CS641 - Modern CryptoGraphy
Assignment 2

Shashvat Singham
(200922)

On entering the command read, we find the cipher-text for this level. When we press go on the same page, we see a weird pattern and a message which will be significant later.

The cipher-text again consisted of English alphabets meaning that the cipher for this level is yet another classic alphabet-to-alphabet translation. For a start we ran our Caesar and Substitution cipher decryption code from the previous level on the cipher-text. This didn't yield any result as expected. Then we guessed that it may be a permutation-substitution cipher and modified our code from previous level to break it. However, this didn't work and then we tried another cipher. We considered Vigenere cipher this time and it worked.

1 Permutation-Substitution Cipher

We extended our substitution cipher breaking code from previous level for this. To break this we tried all possible permutation keys and for each permutation we passed the permuted cipher-text to our substitution cipher solver routine. We needed the block size for permutation. The length of cipher-text after all non-alphabetic characters was 185. So, possible block lengths were 5, 37 or 185. Since 37 and 185 size blocks were too large to be brute-forced in reasonable time, we chose the block size to be 5 and tried all 120 permutations. Since for an incorrect permutation, the substitution routine is not expected to terminate, we put an upper limit on the number of restarts (substitution routine is detailed in write-up for chapter 1). This code didn't result in a successful decryption and since other possible block-sizes were too large, we decided to try another cipher. The code for permutation-substitution is sub_perm_decrypt.py, provided in submission package.

2 Vigenere Cipher

We next moved on to Vigenere cipher. Security of Vigenere cipher is greatly compromised, if we can guess a word in the cipher-text. Based on the plain-text from the previous we guessed that the word "password" must occur somewhere in the message to refer to the answer of this round. Now, there were only two 8-letter words in the cipher-text: "sfuuxytj" and "uzthsivi". The proximity of "sfuuxytj" to the quoted text in the cipher-text strongly suggested that this is the encrypted form of password. Using this we can derive partial Vigenere key (assuming it really is Vigenere). The partial key is: [3, 5, 2, 2, 1, 10, 2, 6]

Now we iterate over all possible key length from 8 (from the partial-key derived from password we can see that the key size is at minimum 8) to higher. The key under consideration must have the guessed part-key as it's part. For example, assume that the part key was [1,2,3] and as part of enumerating all possible keys we want to extend it with [4,5], then [1, 2, 3, 4 ,5] [4, 1, 2, 3, 5] [3, 4, 5, 1, 2] are valid as [1, 2, 3] is part of extended key when the extended key when the key is circularly wrapped. But [1, 4, 2, 3, 5] is invalid as [1, 2, 3] doesn't appear even when wrapped.

Considering the above rules for extension, our code extend the key to length 9, 10, 11, and so on. For each key it decrypts using that key and check if the decrypted text is valid English using dictionary matching and considering the fraction of words found in the dictionary. When shifting we left non-alphabets as shifting them can result in weird or even non-printable characters.

On running the code, we found that a suitable decryption was indeed found and that the key was of length 9. The key that we found was: [10, 2, 6, 2, 3, 5, 2, 2, 1] and the password was "the_cave_man_be_pleased".

Interestingly, the patterns we found in the the home screen on pressing ’go’ was the key for Vigenere cipher. The number of characters in each line of the image when counted from bottom-to-up was [10, 2, 6, 2, 3, 5, 2, 2, 1], exactly equal to the key our code found.

3 Code

The submission package has the code we wrote.

decrypt.py : Substitution cipher decoder from previous level.
sub_perm_decrypt.py: Permutation-Substitution Cipher decoder. Uses substitution routine from decrypt.py
vigenere.py: Vigenere Cipher decoder.

All the codes are well commented and implements the algorithms described above.