README

Name	ID	Sec	BN
Walid Osama Khamees	9213035	2	31
Mohamed Maher Hasan Amin	9213347	2	12
Ali Mohamed Farid	9210684	1	25
Mostafa El-Sayed	9211164	2	20

1. Cryptanalysis

From the histogram of text, it was obvious that the cipher used is a mono-alphabetic cipher, to formulate the key, We used the histogram and the most common one-letter words, two-letter words and three-letter words.

Key is `crljaqphextknfdovigbmsyuwz`

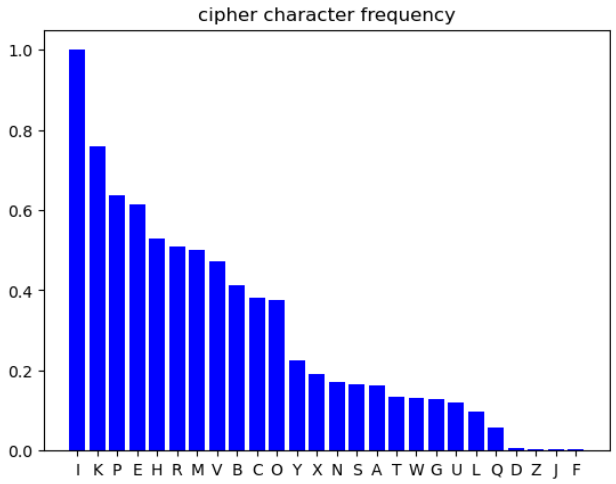
```
# cryptanalysis resource :
https://www3.nd.edu/~busiforc/handouts/cryptography/cryptography%20hints.h
tml
def read_file(file):
  words = []
  with open(file, 'r') as f:
  for line in f:
  if line.strip() and line[0] != '\n':
    for word in line.strip().split(" "):
      words.append(word)
  return words

def get_common_words_by_length(cipher ,i):
  cipher = list(filter(lambda x: len(x) == i, cipher))
  common = {}
  for i in range(len(cipher)):
    common[cipher[i]] = common.get(cipher[i], 0) + 1
    common = dict(sorted(common.items(), key=lambda item: item[1],
reverse=True))
  return common

for i in range(1,5):
  print(get_common_words_by_length(read_file("encrypted_text.txt"), i))
```

Code for getting specific length words

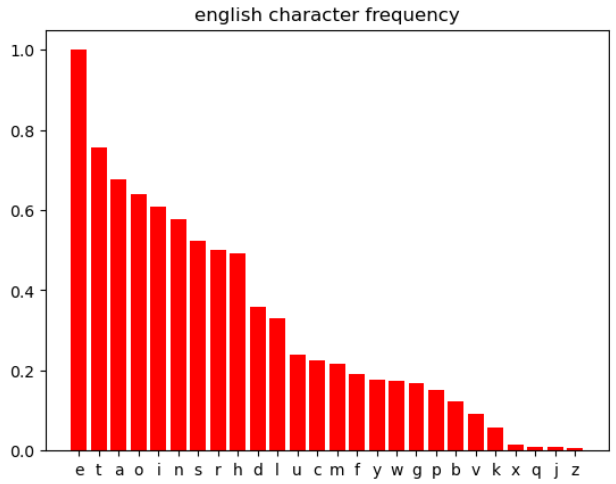
```
english_freq = {
'e': 12.02, 't': 9.10, 'a': 8.12, 'o': 7.68, 'i': 7.31, 'n': 6.95, 's':
6.28, 'r': 6.02, 'h': 5.92, 'd': 4.32, 'l': 3.98, 'u': 2.88, 'c': 2.71,
'm': 2.61, 'f': 2.30, 'y': 2.11, 'w': 2.09, 'g': 2.03, 'p': 1.82, 'b':
1.49, 'v': 1.11, 'k': 0.69, 'x': 0.17, 'q': 0.11, 'j': 0.10, 'z': 0.07
}

def get_cosets(array, offset=0):
  result = []
  for i in range(offset):
    result.append(array[i::offset])
  return result
def read_file_continuous(file):
  words = ""
  lenOfWords = []
  with open(file, 'r') as f:
    for line in f:
      if line.strip() and line[0] =≠ '\n':
        for word in line.strip().split(" "):
          words += word
          lenOfWords.append(len(word))
  return words, lenOfWords

cipher_text, lenOfWords = read_file_continuous("encrypted_text.txt")
i = 1
key = ""
cosets = get_cosets(cipher_text, i)
histograms = []

for coset in cosets:
  hist = {}
  for char in coset:
    hist[char] = hist.get(char, 0) + 1
  histograms.append(hist)

for i, histogram in enumerate(histograms):
  histogram = dict(sorted(histogram.items(), key=lambda item: item[1],
reverse=True))
  plt.figure(i)
  plt.bar(histogram.keys(), [value/max(histogram.values()) for value in
histogram.values()],color = 'b')
  plt.show()

plt.bar(english_freq.keys(), [value/max(english_freq.values()) for value
in english_freq.values()], color = 'r')
plt.show()
```

code for displaying English and cipher text letter frequency

Cipher character frequency



English character frequency



2. Packet Analysis

Opening the packet in wireshark and looking at the protocol hierarchy reveals two packets which are entirely text-based which are as follows

1. none
2. Gur synt vf cvpbPGS{c33xno00_1_f33_h_qrnqorrs}
   the second one looks like the flag, guessing the cvpbPGS equates to picoCTF leaves us with the fact the cipher is Caesar's cipher with a key of 13 then the plain text is `THE FLAG IS PICOCTF{P33KAB00_1_S33_U_DEADBEEF}`

```
def CaesarCipher(text: str, shift: int, encrypt: bool) -> str:
  result = ""
  for i in range(len(text)):
  char = text[i].lower()
  if char == " ":
    result += " "
  else:
    if encrypt:
      result += chr(((ord(char) + shift - 97) % 26) + 97)
    else:
      if ord(char) ⩾ ord('a') and ord(char) ⩽ ord('z'):
        result += chr(((ord(char) - shift - 97) % 26) + 97)
      else:
        result += char
  return result

print(CaesarCipher("Gur synt vf cvpbPGS{c33xno00_1_f33_h_qrnqorrs}", 13,
False).upper())
```

Deciphering Caesar

3. Image Manipulation

We xored the two images and raised the gamma of the resulting image which produced this key `picoCTF{d72ea4af}`
after a while We discovered that adding the two images produce the same effect.


```
image1 = cv2.imread('first.png')
image2 = cv2.imread('second.png')
xor = cv2.bitwise_xor(image1, image2)
plt.imsave('xor.png', xor)
gamma = 100
# Apply gamma correction
corrected_image = np.clip((xor / 255.0) ** gamma * 255.0, 0,
255).astype(np.uint8)
corrected_image[corrected_image < 255] = 0
plt.imsave('output.png', corrected_image)
```

4. Bit Shifting

We first tried individually shifting each byte but that went nowhere, shifting the entire text at once and trying different shift left values we get the following flag
`Hello and welcome to file11 forensic challenge. This is just filler text to make it longer. fastctf{a_bit_tricky}`
the shift value = `1`
shift direction = `left`

5. Search

```
grep "grep" flag.txt
picoCTF{grep_is_good_to_find_things_dba08a45}
```

6. New Encryption

Decipher : `The enemies are making a move. We need to act fast.`
key : `e`

```
def decode_b16(text):
  decode=""
  for i in range(0, len(text), 2):
    text_bin = CHARSET.index(text[i]) * 16 + CHARSET.index(text[i+1])
    decode += chr(text_bin)
  return decode

def inverse_caesar_shift(c, k):
  # c is the output of caesar after getting index of charset so we must
get inverse of charset[] operation with .index
  # - ord(k) standard modular operations
  # + Start is because there is one character that needs an offset (k)
unlike above were there were two (c_orig, k)
  return CHARSET[((CHARSET.index(c) - ord(k) + START) % len(CHARSET))]

# because a the encryption is done by encoding first and ciphering second
the decryption is done in the opposite direction
def decipher(cipher, key):
  inverse = ""
  for i, c in enumerate(cipher):
    inverse += inverse_caesar_shift(c, key[i % len(key)])
  return decode_b16(inverse)

encrypted=""
with open("cipher.txt", "r") as f:
encrypted = f.read()
for key in CHARSET:
  print(decipher(encrypted, key))
```

7. Stenography

Searching lead me to `steghide` which is used for stenography, using it on the picture it asks for a password the document has `HIDING` in all caps trying it reveals the flag `Hello, the flag is CMPN{Spring2024}`.

8. Can You Help Me?

The audio itself is morse code decoding it using online sources reveals this message
`THE RUSSIAN TERRORISTS ARE THE ONES WHO STARTED THIS, THEY ARE THE KEY. PLEASE YOU MUST EXTRACT ME`
doing further analysis on the audio file using `strings` reveals the following
`https://en.wikipedia.org/wiki/Nihilist_cipher?keyword=polybius`
`96 57 47 66 62 38 55 67 55 35 68 44 48 95 66 65 57 65 53 75 78 77 55 36 47 55 45 66 87 34 46 48 33 77`

cipher algorithm → `Nihilist`
keyword → `polybius`
ciphertext → `96 57 47 66 62 38 55 67 55 35 68 44 48 95 66 65 57 65 53 75 78 77 55 36 47 55 45 66 87 34 46 48 33 77`
from the audio file trying the key → `RUSSIAN`
reveals the flag `THANK YOU FOR SAVING ME THE FLAG IS MOSCOW`