NSC : Assignment 1

Project 0 :

Encryption & decryption using monoalphabetic substitution of a pair of characters at a time

Submitted by
Naman Kaushik 2020088
Yashika Singh 2020161

How to use the system

Step 1 : In Helper.py, We can hardcode the strings for which we need to perform the encryption, We need to run helper.py, this forms a file Plaintext.txt which contains the original strings + hash value

Step 2 : Run the file Encrypt.py, the key is hardcoded in the file.

Step 3 : Run the file Decrypt.py it displays the decoded strings in AfterDecryption.txt.

Step 4 : Run the program BruteForce.py, It uses all the possible keys in order to find the one that decodes our set of strings.

- using the key

How it Works

First we encode the strings using the key, Key shows mapping for all permutations of pq where p,q belongs to the set {“A”,”B”,”C”}. We iterate through the string two characters at a time and at the end if there are odd no. of characters we add “_”, We similarly encrypt the hash value. Hash and plaintext are separated by a “.”

Then using reverse key we again map the characters of the encrypted string which gives us the original value.

“Recognizable” Plain Text

Plaintext consists of Original Value and Hash of the Original Value seperated by a “.” If the Hash matches the corresponding original value then the text is considered recognizable.

Hash is calculated in the following way:

We start a value variable from 0 the keep on adding 0,1,2 for each occurence of ‘A’, ‘B’ or C respectively. After each character value is multiplied by three.

Then to this we append the No. of times A, B, C occur respectively.

So for A hash would be “0”+”1”+”0”+”0” = “0100”

for AB hash would be “1” + “1” + “1” + “0” = “1110”

Now they are mapped to their character encoding defined in the Helper.py file

So for A hash is AAAAA and AA hash is AAAAAAA as 0 is mapped to A and 1 is mapped to AA

Brute Force

To find the key:

- We first generated all the possible keys that are possible with the given character set and constraints
- Now using the keys one by one (the reverse dictionary mapping) we decrypted the two parts of the cipher text individually (by two parts, we mean the text before and after the dot)
- Now to check if the decrypted text is indeed the correct plaintext, we used our self-defined function called ‘is_recognisable’ which hashes our first part of the deciphered text (the text before the dot) and compares it with the text after the dot
- If the two are found to be the same, it implies that the deciphered text is indeed right and the key is right.
- We now use this key for the remaining ciphertexts, if all the deciphered texts come out to be recognisable, it implies that the key we got it correct
- Hence the brute force attack would have succeeded.