lab2

Lab - 2 : Attacking Classic Crypto Systems

Task - 1 : Decrypting Caesar Cipher

The given cipher is : `"odroboewscdrolocdcwkbdmyxdbkmdzvkdpybwyeddrobo"`

The following bruteforce python code is used to decrypt it.

```
def decrypt_caesar_cipher(cipher_text):
    for shift in range(26):
        plain_text = ""        for char in cipher_text:
            if char.isalpha():
                shifted = ord(char) - shift
                if char.islower():
                    if shifted < ord('a'):
                        shifted += 26                elif char.isupper():
                    if shifted < ord('A'):
                        shifted += 26                plain_text += chr(shifted)
            else:
                plain_text += char
        print('Shift #%s: %s' % (shift, plain_text))
message = 'odroboewscdrolocdcwkbdmyxdbkmdzvkdpybwyeddrobo'decrypt
_caesar_cipher(message)
```

Output :

```
Shift #0: odroboewscdrolocdcwkbdmyxdbkmdzvkdpybwyeddrobo
Shift #1: ncqnandvrbcqnknbcbvjaclxwcajlcyujcoxavxdccqnan
.....
Shift #10: ethereumisthebestsmartcontractplatformoutthere
.....
Shift #25: pespcpfxtdespmpdedxlcenzyeclneawleqzcxzfeespcp
```

From all possible 26 shifts, shift 10 is the probable plain text. For shift 26, deciphered text is

`ethereum is the best smart contract platform out there`

Task - 2 : Substitution Cipher

Two cipher text and a frequency distribution of english characters are given. Need to decipher them.

Here, most frequent character in the cipher text will be replaced with the most frequent english character from the distribution table.

- Sort frequency distribution in descending order.
- Find frequency distribution of letters in cipher text and sort in descending or der.
- Map the letters according to frequency distribution to decrypt the text.

Substitution Cipher Breaker :

```
# Checking for frequency distributiondef calculate_char_frequency(text):
    text = text.lower()
    char_counts = collections.Counter(c for c in text if c.isalpha())
    total_alpha_chars = sum(char_counts.values())
    # Calculate percentages    frequencies = [
        (char, count, (count / total_alpha_chars) * 100)
        for char, count in char_counts.items()
    ]
    # Sort by percentage in descending order    frequencies.sort(key=lambda x:
x[2], reverse=True)
    return frequencies
# Calculate frequenciesfreq1_analysis = calculate_char_frequency(CIPHER_1)
freq2_analysis = calculate_char_frequency(CIPHER_2)
freq1_list = [(char, f'{percentage:.2f}%') for char, count, percentage in freq1_a
nalysis]
english_freq_list = [(char, f'{percentage:.2f}%') for char, percentage in SORTE
D_ENGLISH_FREQ]
```

```
# Determine the maximum length for consistent printingmax_len = max(len(fre
q1_list), len(english_freq_list))
for i in range(max_len):
    cipher_char_info = freq1_list[i] if i < len(freq1_list) else (' ', '      ')
    eng_char_info = english_freq_list[i] if i < len(english_freq_list) else (' ', '      ')
    print(f"  '{cipher_char_info[0]}': {cipher_char_info[1].ljust(7)} |   '{eng_char_i
nfo[0]}': {eng_char_info[1]}")
final_key_map_cipher1 = {
    'a': 'i', 'c': 't', 'd': 'o', 'e': 'h', 'f': 'n', 'g': 'd', 'h': 'b', 'i': 'e',
    'j': 'q', 'k': 'r', 'l': 'k', 'm': 'g', 'n': 'l', 'o': 'm', 'p': 'a', 'q': 'c',
    'r': 's', 's': 'j', 't': 'w', 'u': 'f', 'v': 'u', 'w': 'y', 'x': 'p'}
print("\n--- Final Key Map for Cipher-1 ---")
for k, v in sorted(final_key_map_cipher1.items()):
    print(f"'{k}' → '{v}'")
#------------------------------------------------------------## Similarly repeat th
ose step for the second cipher text
```

First cipher text :

```
af p xpkcaqvnpk pfg, af ipqe qpri, gauuikifc tpw, ceiri udvk tiki afgarxifrphni c
d eao--wvmd popkwn, hiqpvri du ear jvaql vfgikrcpfgafm du cei xkafqaxnir du
xrwqedearcdkw pfg du ear aopmafpcasi xkdhafmr afcd fit pkipr. ac tpr qdoud
kcafm cd lfdt cepc au pfwceafm epxxifig cd ringdf eaorinu hiudki cei opceiop
caqr du cei uaing qdvng hi qdoxnicinw tdklig dvc--pfg edt rndtnw ac xkdqiigi
g, pfg edt odvfcpafdvr cei dhrcpqnir--ceiki tdvng pc niprc kiopaf dfi mddg oaf
g cepc tdvng qdfcafvi cei kiripkqe
```

Output :

```
--- Final Key Map for Cipher-1 ---
'a' → 'i'
'c' → 't'
'd' → 'o'
'e' → 'h'
'f' → 'n'
```

'g' → 'd'
'h' → 'b'
'i' → 'e'
'j' → 'q'
'k' → 'r'
'l' → 'k'
'm' → 'g'
'n' → 'l'
'o' → 'm'
'p' → 'a'
'q' → 'c'
'r' → 's'
's' → 'j'
't' → 'w'
'u' → 'f'
'v' → 'u'
'w' → 'y'
'x' → 'p'

Deciphered Text : in a particular and, in each case, different way, these four w ere indispensable to him--yugo amaryl, because of his quick understanding of the principles of psychohistory and of his imaginatije probings into new areas. it was comforting to know that if anything happened to seldon himself before t he mathematics of the field could be completely worked out--and how slowly it proceeded, and how mountainous the obstacles--there would at least remai n one good mind that would continue the research

Second cipher text :

"aceah toz puvg vcdl omj puvg yudqecov, omj loj auum klu thmjuv hs klu zlcv u shv zcbkg guovz, upuv zcmdu lcz vuwovroaeu jczoyyuovomdu omj qmubyu dkuj vukqvm. klu vcdluz lu loj avhqnlk aodr svhw lcz kvopuez loj mht audhwu o ehdoe eunumj, omj ck toz yhyqeoveg auecupuj, tlokupuv klu hej sher wcnlk zog, klok klu lcee ok aon umj toz sqee hs kqmmuez zkqssuj tckl kvuozqvu. om j cs klok toz mhk umhqnl shv sowu, kluvu toz oezh lcz yvhehmnuj pcnhqv kh wovpue ok. kcwu thvu hm, aqk ck zuuwuj kh lopu eckkeu ussudk hm wv. aon

ncmz. ok mcmukg lu toz wqdl klu zowu oz ok scskg. ok mcmukg-mcmu klug a unom kh doee lcw tuee-yvuzuvpuj; aqk qmdlomnuj thqej lopu auum muovuv k lu wovr. kluvu tuvu zhwu klok zlhhr klucv luojz omj klhqnlk klcz toz khh wqdl h s o nhhj klcmn; ck zuuwuj qmsocv klokomghmu zlhqej yhzzuzz (oyyovumkeg) yuvyukqoe ghqkl oz tuee oz (vuyqkujeg) cmubloqzkcaeu tuoekl. ck tcee lopu kh au yocj shv, klug zocj. ck czm'k mokqvoe, omj kvhqaeu tcee dhwu hs ck! a qk zh sov kvhqaeu loj mhk dhwu; omj oz wv. aonncmz toz numuvhqz tckl lcz whmug, whzk yuhyeu tuvu tceecmn kh shvncpu lcw lcz hjjckcuz omj lcz nhhj shvkqmu. lu vuwocmuj hm pczckcmn kuvwz tckl lcz vueokcpuz (ubduyk, hs d hqvzu, klu zodrpceeu- aonncmzuz), omj lu loj womg juphkuj ojwcvuvz owhmn klu lhaackz hs yhhv omj qmcwyhvkomk sowcecuz. aqk lu loj mh dehzu svcum jz, qmkce zhwu hs lcz ghqmnuv dhqzcmz aunom kh nvht qy. klu uejuzk hs klu zu, omj aceah'z sophqvcku, toz ghqmn svhjh aonncmz. tlum aceah toz mcmu kg-mcmu lu ojhykuj svhjh oz lcz lucv, omj avhqnlk lcw kh ecpu ok aon umj; o mj klu lhyuz hs klu zodrpceeu- aonncmzuz tuvu scmoeeg jozluj. aceah omj sv hjh loyyumuj kh lopu klu zowu acvkljog, zuykuwauv 22mj. ghq loj aukkuv dhw u omj ecpu luvu, svhjh wg eoj, zocj aceah hmu jog; omj klum tu dom dueuavo ku hqv acvkljog-yovkcuz dhwshvkoaeg khnukluv. ok klok kcwu svhjh toz zkce e cm lcz ktuumz, oz klu lhaackz doeeuj klu cvvuzyhmzcaeu ktumkcuz auktuu m dlcejlhhj omj dhwcmn hs onu ok klcvkg-klvuu"

Output :

```
--- Final Key Map for Cipher-2 ---
'a' → 'b'
'b' → 'x'
'c' → 'i'
'd' → 'c'
'e' → 'l'
'g' → 'y'
'h' → 'o'
'i' → 'j'
'j' → 'd'
'k' → 't'
'l' → 'h'
'm' → 'n'
```

'n' → 'g'
'o' → 'a'
'p' → 'v'
'q' → 'u'
'r' → 'k'
's' → 'f'
't' → 'w'
'u' → 'e'
'v' → 'r'
'w' → 'm'
'y' → 'p'
'z' → 's'

Deciphered Text : bilbo was very rich and very peculiar, and had been the wo nder of the shire for sixty years, ever since his remarkable disappearance and unexpected return. the riches he had brought back from his travels had now b ecome a local legend, and it was popularly believed, whatever the old folk mig ht say, that the hill at bag end was full of tunnels stuffed with treasure. and if t hat was not enough for fame, there was also his prolonged vigour to marvel a t. time wore on, but it seemed to have little effect on mr. baggins. at ninety he was much the same as at fifty. at ninety-nine they began to call him well-pres erved; but unchanged would have been nearer the mark. there were some tha t shook their heads and thought this was too much of a good thing; it seemed unfair that anyone should possess (apparently) perpetual youth as well as (re putedly) inexhaustible wealth. it will have to be paid for, they said. it isn't natur al, and trouble will come of it! but so far trouble had not come; and as mr. bag gins was generous with jis money, most people were willing to forgive him jis oddities and jis good fortune. he remained on visiting terms with jis relatives (except, of course, the sackville- bagginses), and he had many devoted admir ers among the hobbits of poor and unimportant families. but he had no close f riends, until some of jis younger cousins began to grow up. the eldest of thes e, and bilbo's favourite, was young frodo baggins. when bilbo was ninety-nine he adopted frodo as jis heir, and brought jim to live at bag end; and the hopes of the sackville- bagginses were finally dashed. bilbo and frodo happened to have the same birthday, september 22nd. you had better come and live here, f rodo my lad, said bilbo one day; and then we can celebrate our birthday-parti

```
es comfortably together. at that time frodo was still in jis tweens, as the hobbit
s called the irresponsible twenties between childhood and coming of age at th
irty-three
```

Observation:

The **second cipher** is easier to break.

Why (short reasons):
- It contains many **short repeated words** (e.g. `omj`, `klu`, `toz`, `ok`, `ck`) that map cleanly to high-frequency function words like **“the”, “and”, “to”, “of”, “be”**, which gives strong starting points for substitution.
- There are **recognizable word patterns and endings** (repeated suffixes, apostrophe forms like `czm'k`) that suggest common English forms (`n't`, `'s`, etc.), helping rapid pattern matching.
- The text has **clear sentence structure and punctuation**, providing context to confirm guesses (capitalization, commas, parentheses, numbers like `22mj`), so trial mappings can be validated against many occurrences.
- Frequency ranks in cipher-2 more closely match English letter-frequency order, so a frequency-analysis initial key yields readable fragments quickly.

Why cipher-1 is harder:
- It has many **longer, less-repetitive tokens** and hyphenated compounds that obscure common function-word positions.
- Fewer obvious short-word anchors and more varied letter clusters make frequency-based guesses less reliable.
- More noise for contextual confirmation, so manual refinement after a frequency guess is much harder.

Conclusion: Cipher-2 is more likely to be broken by simple frequency analysis + pattern matching; Cipher-1 is harder and would require more advanced techniques or more ciphertext/context.