Solution Report 1

Modern Cryptology (CS641)

Computer Science and Engineering

Level 1 + Level 2

Team: **team58**
Ayush Bansal (160177)
Gunjan Jalori (170283)
Siddharth Nohria (160686)

Date: January 22, 2020

1 Chapter 1 (The Entry)

There are 5 sub-levels in the chapter, first 4 of these don't have any cipher which needs to be decrypted.

The last sub-level is a **Substitution Cipher**, the answer to - "how it was recognised and solved" is explained in the subsection after the following list of commands.

Below is the solution to each of the sub-levels:

1. go
2. read
3. enter
4. read
5. cyLe70Lecy

1.1 Substitution Cipher

The ciphertext given was:

```
Nwy dejp pmcplpz cdp sxlrc adegipl ws cdp aejpr. Er nwy aem rpp cdplp xr mwcdxmv ws xmcplprc
xm cdp adegipl. Rwgp ws cdp qecpl adegiplr fxqq ip gwlp xmcplprcxmv cdem cdxr wmp, x
eg rplxwyr. Cdp awzp yrpz swl cdxr gprrevp xr e rxgbqp ryircxcycxwm axbdpl xm fdxad zxvxcr
dejp ippm rdxscpz in 2 bqeapr. Swl cdxr lwymz berrfwlz xr vxjpm ipqwf, fxcdwyc cdp hywcpr.
```

For identifying what kind of cipher is applied in the above text, we will use the **Index of Coincidence**.

The detailed explanation on *Index of Coincidence* can be found in section 3.1.

The *Index of Coincidence* of the above ciphertext is about 0.07, which is approximately same as a valid English text, this suggests that the cipher used is *Mono-alphabetic* such as *Substitution Cipher*.

For Solving the *Substitution Cipher*, the following steps were employed:

1. Calculate the frequency of each of the characters in the ciphertext, ignoring anything which is not an english alphabet.
2. The Character with the highest frequency is most probably 'e' or 'a', which can be placed in its place and identified further.
3. As the places get revealed, played hangman to find out what the other characters might be looking at one-letter, 2-letter, 3-letter words with highest number of characters revealed.
4. Built the decryption key by keeping a map of characters as they are being replaced.
5. Finally used the decryption key to decrypt the code given for the solution.

The code used in this part is in the file - `break_substitution.py`.

The Steps employed in the hangman game and building the key are mentioned below:

```
key = {}
key['p'] = 'e'    # Because 'p' has very high frequency
key['r'] = 's'    # _ee word exists, matches with "see"
key['i'] = 'b'    # _e word exists, matches with "be"
key['n'] = 'y'    # b_ word exists, matches with "by"
key['m'] = 'n'    # bee_ word exists, matches with "been"
key['w'] = 'o'    # _ne word exists, matches with "one"
key['s'] = 'f'    # o_ word exists, 'n' is already taken, matches with "of"
```

```
key['l'] = 'r'    # fo_ word exists, matches with "for"
key['y'] = 'u'    # yo_ word exists, matches with "you"
key['g'] = 'm'    # so_e word exists, matches with "some"
key['z'] = 'd'    # use_ word exists, 'r' is already taken, matches with "used"
key['c'] = 't'    # en_ered word exists, matches with "entered"
key['d'] = 'h'    # t_e word exists, matches with "the"
key['x'] = 'i'    # f_rst word and _ (single letter word) exist, matches with "first" and "i"
key['e'] = 'a'    # single letter word exists, 'i' is already taken, matches with "a"
key['j'] = 'v'    # ha_e word exists, matches with "have"
key['a'] = 'c'    # _hamber word exists, matches with "chamber"
key['v'] = 'g'    # nothin_ word exists, matches with "nothing"
key['f'] = 'w'    # _hich word exists, matches with "which"
key['q'] = 'l'    # be_ow and wi__ word exists, matches with "below" and "will"
key['b'] = 'p'    # sim_le and ci_her word exists, matches with "simple" and "cipher"
key['h'] = 'q'    # _uotes word exists, matches with "quotes"
```

The plaintext revealed after using the above decryption key is:

```
You have entered the first chamber of the caves. As you can see there is nothing of interest
in the chamber. Some of the later chambers will be more interesting than this one, i
am serious. The code used for this message is a simple substitution cipher in which digits
have been shifted by 2 places. For this round password is given below, without the quotes.
```

```
# For the case of integer digits, "1" must be subtracted from each digit, as mentioned
# text after decryption, it was "2" but it itself was shifted so
# x+x = 2, this gives x = 1
```

So, final plaintext is:

```
You have entered the first chamber of the caves. As you can see there is nothing of interest
in the chamber. Some of the later chambers will be more interesting than this one, i
am serious. The code used for this message is a simple substitution cipher in which digits
have been shifted by 1 places. For this round password is given below, without the quotes.
```

Using the above decryption key and the logic for digit, we can decipher the code for the answer as well:

Code: `anQp81Qpan`
Solution: `cyLe70Lecy`

2 Chapter 2 (The Caveman)

There are 2 sub-levels in the chapter, first one doesn't have any cipher which needs to be decrypted.

The second sub-level is a **Vigenere Cipher**, the answer to - "how it was recognised and solved" is explained in the subsection after the following list of commands.

The detailed explanation on *Vigenere Cipher* can be found in section 3.2.

Below is the solution to each of the sub-levels:

1. read
2. the_cave_man_be_pleased

2.1 Vigenere Cipher

The ciphertext given was:

```
Lg ccud qh urg tgay ejbwdkt, wmgtf su bgud nkudnk lrd vjfbg. Yrhfm qvd vng sfuuxytj
"vkj_ecwo_ogp_ej_rnfkukf" wt iq urtuwjm. Ocz iqa jdag vio uzthsivi pqx vkj pgyd encpggt.
Uy hopg yjg fhkz arz hkscv ckoq pgfn vu wwygt nkioe zttft djkth.
```

For identifying what kind of cipher is applied in the above text, we will use the **Index of Coincidence**.

The detailed explanation on *Index of Coincidence* can be found in section 3.1.

The *Index of Coincidence* of the above ciphertext is about 0.042, which is closer to the uniform distribution of English text, this suggests that the cipher is *Poly-alphabetic* such as *Vigenere Cipher*, it may be some other Poly-alphabetic cipher as well but we still have to give it a shot.

For solving the *Vigenere Cipher*, the following steps were employed:

1. Remove all characters from the text which are not part of the English alphabets and capitilize all characters.
2. Partition the text according to different key lengths and sort them according to the *Index of Coincidences* achieved, since higher the IC, closer it is to valid English Text.
3. For each keylen, perform frequency analysis to get the best key possible with the given length.
4. Try out all the keys retrieved and see which one gives some valid English text.

The code used in this part is in the file - `break_vigenere.py`.

The plaintext revealed after using the above decryption key is:

```
Be wary of the next chamber, there is very little joy there. Speak out the password
"the_cave_man_be_pleased" to go through. May you have the strength for the next chamber.
To find the exit you first will need to utter magic words there.
```

From the above, the solution is revealed: `the_cave_man_be_pleased`.

3 Appendix

This section explains each of the things used in between the solutions without proper explanation.

3.1 Index of Coincidence

The **Index of Coincidence** is a measure of how similar a frequency distribution is to the uniform distribution.

$$I.C. = \frac{\sum_{i=A}^{i=Z} f_i(f_i - 1)}{N(N-1)}$$

where f_i is the count of letter i (where $i = A, B, ..., Z$) in the ciphertext, and N is the total number of letters in the ciphertext.

Important facts about the *Index of Coincidence*:

- The *Index of Coincidence* of valid English text is about 0.07.
- The *Index of Coincidence* for uniform distribution of English text is about 0.038.
- The *Index of Coincidence* remains the same for the ciphertext and plaintext if cipher is **Monoalphabetic** (i.e. Substitution Cipher).
- The *Index of Coincidence* of ciphertext is closer to uniform distribution if cipher is **Polyalphabetic** (such as Vigenere Cipher).

We can get an approximate idea of what kind of cipher is used to generate the ciphertext by using the *Index of Coincidence*.

3.2 Vigenere Cipher

The Vigenere Cipher is a polyalphabetic substitution cipher.

Suppose, the length of the encryption key is k, then the string formed by picking out each letter with a multiple of k letters in between them will be a *Caesar Cipher*.

Since each such string is a *Caesar Cipher*, the *Index of Coincidence* of this string will be closer to that of valid English text rather than closer to uniform distribution.

Using the above principle, we can crack the *Vigenere Cipher*.