Cryptography and Network Security

Name: Piyush Mhaske Batch: B3
PRN : 2019BTECS00089

Assignment 2

Objective:

Cryptanalysis of Caesar Cipher

Theory :

It is simply a brute force approach towards finding out the plain text using cipher text. Here checking of plain test from all the 26 combination of the cipher text using the all the shift is done.

In Cryptanalysis , the objective of attacking an encryption system is to recover the key in use rather than simply to recover the plaintext of a single charachter

1) The Cryptanalysis :

Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

2) Brute Force Attack:

The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

```
//code by :- Piyush Mhaske
// using key and find out the key
#include <bits/stdc++.h>
#include <fstream>
#include <cstdlib>
```

```
#define ll long long
#define ul unsigned long long
#define pb emplace_back
#define po pop_back
#define vi vector<ll>
#define vii vector<vector<ll>>
using namespace std;
void file(){
     ios_base::sync_with_stdio(false);
     cin.tie(NULL);}
ll M = 1e9 + 7;
string caeserCipherEnc(string input, int shift){

    for(int i=0;i<input.size();i++){
        char num = input[i] - 'a';

        input[i] = 'a' + (num + shift)%26;
    }
   return input;
}
string caeserCipherDecWithShift(string output, int shift){
    vector<string> ans;
        string temp;
        for(int j=0;j<output.size();j++){
            char num = output[j] - 'a';
            temp += 'a' + (num + (26-shift))%26;
        }

        // for(auto x:dictionary){
        //     if(x==temp){
        //         return temp;
        //     }
        // }
        // cout<<temp;

   return temp;
}
vector<string> caeserCipherDec(string output){
    vector<string> ans;
    for(int i=1;i<26;i++){
        string temp;
        for(int j=0;j<output.size();j++){
            char num = output[j] - 'a';
            temp += 'a' + (num + (26-i))%26;
        }
        ans.push_back(temp);
    }
   return ans;
}
int main()
{   file();
    string input;
    cin>>input;
```

```
    int shift;
    cin>>shift;

    // encrypt
    string ans = caeserCipherEnc(input,shift);
    cout<<"Encryption of above input: ";
    cout<<ans<<"\n";

    // decrypt
    string dec = caeserCipherDecWithShift(ans,shift);
    cout<<"Decryption of above input: ";
   cout<<dec<<"\n";

    // crack the key
    vector<string> arr = caeserCipherDec(ans);
    cout<<"Decryption of above input: ";
    int i=1;
    for(auto x:arr){
        cout<<"for shift"<<i<<" :";
        cout<<x<<"\n";
        i++;
    }

    return 0;
}
```

Crack the Code

```
import enchant
from numpy.core.defchararray import lower

dic = enchant.Dict("en-US")

#function to decrypt
def decrypt(text, shift):
    cipher = ""
    text = lower(text)
    text = str(text)
    for c in text:
        # if c == 32:
        #     continue
        t = (ord(c) - shift - 65) % 26;
        if t<0:
            t = 26 + t
        cipher += chr(t % 26 + 65)
    #print(cipher)
    return cipher
```

```
#main body

cipheredText =  input("Enter secret message: ")

shift = 26

for i in range(shift):
    plainText = decrypt(cipheredText,i)
    if dic.check(plainText):
         print(plainText)
         exit()
print("Cannot Crack the code")
```

Output:

input :

```
abcdef
7
```

Output:

Conclusion: Cryptanalysis helps to find out the key from cipher text

```
Encryption of above input: hijklm
Decryption of above input: abcdef
Decryption of above input: for shift1 :ghijkl
for shift2 :fghijk
for shift3 :efghij
for shift4 :defghi
for shift5 :cdefgh
for shift6 :bcdefg
for shift7 :abcdef
for shift8 :zabcde
for shift9 :yzabcd
for shift10 :xyzabc
for shift11 :wxyzab
for shift12 :vwxyza
for shift13 :uvwxyz
```

```
for shift14 :tuvwxy
for shift15 :stuvwx
for shift16 :rstuvw
for shift17 :qrstuv
for shift18 :pqrstu
for shift19 :opqrst
for shift20 :nopqrs
for shift21 :mnopqr
for shift22 :lmnopq
for shift23 :klmnop
for shift24 :jklmno
for shift25 :ijklmn
```