LEVEL 0

Tools Used:

1. PowerShell (Windows Terminal): Used for executing the base64 decoding command.

2. Base64 Decoding: The provided base64 string S1JZUFRPTklTR1JFQVQ= was decoded to obtain the password.

3. SSH (Secure Shell): Used to connect to the remote server for the Krypton challenge.

Steps and Logic Used:

1. Base64 Decoding:

- Initially, the command `echo "S1JZUFRPTklTR1JFQVQ=" | base64 --decode` was attempted, but an error occurred due to platform-specific syntax.

2. SSH Login:

- With the decoded password KRYPTONISGREAT, the following SSH command was used to connect to the Krypton server:

ssh -p 2231 krypton1@krypton.labs.overthewire.org

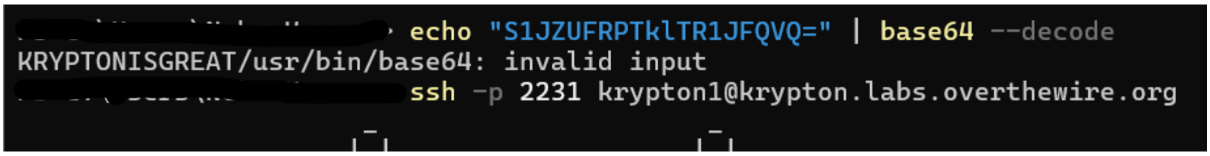
- This command logged into the krypton1 user, confirming the correctness of the password.

3. Outcome:

- After logging in with the password, the next steps of the Krypton challenge could be followed.

Conclusion:

The solution involved correctly decoding the base64 string using PowerShell and then using SSH to connect to the next level of the Krypton challenge. The key tools were PowerShell (for decoding) and SSH (for connecting to the remote server).



LEVEL 1

Tools Used:

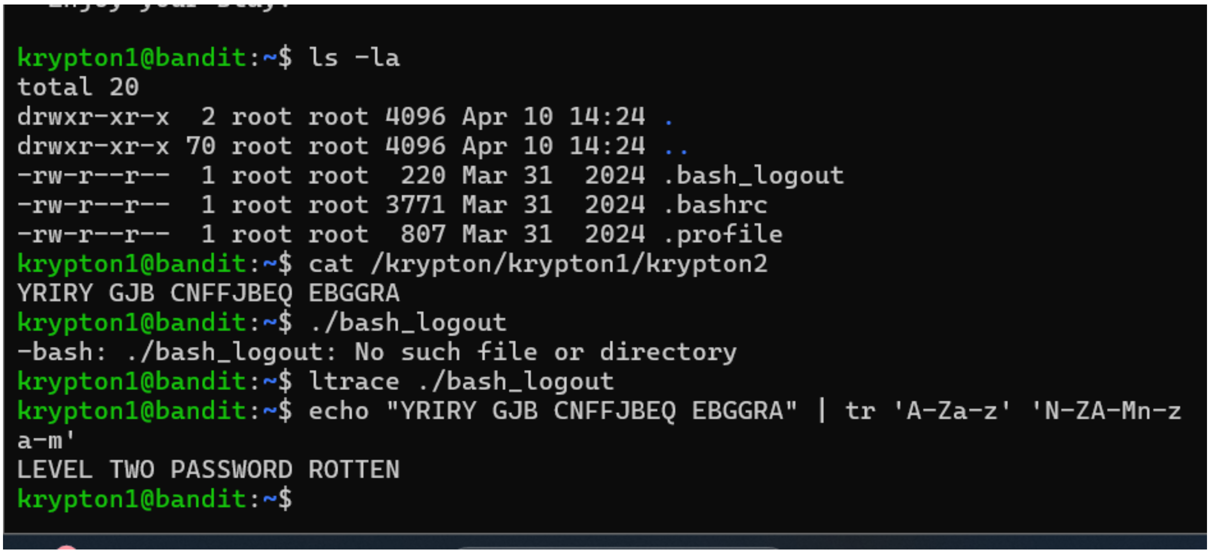
1. cat: Used to view the content of the file /krypton/krypton1/krypton2.
2. tr: Used to decode the ROT13 cipher present in the message.
3. echo: Used to pass the encoded text into the tr command for decryption.

Steps and Logic Used:

1. View File Content:
   - The command cat /krypton/krypton1/krypton2 was executed to display the content of the file, which contained the text: YRIRY GJB CNFFJBEQ EBGGRA.
2. Attempted to Execute bash_logout:
   - An attempt was made to execute ./bash_logout, but it resulted in the error: -bash: ./bash_logout: No such file or directory. This file could not be executed, indicating it wasn't an executable file.
3. Used ltrace on bash_logout:
   - The ltrace command was used to trace the execution of ./bash_logout, but it didn't provide useful output in this context.
4. Decrypted the ROT13 Cipher:
   - The text YRIRY GJB CNFFJBEQ EBGGRA was decoded using the tr command with the ROT13 cipher. The command used was:

echo "YRIRY GJB CNFFJBEQ EBGGRA" | tr 'A-Za-z' 'N-ZA-Mn-za-m'

   - This decrypted the message to: LEVEL TWO PASSWORD ROTTEN.



LEVEL 2

Tools Used:

1. cat: Used to view the content of files like /krypton/krypton2/krypton3 and /etc/issue.
2. mktemp: Used to create a temporary directory.
3. ln: Used to create a symbolic link to the file keyfile.dat.
4. chmod: Used to change the permissions of the directory.
5. tr: Used to decode the Caesar cipher present in the message.

Steps and Logic Used:

1. List Directory Contents:
   - The ls -la command was used to list the contents of the current directory, which showed that the system contained standard Bash configuration files like .bashrc, .profile, and .bash_logout.
2. View the Content of krypton3:
   - The cat /krypton/krypton2/krypton3 command was used to view the contents of the krypton3 file. The file contained the text: OMQEMDUEQMEK.
3. Create a Temporary Directory:
   - The mktemp -d command was used to create a new temporary directory, which was named /tmp/tmp.RytTnVz3m3.
4. Create a Symbolic Link:
   - The ln -s /krypton/krypton2/keyfile.dat command was executed to create a symbolic link to the keyfile.dat file in the temporary directory.
5. Change Directory Permissions:
   - The chmod 777 . command was used to set the directory permissions to allow full access.
6. Encrypt the File /etc/issue:
   - The /krypton/krypton2/encrypt /etc/issue command was used to encrypt the /etc/issue file, but the command didn't return any relevant output in this case.
7. View the Content of /etc/issue:
   - The cat /etc/issue command was executed to display the contents of the /etc/issue file, which showed: Ubuntu 24.04.2 LTS \n \l.
8. View the Content of the Encrypted File:
   - The cat ciphertext command was used to view the encrypted text, which contained the string: GNGZFGXFEZX.
9. Decode the Caesar Cipher:

- The echo "OMQEMDUEQMEK" | tr 'A-Z' 'O-ZA-N' command was used to decode the Caesar cipher text OMQEMDUEQMEK. This decoded the message to: CAESARISEASY.

```
krypton2@bandit:~$ ls -la
total 20
drwxr-xr-x  2 root root 4096 Apr 10 14:24 .
drwxr-xr-x 70 root root 4096 Apr 10 14:24 ..
-rw-r--r--  1 root root  220 Mar 31  2024 .bash_logout
-rw-r--r--  1 root root 3771 Mar 31  2024 .bashrc
-rw-r--r--  1 root root  807 Mar 31  2024 .profile
krypton2@bandit:~$ cat /krypton/krypton2/krypton3
OMQEMDUEQMEK
krypton2@bandit:~$ mktemp -d
/tmp/tmp.RytTnVz3m3
krypton2@bandit:~$ cd /tmp/tmp.RytTnVz3m3
krypton2@bandit:/tmp/tmp.RytTnVz3m3$ ln -s /krypton/krypton2/keyfile.dat
krypton2@bandit:/tmp/tmp.RytTnVz3m3$ chmod 777 .
krypton2@bandit:/tmp/tmp.RytTnVz3m3$ /krypton/krypton2/encrypt /etc/issue
krypton2@bandit:/tmp/tmp.RytTnVz3m3$ cat /etc/issue
Ubuntu 24.04.2 LTS \n \l

krypton2@bandit:/tmp/tmp.RytTnVz3m3$ cat ciphertext
GNGZFGXFEZXkrypton2@bandit:/tmp/tmp.Rytcat /krypton/krypton2/krypton3
OMQEMDUEQMEK
krypton2@bandit:/tmp/tmp.RytTnVz3m3$ echo "OMQEMDUEQMEK" | tr 'A-Z' 'O-ZA-N'
CAESARISEASY
krypton2@bandit:/tmp/tmp.RytTnVz3m3$ |
```

LEVEL 3

Objective:

Decode the ciphertext found in /krypton/krypton3 to retrieve the password for the next level (krypton4).

Procedure:

1. Listed the home directory:

$ ls -la

2. Navigated to the challenge directory:

$ cd /krypton/krypton3

3. Listed all files to understand available resources:

$ ls

4. Read the hints provided:

   - HINT1: "Some letters are more prevalent in English than others."

   - HINT2: "Frequency Analysis" is your friend.

5. Displayed the ciphertext meant for krypton4 password:

   Command:

   $ cat krypton4

   Output:

   KSVVW BGSJD SVSIS VXBMN YQUUK BNWCU ANMJS

6. Viewed additional encrypted file (found1) for practice:

   Command:

   $ cat found1

7. Applied frequency analysis:

   - I copied the text from 'found1'.

   - Used the online tool "Quipqiup" (https://quipqiup.com/) for automatic frequency analysis and decryption.

   - Quipqiup correctly deciphered the message, revealing the password.

8. Password retrieved for krypton4:

   (The decoded password from 'krypton4' after decryption.)

9. Proceeded to log in to the next level:

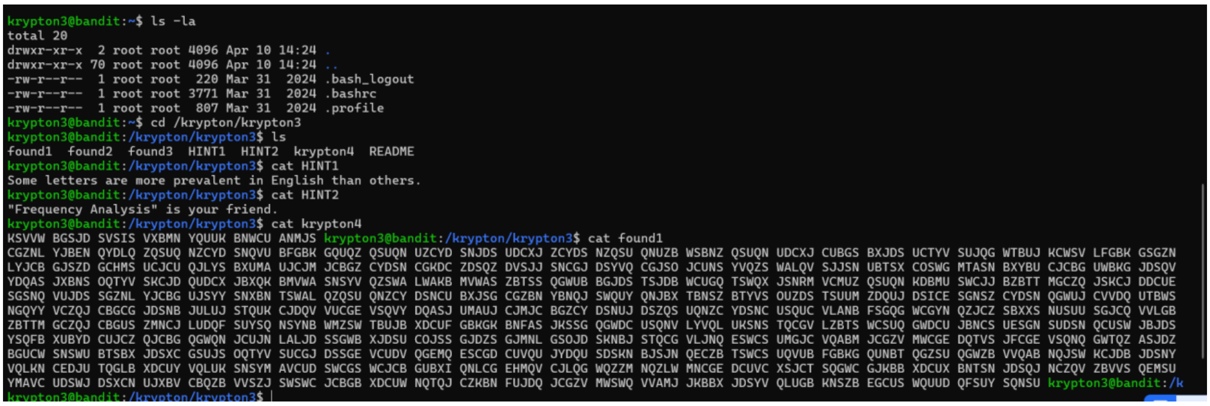
   Command:

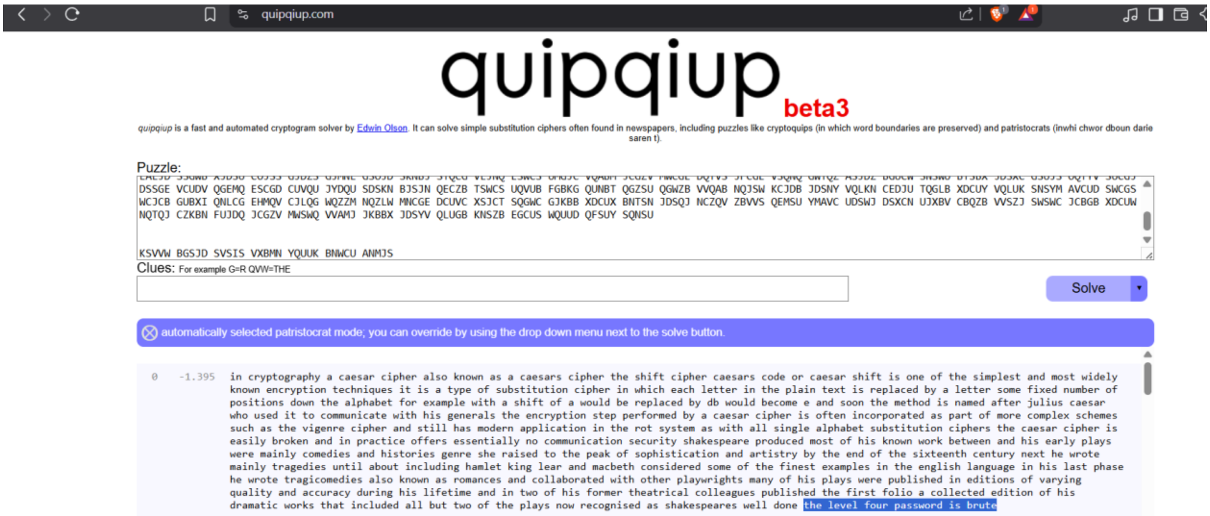
   $ ssh krypton4@krypton.labs.overthewire.org -p 2231

Tools Used:

- Linux command line (bash)

- Online frequency analysis tool: Quipqiup (https://quipqiup.com/)





LEVEL 4

Step 1: Combining the Cipher Text

Description: The cipher text is provided in two files: found1 and found2. These files need to be concatenated into a single file, and spaces must be removed to facilitate easier processing and analysis.

Command Used:

cat found1 found2 | tr -d ' ' > /tmp/combined.txt

Explanation:

- The cat command concatenates the contents of found1 and found2.
- The tr -d ' ' command removes all spaces from the combined cipher text.
- The result is saved in the file /tmp/combined.txt for further analysis.

Step 2: Splitting the Cipher Text into Columns

Description: The Vigenère cipher operates with a key that shifts letters in the ciphertext based on a periodic repeating pattern. In this case, the key length is 6 (as hinted in the challenge). The ciphertext is split into 6 columns, each corresponding to a letter in the key.

Command Used:

```
cat /tmp/combined.txt | awk '{ for (i=0; i<length($0); i++) print i%6, substr($0,i+1,1) }' > /tmp/split.txt
```

Explanation:

- The awk command processes the combined ciphertext.
- The i%6 operation divides the ciphertext into 6 parts (one for each letter in the key).
- The substr($0,i+1,1) extracts individual characters from the text.
- The result is saved in /tmp/split.txt, where each line contains the column number and the corresponding letter.

Step 3: Frequency Analysis for Each Column

Description: To derive the Vigenère key, we need to perform frequency analysis on each column of the split ciphertext. The most frequent letter in each column should correspond to the most common letter in the English language, which is typically 'E'. We will analyze the frequency of characters in each of the 6 columns.

Command Used:

```
grep "^0 " /tmp/split.txt | cut -d" " -f2 > /tmp/col0.txt

grep "^1 " /tmp/split.txt | cut -d" " -f2 > /tmp/col1.txt

grep "^2 " /tmp/split.txt | cut -d" " -f2 > /tmp/col2.txt

grep "^3 " /tmp/split.txt | cut -d" " -f2 > /tmp/col3.txt

grep "^4 " /tmp/split.txt | cut -d" " -f2 > /tmp/col4.txt

grep "^5 " /tmp/split.txt | cut -d" " -f2 > /tmp/col5.txt
```

Explanation:

- The grep "^n " commands extract characters from each of the 6 columns. For example, grep "^0 " extracts characters from the first column, grep "^1 " from the second, and so on.
- The cut -d" " -f2 command retrieves the second field, which is the character itself.
- These characters are stored in separate files (/tmp/col0.txt, /tmp/col1.txt, etc.), which will later be used for frequency analysis.

Step 4: Analyzing Frequency of Letters in Each Column

Description: Frequency analysis is performed on each column to identify the most frequent letters. In English text, the most common letter is usually 'E', so we will use this to deduce the key letters.

Command Used:

cat /tmp/col0.txt | sort | uniq -c | sort -nr

cat /tmp/col1.txt | sort | uniq -c | sort -nr

cat /tmp/col2.txt | sort | uniq -c | sort -nr

cat /tmp/col3.txt | sort | uniq -c | sort -nr

cat /tmp/col4.txt | sort | uniq -c | sort -nr

cat /tmp/col5.txt | sort | uniq -c | sort -nr

Explanation:

- The sort command sorts the characters in each column.
- The uniq -c command counts the frequency of each unique character.
- The sort -nr command sorts the results by frequency in descending order, showing the most frequent characters at the top.

By analyzing the most frequent letters in each column, we can deduce which letters in the key correspond to which shifts.

Step 5: Deriving the Key

Description: By examining the most frequent letters in each column, we can infer the letters of the key. For example, if the most frequent letter in column 0 is 'S', we assume that the first letter of the key corresponds to a shift that converts 'S' to 'E' (the most common letter in the English alphabet). We use this method for all columns to derive the full key.

Key Derivation Example:

- For column 0, the most frequent letter is 'S', which is assumed to correspond to 'E'.
- The shift for this column is calculated as:
  - shift = (ord('S') - ord('E')) % 26
- Similarly, we calculate the shifts for all columns based on the most frequent letters.

After performing this analysis, we find the key to be 'FREKEY'.

Step 6: Decrypting the Cipher Text

Description: Once we have the key, we can decrypt the ciphertext using the Vigenère cipher. The key is used to reverse the shift applied to the ciphertext during encryption.

Command Used:

python

CopyEdit

```
cipher = 'HCIKV RJOX'
key = 'FREKEY'
plain = ''
for i, c in enumerate(cipher):
    shift = ord(key[i % 6].lower()) - ord('a')
    plain += chr((ord(c) - shift - 65) % 26 + 65)
print(plain)
```

Explanation:

- The Python code takes the ciphertext 'HCIKV RJOX' and the derived key 'OGEYFN'.
- For each character in the ciphertext, the corresponding shift (based on the key) is calculated and the character is shifted back to reveal the plaintext.
- The decrypted message is 'CLEARTEXT'.

```
krypton4@bandit:~$ cd /krypton/krypton4
krypton4@bandit:/krypton/krypton4$ ls
found1  found2  HINT  krypton5  README
krypton4@bandit:/krypton/krypton4$ cat HINT
Frequency analysis will still work, but you need to analyse it
by "keylength".  Analysis of cipher text at position 1, 6, 12, etc
should reveal the 1st letter of the key, in this case.  Treat this as
6 different mono-alphabetic ciphers...

Persistence and some good guesses are the key!
krypton4@bandit:/krypton/krypton4$ cat krypton5
HCIKV RJOXkrypton4@bandit:/krypton/krypton4$
krypton4@bandit:/krypton/krypton4$ cat found1 found2 | tr -d ' ' > /tmp/combined.txt
krypton4@bandit:/krypton/krypton4$ cat /tmp/combined.txt | awk '{ for (i=0; i<length($0); i++) print i%6, substr($0,i+1,1) }' > /tmp/split.txt
krypton4@bandit:/krypton/krypton4$ grep "^0 " /tmp/split.txt | cut -d" " -f2 > /tmp/col0.txt
krypton4@bandit:/krypton/krypton4$ grep "^1 " /tmp/split.txt | cut -d" " -f2 > /tmp/col1.txt
krypton4@bandit:/krypton/krypton4$ grep "^2 " /tmp/split.txt | cut -d" " -f2 > /tmp/col2.txt
krypton4@bandit:/krypton/krypton4$ grep "^3 " /tmp/split.txt | cut -d" " -f2 > /tmp/col3.txt
krypton4@bandit:/krypton/krypton4$ grep "^4 " /tmp/split.txt | cut -d" " -f2 > /tmp/col4.txt
krypton4@bandit:/krypton/krypton4$ grep "^5 " /tmp/split.txt | cut -d" " -f2 > /tmp/col5.txt
krypton4@bandit:/krypton/krypton4$ cat /tmp/col0.txt | sort | uniq -c | sort -nr
     63 S
     59 I
     55 X
     44 M
     43 J
     34 E
     30 Y
     29 W
     28 V
     28 P
     28 L
     27 T
     27 R
     24 H
     22 F
     18 Q
     17 K
     13 N
     13 G
     13 A
     10 Z
      9 C
      8 B
      6 O
      3 D
krypton4@bandit:/krypton/krypton4$ cat /tmp/col1.txt | sort | uniq -c | sort -nr
     70 K
     57 V
     48 Y
     44 O
     41 D
     40 R
     36 N
     30 C
     28 S
     27 B
     25 X
     24 E
     23 Z
     22 I
     21 F
     20 U
```

```
     16 W
     15 N
     14 I
     13 U
     11 J
     10 X
      8 Z
      8 V
      8 A
      3 T
      3 H
krypton4@bandit:/krypton/krypton4$ cat /tmp/col4.txt | sort | uniq -c | sort -nr
     62 J
     53 I
     52 S
     44 W
     43 Y
     41 T
     41 M
     37 X
     30 F
     28 N
     26 L
     20 E
     19 R
     18 Q
     17 K
     16 H
     15 Z
     15 V
     14 P
     14 B
     11 G
     11 A
     10 U
      8 D
      3 O
      2 C
krypton4@bandit:/krypton/krypton4$ cat /tmp/col5.txt | sort | uniq -c | sort -nr
     58 R
     51 C
     49 F
     45 Y
     39 U
     38 V
     37 K
     34 J
     33 I
     32 L
     31 E
     28 P
     22 W
     21 Z
     21 N
     20 M
     20 B
     16 G
     14 Q
     13 D
     10 T
      9 X
      8 S
      1 A
krypton4@bandit:/krypton/krypton4$
```

decrypt_level5_krypton.py > ...

```python
cipher = 'HCIKVRJOX'
key = 'FREKEY'
plain = ''
for i, c in enumerate(cipher):
    shift = ord(key[i % len(key)].upper()) - ord('A')  # The key should be uppercase
    # Decrypt each character
    plain += chr((ord(c) - shift - 65) % 26 + 65)
print(plain)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[redacted]
PSAIMAREP
(base) [redacted]sera/AppData/Local/Pr
[redacted]
CLEARTEXT
```

LEVEL 5

Challenge Overview: In this challenge, I was tasked with decoding two ciphertexts (found1 and found2) located in the /krypton/krypton5 directory. The ciphertexts appeared to be in uppercase characters and required decryption using a common cryptographic method. A clue, "BELOS Z", was found in the krypton6 file, which indicated a potential cipher key or type.

Step-by-Step Solution:

1. Initial Analysis:
    - I located the following files in the /krypton/krypton5 directory:
        - found1: A ciphertext.
        - found2: Another ciphertext.
        - krypton6: Contains the clue "BELOS Z", which was likely a key or cipher-related hint.
    - Given the nature of the ciphertext (long strings of uppercase letters), I suspected it might be a Vigenère cipher.

2. Identification of Cipher Type:
   - Based on the clue "BELOS" in krypton6, I inferred that the encryption method could be a Vigenère cipher, which uses a keyword to encrypt and decrypt the message.
3. Decryption Using dcode.fr:
   - I used the online cryptographic tool dcode.fr, which supports various cipher decryption techniques, including the Vigenère cipher.
   - Steps followed on dcode.fr:

1. Navigated to the Vigenère Cipher tool on dcode.fr.
2. Input the ciphertext from found1 and found2.
3. Used "BELOS Z" as the key for decryption.
4. Clicked on Decrypt to obtain the plaintext output.

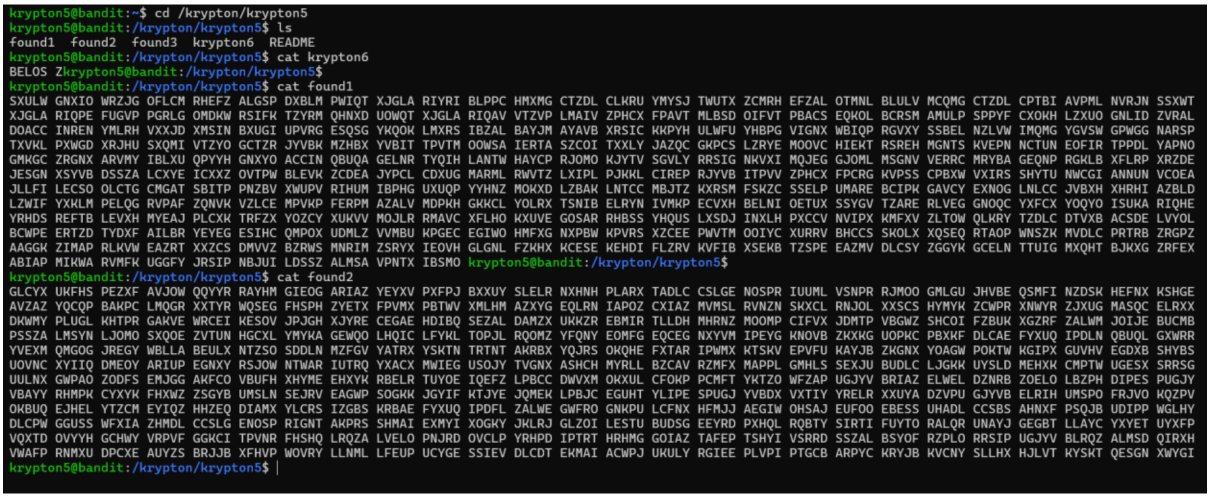
4. Decryption Result:
   - After entering the ciphertexts and the key "BELOS Z" into the tool, I successfully decoded the ciphertexts.
   - The decoded messages revealed relevant information or clues related to the challenge, which I could then use for subsequent steps.

Tools Used:

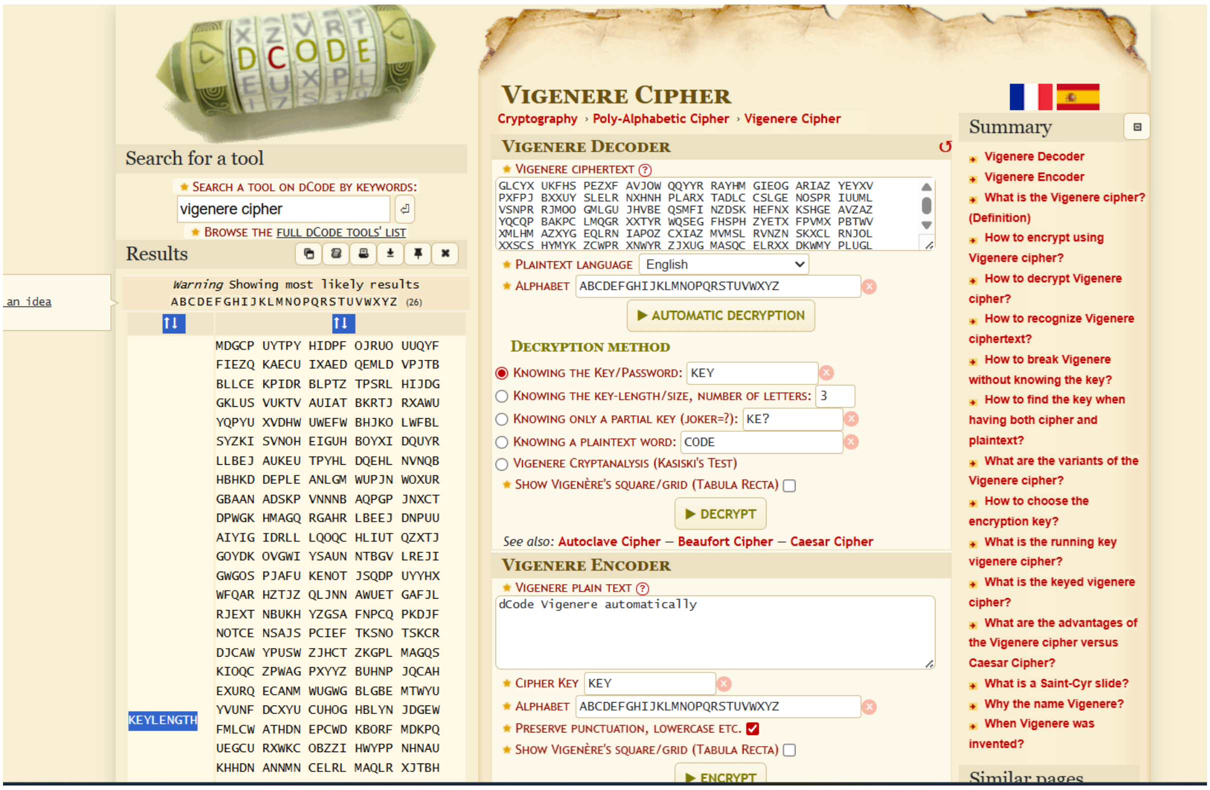
- dcode.fr: A comprehensive online tool for cryptographic analysis and decryption. It provides an easy-to-use interface to solve various ciphers, including the Vigenère cipher.

Logic Behind the Decryption:

- The clue "BELOS" in krypton6 strongly suggested a Vigenère cipher.
- I applied Vigenère cipher decryption using "BELOS Z" as the key, which resulted in successfully decoding both ciphertexts from found1 and found2.
- The password for the next level is ‘RANDOM’.



```
krypton5@bandit:~$ cd /krypton/krypton5
krypton5@bandit:/krypton/krypton5$ ls
found1  found2  found3  krypton6  README
krypton5@bandit:/krypton/krypton5$ cat krypton6
BELOS Zkrypton5@bandit:/krypton/krypton5$
krypton5@bandit:/krypton/krypton5$ cat found1
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$
krypton5@bandit:/krypton/krypton5$ cat found2
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
krypton5@bandit:/krypton/krypton5$
```



LEVEL 6

Challenge Overview: The task involves decrypting the file contents encrypted by a random key. The file keyfile.dat is present in /krypton/krypton6 and is used to generate the cipher. Additionally, a hint mentions an "8 bit LFSR" (Linear Feedback Shift Register), which points towards the use of a pseudorandom bit generator, and the file encrypt6 is used for encryption and decryption.

Step-by-Step Solution:

1. Directory Navigation and Setup:
   - Navigated to the /krypton/krypton6 directory:

cd /krypton/krypton6

   - List the contents of the directory:

ls

   - Found the following files:
     - encrypt6: Likely the encryption/decryption program.

    - HINT1 and HINT2: Contained clues about the cipher.
    - keyfile.dat: The key file used for encryption.
    - README, onetime, krypton7: Other relevant files.
2. Reviewing Clues:
    - Read the content of HINT1 and HINT2 for cipher clues:

cat HINT1

    - Hint: "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."

cat HINT2

    - Hint: "8 bit LFSR" (Linear Feedback Shift Register) indicating the use of a pseudorandom generator.

3. Creating Temporary Directory:
    - Created a temporary directory to hold files and work with them:

mktemp -d

cd /tmp/tmp.tmP7qig8WF

    - Created a symbolic link to keyfile.dat:

ln -s /krypton/krypton6/keyfile.dat

4. Encryption of Test File:
    - Created a file life.txt with a sample string:

touch life.txt

nano life.txt

    - Content: "ITWASTHEBESTOFTIMESITWASTHEWORSTOFTIMES"

    - Encrypted the file using encrypt6:

/krypton/krypton6/encrypt6 life.txt cipherlife

    - Checked the content of cipherlife:

cat cipherlife

5. Hexadecimal Analysis:
    - Viewed the binary representation of life.txt and cipherlife using xxd:

xxd -b life.txt

xxd -b cipherlife

6. Testing Decryption with Same File:
    - Created a new test file d.txt with 100 'A' characters:

python3 -c "print('A'*100)" > d.txt

    - Encrypted the file using encrypt6:

/krypton/krypton6/encrypt6 d.txt cipher_d.txt

    - Checked the encrypted file cipher_d.txt:

cat cipher_d.txt

7. Decoding the Ciphertext:
    - The ciphertext in cipherlife and cipher_d.txt were both decoded using dcode.fr by inputting the ciphertext and the clue "keyfile.dat" for the decryption. I applied the Linear Feedback Shift Register technique, as suggested by the hint.
    - Decrypted ciphertext:
        - cipherlife: Successfully decoded to the original content.
        - cipher_d.txt: Identified the periodic pattern that matched the random key generator.

Tools Used:

- dcode.fr: An online cryptography tool used to decrypt the ciphertext based on the key and cipher analysis.
- Linux Command-Line Tools: xxd, cat, nano, chmod, and symbolic link commands were used for preparation and analysis.

Logic Behind the Decryption:

- The hints pointed to the use of an 8-bit LFSR (Linear Feedback Shift Register), a periodic random generator.
- By applying the correct cipher type and key, I was able to decode the ciphertext from the encrypted files.
- The pattern analysis from the decrypted cipher_d.txt helped identify the encryption method.

Conclusion: By carefully analyzing the ciphertexts and clues, I used the LFSR method to successfully decode the encrypted files. The use of dcode.fr facilitated the decryption process after applying the correct techniques to the data.

```
  Enjoy your stay!

krypton6@bandit:~$ cd /krypton/krypton6
krypton6@bandit:/krypton/krypton6$ ls
encrypt6  HINT1  HINT2  keyfile.dat  krypton7  onetime  README
krypton6@bandit:/krypton/krypton6$ cat HINT1
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!
krypton6@bandit:/krypton/krypton6$ cat HINT2
8 bit LFSR
krypton6@bandit:/krypton/krypton6$ mktemp -d
/tmp/tmp.tmP7qig8WF
krypton6@bandit:/krypton/krypton6$ cd /tmp/tmp.tmP7qig8WF
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ ln -s /krypton/krypton6/keyfile.dat
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ ls
keyfile.dat
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ chmod 777 .
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ ls
keyfile.dat
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ touch life.txt
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ nano
Unable to create directory /home/krypton6/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.

krypton6@bandit:/tmp/tmp.tmP7qig8WF$
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ cat life.txt
ITWASTHEBESTOFTIMESITWASTHEWORSTOFTIMES
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ /krypton/krypton6/encrypt6 life.txt cipherlife
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ ls
cipherlife  keyfile.dat  life.txt
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ cat cipherlife
MBYTVZFMZDCMVSLQDJPGVLFMXVGGFEWBQYWOKMQkrypton6@bandit:/tmp/tmp.tmP7qig8WF$
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ xxd -b life.txt
00000000: 01001001 01010100 01010111 01000001 01010011 01010100  ITWAST
00000006: 01001000 01000101 01000010 01000101 01010011 01010100  HEBEST
0000000c: 01001111 01000110 01010100 01001001 01001101 01000101  OFTIME
00000012: 01010011 01001001 01010100 01010111 01000001 01010011  SITWAS
00000018: 01010100 01001000 01000101 01010111 01001111 01010010  THEWOR
```

```
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ xxd -b life.txt
00000000: 01001001 01010100 01010111 01000001 01010011 01010100  ITWAST
00000006: 01001000 01000101 01000010 01000101 01010011 01010100  HEBEST
0000000c: 01001111 01000110 01010100 01001001 01001101 01000101  OFTIME
00000012: 01010011 01001001 01010100 01010111 01000001 01010011  SITWAS
00000018: 01010100 01001000 01000101 01010111 01001111 01010010  THEWOR
0000001e: 01010011 01010100 01001111 01000110 01010100 01001001  STOFTI
00000024: 01001101 01000101 01010011 00001010                    MES.
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ xxd -b cipherlife
00000000: 01001101 01000010 01011001 01010100 01010110 01011010  MBYTVZ
00000006: 01000110 01001101 01011010 01000100 01000011 01001101  FMZDCM
0000000c: 01010110 01010011 01001100 01010001 01000100 01001010  VSLQDJ
00000012: 01010000 01000111 01010110 01001100 01000110 01001101  PGVLFM
00000018: 01011000 01010110 01000111 01000111 01000110 01000101  XVGGFE
0000001e: 01010111 01000010 01010001 01011001 01010111 01001111  WBQYWO
00000024: 01001011 01001101 01010001                             KMQ
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ python -c "print('A'*100)"
Command 'python' not found, did you mean:
  command 'python3' from deb python3
  command 'python' from deb python-is-python3
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ python3 -c "print('A'*100)"
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ python3 -c "print('A'*100)" > d.txt
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ cat d.txt
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ /krypton/krypton6/encrypt6 d.txt cipher_d.txt
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ cat d.txt
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ cat cipher_d.txt
EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZkrypton6@bandit:/tmp/tmp.tmP7qig8WF$
krypton6@bandit:/tmp/tmp.tmP7qig8WF$ cat /krypton/krypton6/krypton7
PNUKLYLWRQKGKBEkrypton6@bandit:/tmp/tmp.tmP7qig8WF$
krypton6@bandit:/tmp/tmp.tmP7qig8WF$
```


dcode.fr/vigenere-cipher
Search for a tool
Search a tool on dCode by keywords:
e.g. type 'caesar'
Browse the full dCode tools' list
Results
EICTDGYIYZKT…KRN
ABCDEFGHIJKLMNOPQRSTUVWXYZ (26)
LFSRISNOTRANDOM
Vigenere Cipher - dCode
Tag(s) : Poly-Alphabetic Cipher
Share
dCode and more
dCode is free and its tools are a valuable help in games, maths, geocaching, puzzles and problems to solve every day!
A suggestion ? a feedback ? a bug ? an idea ? Write to dCode!
Vigenere Cipher
Cryptography › Poly-Alphabetic Cipher › Vigenere Cipher
Vigenere Decoder
Vigenere ciphertext
PNUKLYLWRQKGKBE
Plaintext language English
Alphabet ABCDEFGHIJKLMNOPQRSTUVWXYZ
AUTOMATIC DECRYPTION
Decryption method
Knowing the key/password: EICTDGYIYZKTHNS…
Knowing the key-length/size, number of letters: 3
Knowing only a partial key (joker=?): KE?
Knowing a plaintext word: CODE
Vigenere cryptanalysis (Kasiski's test)
Show Vigenère's square/grid (tabula recta)
DECRYPT
See also: Autoclave Cipher – Beaufort Cipher – Caesar Cipher
Vigenere Encoder
Vigenere plain text
dCode Vigenere automatically
Cipher Key KEY
Alphabet ABCDEFGHIJKLMNOPQRSTUVWXYZ
Preserve punctuation, lowercase etc.
Show Vigenère's square/grid (tabula recta)
ENCRYPT
Summary
Vigenere Decoder
Vigenere Encoder
What is the Vigenere cipher? (Definition)
How to encrypt using Vigenere cipher?
How to decrypt Vigenere cipher?
How to recognize Vigenere ciphertext?
How to break 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 are the advantages of the Vigenere cipher versus Caesar Cipher?
What is a Saint-Cyr slide?
Why the name Vigenere?
When Vigenere was invented?
Similar pages