OverTheWire Krypton Walkthrough

Introduction

This is a comprehensive walkthrough of the Krypton wargame offered by the OverTheWire platform. Krypton is a series of cybersecurity challenges that test the player's knowledge and understanding of cryptography. My walkthrough of itt covers:

- The research I undertook to understand the concepts covered by each level,
- And the methodology I utilized to finally solve the problem.

Connection

Each Krypton levels can be accessed via SSH, specifically the command:
'ssh kryptonX@krypton.labs.overthewire.org -p 2231', where X is the level you are attempting to access.

Code Repository

All of the code I've written to complete the challenges can be found in this GitHub repository. Most of the programs are quite simple, but if you'd be interested in reading about how they work, I've written comprehensive explanations for each program In the README file.

Level 0 - Level 1

Link: https://overthewire.org/wargames/krypton/krypton0.html

Level Info

Welcome to Krypton! The first level is easy. The following string encodes the password using Base64:

S1JZUFRPTklTR1JFQVQ=

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/

Research

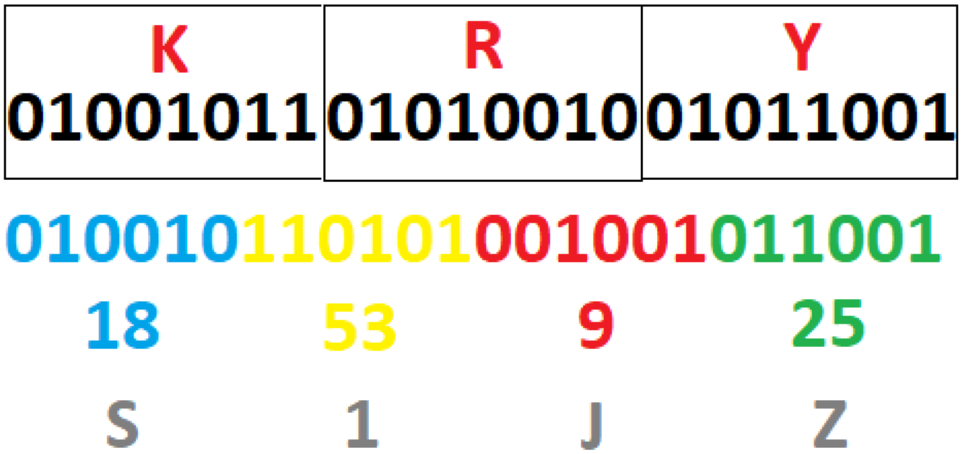
Base64 is known as a "binary-to-text" method of encryption. It represents binary data in blocks of 24 bits, split into four 6-bit long Base64 characters. The available Base64 characters are shown in the following table:

Table 1: The Base 64 Alphabet

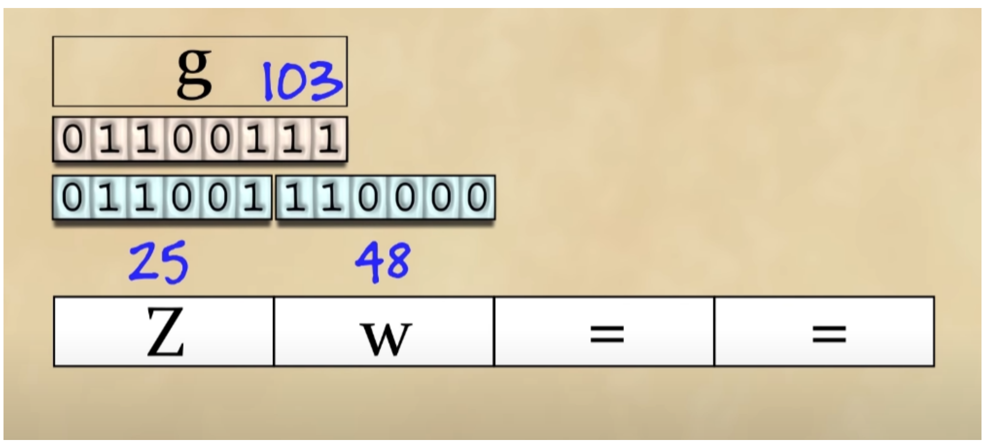
Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Sourced from RFC 4648

The way that Base64 works is best shown using an example. Therefore, let's try encoding the characters KRY from Krypton.



In the above diagram, the characters K, R, and Y are converted into the 8-bit binary values in black. This can be extracted from an ASCII table. Afterwards, we line up these bits next to each other and split them up into the four color-coded 6-bit binary blocks. The binary values can then be converted to decimals. Using the Base64 table above, we can then reach the conclusion that the string ‘KRY’ is encrypted into ‘S1JZ. A string will not always be long enough to be perfectly divided into 6-bit sequences. Hence, Base64 encryption features the use of padding characters, namely the “=” character. Examine the following example by Youtuber ‘schenken’ in his video covering Base64:



The first 6 bits of ‘g’ are encoded into the letter ‘Z’. The next 2 bits are padded with 0’s to form the 6-bit sequence ‘110000’, which is converted into ‘w’. As there is nothing left to encode, Base64 pads the rest of the string with ‘=’ characters. Hence, the letter ‘g’ on its own is encoded as ‘Zw==’ using Base64

Method

Any Base64 decoder can be used to solve this problem. I used the Bash shell’s built-in Base64 command like so:

```
ditto@Roblox-Hacking-Station:/mnt/c/Users/Kevin Ngo$ base64 -d -
S1JZUFRPTklTR1JFQVQ=
KRYPTONISGREATditto@Roblox-Hacking-Station:/mnt/c/Users/Kevin Ngo$
```

The password for the next level is revealed to be **KRYPTONISGREAT**. I didn’t actually know about the Bash shell’s Base64 command, so I simply looked it up on Google and accessed the documentation.

Level 1 - Level 2

Link: https://overthewire.org/wargames/krypton/krypton1.html

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!

Preamble

Before starting the level ,we must connect to it using the password we obtained in the previous level and the connection instructions at the start of the document. Afterwards we'll be connected to the Krypton virtual machine, which is running a Linux distro. The level info above suggests that the password is stored in a file called "krypton2", and so we must navigate to it like so:

```
krypton1@bandit:~$ ls
krypton1@bandit:~$ cd /krypton
krypton1@bandit:/krypton$ ls
krypton1  krypton2  krypton3  krypton4  krypton5  krypton6
krypton1@bandit:/krypton$ cd krypton1
krypton1@bandit:/krypton/krypton1$ ls
krypton2  README
krypton1@bandit:/krypton/krypton1$
```

Examining the README file provides us with further information about the level, specifically that the password is encrypted using **ROT13:**

```
krypton1@bandit:/krypton/krypton1$ cat README
Welcome to Krypton!

This game is intended to give hands on experience with cryptography
and cryptanalysis.  The levels progress from classic ciphers, to modern,
easy to harder.

Although there are excellent public tools, like cryptool,to perform
the simple analysis, we strongly encourage you to try and do these
without them for now.  We will use them in later excercises.

** Please try these levels without cryptool first **


The first level is easy.  The password for level 2 is in the file
'krypton2'.  It is 'encrypted' using a simple rotation called ROT13.
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!
krypton1@bandit:/krypton/krypton1$
```

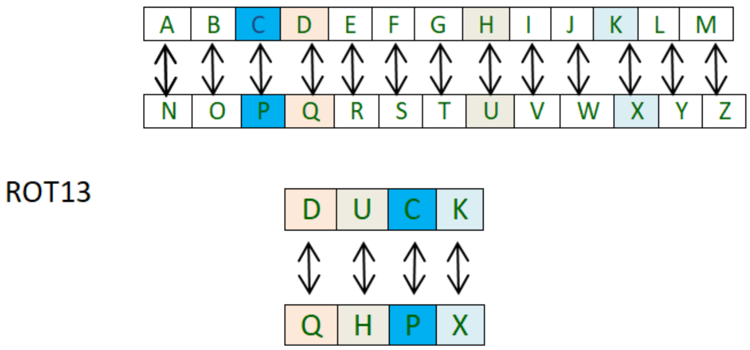
Now, having a look at the krypton2 file gives us the encrypted password: **YRIRY GJB CNFFJBEQ EBGGRA.**

```
krypton1@bandit:/krypton/krypton1$ cat krypton2
YRIRY GJB CNFFJBEQ EBGGRA
krypton1@bandit:/krypton/krypton1$
```

From this initial reconnaissance, we've gathered that we need to understand how ROT13 works, and how to decrypt it to obtain the password. Therefore, I've conducted some research on it and have documented my findings below:

Research

ROT13 is a substitution cipher where each plaintext letter is replaced with the letter that is 13 places after it in the alphabet. For example, the letter ‘A’ becomes ‘N’ because ‘A’ is the 1st letter in the alphabet, whereas ‘N’ is the 14th letter. Since 13 is half of the number of letters in the alphabet, the inverse would also be true. Encrypting ‘N’ using ROT13 would result in it being substituted for ‘A’ (14th letter + 13 = 27 = 26 + 1st letter in the alphabet).



Sourced from https://www.geeksforgeeks.org/rot13-cipher/

From this, it should be clear that decrypting ROT13 is extremely easy; we simply shift every letter forward by 13 places.

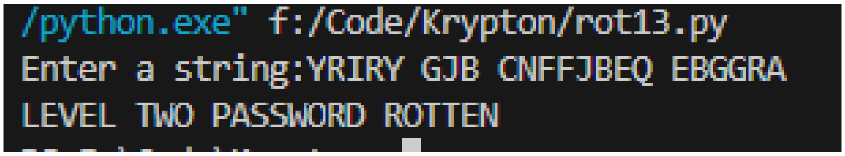
Method

There are a plethora of websites out there that will easily decipher a ROT13 cipher, such as this [one](). However, to challenge myself and improve my programming skills, I decided to write a [Python program]() to accomplish this:

```python
#!/usr/bin/env python3

lower = {
  "a" : 1, "b" : 2, "c" : 3, "d" : 4, "e" : 5, "f" : 6, "g" : 7, "h":8,
  "i" : 9, "j" : 10, "k" : 11, "l" : 12, "m" : 13, "n" : 14, "o" : 15,
  "p" : 16, "q" : 17, "r" : 18, "s" : 19, "t" : 20, "u" : 21, "v" : 22,
  "w" : 23, "x" : 24, "y" : 25, "z" : 26
}

l_key_list = list(lower.keys())

upper = {
  "A" : 1, "B" : 2, "C" : 3, "D" : 4, "E" : 5, "F" : 6, "G" : 7, "H":8,
  "I" : 9, "J" : 10, "K" : 11, "L" : 12, "M" : 13, "N" : 14, "O" : 15,
  "P" : 16, "Q" : 17, "R" : 18, "S" : 19, "T" : 20, "U" : 21, "V" : 22,
  "W" : 23, "X" : 24, "Y" : 25, "Z" : 26
}

u_key_list = list(upper.keys())

SHIFT = 13
LETTERS = 26

message = input("Enter a string:")
decoded = ''


for character in message:
  # calculate (charpos + 13) mod 26
  if character in lower:
    new_pos = (lower[character] + SHIFT) % LETTERS
    decoded = decoded + l_key_list[new_pos - 1]
  elif character in upper:
    new_pos = (upper[character] + SHIFT) % LETTERS
    decoded = decoded + u_key_list[new_pos - 1]
  else:
    decoded = decoded + character

print(decoded)
```

The above code works using two dictionaries, one to store lower case letters along with their indexes, and the other to store upper case letters. The user inputs a message via STDIN, then the program loops through each character in the string. Each character is shifted forward by 13 characters using the operation: $(characterIndex\ +\ 13)\ mod\ 26$. The shifted character is concatenated to the "decoded" string and printed after all characters have been looped over. Due to the limitations of ROT13, any character that isn't in the alphabet is not shifted.

Now, let's input the cipher **YRIRY GJB CNFFJBEQ EBGGRA** into the program:



Hence, the password for this level is **ROTTEN.**

Level 2 - Level 3

Link: https://overthewire.org/wargames/krypton/krypton2.html

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.

<u>Additional Information:</u>
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.

<u>Here is an example:</u>
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

Research

Our earlier research on ROT13 applies here, since it's actually a type of Caesar Cipher. The level info also provides sufficient information on the cipher. However, unlike ROT13, unless the shift is 13 letters, we cannot decrypt the cipher by simply running the encryption algorithm again on the ciphertext. Instead, we must perform a **left shift** to reverse the encryption's right shift.. Recall the ROT13 program I wrote earlier, where shifting a letter was performed with the following calculation:

$$.Encrypted = (characterIndex + N)\ mod\ 26,\ where\ N = shift$$

A left shift is performed with a similar algorithm, simply changing the addition to a subtraction.

$$.Decrypted = (characterIndex - N)\ mod\ 26,\ where\ N = shift$$

Method

We are provided a program that will encrypt anything with the same key used to encrypt the password. In terms of a Caesar Cipher, the key would simply refer to the number of positions each letter is shifted. Using this, we can find the key:

1. Follow the example provided in the level info to create a temporary working directory.
2. Create a file that simply contains the letter 'A':

```
krypton2@bandit:/tmp/tmp.xDNBQOSI0c$ echo 'A' > a.txt
krypton2@bandit:/tmp/tmp.xDNBQOSI0c$ ls
a.txt  keyfile.dat
krypton2@bandit:/tmp/tmp.xDNBQOSI0c$ cat a.txt
A
krypton2@bandit:/tmp/tmp.xDNBQOSI0c$
```

3. Encrypt a.txt and open the produced ciphertext:

```
krypton2@bandit:/tmp/tmp.xDNBQOSI0c$ /krypton/krypton2/encrypt a.txt
krypton2@bandit:/tmp/tmp.xDNBQOSI0c$ ls
a.txt  ciphertext  keyfile.dat
krypton2@bandit:/tmp/tmp.xDNBQOSI0c$ cat ciphertext
Mkrypton2@bandit:/tmp/tmp.xDNBQOSI0c$
```

4. Encrypting 'A" shifts it to 'M'. **Therefore, we can deduce that the Caesar Cipher shifts letters by 12 positions.**

I've written a simple program to decipher Caesar Ciphers, it's essentially the same as the ROT13 program with modified shift operations and allows for user-inputted shift amounts:

```python
#!/usr/bin/env python3

lower = {
  "a" : 1, "b" : 2, "c" : 3, "d" : 4, "e" : 5, "f" : 6, "g" : 7, "h":8,
  "i" : 9, "j" : 10, "k" : 11, "l" : 12, "m" : 13, "n" : 14, "o" : 15,
  "p" : 16, "q" : 17, "r" : 18, "s" : 19, "t" : 20, "u" : 21, "v" : 22,
  "w" : 23, "x" : 24, "y" : 25, "z" : 26
}

l_key_list = list(lower.keys())

upper = {
  "A" : 1, "B" : 2, "C" : 3, "D" : 4, "E" : 5, "F" : 6, "G" : 7, "H":8,
  "I" : 9, "J" : 10, "K" : 11, "L" : 12, "M" : 13, "N" : 14, "O" : 15,
  "P" : 16, "Q" : 17, "R" : 18, "S" : 19, "T" : 20, "U" : 21, "V" : 22,
  "W" : 23, "X" : 24, "Y" : 25, "Z" : 26
}

u_key_list = list(upper.keys())

LETTERS = 26

message = input("Enter the ciphertext: ")
shift = int(input("Enter the shift amount: "))
decoded = ''


for character in message:
  # calculate (charpos - 13) mod 26
  if character in lower:
    new_pos = (lower[character] - shift) % LETTERS
    decoded = decoded + l_key_list[new_pos - 1]
  elif character in upper:
    new_pos = (upper[character] - shift) % LETTERS
    decoded = decoded + u_key_list[new_pos - 1]
  else:
    decoded = decoded + character

print(decoded)
```

Now, feeding it the ciphertext **OMQEMDUEQMEK,** and the shift amount of 12, we get the password **'CAESARISEASY'**.

```
Enter the ciphertext: OMQEMDUEQMEK
Enter the shift amount: 12
CAESARISEASY
```

Now that we've completed the level using this more long winded route, I wanted to point out that it's also extremely easy to complete via simple brute forcing. In the English language, there are only 25 possible shifts for the Caesar Cipher (since there are 26 letters in the alphabet). Therefore, you could ignore the process of using the encryption program and just try all shifts until it works. This is made easy using readily accessible websites such as Cryptii.

Level 3 - Level 4

Link: https://overthewire.org/wargames/krypton/krypton3.html

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.

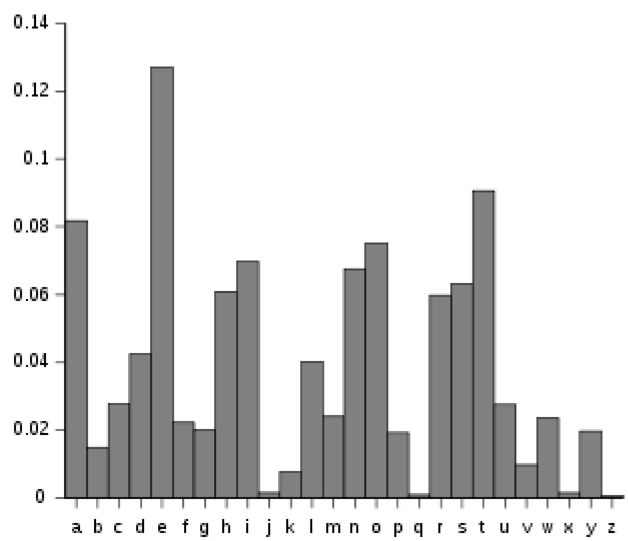
Preamble

This challenge is slightly more difficult than the previous ones because the cipher mechanism is not provided to us. Therefore to complete it we must learn about and apply cryptanalysis techniques, namely **Frequency Analysis** along with trial and error.

Research

According to the [OWASP Foundation](), "Cryptanalysis is a process of finding weaknesses in cryptographic algorithms and using these weaknesses to decipher the ciphertext without knowing the secret key (instance deduction)". We will be learning how to conduct some basic cryptanalysis to determine the cipher mechanism.

A common cryptanalysis technique is "frequency analysis". It involves counting the frequencies of letters in the ciphertext, and comparing them to the expected frequencies of letters in the corresponding alphabet. All written languages have certain patterns; for example, 'E' is the most common letter in the English alphabet. Below is a graph depicting a typical distribution of letters in written English:



Source: https://en.wikipedia.org/wiki/Frequency_analysis

It's also possible to do frequency analysis on pairs of letters (bigrams), triplets (trigrams) and other N-letter combinations (N-grams).

Using this information, it is possible to make certain deductions about the cipher. For example, if the frequency of letters were consistently shifted then it could be concluded that a Caesar Cipher was used as well as the number of shifts. For example, suppose that the frequency analysis of a ciphertext showed that 'H' was the most common letter, 'W' the second most common, 'D' the third most common etc. Therefore, we could conclude that a Caesar Cipher with a 2 letter right shift was used.

Method

Firstly, we'll perform a frequency analysis on the ciphertext to obtain information about its language patterns. Using this information, we'll brute force the solution.

I wrote a simple [Python program](#) to perform single letter frequency analysis. It prints out each character sorted in order of the number of times they appear in the ciphertext.

```python
#!/usr/bin/env python3

frequencies = {}

ciphertext = input("Enter the Ciphertext: ")
ciphertext = ciphertext.replace(" ", "")
ciphertext = ciphertext.upper()

for character in ciphertext:
    if character.isalpha():
        frequencies[character] = frequencies.get(character, 0) + 1

for character, frequency in sorted(frequencies.items(), key = lambda x:x[1], reverse = True):
    print(f"{character}: {frequency} ")
```

Examining the krypton4 file, we see that the ciphertext is **KSVVW BGSJD SVSIS VXBMN YQUUK BNWCU ANMJS.** Furthermore, there are a number of other files that also contain 5-letter group ciphers, encrypted using the same key.

Inserting the ciphertext into the cryptanalysis program, we obtain the following result:

```
Enter the Ciphertext: KSVVW BGSJD SVSIS VXBMN YQUUK BNWCU ANMJS
S: 6
V: 4
B: 3
N: 3
U: 3
K: 2
W: 2
J: 2
M: 2
G: 1
D: 1
I: 1
```

The letter 's' appears most commonly. However the frequency of the other letters are too close to each other to make any conclusions. Therefore, I also analysed the found files. Their results of the first two are pasted below:

found1:

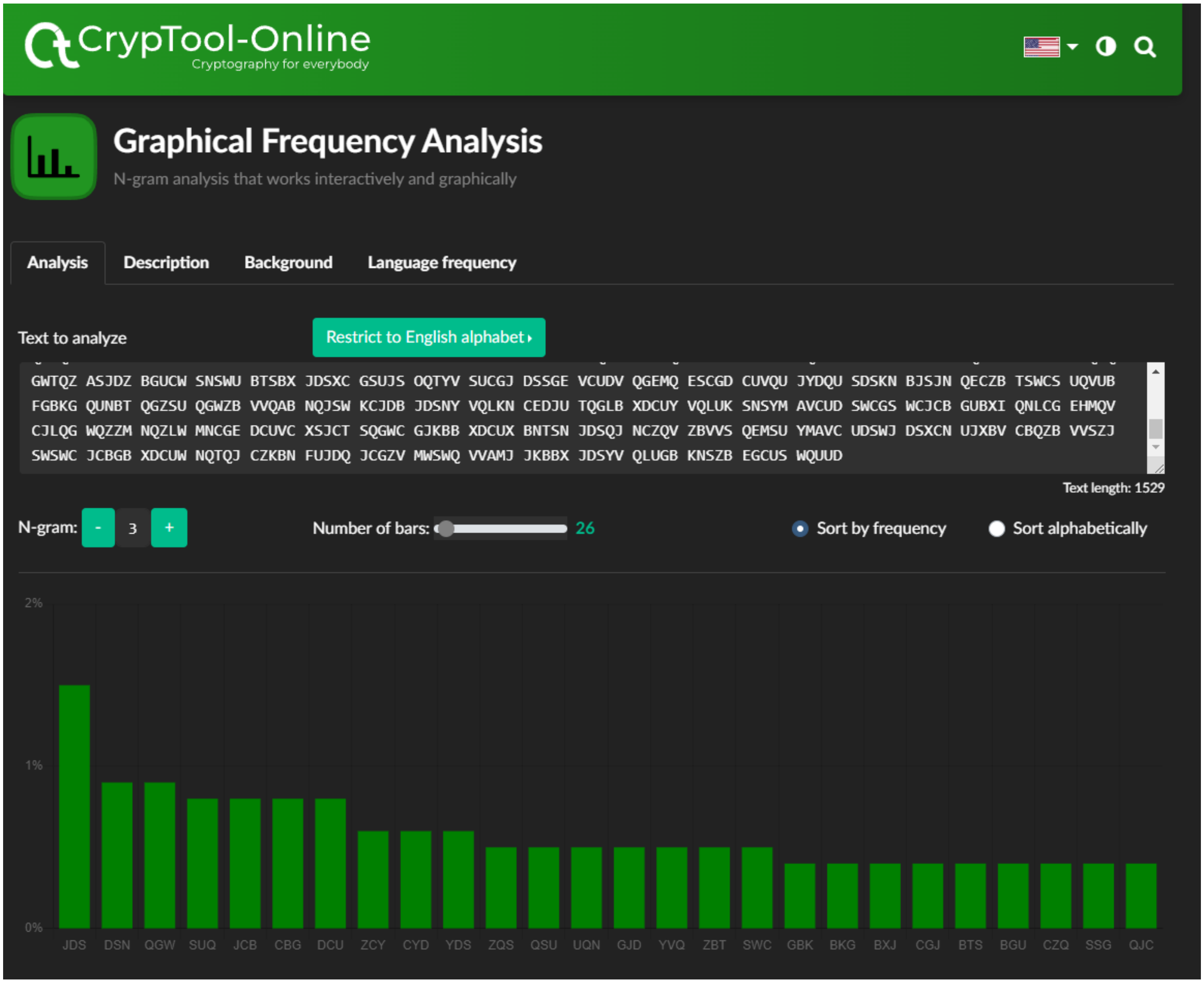
```
Enter the Ciphertext: CGZNL YJBEN QYDLQ ZQSUQ NZCYD SNQVU BFGBK GQUQZ QSUQN UZCYD SNJDS UDCXJ ZCYDS NZQSU QNUZB WSBNZ QSUQN
JCBGZ CYDSN CGKDC ZDSQZ DVSJJ SNCGJ DSYVQ CGJSO JCUNS YVQZS WALQV SJJSN UBTSX COSWG MTASN BXYBU CJCBG UWBKG JDSQV YDQAS JXBN
M VCMUZ QSUQN KDBMU SWCJJ BZBTT MGCZQ JSKCJ DDCUE SGSNQ VUJDS SGZNL YJCBG UJSYY SNXBN TSWAL QZQSU QNZCY DSNCU BXJSG CGZBN YB
GCG JDSNB JULUJ STQUK CJDQV VUCGE VSQVY DQASJ UMAUJ CJMJC BGZCY DSNUJ DSZQS UQNZC YDSNC USQUC VLANB FSGQG WCGYN QZJCZ SBXXS
QGWDC USQNV LYVQL UKSNS TQCGV LZBTS WCSUQ GWDCU JBNCS UESGN SUDSN QCUSW JBJDS YSQFB XUBYD CUJCZ QJCBG QGWQN JCUJN LALJD SSGW
Q GWTQZ ASJDZ BGUCW SNSWU BTSBX JDSXC GSUJS OQTYV SUCGJ DSSGE VCUDV QGEMQ ESCGD CUVQU JYDQU SDSKN BJSJN QECZB TSWCS UQVUB FG
JCB GUBXI QNLCG EHMQV CJLQG WQZZM NQZLW MNCGE DCUVC XSJCT SQGWC GJKBB XDCUX BNTSN JDSQJ NCZQV ZBVVS QEMSU YMAVC UDSWJ DSXCN
EGCUS WQUUD QFSUY SQNSU
S: 155
C: 107
Q: 106
J: 102
U: 100
B: 87
G: 81
N: 74
D: 69
Z: 57
V: 56
W: 47
Y: 42
T: 32
X: 29
```

found2:

```
Enter the Ciphertext: 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 NQAM
J JDSZM NNCZM VMTKQ UWCZJ QJSWA LVQKJ DNBME DBMJS GEVQG WQGWJ DSUZD BBVKB MVWDQ ISYNB ICWSW QGCGJ SGUCI SSWMZ QJCBG CGVQJ CGENQ TTQNQ GWJDS ZVQUU CZ
UQJ JDSQE SBXUD QFSUY SQNST QNNCS WJDSL SQNBV WQGGS DQJDQ KQLJD SZBGU CUJBN LZBMN 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 JBJDS TQNNC QESJD SZBMY VSTQL DQISQ NNQGE SWJDS ZSNS
T BGLCG UBTSD QUJSU CGZSJ DSKBN ZSUJS NZDQG ZSVVB NQVVB KSWJD STQNN CQESA QGGUJ BASNS QWBGZ SCGUJ SQWBX JDSMU MQVJD NSSJC TSUQG GSUYN SEGQG ZLZBM VW
DQI 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 SACN
J DBXJD SJKCG UJDSN SQNSX SKDCU JBNCZ QVJNQ ZSUBX UDQFS UYSQN SMGJC VDSCU TSGJC BGSWQ UYQNJ BXJDS VBGWB GJDSQ JNSUZ SGSCG ASZQM USBXJ DCUEQ YUZDB VQ
NUN 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 GXBN
W SSNYB QZDCG EQGBJ DSNSC EDJSS GJDZS GJMNL UJBNL DQUUD QFSUY SQNSU JQNJC GEDCU JDSQJ NCZQV ZQNSS NTCGW CGEJD SDBNU SUBXJ DSQJN SYQJN BGUCG VBGWB GR
BDG 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 ICWS
G ZSUMA UJQGJ CQJSU UMZDU JBNCS UBJDS NJDQG DSQNU QLZBV VSZJS WQXJS NDCUW SQJD
S: 243
Q: 186
J: 158
N: 135
U: 130
B: 129
D: 119
G: 111
C: 86
W: 66
Z: 59
V: 53
M: 45
T: 37
E: 34
X: 33
Y: 33
K: 30
L: 27
```

The results consistently show that ‘S’ is the most common letter in the ciphers. Therefore, it is likely that the letter **‘E’ has been substituted for ‘S’**. However the results for the other characters are still inconsistent. The encrypted password’s second-most common letter is ‘V’, while the ‘found’ files’ most common letters are either C or Q. These inconsistencies rule out the possibility of the cipher being a simple shift substitution cipher (such as a Caesar cipher). This is because we’d expect to see essentially the same frequency distribution as the usual English alphabet but shifted to different letters.

Having stumbled upon a dead end, I decided to look towards the trigram analysis of the found files. Since my program isn’t capable of doing the analysis, I used the CrypTool website. Via this, we see that the most common trigram is JDS:



Now, the most common trigram in the English language is ‘THE’. Therefore, we can deduce that **‘THE’ is mapped to ‘JDS’.**

Enough information has been obtained such that we can now brute force the solution. I’ll highlight a few of my steps below:

- ☐ Use the ‘tr’ command to undo the ‘THE’ - > ‘JDS’ mapping:

```
ditto@Roblox-Hacking-Station:~$ echo 'KSVVW BGSJD SVSIS VXBMN YQUUK BNWCU ANMJS' | tr 'JDS' 'THE'
KEVVW BGETH EVEIE VXBMN YQUUK BNWCU ANMTE
```

- ☐ Seeing as ‘THE’ is a word, let’s try rearranging the ciphertext a bit: ‘KEVVW BGE **THE** VEIE VXBMN YQUUK BNWCU ANMTE’
- ☐ Consider KEVV, if we assume it is a 4 letter word with its 2nd letter being E, then it’s possible that ‘L’ was mapped to ‘V’. Undoing this, the ciphertext is now: ‘**KELL** W BGE **THE** LEIE LXBMN YQUUK BNWCU ANMTE’.
- ☐ Substitute ‘K’ for ‘W’ since it looks like the first word could be ‘well’: ‘**WELL** WBGE **THE** LEIE LXBMN YQUUW BNWCU ANMTE’.
- ☐ Continue this process, and we finally reach the conclusion that the plaintext is **‘WELL DONE THE LEVEL FOUR PASSWORD IS BRUTE’.**

Level 4 - Level 5

Link: https://overthewire.org/wargames/krypton/krypton4.html

Level Info

Good job!

You more than likely used some form of FA 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 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 (American English). 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!

Research

Vignere ciphers are a type of polyalphabetic substitution ciphers, where each alphabetic character in the plaintext is encrypted using its position and the position of the corresponding letter in the key. This is perfectly demonstrated in the level info, where the plaintext ‘PROCEED MEETING AS AGREED’, is encrypted using the key ‘GOLD’.

When counting the position of letters in the context of Vignere Ciphers, it is important to note that A is the 0th letter, not the 1st.

Method

Examining the krypton5 file, we see that the password's ciphertext is **HCIKV RJOX.** The two intercepted messages are:

```
krypton4@bandit:/krypton/krypton4$ cat found1
YYICS JIZIB AGYYX RIEWV IXAFN JOOVQ QVHDL CRKLB SSLYX RIQYI IOXQT WXRIC RVVKP BHZXI YLYZP DLCDI IKGFJ UXRIP TFQGL CWVXR
IEZRV NMYSF JDLCL RXOWJ NMINX FNJSP JGHVV ERJTT OOHRM VMBWN JTXKG JJJXY TSYKL OQZFT OSRFN JKBIY YSSHE LIKLO RFJGS VMRJC
CYTCS VHDLC LRXOJ MWFYB JPNVR NWUMZ GRVMF UPOEB XKSDL CBZGU IBBZX MLMKK LOACX KECOC IUSBS RMPXR IPJZW XSPTR HKRQB VVOHR
MVKEE PIZEX SDYYI QERJJ RYSLJ VZOVU NJLOW RTXSD LYYNE ILMBK LORYW VAOXM KZRNL CWZRA YGWVH DLCLZ VVXFF KASPJ GVIKW WWVTV
MCIKL OQYSW SBAFJ EWRII SFACC MZRVO MLYYI MSSSK VISDY YIGML PZICW FJNMV PDNEH ISSFE HWEIJ PSEEJ QYIBW JFMIC TCWYE 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 BRROO WJFMI CSDDF YKWQM VLKWM KKLOV CXKFE XRFBI MEPJW SBWFJ ZWGMA PVHKR BKZIB
GCFEH WEWSF XKPJT NCYYR TUICX PTPLO VIJVT 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 IPJNM DLBFZ WSIQF EHLYR
YVVMY NXUSB SRMPW DMJQN SBIRM VTBIR YPWSP IIIIC WQMVL KHNZK SXMLY YIZEJ FTILY RSFAD SFJIW EVNWZ WOWFJ WSERB NKAKW LTCSX
KCWXV OILGL XZYPJ NLSXC YYIBM ZGFRK VMZEH DSRTJ ROGIM RHKPQ TCSCX GYJKB ICSTS VSPFE HGEQF JARMR JRWNS PTKLI WBWVW CXFJV
QOVYQ UGSXW BRWCS MSCIP XDFIF OLGSU ECXFJ PENZY STINX FJXVY YLISI MEKJI SEKFJ IEXHF NCPSI PKFVD LCWVA OVCSF JKVKX ESBLM
ZJICM LYYMC GMZEX BCMKK LOACX KEXHR MVKBS SSUAK WSSKM VPCIZ RDLCF WXOVL TFRDL CXLRC LMSVL YXGSK LOMPK RGOWD TIXRI PJNIB
ILTKV OIQYF SPJCW KLOQQ MRHOW MYYED FCKFV ORGLY XNSPT KLIEL IKSDS YSUXR IJNFR GIPJK MBIBF EHVEW IFAXY NTEXR IEWRW CELIW
IVPYX CIOTU NKLDL CBFSN QYSRR NXFJJ GKVCH ISGOC JGMXK UFKGR krypton4@bandit:/krypton/krypton4$
```

```
krypton4@bandit:/krypton/krypton4$ cat found2
YYIIA CWVSL PGLVH DSAFD TYYRY YEDRG LYXER BJIEV EPLVX BICNE XRIDT IICXD TIXRI PJNIB ILTYS EWCXE IKVRM VXBIC RRHOE ETFHD
LGHBG YZCWZ RQXMU ISDIA YKLOQ DWFQD LCIVA KRBYY IDMLB FSNQY STLYT NJUEQ VCFKT SPCTW AYSBB ZXRLG XRBOE LIUSB SRMPF EMJYR
WZPCS UMNJG WVXRE RBRVW IBMVV KRBRR HOLCW WIOPJ JJWVS LJCCC LCFEH DSRTR XOXFJ CECXM KKLOM PGIIK HYSUR YAQMV HSHLT KOXSU
BYEDX FJPAY YJIUS PSPGI IKODF JXSJW TLASW FXRMN XFJCM YRGBZ PVKMN EXYXF JWSBI QYRRN OGQCE NICWW SBCMZ PSEGY SISKW RNKFI
XFJWM BIQNE GOCMZ IXKWR JJEBI QTGIM YJNRV DLYYP SETPJ WIBGM TBINJ MTUEX HRMVR ISSBZ PVLYA VEFIP DXSYH ZWVEU JYXKH YRRUC
IKWCI FRDFC LXINX FJKMX AMTUQ KRGXY SEPBH VVDEG SCCGI CUZJI SSPZP VIBFG SYVBJ VVKRB YYIXQ WORAC AMZCH BYQYR KKMLG LXDLC
QZSXA CSKEG EWNEX YXFJW SBIQY RRNJM ZEHRM QTNRC YNUVV KRBSF SXICA VVURC BNLKX GYNEC JMWYI NMBSK QORRN FRSXY SUXRI QHRVO
GPTNJ YYLIR XBICK LPVSD SLXCE LIWMV PCIUS BSRMP WLEQP VXGMR MKLOQ QTKLK XQMVA YYJIE SDFCM LRQVW KFVKP MSXXS QCXYI DLMZX
LDXFN JAKWT JICUM LIRRN XFTLK RXDZC SPXFJ JGKVC HISGF SYJLO PYZXL OHFJR VDMJD RXDLC FNOGE PINEI MLBYM MLRMV TYSPH IIKXS
WVTSG IJUYZ XFJEY DWFNJ TKHBJ ULKRB XNIBI QTTPE QQDRR NXFJE YDWUJ IICSQ RRPVX FFKLO HPTGT OHYQD SCXYX DEXCY XYIZY RNEXR
IZFJO OXZZK XRIQH 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 KKMTU HSWRF EGOEU FPJBS KYYIP PYRVW KRBTE PIGYR VROEP
YFGYZ CWUSB 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 UGSXW YYMCA YXKSL IYSVZ ORRKL PNEWK FVDLC YIEFI JJIWD LCDYE NLYWU PIFCJ EAKPI NEKKR FTLVG LCSKL OCQFN FOJMW VXRIK
FXVOE RIZXM LRMRX MVMXJ INXFJ ISKHY SUHSZ GIVHD LCKFV OWRFJ JKVYX KLOCA TLPNW CJFRO MRMVV CMBJZ XGEQF MIBCU NUINM RHYEX
HUMVR DLCDT VOTRZ GXYXF JVHQI YSUPY SIJUM XXMNK XRIWH FYVHQ JVMDA YXRPC STJIC NICUR RNXFJ IIGIP JDEXC ZNXNK KEJUV YGIXR
XDLCG FXDSK YYICM BJJAO VCXFW DICUK LKXLT EIYJR MVQMS SQUGV MKGUS GRYSU JYVYR FQORR NKWOI KJUXR ERYYI SVHTL VXIWR LWDIL
INLKX QMRPV ACIFE COCIU SBSRM PHOWN FZVSR EQPMR ETJEX DLCKR MXXCX KMNIY XRMNX FJKMX AMTUQ KRYSU XRIJN FRCLM TBLSW QMRKQ
CKFEI KRBQF SUIBY YSEKF YWYVF SYKLO WAFII MVMBJ ESHUJ TEXRM 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 PIPGT EEPXC
WDIKW SWZRQ XFJUM CXYST IMEPJ WYVPW NELSW KNEHD LCSNI 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 DDFYP SSSUX RERDF
YMSSQ URYXH ZDTYZ CWKLO KSQWH YVMYY CGSSQ UFOOG QCINS PYYID MLBFS NQYSS ENPWI VRDIB TEXRI PTTOC FCQFA LYRNW MKQMS PSEVZ
FTOSX UNCPX SRRRX DIPXF QEGFK FVDLC KRPVA MZCHX SRMLV DQCFK EVPkrypton4@bandit:/krypton/krypton4$
```

We know that the key length is 6 letters long from the level info. Therefore, it is possible to deduce that every 6 letters in the cipher, we'll see the same shift. For example, have a look at the example Vignere Cipher in the level info:

P R O C E E D M E E T I
G O L D G O L D G O L D
V F Z F K S O P K S E L

In this case, every 4 letters from the start of the key, the plaintext is shifted by the letter 'G'. Shifting over one position, every 4 letters the plaintext is shifted by 'O'. This applies for every letter in the key.

In other words, a simple Caesar cipher shift occurs every n letters in a Vignere cipher, where n refers to the length of the key.

Therefore, if we have the length of the key, we can form individual strings compiled from the letters in the ciphertext that have been shifted by the same letter in the key. In the example, the start of two of these strings would be: ["VKK", "FSS"], as highlighted above. Performing a basic frequency analysis on one of these individual strings will theoretically yield the letter that it was shifted by. Hence, performing frequency analysis on all of the strings will yield the key.

Now, I've written a [Python program](#) to perform this task for us. Here it is below:

```python
#!/usr/bin/env python3

key_length = input("Enter the key length: ")
key_length = int(key_length)
ciphertext = input("Enter the ciphertext: ")
ciphertext = ciphertext.replace(" ", "")

string_list = []

for step in range(key_length):
    curr_string = ''
    for pos in range(step, len(ciphertext), key_length):
        curr_string = curr_string + ciphertext[pos]
    string_list.append(curr_string)

print(string_list)
```

I inserted a substring from the found1 file into the program to demonstrate it:

```
ython.exe" f:/Code/Krypton/split2.py
Enter the key length: 6
Enter the ciphertext: YYICS JIZIB AGYYX RIEWV IXAFN JOOVQ QVHDL CRKLB SSLYX RIQYI I
OXQT WXRIC RVVKP BHZXI YLYZP DLCDI IKGFJ UXRIP TFQGL CWVXR
['YIYWNQRLYTRHYDJTW', 'YZYVJVKYIWVZZIUFV', 'IIXIOHLXIXVXPIXQX', 'CBRXODBRORKIDKRGR'
, 'SAIAVLSIXIPYLGIL', 'JGEFQCSQQCBLCFPC']
PS F:\Code\Krypton> 
```

The first string in the array is made of letters in the ciphertext that are shifted by the first letter in the key, the second string made of letters shifted by the second letter in the key, etc.

Now, inputting all of found1's ciphertext into the program gives us:

```
P XDFIF OLGSU ECXFJ PENZY STINX FJXVY YLISI MEKJI SEKFJ IEXHF NCPSI PKFVD LCWVA OVCSF JKVKX ESBLM ZJICM LYYMC GMZEX BCMKK LO
ACX KEXHR MVKBS SSUAK WSSKM VPCIZ RDLCF WXOVL TFRDL CXLRC LMSVL YXGSK LOMPK RGOWD TIXRI PJNIB ILTKV OIQYF SPJCW KLOQQ MRHOW
MYYED FCKFV ORGLY XNSPT KLIEL IKSDS YSUXR IJNFR GIPJK MBIBF EHVEW IFAXY NTEXR IEWRW CELIW
['YIYWNQRLYTRHYDJTWZSLNNHTMJJYFNYIJJSLWNMFXBBKXIMJTBMIYJJNTYBWKWWLFGWISJSZYSYPJNFJQFWTYWKJJMMNSYWKYSAYMTSQZJRFDMKXFJJPKFSTTT
JMBMJDSQIJPFJTSJWJPJIKXISJFFYXMQMYIMZYFSJWJNTWGJYGZTMTYSFFJTWJQBSFSJSJIJJNKWSXZYZKXMSSIFTXSSKTJTYWMYKLTISNJFITWI', 'YZYVJVKY
IWVZZIUFVRFRMJVTVTJKTJSKGCVRFVZUKZZKKUPZRVVZYJVJXNKVZZVZKVVKWEFRYKYZNEEPYMYKYVFRJVRWKYUKVVVIVIRCXNRMFVKKBWZVZEFNUPVVYVKFFKYP
VEYUUUUJZIJKUYUNZEVUPNVPIVKYTFIZWKCXLNYFEJRCJTEJRKVVURCIUPTXSIICFVFEJYEKKVUKZWFLVKRINKFKRYFYKKUFKEFERW', 'IIXIOHLXIXVXPIXQXV
JXISVOMXXLOKSLSCHXYRGPSGXLESXWHVKEIRZLSELARRHVAITLSWAVIVIIMHHSIIEWIGVRXVXJWIMLWLVSVIXCLSOIYLLFISWHIHXCILTAIWWYXSIXHHMFXXXXRJ
SKSIXMWHVSWSTWILSIIAWWSASVXLIRHRHSKSHAWLWQGWIFEEIVISEPVAJSIMXLEKAMRXRRLLGXIVSLHEVXLSXRMHAXW', 'CBRXODBRORKIDKRGRNDONPEOBKYOS
BHOVYDOBNRODUMOCBRXKOEXQYOODIOONADVSKVOBRCOMIGCVIWEBCZKCBGDBYSOKQKOXKNILCEDDEOCKKOEMBGKBWKYCODOBEKMPPBRRLWIEYRGQSPBDBRDSLMBD
BBSCKXZLDEOEKXOZSBKDOKCBVGRNICOSCPOCNNYMEXSDOKBCCBOXBKVDODCYOORBOPOODONIDRGBVXRC', 'SAIAVLSIXIPYLGILIMLWXJRHWGTQRIERMTLJJWVE
LILAOSISRHESESVWLLRXLYLXPWMQAICMSSMWPSEEWTWMGSISSLWVMYRKSWIIISVLWPWSWWVXEWMRGEPYXVSWIRREEMXEMEKIVAIESIWEXIILIYYSMIIPWHMEYSVW
RWKIYXMVSGPXISEMSWXVXSXLXZXYEKHILVVLMGCAHSWPLVLLXMWIIIJQWFRSESIIIEYIE', 'JGEFQCSQQCBLCFPCEYCJFGJRNJSZFYLFRCCMPUMBCBMCCRPPQRP
DRLURYMYMCGCFJWCYFIMLSDLFDSIJJCLBYUKBWRBCBYMYMCQBDGYMFBJDQMCRPFABCWJRPIRRRQBLGLFLKBMYLYUJTJMQCYPBQRNRJRRIQNLJRFNFBLCLPCZMRIQ
GCPQRPBFYWMDGFYFLKFFPCCKMLMMCRSSCCLCMGPDPLQCQMCGPLYJPBWNEL']
PS F:\Code\Krypton> 
```

You may need to zoom into the image to see the output

Let's perform a frequency analysis on each string using the frequency analysis program, like we did in the previous level.

First String:
YIYWNQRLYTRHYDJTWZSLNNHTMJJYFNYIJJSLWNMFXBBKXIMJTBMIYJJNTYBWKWWLFGWI SJSZYSYPJNFJQFWTYWKJJMMNSYWKYSAYMTSQZJRFDMKXFJJPKFSTTTJMBMJDSQIJPFJT SJWJPJIKXISJFFYXMQMYIMZYFSJWJNTWGJYGZTMTYSFFJTWJQBSFSJSJIJJNKWSXZYZKX MSSIFTXSSKTJTYWMYKLTISNJFITWI

The single letter frequency analysis is:

```
Enter the Ciphertext: YIYWNQRLYTRHYDJTWZSLNNHTMJJYFNYIJJSLWNMFXBBKXIMJTBMIYJJNTYBWKWWLFGWISJSZYSYPJNFJQFWTYWKJJMMNSYWKYSAYMTSQ
ZJRFDMKXFJJPKFSTTTJMBMJDSQIJPFJTSJWJPJIKXISJFFYXMQMYIMZYFSJWJNTWGJYGZTMTYSFFJTWJQBSFSJSJIJJNKWSXZYZKXMSSIFTXSSKTJTYWMYKLTISNJF
ITWI
J: 35
S: 23
Y: 22
T: 20
W: 17
F: 17
M: 16
I: 14
N: 11
K: 11
X: 8
Z: 7
Q: 6
B: 6
L: 5
P: 4
R: 3
D: 3
G: 3
H: 2
A: 1
```

The most common letter in the string is J’. In typical plaintext English, the most common letter is ‘E’. Therefore, we can deduce that E has been shifted to J in the ciphertext. E is the 4th letter in the alphabet, whereas J is the 9th letter (remembering that A is the 0th letter) . Therefore, a Caesar shift of 5 positions has been performed on this string; The 5th letter in the alphabet is F, so we can conclude that **every letter in the first string has been shifted by F and so the first letter of the Vignere cipher key is F.**

We now repeat this process for the other 5 strings in our list. I’ll show the process for the next string and move on.

Second String:
YZYVJVKYIWVZZIUFVRFRMJVTVTJKTJSKGCVRFVZUKZZKKUPZRVVZYJVJXNKVZZVZKVVKWEFRYKYZNEEPYMYKYVFRJVRWKYUKVVVIVIRCXNRMFVKKBWZVZEFNUPVVYVKFFKYPVEYUUUUJZIJKUYUNZEVUPNVPIVKYTFIZWKCXLNYFEJRCJTEJRKVVURCIUPTXSIICFVFEJYEKKVUKZWFLVKRINKFKRYFYKKUFKEFERW

```
Enter the Ciphertext: YZYVJVKYIWVZZIUFVRFRMJVTVTJKTJSKGCVRFVZUKZZKKUPZRVVZYJVJXNKVZZVZKVVKWEFRYKYZNEEPYMYKYVFRJVRWKYUKVV
VIVIRCXNRMFVKKBWZVZEFNUPVVYVKFFKYPVEYUUUUJZIJKUYUNZEVUPNVPIVKYTFIZWKCXLNYFEJRCJTEJRKVVURCIUPTXSIICFVFEJYEKKVUKZWFLVKRINK
FKRYFYKKUFKEFERW
V: 35
K: 30
Y: 19
Z: 18
F: 18
U: 16
R: 15
J: 13
E: 12
I: 11
N: 8
W: 7
P: 7
T: 6
C: 6
X: 4
M: 3
S: 2
L: 2
G: 1
B: 1
```

V is the most common letter. E is the 4th letter in the alphabet, whereas V is the 21st letter. $21 - 4 = 17$, and hence a Caesar Shift of 17 positions was performed on the string. **The 17th letter is R and so we can conclude that the second letter of the key is R.**

After analyzing all of the other strings, **the key is discovered to be FREKEY.** We can now easily decipher the password by subtracting the index of each letter in the ciphertext with the corresponding letter in the key (similar to what we just did):

Ciphertext: H C I K V R J O X
Key: F R E K E Y F R E
Password: C L E A R T E X T

H - F = 7 - 5
= 2
= C

C - R = 2 - 17
= -15
= 26 - 15
= 11
= L

I - E = 8 - 4
= 4
= E

Note that what we've done above is actually equivalent to performing a modulo 26 operation. For example:
$(7 - 5)\ mod\ 26 = 2$
$(2 - 17)\ mod\ 26 = 11$

Level 5 - Level 6

Link: https://overthewire.org/wargames/krypton/krypton5.html

Level Info

FA can break a known key length as well. Lets try one last polyalphabetic cipher, but this time the key length is unknown. Note: the text is written in American English

Enjoy.

Research

A method of deducing the key length of a polyalphabetic substitution cipher is the Kasiski Examination. It involves searching for repeated strings in the ciphertext, then finding the distance between the beginning of each of the found strings. The factors of the distance value are then considered; the key length is one of these factors. An analyst will narrow down the factors based on logic and trial and error.

Kasiski Examination works based on the assumption that if a repeated string occurs in the plaintext, and the distance between said string is a multiple of the key length, then they'll be encrypted in the exact same way.

Consider the following example from the Wikipedia entry on Kasiski Examination:

```
Key:        ABCDABCDABCDABCDABCDABCDABCD
Plaintext:  cryptoisshortforcryptography
Ciphertext: CSASTPKVSIQUTGQUCSASTPIUAQJB
```

In the ciphertext, we can see that the strings CSASTP are repeated. By counting each letter from the start of the first CSASTP string, we deduce that their distance is 16. The factors of 16 are 16, 8, 4, 2, 1. The key length is likely not 16 as this seems too long. On the other hand, the key length also likely isn't 1 or 2 as these are too short. Hence, we only need to try 8 and 4 as the key length.

If the ciphertext is longer, and multiple instances of different repeated strings are found, we can count the distance between each of these strings. The greatest common factor of the distances is likely to be the correct key length. An example of this is demonstrated in the method below.

Method

Examining the found1 file returns the following ciphertext.

```
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
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 ARVMY IBLXU QPYYH GNXYO ACCIN QBUQA GELNR TYQIH LANTW HAYCP RJOMO KJYTV SGVLY RRSIG
NKVXI MQJEG GJOML MSGNV VERRC MRYBA GEQNP RGKLB XFLRP XRZDE JESGN XSYVB DSSZA LCXYE ICXXZ OVTPW BLEVK ZCDEA JYPCL CDXUG
MARML RWVTZ LXIPL PJKKL CIREP RJYVB 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 UMARE BCIPK GAVCY EXNOG LNLCC JVBXH
XHRHI AZBLD LZWIF YXKLM PELQG 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 LXSDJ INXLH PXCCV NVIPX KMFXV ZLTOW QLKRY TZDLC DTVXB ACSDE LVYOL BCWPE ERTZD TYDXF AILBR YEYEG ESIHC
QMPOX UDMLZ VVMBU KPGEC EGIWO HMFXG NXPBW KPVRS XZCEE PWVTM OOIYC XURRV BHCCS SKOLX XQSEQ RTAOP WNSZK MVDLC PRTRB ZRGPZ
AAGGK ZIMAP RLKVW EAZRT XXZCS DMVVZ BZRWS MNRIM ZSRYX IEOVH GLGNL FZKHX KCESE KEHDI FLZRV KVFIB XSEKB TZSPE EAZMV DLCSY
ZGGYK GCELN TTUIG MXQHT BJKXG ZRFEX ABIAP MIKWA RVMFK UGGFY JRSIP NBJUI LDSSZ ALMSA VPNTX IBSMO krypton5@bandit:/krypton
/krypton5$
```

As circled in red and blue, there are two repeating words: MGCTZDLC and TXJGLARI. Both of these words are 8 letters long, and have a **distance of 45 characters** in between them, excluding whitespaces. The factors of 45 are: 45, 15, 9, 5, 3, and 1. 45 and 15 are likely too long for the key length, whereas 1 and 3 are likely too short. Hence, by method of elimination, **the key length is likely 9 characters long.** To confirm this, let's examine the ciphertext in the found2 file:

```
GLCYX UKFHS PEZXF AVJOW QQYYR RAYHM GIEOG ARIAZ YEYXV PXFPJ BXXUY SLELR NXHNH PLARX TADLC CSLGE NOSPR IUUML VSNPR RJMOO
GMLGU JHVBE QSMFI NZDSK HEFNX KSHGE AVZAZ YQCQP BAKPC LMQGR XXTYR WQSEG FHSPH ZYETX FPVMX PBTWV XMLHM AZXYG EQLRN IAPOZ
CXIAZ MVMSL RVNZN SKXCL 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 SHCOI FZBUK XGZRF ZALWM JOIJE BUCMB PSSZA LMSYN
LJOMO SXQOE ZVTUN HGCXL YMYKA GEWQO LHQIC LFYKL TOPJL RQOMZ YFQNY EOMFG EQCEG NXYVM IPEYG KNOVB ZKXKG UOPKC PBXKF DLCAE
FYXUQ IPDLN QBUQL GXWRR YVEXM QMGOG JREGY WBLLA BEULX NTZSO SDDLN MZFGV YATRX YSKTN TRTNT AKRBX YQJRS OKQHE FXTAR IPWMX
KTSKV EPVFU KAYJB ZKGNX YOAGW POKTW KGIPX GUVHV EGDXB SHYBS UOVNC XYIIQ DMEOY ARIUP EGNXY RSJOW NTWAR IUTRQ YXACX MWIEG
USOJY TVGNX ASHCH MYRLL BZCAV RZMFX MAPPL GMHLS SEXJU BUDLC LJGKK UYSLD MEHXK CMPTW UGESX SRRSG UULNX GWPAO ZODFS EMJGG
AKFCO VBUFH XHYME EHXYK RBELR TUYOE IQEFZ LPBCC DWVXM OKXUL CFOKP PCMFT YKTZO WFZAP UGJYV BRIAZ ELWEL DZNRB 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 EUFOO 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 TPVNR FHSHQ LRQZA LVELO PNJRD OVCLP YRHPD IPTRT HRHMG GOIAZ TAFEP TSHYI VSRRD SSZAL BSYOF
RZPLO RRSIP UGJYV BLRQZ ALMSD QIRXH VWAFP RNMXU DPCXE AUYZS BRJJB XFHVP WOVRY LLNML LFEUP UCYGE SSIEV DLCDT EKMAI ACWPJ
UKULY RGIEE PLVPI PTGCB ARPYC KRYJB KVCNY SLLHX HJLVT KYSKT QESGN XWYGI PXFVT ZCIBL PBTZV XLGDA NEMVR MQMVR GDMKW Rkrypt
on5@bandit:/krypton/krypton5$
```

As circled in red, the word SPUGJYVB is repeated. It is 8 letters long, and there is a distance of 81 characters in between them. The factors of 81 are: 81, 27, 9, 3 and 1. **9 is the greatest common divisor of the two distances, 81 and 45.** Hence, we have obtained evidence to further reinforce the proposition that the key length is 9.

From here on, the steps to decipher the password ciphertext, which is **BELOS Z**, are identical to the method utilized in the previous level. I'll briefly illustrate the steps here

Insert the ciphertext of one of the found files into the split program:

```
Enter the key length: 9
Enter the ciphertext: SXULW GNXIO WRZJG OFLCM RHEFZ ALGSP DXBLM PWIQT XJGLA RIYRI BLPPC HMXMG CTZDL CLKRU YMYSJ TWUTX ZC
MRH EFZAL OTMNL BLULV MCQMG CTZDL CPTBI AVPML NVRJN SSXWT XJGLA RIQPE FUGVP PGRLG OMDKW RSIFK TZYRM QHNXD UOWQT XJGLA RI
QAV VTZVP LMAIV ZPHCX FPAVT MLBSD OIFVT PBACS EQKOL BCRSM AMULP SPPYF CXOKH LZXUO GNLID ZVRAL DOACC INREN YMLRH VXXJD XM
SIN BXUGI UPVRG ESQSG YKQOK LMXRS IBZAL BAYJM AYAVB XRSIC KKPYH ULWFU YHBPG VIGNX WBIQP RGVXY SSBEL NZLVW IMQMG YGVSW GP
WGG NARSP TXVKL PXWGD XRJHU SXQMI VTZYO GCTZR JYVBK MZHBX YVBIT TPVTM OOWSA IERTA SZCOI TXXLY JAZQC GKPCS LZRYE MOOVC HI
EKT RSREH MGNTS KVEPN NCTUN EOFIR TPPDL YAPNO GMKGC ZRGNX ARVMY IBLXU QPYYH GNXYO ACCIN QBUQA GELNR TYQIH LANTW HAYCP RJ
OMO KJYTV SGVLY RRSIG NKVXI MQJEG GJOML MSGNV VERRC MRYBA GEQNP RGKLB XFLRP XRZDE JESGN XSYVB DSSZA LCXYE ICXXZ OVTPW BL
EVK ZCDEA JYPCL CDXUG MARML RWVTZ LXIPL PJKKL CIREP RJYVB ITPVV ZPHCX FPCRG KVPSS CPBXW VXIRS SHYTU NWCGI ANNUN VCOEA JL
LFI LECSO OLCTG CMGAT SBITP PNZBV XWUPV RIHUM IBPHG UXUQP YYHNZ MOKXD LZBAK LNTCC MBJTZ KXRSM FSKZC SSELP UMARE BCIPK GA
VCY EXNOG LNLCC JVBXH XHRHI AZBLD LZWIF YXKLM PELQG RVPAF ZQNVK VZLCE MPVKP FERPM AZALV MDPKH GKKCL YOLRX TSNIB ELRYN IV
MKP ECVXH BELNI OETUX SSYGV TZARE RLVEG GNOQC YXFCX YOQYO ISUKA RIQHE YRHDS REFTB LEVXH MYEAJ PLCXK TRFZX YOZCY XUKVV MO
JLR RMAVC XFLHO KXUVE GOSAR RHBSS YHQUS LXSDJ INXLH PXCCV NVIPX KMFXV ZLTOW QLKRY TZDLC DTVXB ACSDE LVYOL BCWPE ERTZD TY
DXF AILBR YEYEG ESIHC QMPOX UDMLZ VVMBU KPGEC EGIWO HMFXG NXPBW KPVRS XZCEE PWVTM OOIYC XURRV BHCCS SKOLX XQSEQ RTAOP WN
SZK MVDLC PRTRB ZRGPZ AAGGK ZIMAP RLKVW EAZRT XXZCS DMVVZ BZRWS MNRIM ZSRYX IEOVH GLGNL FZKHX KCESE KEHDI FLZRV KVFIB XS
EKB TZSPE EAZMV DLCSY ZGGYK GCELN TTUIG MXQHT BJKXG ZRFEX ABIAP MIKWA RVMFK UGGFY JRSIP NBJUI LDSSZ ALMSA VPNTX IBSMO
['SOCGWRCDYCOVDPSRPKYORPCBBBPKLOYDGQMBBYBBSWSRXSOBBOSYCOSKNDKRQOQQAKYXOEGBDYCOKCRXCBCPXNNFCBXMQKNKCRVNXDLPZFVCNVBXROORSX
COMCVBXPXODCBDBIDKOBCOBXODRZWCRSGXDFZDKIKBRYUMB', 'XWMSIIHLSMTMLMXIPWRWILXSACSHIAMXISXAXHPISIWSWXGKIWZJSVRVELGVPAAIYJRIM
REXEVXVZLMIIIXSIWVITIWIPXTXSECLHLMALEMLIMESEQQIRHXZOXESSXKWLSCTRHMPHWEIHXPLGIESWRLKIISLGGXIVJISS', 'URRPQYMCJRMCCLWQGRMQ
QMFDCRPLDCLMUGRYRUGQBMGPGQCMTSCALCEEOYCMYCGHCYRMLRQFJBYTCCLPRTFSRCCLGTUBYDCRSBYCRZPFCRDYBKLSRCYQEMKCJFGSDCMQCDWYYCLGMKEY
CQWCPMADSYGCFBPCCMGAMRLAM', 'LZHDTRXLTHNQPNTPRSQTAAPOSSPZZCRSPYSJSLVPEQPTDMTZTAOZZHHPFAZYYCELPTSQMCNLEDEPDDRLEPPCSGOECPP
PYLCSECECHWEZEPPOEPNYLYOHFYTYLLOYJCFLDEPDEQZEFPPCCSNPZAZMMXNELXESEXZPFSDVO', 'WJEXXIMKWELMTVXELIHXVIAIEMYXVIHIVKIMIWIRLM
WXXIZHPIIQRIMNIPRIHILARVIJSMPRSSIWEXWPPVCPSIECMPVHHZMMLIXJIILQMMKLLEIGVXIETERXRHSHIVXKTLEXYMVCXVWXSESRAPRVNILSZSEYLQRMKI
SP', 'GGFBJBGRUFBGBRJFGFNJVVVFQAFURNVNRQBACFGGNGGVRVRBVETCYEGNRNGBGNNNJSGEGRRPGSCBAUVJRVRBHAASGNRGNBBFPPNVAFQNPAHRRCOVEF
SYBAFUROAQNNVRVVEFEPVEGRVUSQZTARTVREFEREAZNHFIUPSN', 'NOZLGLCUTZLCIJGUOKXGTZTVKMCOARXBGOZYKUNVZYGKJTJXTRXGEKNCTONLNQRTOG
NGNYGXNZXLJGTKJZGXYNJOAZIUZAJSUKOBZYGVVZGXYVETGCURLJZKMKRUXVZYXYRAGOMGNSTRKRKRGLXZIOZKVKZGTTEKGNZT', 'XFAMLPTYXAUTANLGMT
DLZPMTOUXGLEXXEKAAKYXXLGNLHZYYMTXKMTTTPGXXXBTWMVKGVBKRXAXEYMZKYPKWTNLOTBHXMKTKMGGXBXRKKAKTNXTZGXKHEPXVAXRSLILTBOTIEXBIXX
MROTMBGKXBMVKEKBMGTBXWGBAX', 'ILLPAPZMZLLZVSAVDZUAVHLPLLONDNJUSLLVPHWYVVAPUYVVOALPORSUPMAUYUYHOLVJVALZSLZVPALLVHVVUULLSV
UUOLZZAALHLKVVPLKSIHUANYADVLYVVUHLHPTZALZLSUUWPZOVLAVZKVZZZHHHVTVYUJAAFJLI']
```

Perform a frequency analysis on each of the resultant strings. For example, inputting the second string into the frequency analysis program shows that I is the most common letter, meaning that E has been Caesar shifted to I. This is a shift of 4 positions, a shift by the letter E in other words. Hence, the second letter of the key is E.

After analysing all 9 strings, we find that the key is supposedly XEYLENGTH. This can obviously be corrected to **KEYLENGTH.** I can't clearly explain why the first letter turned out to be X, but luckily it was an easy correction.

Now, the ciphertext, **BELOS Z**, can easily be deciphered:

Ciphertext = B E L O S Z

Key = K E Y L E N G T H

Password = R A N D O M

B - K = 1 - 10
= - 9
= 26 - 9
= 17
= R

E - E = 4 - 4
= 0
= A

L - Y = 11 - 24
= -13
= 26 - 13
= 13
= N

Level 6 - Level 7

Link: https://overthewire.org/wargames/krypton/krypton6.html

Level Info

Examine the link since the contents are really long.

Preamble

The final level involves cracking a **stream cipher**. Similar to the second level, we are given a program, encrypt6, that will encrypt any plaintext using the same key that was used to encrypt the password.

Research

In modern cryptography, there are two types of ciphers: block ciphers and stream ciphers. The substitution ciphers we dealt with in previous levels share similarities with modern block ciphers which work by grouping plaintext into blocks of specific sizes (128 bits for example) and encrypting them all at once. Stream ciphers on the other hand, encrypt the plaintext one bit at a time. Stream ciphers consist of three main parts:

- Plaintext
- Keystream: A pseudorandom sequence of bits produced by inputting a specific key into a pseudorandom number generator.
- Ciphertext: The ciphertext is produced by performing XOR operations on the plaintext bit-by-bit with the keystream.

The keystream being pseudorandom is an important detail. Pseudorandom number generators create statistically random digit strings, but these strings are not truly random. This is because they are determined via an initial value, known as the “seed value” . In this case, the seed value is the key. Therefore, if the key used to generate the keystream is obtained, then the keystream itself can easily be found.

XOR operations are a type of Linear-feedback Shift Registers (LFSR) that when used on binary bits, work like so:

Boolean Math: XOR ($\oplus$)

X	Y	X $\oplus$ Y
0	0	0
0	1	1
1	0	1
1	1	0

Source: https://www.secureideas.com/blog/2020/10/boolean-math-xor-logic-cissp-domain-3.htm

Method

Let us observe how the encrypt6 function encrypts a string consisting of only A’s:

```
krypton6@bandit:/krypton/krypton6$ python3 -c "print('A'*200)" > /tmp/plainA.txt
krypton6@bandit:/krypton/krypton6$ cat /tmp/plainA.txt
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
krypton6@bandit:/krypton/krypton6$ ./encrypt6 /tmp/plainA.txt /tmp/ciperA.txt
krypton6@bandit:/krypton/krypton6$ cat /tmp/ciperA.txt
EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRN
EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYkrypton6@bandit:/krypton/krypton6$
```

The number of A’s to test is arbitrary, I simply chose 200 because I felt that a longer plaintext would better illustrate potential patterns. This assumption was correct; the ciphertext repeats several times. To be precise, the word “**EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRN”** is repeated. It is my suspicion that this is simply the key.

Let’s test this theory by observing the program’s behaviour on 200 B’s instead:

```
PNUKLYLWRQKGKBEkrypton6@bandit:/krypython3 -c "print('B'*100)" > /tmp/plainB.txt
krypton6@bandit:/krypton/krypton6$ python3 -c "print('B'*200)" > /tmp/plainB.txt
krypton6@bandit:/krypton/krypton6$ ./encrypt6 /tmp/plainB.txt /tmp/cipherB.txt
krypton6@bandit:/krypton/krypton6$ cat /tmp/cipherB.txt
FJDUEHZJZALUIOTJSGYZDQGVFPDLSOFJDUEHZJZALUIOTJSGYZDQGVFPDLSOFJDUEHZJZALUIOTJSGYZDQGVFPDLSOFJDUEHZJZALUIOTJSGYZDQGVFPDLSO
FJDUEHZJZALUIOTJSGYZDQGVFPDLSOFJDUEHZJZALUIOTJSGYZDQGVFPDLSOFJDUEHZJZALUIOTJSGYZkrypton6@bandit:/krypton/krypton6$
```

We get repetitions of the word
“**FJDUEHZJZALUIOTJSGYZDQGVFPDLSOFJDUEHZJZALUIOTJSGYZDQGVFPDLSO”** instead. Notice that this is simply the first repeated word with each letter shifted forward by one position. Therefore, we can confirm that plainB.txt has been encrypted with the first word. From the start of the string, if B is encrypted by E, we get F (B is Caesar shifted to the right by E), if B is encrypted by I we get J etc.

Therefore, the ciphertext, **PNUKLYLWRQKGKBE**, can be decrypted easily using the same method used in the past two levels.

Ciphertext = P N U K L Y L W R Q K G K B E
Key = E I C T D G Y I Y Z K T H N S
Password = L F S R I S N O T R A N D O M

P - E = 15 - 4
= 11
= L

N - I = 13 - 8
= 5
= F

U - C = 20 - 2
= 18
= S

K - T = 10 - 19
= -9
= 26 - 9
= 17
= R

The level info made this level much more difficult than it needed to be. I spent a while going in circles, trying to make sense of hexdump output to detect patterns in it and messing with online pseudorandom generators. I’m sure it is possible to complete this level using hexdump and other similar tools but I believe the method I used here is the most straightforward.

Conclusion

With that, we have completed all of Krypton, congratulations! I believe that Krypton is a great resource for beginners to start their journey into cryptography. Through both self-research and practice, players learn the mechanisms of classical substitution ciphers and the cryptanalysis techniques used to crack them. Most importantly, the challenges force players to adopt an attacker’s mindset which is something that all security people must learn to understand.

Thank you for following my guide, I hope that it was easy to understand and that you learnt something from it.

Sources

Caesar cipher in cryptography (2023) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/ (Accessed: 15 October 2023).

Classical cipher (2023) *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Classical_cipher#Cryptanalysis_of_classical_ciphers (Accessed: 07 October 2023).

Cryptanalysis (no date) *Cryptanalysis | OWASP Foundation*. Available at: https://owasp.org/www-community/attacks/Cryptanalysis#:~:text=Cryptanalysis%20is%20a%20process%20of,secret%20key%20(instance%20deduction) (Accessed: 16 October 2023).

Frequency analysis (2023) *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Frequency_analysis (Accessed: 16 October 2023).

Graphical frequency analysis (no date) *CrypTool Portal*. Available at: https://www.cryptool.org/en/cto/frequency-analysis (Accessed: 16 October 2023).

Josefsson, S. (2006) *RFC 4648: The base16, Base32, and base64 data encodings*, *IETF Datatracker*. Available at: https://datatracker.ietf.org/doc/html/rfc4648#section-4 (Accessed: 10 October 2023).

Kasiski examination (2022) *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Kasiski_examination (Accessed: 25 October 2023).

Rot13 cipher (2023) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/rot13-cipher/ (Accessed: 14 October 2023).

schenken (2018) *Base64 encoding*, *YouTube*. Available at: https://www.youtube.com/watch?v=aUdKd0IFI34 (Accessed: 10 October 2023).

Stream ciphers (2020) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/stream-ciphers/ (Accessed: 27 October 2023).

Tilliksew, B. (no date) *Vigenère cipher*, *Brilliant Math & Science Wiki*. Available at: https://brilliant.org/wiki/vigenere-cipher/#:~:text=The%20Vigen%C3%A8re%20cipher%20is%20a,is%20much%20harder%20to%20crack. (Accessed: 04 November 2023).