(Team details can be found at the end of the assignment)

Cipher Method: *Vignere Cipher*

Passcode: *the_cave_man_be_pleased*

Justification:

i. At the first screen we notice some funny patterns, thus ***read*** was the command.
ii. First try we made was to check for substitution cipher. We checked the frequency of individual letters and didn't notice any similarity to the standard pattern of frequency of letters, so it wasn't substitution.
iii. We knew there are some cipher that work on frequency of bigrams and trigrams rather than individual letters. So, we checked the frequency of bigrams and trigrams. First, we thought it was *'Playfair Cipher'* but then *Playfair* has a property that either i or j appears in ciphertext not both but then our ciphertext has so it wasn't *Playfair*.
iv. Second try we made is for '*Vignere Cipher'*. For *Vignere* first we need to determine the length of key. We noticed ***'vkj'*** was repeating in the cipher text at a gap of 54. So, all factors of 54 are possible candidate for key length.
v. We created a C++ code (**KeyGen.cpp**) which takes cipher text and key length and gives us the possible key.
vi. We iterated over all possible key length on our program. After we got the key, we converted the cipher text to plain text and if generated plain text makes sense then current iteration is our key length and generated key is our required key.
vii. After the above step we find length of key is 9 and the key found was *[k c g c d f c c b] where index of a is 0*.

Explanation for KeyGen.cpp:

Input: key length

Output: key

Algorithm:

1. We divided the cipher text into blocks of key length each.
2. For every element of the key we fetched the corresponding letter from each block of cipher text and computed frequencies.
3. Now, we try to make the best possible matching of the computed frequencies and the standard frequencies of the letters. We do this by performing the sum of products of standard frequencies and computed frequencies. After that we shift computed frequency by one place and compute sum of products again.
4. We do this for 25 shifts and record the number of shifts which give maximum sum of products. This number of shifts is our key element.
5. After we do above steps for every key element, we get our required key.

Team Details:

- Ashish Pal (18111010)
- Darshit Vakil (18111013)
- Mayank Rawat (18111040)