Lecture 3

Monday, Jan 9, 2012

The Frequency Attack

As we saw in the last two lectures, the Caesar cipher was trivial to break by simple brute force search. For each of the 26 possibilities for the key, compute a candidate message from the ciphertext and output the one that "makes sense".

Today, we will see a new and much more powerful attack on the Caesar Cipher called the "frequency analysis attack". There are two key ideas underlying the attack:

1. The Caesar cipher shifts each character of a message stream *by the same amount.*

   Thus, if a character 'E' occurs at two points in the text, they will both be encrypted the same character in the ciphertext, say 'Z'.

2. The probability distribution of the underlying message space (in this case, English) is not uniform.

   For example, in a typical English text, 'E' is the most frequent letter, and the letters 'Q' and 'Z' are among the least. See Figure **??** for the English frequency table.

   Note that the figure corresponds to a "typical" English text. Different languages, or even different types of English documents, have different underlying frequency distributions. For example, a legal document might have a different frequency distribution than Shakespeare's "Tempest".

Put together, this means that given a ciphertext, we can infer the following: *if the ciphertext contains an overwhelming number of 'Z's, it must mean that the encryption process mapped 'E's into 'Z's, and thus the shift was* 21.

We will make this more rigorous in a little bit.

Frequency Analysis, Mathematically

Let the English frequency vector be denoted as $\vec{p} = (p_a, p_b, \ldots, p_z)$, or equivalently as $\vec{p} = (p_0, p_1, \ldots, p_{25})$ – where a is mapped to 0, b is mapped to 1 etc.. In essence, this is the fraction of the time that each letter occurs in a typical English text.

Assume now that you are given a relatively large quantity of ciphertext characters encrypted using the Caesar cipher, and let the ciphertext frequency vector be $\vec{q} = (q_0, q_1, \ldots, q_{25})$. This is essentially the fraction of the time each letter occurs in the given ciphertext.

Of course, since this is the Caesar cipher, the vector $\vec{q}$ is approximately a shift of the vector $\vec{p}$. (The quality of the approximation depends on how large a ciphertext we are given. If we only have a couple of characters of the ciphertext, then the distribution $\vec{q}$ will be too skewed to be useful.)

We now do the following:

1. For each i, compute the following quantity which is the confidence by which we will guess that the Caesar key $K = i$.

$$\mathsf{Guess}_i = \sum_{j=0}^{25} p_j q_{j-i}$$

2. Output the number i for which Guess_i is the largest.

The idea is that if the key is indeed $K = i$, then $p_j \approx q_{j-i}$ ($\approx$ means "approximately equal to"). The expression $\sum_{j=0}^{25} p_j q_{j-i}$ is maximized when this happens (as we saw in class).

"Large Key Length" Principle

To avoid a brute force search attack, any secure encryption scheme needs to have a "long enough" key. This is a *necessary* condition, but not *sufficient*. That is, schemes with short keys can certainly be broken with brute force search. However, schemes with long keys are not necessarily secure. We will see an example next, with the Vigenere Cipher.

Vigenere Cipher

The Vigenere Cipher is very much like the Caesar Cipher except that it uses many different (random) shifts as the key. Here is an example of a Vigenere cipher with two shifts:

1. KeyGen: Generate two random shifts $\vec{K} = (K_0, K_1) \in_R \mathbb{Z}_{26}^2$.
2. $\mathsf{Enc}(\vec{K}, M_0 M_1 M_2 M_3 \ldots)$: Shift M_0 by K_0, M_1 by K_1, M_2 by K_0, M_3 by K_1 and so forth. That is, for even positions, use the shift K_0, and for odd positions, use K_1.
3. $\mathsf{Dec}(\vec{K}, M_0 M_1 M_2 M_3 \ldots)$: Obvious.

In class, you will show that this can be broken with a frequency attack as well (albeit using a longer ciphertext).

The notes will be updated with an example post lecture.