Lezione3

Breaking Vigenére cipher

Recovering the length of the key

This is done with the Friedman method:
Using the index of coincidence:
$I_c(x) = \frac{\sum_{i=1}^{26} f_i(f_i-1)}{n(n-1)} \approx \sum_{i=1}^{26} p_i^2$
Where:

- n is the length of the text.
- f_i is the frequency of the i-th letter (i.e. the number of times it occurs in a text)
- p_i is the probability of the i-th letter, $p_i = \frac{f_i}{n}$, this gives the probability that two letters, randomly chosen from a text, are the same.

  The IC of the text bmqvszfpjtcsswgwvjlio would be given by:

  b(1*0)+ c(1*0)+ f(1*0)+ g(1*0)+ i(1*0)+ j(2*1)+ l(1*0)+ m(1*0)+ o(1*0) + p(1*0)+ q(1*0)+ s(3*2)+ t(1*0)+ v(2*1)+ w(2*1)+ z(1*0) =12

  divided by N*(N-1) = 21*20 = 420

  which gives us an IC of 12/420 = 0.0286

  The value of the index of coincidence can range from $\frac{1}{26}$ (if the letters have the same probability, basically random text) to 1 (single letter text e.g. AAAAA).
  After computing the I_c, if the value is ≈ 0.038, we're trying to break a poly alphabetic cipher; if the value is ≈ 0.065, we're trying to break a mono alphabetic cipher. (I believe this is assuming we're using the English language).

```
m = 1
LIMIT=0.06 # this is to check that ICs are above 0.06 and thus close to 0.065
found = False
while(not found):
    sub = subciphers() # takes the m subciphertexts sub[m] obtained by selecting one letter every m
    found = True
    for i in range(0,m): # compute the Ic of all subtexts
        if Ic(sub[i]) < LIMIT:
            # if one of the Ic is not as expected try to increase length
            found = False
            m += 1
```

```
                break
    # survived the check, all Ic's are above LIMIT
    output(m)
```

Once we find a value close to the original I_c, we can try to recover the original key.

Recovering the key

Using the mutual index of coincidence:
$MI_c(x,x') = \frac{\sum_{i=1}^{26} f_i f_i'}{nn'} = \sum_{i=1}^{26} p_i p_i'$
where x and x' are two subtexts (two sub ciphers basically).
Given the previous sub ciphers, we can use the mutual index of coincidence to find the shift for each letter, granting us the key.

```
key = [] # empty list
for i in range(0,m): # for any letter of the key
    k = 0       # current relative shift
    mick = 0    # maximum index so far (we start with 0)
    for j in range(0,26): # for any possible relative shift
        # compute the mutual index of coincidence between the first
subcipher
        # sub[0] and the i-th subcipher shifted by j
        mic = MIc(sub[0], shift(j,sub[i]))
        if mic > mick: # if it is the biggest so far
            k = j      # we remember the relative shift
            mick = mic # ... and the new maximum
    key.append(k)      # we append to the list the shift we have found
```

With this algorithm we find the list of relative shifts starting from key[0], for example

```
key=[0,4,6,3,9]
```

This means that the second letter of the key is the first letter $+4$ and so on.
Lastly, we brute force the first letter (only 26 possible letters so it's feasible) and we retrieve the key.

Known plain text attacks

Usually, an attacker can guess part of the plain text, for example a header. This is a reasonable assumption if the messages are encrypted using the same key, thus an attacker can try to guess part of the plain text (e.g. sniffing traffic) and then he'll know some pairs (x, y) where $x =$ plain text and $y =$ cipher text.

Hill cipher

Poly alphabetic cipher, "based on Vigenére".
$P = C = Z_m^{26}$ and $K = \{K | K$ is a invertible mod 26 matrix m $\times$ n $\}$
$E_k(x_1, \dots, x_m) = (x_1, \dots, x_m)K\%26$
$D_k(y_1, \dots, y_m) = (y_1, \dots, y_m)K^{-1}\%26$

Encryption using Hill cipher

To encrypt a message M, we have to
-take $M=(x_{1.},..,x_m)$ and the K which is a matrix
-compute $(x_{1,}...,x_m)$ K mod 26

Example: Let us assume $M=(x_1,x_2)=(5,9)$ we have

$$K = \begin{bmatrix} 5 & 11 \\ 8 & 3 \end{bmatrix}$$

Thus, $E_k(5,9)= (5,9)\text{x}\begin{bmatrix} 5 & 11 \\ 8 & 3 \end{bmatrix}$mod 26=(5x5+9x8,5x11+9x3) mod 26=(25+72,55+27) mod 26= (97,82) mod 26= (19,4)

Decryption using Hill cipher

First of all we have to compute K^{-1}:
To do so we first have to compute the inverse of the matrix K, in a 2×2 matrix this is done like this, assuming that the starting matrix is:

$$K = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$K^{-1} = det^{-1}(K) \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \%26$$

HINT: Keep the values of the matrix positive (modular arithmetic wonders e.g. -11%26 = 15)
How do we compute $det^{-1}(K)$?
First we compute normally $det(K) = ad - bc$ then we have to find a number in the range $[0, 25]$(generally speaking a number in the range of $(0, n-1)$). that multiplied by

$det(K)\%26$ gives us 1.
What's left is to multiply the message $(y_1, \dots, y_m)$ and the K^{-1} matrix, thus giving us the original message $(x_1, \dots, x_m)$.

Breaking Hill Cipher

Let's assume that an attacker knows some pairs $(x, y), (x', y'), \dots$ of plain text - cipher text.
We know that encryption using Hill Cipher is done by multiplying the message with the key matrix: $Y = KX$ where $Y = (y_1, \dots, y_m)$, $X = (x_1, \dots, x_m)$ and K is the matrix $m \times m$ used as a key.
If the attacker knows those (x, y) pairs, he can try to calculate X^{-1} and then obtain the key K like this:
We know that $Y = KX$, so if we find the inverse of X, namely X^{-1} we can do the following steps to retrieve the orignal key K:

$$X^{-1}Y = X^{-1}XK$$

Since multiplying a matrix by it's inverse we obtain the identity matrix:

$$X^{-1}Y = IK$$

Multiplying a matrix by the identity I, we obtain the original matrix:

$$X^{-1}Y = K$$

NOTE: The identity matrix is the matrix that has all 0 except the main diagonal which has 1.