", "r") # For each of these files the input exits
f2 = open("/krypton/krypton3/found2", "r") # all on one line, which is actually very
f3 = open("/krypton/krypton3/found3", "r") # convinient for us!

s1 = f1.readline()
s2 = f2.readline()
s3 = f3.readline()

# ord() --> a function to turn a character into an ascii code number

# To make the indexing simpler and cleaner, we are going to use the ASCII values of each character to index them, so we don't have to
# Use a huge if else block. Using the ASCII code, it will allow us to have an easy way to index every character into the list.

# First lets create a list of 26 zeros which will corespond to the number of times each letter appears in a string and later will store frequencies (that is why the 0's are floats)
freq = [0.0] * 26

# Lets also make a variable that counts that total number of letters so that we can do a frequency calculation later
totLetters = 0

# Now lets work with s1 and calculate the number of all characters present.

for i in s1:
        # This is the formula we will use to find out the index for a specific character
        index = ord(i) - 65

        # We don't want spaces because not only will they cause an indexing error with our formula
        # we also just don't care about spaces! They don't matter for our frequences so screw em!
        if(ord(i) == 32):
                continue

        # Now that we have only the characters we care about, we can increment the count and total number of letters
        freq[index] += 1
        totLetters += 1

# Now do the same thing with the other found files:

for j in s2:
        index = ord(j) - 65

        if(ord(j) == 32):
                continue

        freq[index] += 1
        totLetters += 1
```

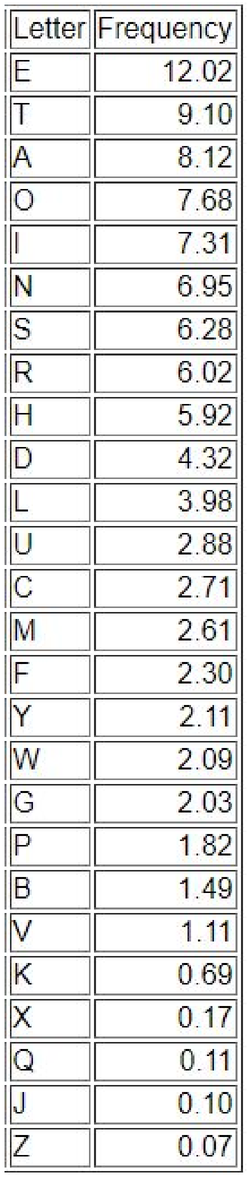


```
for k in s3:
        index = ord(k) - 65

        if(ord(k) == 32):
                continue

        freq[index] += 1
        totLetters += 1

# Now we have all counts of letters, lets change them to frequencies:
for count in range(0, 26):
        freq[count] = (freq[count] / totLetters) * 100

# Now at this point we should have all the letter frequencies stored in our list, which is pretty sick.
# So lets just display the numbers in a way that I find asthetically pleasing!

# unichr() --> a function to turn an ASCII code into a character

for letter in range(0, 26):
        ASCII = letter + 65
        freq[letter] = (freq[letter], str(unichr(ASCII)))

freq.sort(reverse = True)

for tup in freq:
        print( str(tup[1]) + " : " + str(tup[0]) )

# Close the files
f1.close()
f2.close()
f3.close()
```

Now we take this program and run it to get our frequencies and compare with the english alphabet:

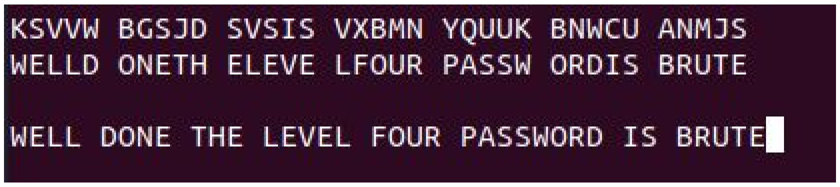


```
S : 12.9738837405
Q : 9.57596180848
J : 8.50884582982
U : 7.30131985397
B : 6.99241786015
N : 6.82392586352
G : 6.40269587195
C : 6.40269587195
D : 5.92530188149
V : 3.76298792474
Z : 3.70682392586
W : 3.67874192643
M : 2.47121595058
Y : 2.38696995226
T : 2.10614995788
X : 2.02190395956
K : 1.93765796125
E : 1.79724796406
L : 1.6849199663
A : 1.57259196855
F : 0.786295984274
I : 0.561639988767
O : 0.33698399326
R : 0.112327997753
H : 0.112327997753
P : 0.0561639988767
```



Letter	Frequency
E	12.02
T	9.10
A	8.12
O	7.68
I	7.31
N	6.95
S	6.28
R	6.02
H	5.92
D	4.32
L	3.98
U	2.88
C	2.71
M	2.61
F	2.30
Y	2.11
W	2.09
G	2.03
P	1.82
B	1.49
V	1.11
K	0.69
X	0.17
Q	0.11
J	0.10
Z	0.07

So this is how the frequency of our alphabet corresponds with the frequency of the regular english alphabet. It is important to remember there is a degree of inaccuracy to our frequencies, so lets try a couple of substitutions before we cement one letter to one spot. So let's make a mapping from our cipher text to the english alphabet based on these frequencies trying different letters for each substitution until we get a string that looks like words:



```
KSVVW BGSJD SVSIS VXBMN YQUUK BNWCU ANMJS
WELLD ONETH ELEVE LFOUR PASSW ORDIS BRUTE

WELL DONE THE LEVEL FOUR PASSWORD IS BRUTE
```

As much as I don't like brute forcing, we kind of had to for this level because we had no guarantee that our frequencies were exactly correct, so it was a lot of trial and error to get our string here which reveals the password!

Krypton 4 → 5:

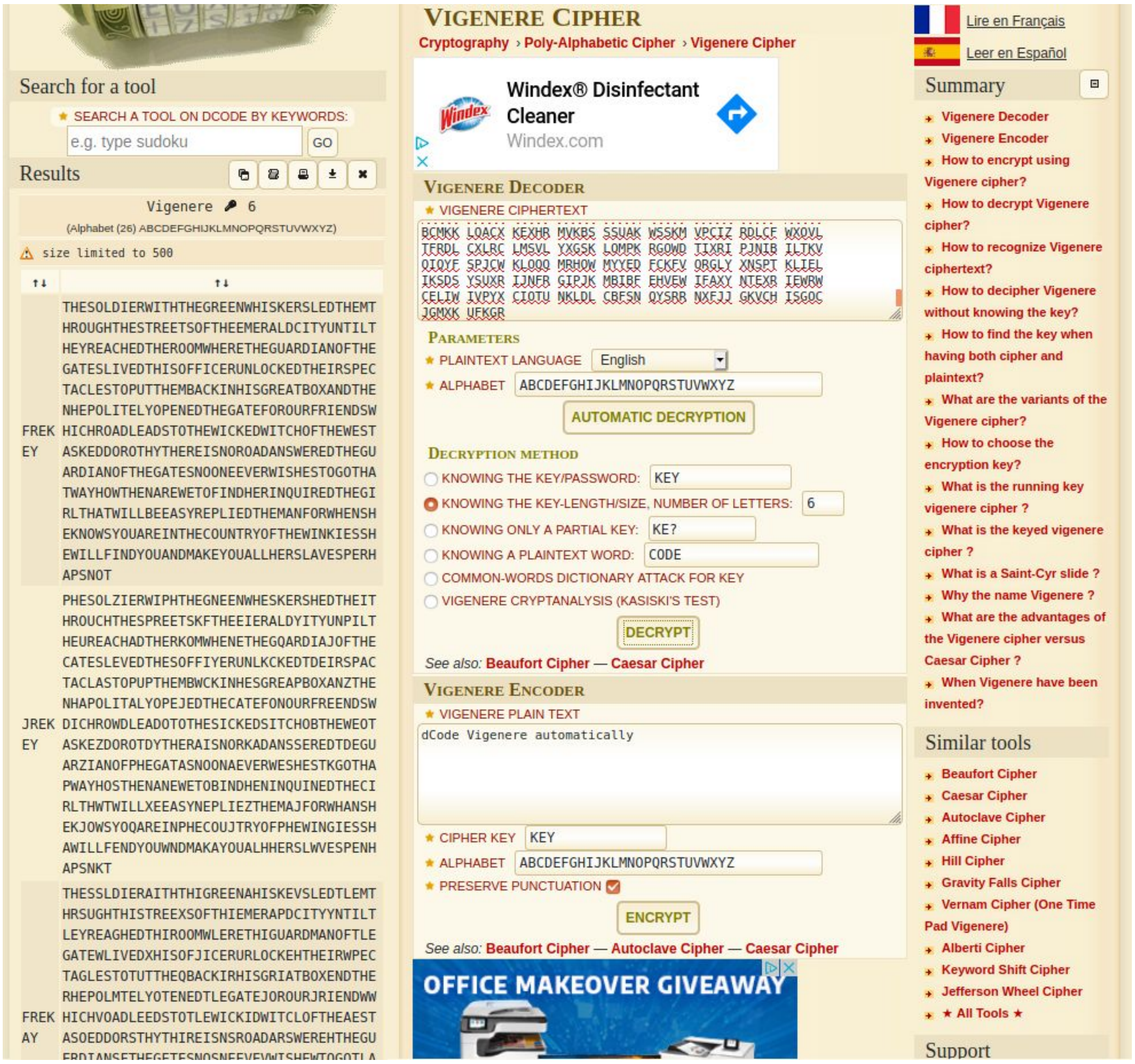
The level goal for the level has some of the following information:
"Another type of substitution cipher is referred to as 'polyalphabetic', where one character of P may map to many, or all, possible ciphertext characters.
An example of a polyalphabetic cipher is called a Vigenère Cipher. It works like 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 Vigenère 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."

For this level they show us a hint file, the password file, and two found files, so I decided to take a look and do some research into the Vingenère Cipher and I found a website which similar to rot13.com also decodes Vigenere Ciphers called https://www.dcode.fr/vigenere-cipher. This website was a fantastic find because I had found it after trying to decode the message and find the key by hand for about an hour. But anyway I copy pasted the text from the found files into the website:



Search for a tool

★ SEARCH A TOOL ON DCODE BY KEYWORDS:

e.g. type sudoku GO

Results

Vigenere 6

(Alphabet (26) ABCDEFGHIJKLMNOPQRSTUVWXYZ)

size limited to 500

```
FREK  THESOLDIERWITHTHEGREENWHISKERSLEDTHEMT
EY    HROUGHTHESTREETSOFTHEEMERALDCITYUNTILT
      HEYREACHEDTHEROOMWHERETHEGUARDIANOFTHE
      GATESLIVEDTHISOFFICERUNLOCKEDTHEIRSPEC
      TACLESTOPUTTHEMBACKINHISGREATBOXANDTHE
      NHEPOLITELYOPENEDTHEGATEFOROURFRIENDSW
      HICHROADLEADSTOTHEWICKEDWITCHOFTHEWEST
      ASKEDDOROTHYTHEREISNOROADANSWEREDTHEGU
      ARDIANOFTHEGATESNOONEEVERWISHESTOGOTHA
      TWAYHOWTHENAREWETOFINDHERINQUIREDTHEGI
      RLTHATWILLBEEASYREPLIEDTHEMANFORWHENSH
      EKNOWSYOUAREINTHECOUNTRYOFTHEWINKIESSH
      EWILLFINDYOUANDMAKEYOUALLHERSLAVESPERH
      APSNOT

JREK  PHESOLZIERWIPHTHEGNEENWHESKERSHEDTHEIT
EY    HROUCHTHESPREETSKFTHEEIERALDYITYUNPILT
      HEUREACHADTHERKOMWHENETHEGQARDIAJOFTHE
      CATESLEVEDTHESOFFIYERUNLKCKEDTDEIRSPAC
      TACLASTOPUPTHEMBWCKINHESGREAPBOXANZTHE
      NHAPOLITALYOPEJEDTHECATEFONOURFREENDSW
      DICHROWDLEADOTOTHESICKEDSITCHOBTHEWEOT
      ASKEZDOROTDYTHERAISNORKADANSSEREDTDEGU
      ARZIANOFPHEGATASNOONAEVERWESHESTKGOTHA
      PWAYHOSTHENANEWETOBINDHENINQUINEDTHECI
      RLTHWTWILLXEEASYNEPLIEZTHEMAJFORWHANSH
      EKJOWSYOQAREINPHECOUJTRYOFPHEWINGIESSH
      AWILLFENDYOUWNDMAKAYOUALHHERSLWVESPENH
      APSNKT

FREK  THESSLDIERAITHTHIGREENAHISKEVSLEDTLEMT
AY    HRSUGHTHISTREEXSOFTHIEMERAPDCITYYNTILT
      LEYREAGHEDTHIROOMWLERETHIGUARDMANOFTLE
      GATEWLIVEDXHISOFJICERURLOCKEHTHEIRWPEC
      TAGLESTOTUTTHEQBACKIRHISGRIATBOXENDTHE
      RHEPOLMTELYOTENEDTLEGATEJOROURJRIENDWW
      HICHVOADLEEDSTOTLEWICKIDWITCLOFTHEAEST
      ASOEDDORSTHYTHIREISNSROADARSWEREHTHEGU
```

VIGENERE CIPHER

Cryptography › Poly-Alphabetic Cipher › Vigenere Cipher

VIGENERE DECODER

★ VIGENERE CIPHERTEXT

```
BCMKK LQACX KEXHR MVKBS SSUAK WSSKM VPCIZ RDLCF WXQVL
TFRDL CXLRC LMSVL YXGSK LQMPK RGOWD TIXRI PJNIB ILTKV
QIQYF SPJCW KLOQQ MRHQW MYYED FCKFV QRGLY XNSPT KLIEL
IKSDS YSUXR IJNFR GIPJK MBIBF EHVEW IFAXY NTEXR IEWRW
CELIW IVPYX CIOTU NKLDL CBFSN QYSRR NXFJJ GKVCH ISGOC
JGMXK UFKGR
```

PARAMETERS

★ PLAINTEXT LANGUAGE: English

★ ALPHABET: ABCDEFGHIJKLMNOPQRSTUVWXYZ

AUTOMATIC DECRYPTION

DECRYPTION METHOD

○ KNOWING THE KEY/PASSWORD: KEY

● KNOWING THE KEY-LENGTH/SIZE, NUMBER OF LETTERS: 6

○ KNOWING ONLY A PARTIAL KEY: KE?

○ KNOWING A PLAINTEXT WORD: CODE

○ COMMON-WORDS DICTIONARY ATTACK FOR KEY

○ VIGENERE CRYPTANALYSIS (KASISKI'S TEST)

DECRYPT

See also: Beaufort Cipher — Caesar Cipher

VIGENERE ENCODER

★ VIGENERE PLAIN TEXT

dCode Vigenere automatically

★ CIPHER KEY: KEY

★ ALPHABET: ABCDEFGHIJKLMNOPQRSTUVWXYZ

★ PRESERVE PUNCTUATION ✓

ENCRYPT

See also: Beaufort Cipher — Autoclave Cipher — Caesar Cipher

Lire en Français

Leer en Español

Summary

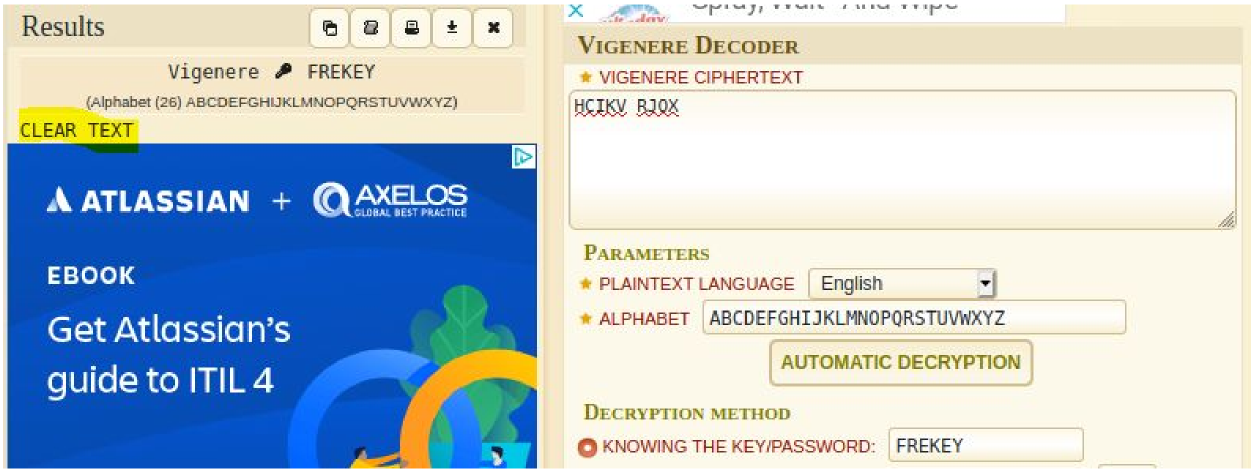
- Vigenere Decoder
- Vigenere Encoder
- How to encrypt using Vigenere cipher?
- How to decrypt Vigenere cipher?
- How to recognize Vigenere ciphertext?
- How to decipher Vigenere without knowing the key?
- How to find the key when having both cipher and plaintext?
- What are the variants of the Vigenere cipher?
- How to choose the encryption key?
- What is the running key vigenere cipher ?
- What is the keyed vigenere cipher ?
- What is a Saint-Cyr slide ?
- Why the name Vigenere ?
- What are the advantages of the Vigenere cipher versus Caesar Cipher ?
- When Vigenere have been invented?

Similar tools

- Beaufort Cipher
- Caesar Cipher
- Autoclave Cipher
- Affine Cipher
- Hill Cipher
- Gravity Falls Cipher
- Vernam Cipher (One Time Pad Vigenere)
- Alberti Cipher
- Keyword Shift Cipher
- Jefferson Wheel Cipher
- ★ All Tools ★

Support

It gives you multiple key possibilities, and shows you what the plain text would look like using that key. We can see that the first key: "FREKEY" gives us a plain text with words that actually make sense. So that is most likely the key. We can put our ciphertext in next, and select that we know the keypassword, and get our plaintext which reveals the password:


Results

Vigenere 🔑 FREKEY

(Alphabet (26) ABCDEFGHIJKLMNOPQRSTUVWXYZ)

CLEAR TEXT

VIGENERE DECODER

★ VIGENERE CIPHERTEXT

HCIKV RJOX

PARAMETERS

★ PLAINTEXT LANGUAGE English

★ ALPHABET ABCDEFGHIJKLMNOPQRSTUVWXYZ

AUTOMATIC DECRYPTION

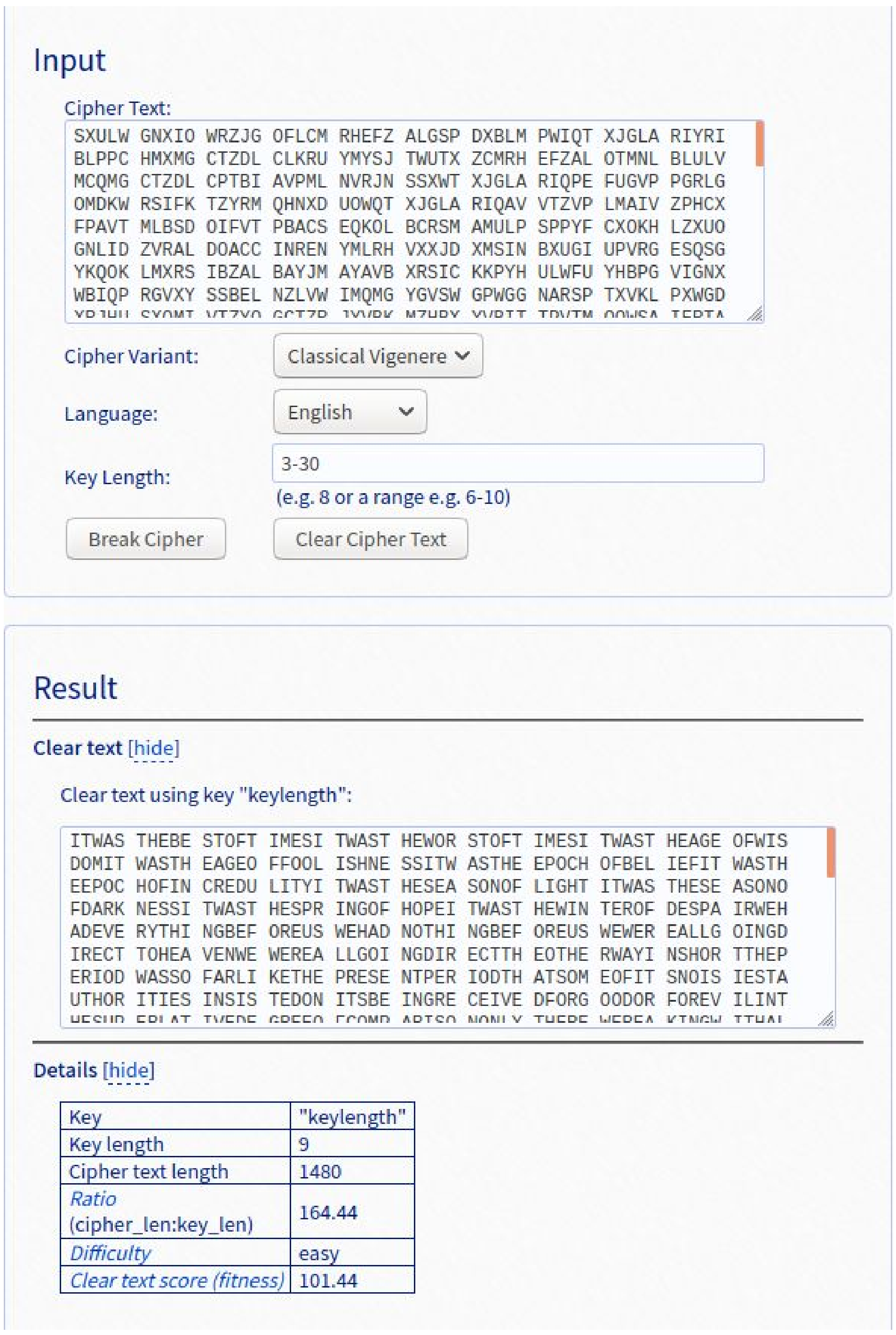
DECRYPTION METHOD

● KNOWING THE KEY/PASSWORD: FREKEY

Krypton 5 → 6:

The level goal states: "FA can break a known key length as well. Let's try one last polyalphabetic cipher, but this time the key length is unknown. Enjoy."

As I started this level I found this website: https://www.guballa.de/vigenere-solver which allows you to not only choose what type of Vigenere cipher, but allows you to choose a language and range of key lengths. After plugging the text from the found files into the input I just had the website solve the cipher for me, it even gave me the key length and key:



Input

Cipher Text:

```
SXULW GNXIO WRZJG OFLCM RHEFZ ALGSP DXBLM PWIQT XJGLA RIYRI
BLPPC HMXMG CTZDL CLKRU YMYSJ TWUTX ZCMRH EFZAL OTMNL BLULV
MCQMG CTZDL CPTBI AVPML NVRJN SSXWT XJGLA RIQPE FUGVP PGRLG
OMDKW RSIFK TZYRM QHNXD UOWQT XJGLA RIQAV VTZVP LMAIV ZPHCX
FPAVT MLBSD OIFVT PBACS EQKOL BCRSM AMULP SPPYF CXOKH LZXUO
GNLID ZVRAL DOACC INREN YMLRH VXXJD XMSIN BXUGI UPVRG ESQSG
YKQOK LMXRS IBZAL BAYJM AYAVB XRSIC KKPYH ULWFU YHBPG VIGNX
WBIQP RGVXY SSBEL NZLVW IMQMG YGVSW GPWGG NARSP TXVKL PXWGD
```

Cipher Variant: Classical Vigenere

Language: English

Key Length: 3-30

(e.g. 8 or a range e.g. 6-10)

Break Cipher | Clear Cipher Text

Result

Clear text [hide]

Clear text using key "keylength":

```
ITWAS THEBE STOFT IMESI TWAST HEWOR STOFT IMESI TWAST HEAGE OFWIS
DOMIT WASTH EAGEO FFOOL ISHNE SSITW ASTHE EPOCH OFBEL IEFIT WASTH
EEPOC HOFIN CREDU LITYI TWAST HESEA SONOF LIGHT ITWAS THESE ASONO
FDARK NESSI TWAST HESPR INGOF HOPEI TWAST HEWIN TEROF DESPA IRWEH
ADEVE RYTHI NGBEF OREUS WEHAD NOTHI NGBEF OREUS WEWER EALLG OINGD
IRECT TOHEA VENWE WEREA LLGOI NGDIR ECTTH EOTHE RWAYI NSHOR TTHEP
ERIOD WASSO FARLI KETHE PRESE NTPER IODTH ATSOM EOFIT SNOIS IESTA
UTHOR ITIES INSIS TEDON ITSBE INGRE CEIVE DFORG OODOR FOREV ILINT
```

Details [hide]

Key	"keylength"
Key length	9
Cipher text length	1480
Ratio (cipher_len:key_len)	164.44
Difficulty	easy
Clear text score (fitness)	101.44

Then we just use the key given to decrypt the password from the krypton6 file.

<u>Krypton 6 → 7:</u>

I will not paste the level goal here due to the fact that it is huge, so please read it on the website. To summarize the hint it gives us a lot of information about the types of ciphers and encryption methods we have seen. You should most definitely read the passage, but I will paste the last part which is the most important piece.

"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.

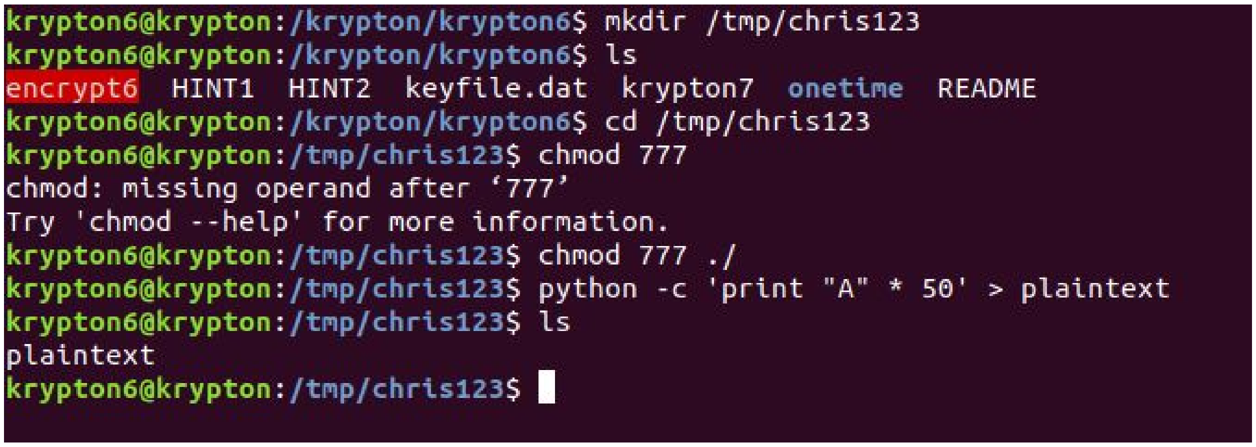
As stated, it is now that we suggest you begin to use public tools, like cryptool, to help in your analysis. You will most likely need a hint to get going. See 'HINT1' if you need a kickstart.

If you have further difficulty, there is a hint in 'HINT2'.

The password for level 7 (krypton7) is encrypted with 'encrypt6'.

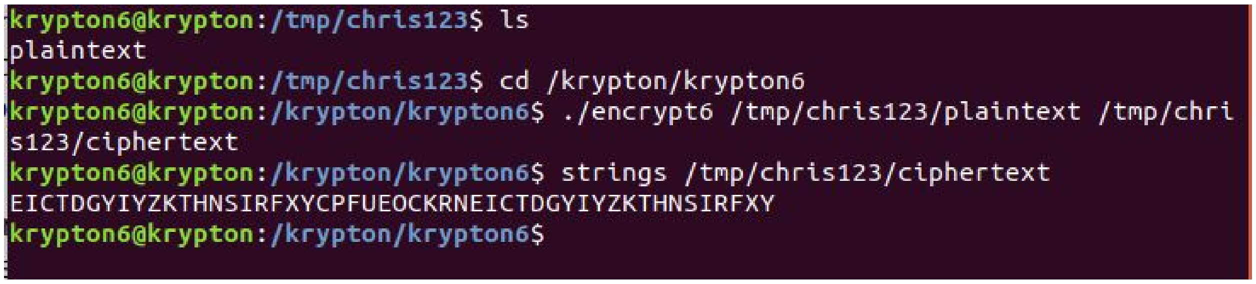
Good Luck!"

The part that really sticks out to me about the level goal is the part where they say the random number generator is weak. So let's just try out the encryption on a long string of the same letter. I will make a temporary directory where I can work, then I can just use a simple python script to make a long string of 'A' s:



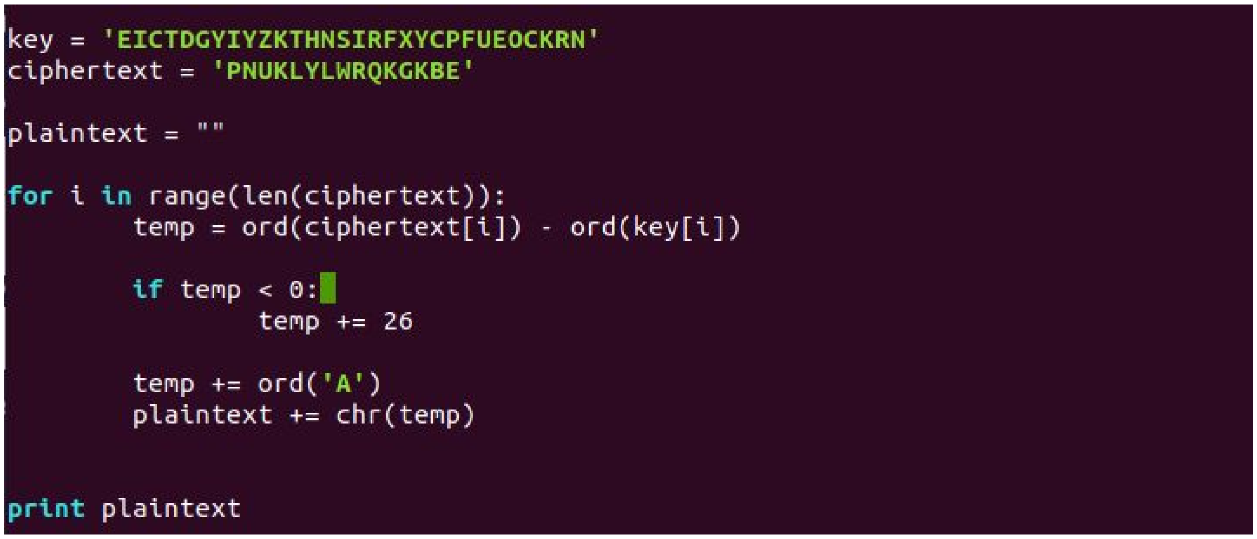
```
krypton6@krypton:/krypton/krypton6$ mkdir /tmp/chris123
krypton6@krypton:/krypton/krypton6$ ls
encrypt6  HINT1  HINT2  keyfile.dat  krypton7  onetime  README
krypton6@krypton:/krypton/krypton6$ cd /tmp/chris123
krypton6@krypton:/tmp/chris123$ chmod 777
chmod: missing operand after '777'
Try 'chmod --help' for more information.
krypton6@krypton:/tmp/chris123$ chmod 777 ./
krypton6@krypton:/tmp/chris123$ python -c 'print "A" * 50' > plaintext
krypton6@krypton:/tmp/chris123$ ls
plaintext
krypton6@krypton:/tmp/chris123$ 
```

Now lets run the encryption on this string of A's:



```
krypton6@krypton:/tmp/chris123$ ls
plaintext
krypton6@krypton:/tmp/chris123$ cd /krypton/krypton6
krypton6@krypton:/krypton/krypton6$ ./encrypt6 /tmp/chris123/plaintext /tmp/chri
s123/ciphertext
krypton6@krypton:/krypton/krypton6$ strings /tmp/chris123/ciphertext
EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXY
krypton6@krypton:/krypton/krypton6$
```

We can see that after about 30 letters, the key starts to repeat "EICTDGYIYZKTHNSIRFXYCPFUEOCKRN" is the repeated number generation. Therefore, we can determine that the key given the string of A's and this repeated string. Subtracting the cipher text from the plain text will give us the key. So we can just do a subtraction with the key pattern we got from doing the A's, and do a subtraction with the cipher text to reveal the plain text as shown below:



```
key = 'EICTDGYIYZKTHNSIRFXYCPFUEOCKRN'
ciphertext = 'PNUKLYLWRQKGKBE'

plaintext = ""

for i in range(len(ciphertext)):
        temp = ord(ciphertext[i]) - ord(key[i])

        if temp < 0:
                temp += 26

        temp += ord('A')
        plaintext += chr(temp)


print plaintext
```

Running my python script revealed the password to level 7!

The End:

And with that we have finished the war game Krypton!