The Vigenère cipher is a polyalphabetic cipher, originally created by Giovan Battista Bellaso in 1553, but named after Blaise de Vigenère. The cipher works similarly to a shift cipher as we begin by assigning each letter in the English alphabet an integer value, starting with $A = 0$ and ending with $Z = 25$. We then come up with the keyword we plan on using to encrypt the plaintext. The plaintext is encrypted in the following manner: the first letter of the plaintext is encrypted with a shift corresponding to the integer value of the first letter of the key. That is to say, we add the integer value of the plaintext letter to the integer value of the keyword letter modulo 26. The outputted value will correspond to the integer value of the first letter in the ciphertext. We do the same for the second letter in the plaintext with the second letter in the keyword, and so on until we run out of letters in the keyword. Then, we cycle back to the first letter in the keyword and begin encrypting. Suppose a keyword has length m, then by indexing the values of the plaintext (starting with 0), we see that the $(0, 0+m, 0+2m, 0+3m\dots)$ letters in the plaintext will all be encrypted with the first letter of the keyword. The $(1, 1+m, 1+2m, 1+3m\dots)$ letters in the plaintext will all be encrypted with the second letter of the keyword, and so on. Deciphering a message given the keyword is extremely straightforward. We do this by following the same procedure as described for the encrypting process, with one adjustment. We subtract the integer value of the key letter instead of adding it.

The cipher seems relatively straightforward and potentially easy to cryptanalyze, but it wound up stumping code breakers for over three hundred years. Notice that the number of possible keywords scales exponentially relative to m. In fact, the number of keywords associated with a given value of m is equal to 26^m. Historically, this number of keys, even for small values of m, was to large for a person to attempt them all by hand. This made the cipher particularly hard to crack. With modern technology, a computer can quickly compute the plaintext corresponding to all possible keys if the value of m is known. The problem is a little more difficult when the value of m is unknown, but still solvable. Without modern computers, the cipher is much more difficult to crack which is why it was thought impossible to cryptanalyze for hundreds of years.

We can now begin to describe the methodologies used to cryptanalyze the Vigenère cipher. The first breakthrough was the Kasiski test, developed in 1863 by Friedrich Kasiski. This was a way to find the value of m, the key length, if given ciphertext encrypted with the Vigenère cipher. Kasiski observed that sections of plaintext that were the same and δ apart, where δ was a multiple of m, would be encrypted into the same ciphertext. Using this, he described a procedure by which we take a three letter sequence in the ciphertext and look for all the places in the ciphertext where it appears. Next, we find the difference between the locations of these sequences. This will give us a set of distances, δ_0, δ_1, ..., δ_i, and it stands to reason that these would all be multiples of m. Thus, if we find the greatest common divisor of the set of δ's, we have a fairly good guess about the length of the key word.

Another development in the cryptanalysis of the Vigenère cipher was the Friedman test, also referred to as the index of coincidence. In 1920, Friedman posited that given a string of n alphabetic characters, $\mathbf{x} = x_1x_2x_3\ldots x_n$, we can define the index of coincidence, $I_c(\mathbf{x})$ as the probability that two random character in $\mathbf{x}$) are the same. The formula for the index of coincidence can be given by

$$I_c(\mathbf{x}) = \frac{\sum_{i=0}^{25} f_i(f_i - 1)}{n(n-1)}$$

where f_i is the corresponding frequency of the i-th English letter, i.e., when $i = 0$ then f_i is the frequency of the letter A. When examining the average string of English text, we find that the $I_c(\mathbf{x}) \approx \sum_{i=0}^{25} p_i^2 = 0.065$, where p_i is the average probability of a letter appearing. That is to say, p_0 is the average probability that a given letter is A. We can now begin to apply these calculations to calculate our most probable key length. The process for this works as follows. We begin by taking our cipher text $y_1y_2y_3\ldots y_k$ and breaking it up into m strings where m is our guess for the key length. We break the cipher text up by taking $\mathbf{y}_1 = y_1y_{1+m}y_{1+2m}\ldots$ for our first string, then $\mathbf{y}_2 = y_2y_{2+m}y_{2+2m}\ldots$ for our second, and so on. We can now calculate $I_c(\mathbf{y}_1)$, $I_c(\mathbf{y}_2)$, and so on. If we have the correct guess for our key length m, then we would expect that each index of coincidence would equal about 0.065. We can keep iterating over different values for m until we find one where our average index of coincidence is equal to about 0.065.

Now that we have a methodology for determining the key length, we are left with the problem of finding the correct key. We begin by breaking up the ciphertext into substrings in the same manner as described in the index of coincidence, with each substring having length equal to $n' = n/m$. We then calculate a new value which we denote M_g. For $0 \leq g \leq 25$, we say that

$$M_g = \sum_{i=0}^{25} \frac{p_i f_{i+g}}{n'}$$

with p_i being the probability of the i-th letter and f_{i+g} being the frequency of the $(i+g)$-th letter. Suppose that a key has length m, then we can write the key as $(k_1, k_2, ..., k_m)$. We can now calculate $M_g(\mathbf{y}_1)$ for all $0 \leq g \leq 25$. It turns out that when $g = k_1$ we will get $M_g(\mathbf{y}_1) \approx 0.065$. We can do this for all substrings of $\mathbf{y}_i$ to find our most likely key values $(k_1, k_2, k_3, \ldots, k_m)$.

Combining these three methodology, we have put together a strong starting point for decrypting cipher text encrypted with the Vigenère cipher. The algorithm implemented on this site uses these foundations, with some more optimization, as describe in what follows.