MATH 587/CSCE 557 - SUMMARY OF CLASS, 2/22/07

I handed back the 1st midterm graded and gave the score distribution. I started by discussing spaces. Since spaces between words are more common even than E, frequency analysis could be used to detect them if they were left in, so they tend to be omitted. If they were left in, we could also use pattern words. So, for example, if ABACDCCA is a word encrypted with some substitution cipher, then it has to be NINETEEN, since that's the only English word with that pattern of letters. I also mentioned online crossword solvers that can find all English words of the form e.g. G*M*C**K* - this can be useful in finishing off decrypting of substitution ciphers (or solving crosswords). Finally, note that it is relatively rare in English for the letter after a vowel to be a vowel.

Then we started ciphertext-only attacks on Vigenere ciphers. The main point is that once we have the keyword length m, then it amounts to m shift ciphers and we can use frequency analysis to find each shift. There are two main techniques for doing this.

1st technique (Friedman's, 1920): For a given passage, consider the probability that a randomly chosen pair of letters consist of the same letter. If the passage is plain English, then this equals (the prob that the 1st letter is A)(the prob the 2nd letter is A)+(the prob the 1st letter is B)(the prob the 2nd letter is B)+...+(the prob the 1st letter is Z)(the prob the 2nd letter is Z), which is $p_0^2 + p_1^2 + ... + p_{25}^2 = 0.066$, where p_i is the frequency with which the ith letter of the alphabet occurs in typical English.

If we have a ciphertext encrypted with a monoalphabetic cipher, then this probability will be the same, since two letters are the same in the ciphertext if and only if they're the same in the plaintext. This is not not going to be true for polyalphabetic ciphers since there the same plaintext letter can be encoded as different ciphertext letters.

Define the index of coincidence, I, to be the proportion of pairs of letters taken from the ciphertext that consist of the same pair of letters. By the last comment, if I greatly differs from 0.066, this is evidence that the cipher is not monoalphabetic.

If there are n_0 A's, n_1 B's, ..., n_{25} Z's in the ciphertext, and $n = n_0 + ... + n_{25}$ is the total number of letters in the ciphertext, then we computed that

$$I = (n_0(n_0 - 1) + n_1(n_1 - 1) + ... + n_{25}(n_{25} - 1))/(n(n - 1))$$

On the other hand, if the keyword has length m, then we can estimate I a different way. Arrange the ciphertext in m columns, each with n/m entries, so that the ith column consists of all letters encrypted by the ith letter of the keyword. We compute the number of pairs of letters taken from the same column - we expect that about 0.066 of these pairs consist of the same letter. Then we compute the number of pairs of letters taken from different columns - we expect that just $1/26 = 0.0385$ of these pairs consist of the same letter. This yields (after some cancellation):

$$I \approx (0.066(n-m) + 0.0385n(m-1))/(m(n-1))$$

If we turn this around and solve for m in terms of I, this gives a guess as to the length of the keyword. Namely:

$$m \approx 0.0275n/((n-1)I + 0.066 - 0.0385n)$$

This is not very reliable, particularly if n is small, since I varies only slowly with m. One extra check would be to work out indices of coincidence for each of the columns and see if they're close to 0.066.

2nd technique (Kasiski's, 1863): Sometimes the ciphertext will contain the same string of letters twice. This could be a random occurrence, but if the string is long enough, it is more likely to have come about because the same string of letters in the plaintext occurred and got encrypted by the same string of letters from the keyword. In particular, if the string starts in the k_1th and k_2th positions, then this implies that the length of the keyword should divide $k_2 - k_1$. By repeating this, you can often nail down a likely value for the length of the keyword.