Homework 2, 550.371 Cryptology and Coding, Spring 2017

Important Instructions: You may discuss this homework (including MATLAB questions) with students currently in the class, with the TAs, and with me, up until the time that you do your write-up; but the solutions and code that you submit should be entirely your own. At no time may you consult any existing written solutions; this would be in violation of the homework rules and, in addition, would be plagiarism if sources are not cited.

Problem 1: First, recall: In DES, Alice computes the ciphertext $c \in \mathbf{Z}_2^{64}$ from the plaintext $m \in \mathbf{Z}_2^{64}$ by evaluating $c := \mathrm{DES}_K(m)$, where $K \in \mathbf{Z}_2^{64}$ is the DES key, and Bob computes m from c by evaluating $m = \mathrm{DES}_{\overline{K}}(c)$, where $\overline{K}$ just reverses the order of the round keys in K. Assume that Eve has a very powerful and fast computer and, if she discovers a known plaintext m and its corresponding ciphertext c, then she can try all of the 2^{64} DES keys in $\mathbf{Z}_2^{64}$ until she finds the right one K that encrypts $c = \mathrm{DES}_K(m)$ and, after finding the key K, Eve can then decrypt all subsequent ciphertexts using K.

a) Knowing this, Alice and Bob agree on two keys $K, L \in \mathbf{Z}_2^{64}$ and they agree that Alice will instead encrypt $c := \mathrm{DES}_K(m) + L$, where + denotes mod 2 addition. Their thinking is that Eve would not be able to try all $2^{64} \cdot 2^{64} = 2^{128}$ possible pairs of keys to make a successful known-plaintext attack. Show how, with two known-plaintexts and their corresponding ciphertexts, Eve can indeed find the pair of keys.

b) Same as part a, except that the encryption is $c := \mathrm{DES}_K(m + L)$.

Problem 2: Write a MATLAB function that decrypts any Hill ciphertext using a snippet from the plaintext. The snippet will be the beginning characters from the plaintext. Assume you are given the blocklength r. Try your code on the four examples in the accompanying file named "hillciphertexts.m"

The name of your MATLAB function should be "crackhill.m" and the first line should read:

```
function [plaintext]=crackhill(snippetplaintext,ciphertext,blocklength)
```

(Note that your MATLAB program may have to try several different collections of r^2 characters from the snippet before your code can successfully continue on its way to computing the whole plaintext.) Submit a hard copy of MATLAB function crackhill.m and a hard copy of a diary in which you use crackhill.m interactively in MATLAB to decipher the four examples in hillciphertexts.m.

Problem 3: Download and save the accompanying file "substitutioncipherexample.mat" on your computer, then open MATLAB, then type the command "load substitutioncipherexample", then type "ciphertext". On your screen is a ciphertext which I encrypted with a substitution cipher; ie, I chose a bijective function f (which I am not showing you) from the English alphabet to itself, and I replaced each plaintext letter x with ciphertext letter $f(x)$. Find f using frequency analysis. You should hand in f as well as a very detailed description of the specific steps you took to find f. MATLAB may be used, but you need not hand in any MATLAB code, diary, or output.

Next are hints and useful tools for Problem 3:

1) The relative frequency of letters a,b,c,d, etc in English can be gotten by typing in the command "englishrelfreq" after you typed in "load substitutioncipherexample" in MATLAB.

2) In a previous set of MATLAB functions that I gave you was a function called zfrequency.m; if you type the command "zfrequency(ciphertext)" then MATLAB will return how many times there are a,b,c,d, etc in the ciphertext. For the plaintext used in Problem 3, the most frequent letter in English is the most frequent letter in the plaintext, the second most frequent letter in English is the second most frequent letter in the plaintext, the third most frequent letter in English is the third most frequent letter in the plaintext, and the fourth most frequent letter in English is the fourth most frequent letter in the plaintext. The fifth through the eleventh most frequent letters in English are the same as the fifth through eleventh most frequent letters of the plaintext, and among them (fifth through eleventh) they are not misordered by more than one. For example, the ninth most frequent letter in English might be either the eighth or ninth or tenth most frequent letter in the plaintext.

3) If you want to play around with substituting letters in various ways into the ciphertext letters then you might find the accompanying MATLAB file "substitute.m" useful. For example, if you type the command "substitute('rpzw**qs**********',ciphertext)" then MATLAB will return a string where each "a" of the ciphertext is replaced by "r", each "b" of the ciphertext is replaced by "p", each "c" of the ciphertext is replaced by "z", each "d" of the ciphertext is replaced by "w", each "g" of the ciphertext is replaced by "q", each "h" of the ciphertext is replaced by "s", and all other letters are replaced with asterisks.

4) Bo Liu (a TA) wrote the accompanying MATLAB function "FindMostFreq.m" which finds the most frequent bigrams (two consecutive letters) trigrams (three consecutive letters) and quadgrams (four consecutive letters) in any string of letters. If you type the command "FindMostFreq(ciphertext,2,10)" then MATLAB will return the ten most frequent bigrams in ciphertext, if you type the command "FindMostFreq(ciphertext,3,10)" then MATLAB will return the ten most frequent trigrams in ciphertext, and if you type the command "FindMostFreq(ciphertext,4,10)" then MATLAB will return the ten most frequent quadgrams in ciphertext. The most frequent bigrams in English are (ordered most to least frequent) *th*, *he*, *in*, *er*, *an*, *re*, *nd*, *at*, *on*, *nt*. You may use the hint that the five most frequent bigrams in our plaintext are the same five most frequent bigrams in English, with just one transposition in their order (meaning that for just one i among $1, 2, 3, 4$, the ith most common English bigram is the $i+1$th most common bigram in the plaintext and the $i+1$th most common English bigram is the ith most common bigram in the plaintext). The most frequent trigrams in English are *the*, *and*, and *tha*; however, only two of these are among the three most frequent trigrams of the plaintext.