Breaking Vigenere

So, the question is, how can the key length for a Vigenere key be obtained, because knowing that will lead to some partial letter frequency information.

There are two standard methods:

1) The Kasiski Test
2) Index of Coincidence

Kasiski Test

The probability that two repeating strings in the ciphertext occurring from two different plaintexts is quite rare. The reason for this is that the keyword has relative "spacing" between its characters. For two different sets of plaintext characters to have "complementary" spacing and to STILL make sense is quite unlikely.

In our example above, MEET encrypted to OSQI. Now, imagine using the part of the key UTER instead of COMP. The corresponding plaintext letters, when added to UTER that would yield the ciphertext OSQI are UZMR, which is NOT an English word. In fact, we are likely to find that if we lined up the ciphertext OSQI under any other portion of the keyword than COMP, that the corresponding plaintext would not be valid English.

Thus, if we saw OSQI in the ciphertext twice, it is overwhelmingly likely that it is the SAME plaintext word, encrypted using the SAME keyword letters. Imagine that the first OSQI appeared at index 110 while the second OSQI appeared at index 158. Taking the difference of these two, we get 48. It's likely then, that the keyword length divides evenly into 48, since the same part of the keyword lined up at these two indexes. We could grab further pairs of repeating ciphertext and note the distance between where these pairs occurred. For example, imagine if another string repeated at indexes 83 and 171. Then we'd also know that the keyword length divided 171 – 83 = 88 as well.

The Kasiski Test is simply finding the lengths between pairs of repetitions of the ciphertext, and then taking the greatest common divisor of each of these values, since the keyword length is likely to divide into each of these values. Thus, list out each starting index of a repeated n-gram in sorted order, and for each successive pair of items on the list, take their gcd. Then, take the gcd of all of these values and that gives you the most likely keyword length.

Index of Coincidence Test

Given a group of letters, their index of coincidence is defined as the probability that two randomly chosen letters in the group are the same.

For example, for text from the English language, the probability that the first letter chosen is 'A' is about 9% and the probability that the second letter chosen is also 'A' is also 9%. Thus, the probability that two randomly chosen letters are both 'A' is .09*.09 = .0081. We can simply sum up the probabilities that two randomly chosen letters are both 'B', 'C', …, 'Z' to calculate their index of coincidence.

Mathematically, for a set of letters, we have: IC = $\sum_{i=1}^{26} \frac{f_i(f_i-1)}{n(n-1)}$, where f_i represents the frequency of the ith letter in the set of letters and n represents the total number of letters in the set of letters. This is equal to the probability that two randonly chosen letters from the set are the same.

It turns out that the Index of Coincidence of English is approximately .065. But, the index of coincidence of random letters is just 1/26 ~ .038. (This is because if each letter is equally likely, then the probability that the second one would match the first is just 1/26, the probability of pulling the appropriate second letter.)

So what you can do with some Vigenere ciphertext is make a guess at the keyword length, k. Then split up the ciphertext letters into k groups. (If k were 8, then the first letter, ninth letter, sixteenth letter, etc. would all be in the same group.) For each of the k groups, calculate the index of coincidence. If most of the groups are around .065, then your guess for k is most likely correct. Otherwise, if these numbers are lower, it is probably not. Try different values of k until you find one that works.

Now the next question is: once you know the keyword length, how do you work to obtain the keyword?

Mutual Index of Coincidence

Once the key length, k, is determined, then create the k groups of letters where each letter in a group was shifted by the same value. What we can do for each group is try each of the 26 possible shifts and see which one creates a set of letter frequencies that are "closest" to English. One easy way to do this is to attempt to minimize the frequencies that line up at the low-frequency letters, J, K, Q, X and Z. This is much easier to visualize than to write in text!

For example, let's look at a simplified version with 5 letters. Let's say our target language, Martian, has the following letter percentages:

A 10%	B 20%	C 35%	D 5%	E 30%

Let's say we have a bin of shifted letters with these percentages:

A 28%	B 11%	C 21%	D 36%	E 4%

If we look at the chart between these two, they don't line up very good:

	A 10%	B 20%	C 35%	D 5%	E 30%
	A 28%	B 11%	C 21%	D 36%	E 4%
Diff	18%	9%	14%	31%	26%

It's pretty unlikely that these both represent the same set of letters.

Now, let’s take the bin of shifted letters from the ciphertext and subtract 1 from each letter, making a ‘B’ from the bin an ‘A’, a ‘C’ from the bin a ‘B’ and so forth. Of course, an ‘A’ from the bin wraps backwards to ‘E’. Our shifted frequencies (if the shift of this bin were to be 1) is:

Shift -1	A 11%	B 21%	C 36%	D 4%	E 28%

Now, let’s compare to the regular Martian frequencies:

Martian	A 10%	B 20%	C 35%	D 5%	E 30%
Shift -1	A 11%	B 21%	C 36%	D 4%	E 28%
Diff	1%	1%	1%	1%	2%

It’s clear that these two bins are really similar. Thus, it’s likely that this bin was shifted by 1 and the corresponding keyword letter was ‘B’.

This idea of “lining up” bins visually is appealing, but also would be time consuming. It would be great if there’s a way to identify the “closest” match quickly without such detailed analysis.

It turns out that if we have two sets of percentages that both add up to 100% and have the same set of values, and the pick pairs to multiply (one from each set) and add these products together (basically a dot product), then we maximize the sum by lining up like terms.

This is where the mutual index of coincidence comes in.

The definition of the mutual index of coincidence between two multisets of letters is as follows:

Given two sets of letters, with frequencies f_i and g_i, with $\sum_{i=1}^{26} f_i = m$ and $\sum_{i=1}^{26} g_i = n$, the mutual index of coincidence of those two sets of letters is defined as the probability that a randomly chosen letter from the first set is equal to a randomly chosen letter from the second set.

To calculate this value, sum over each different letter, the probability of picking that letter from both sets. Doing so yields the following sum: $\sum_{i=1}^{26} \frac{f_i g_i}{mn}$.

Using the intuition previously discussed, the more similar two bins are, the higher the mutual index of coincidence is between them. Thus, here is the strategy:

```
For each bin i:

    For shift in between 0 and 25, inclusive:

        1) Take the letters in bin i and subtract shift from each of them (wrap around).
        2) Calculate the new letter frequencies.
        3) Calculate MIC between English and this set of frequencies.
        4) If it’s larger than the largest seen, update the largest seen and best shift.

Note that steps 1 and 2 can be condensed into:

        1+2) Cyclically left shift the frequency array by 1 spot.
```

A cyclic shift to the left by 1 spot is this operation:

Input: [a0, a1, a2, …, a(k-1)]
Output: [a1, a2, a3, …, a(k-1), a0]

If you use this process, more than likely 50% to 70% of your guesses for keyword characters will be correct. From there, either you can just “see” what the key word ought to be, for example, if you get “DXLPHYNS”, it’s probably “DOLPHINS” if you know a little bit about me =) Alternatively, you can go back to the letters you think are incorrect and look at which shifts created the second or third highest MIC’s and try those out.