

Attacking some classical ciphers

MATH2068 Number Theory & Cryptography
Week 5 Lecture 3

University of Sydney
NSW 2006
Australia

27th August 2008



Vigenère ciphers

Vigenère ciphers have already been described in lectures in Week 3, as well as in the computer tutorials.

Here is another example to illustrate how they work.

Suppose we use a Vigenère cipher with keyword HAMLET and our plaintext message is TOBEORNOTTOBETHATISTHEQUESTION

Identify letters with numbers using the following table:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

The keyword is 7 0 12 11 4 19 and the plaintext is 19 14 1 4 14 17 13 14 19 19 14 1 4 19 7 0 19 8 18 19 7 4 16 20 4 18 19 8 14 13.

Vigenere encryption



Write the keyword underneath the message, repeating it over and over, and use addition modulo 26 to encrypt:

19	14	1	4	14	17	13	14	19	19	14	1	4	19	7	0	19	8	18	19	7	4	16	20	4	18	19	8	14	13		
7	0	12	11	4	19	7	0	12	11	4	19	7	0	12	11	4	19	7	0	12	11	4	19	7	0	12	11	4	19		
<hr/>																															
0	14	13	15	18	10	20	14	5	4	18	20	11	19	19	11	23	1	25	19	19	15	20	13	11	18	5	19	2	6		

In letters, the ciphertext is AONPSKUOFESULTTLXBZTTPUNLSFTCG.

The length of the keyword (6 in this case) is called the *period* of the cipher.

The single most important thing to realize is that if you know the period of a Vigenère cipher then cracking it is almost trivial, provided that the message is long enough.

Decimations



The plaintext M can be thought of as a sequence of residues modulo 26. So $M = c_1 c_2 c_3 \dots c_N$, where each c_i is a natural number less than 26, and N is the length of the message.

The keyword $K = k_1 k_2 \dots k_m$ is also a sequence of residues modulo 26. Here m is the period.

Define $k_{m+1} = k_1$, $k_{m+2} = k_2$, etc.. More precisely, for each $i \in \mathbb{Z}$ let $k_i = k_r$, where r is the residue of $i \pmod{m}$.

The ciphertext is $M' = c'_1 c'_2 c'_3 \dots c'_N$, where the i -th term c'_i is the mod 26 residue of $c_i + k_i$.

In particular the sequence $c'_1 c'_{m+1} c'_{2m+1} \dots$ is simply an "alphabetic shift" of the sequence $c_1 c_{m+1} c_{2m+1} \dots$. To get c'_{am+1} you just add k_1 to $c_{am+1} \pmod{26}$.

We define the *decimation of M with period m and index r* to be the sequence $\text{Dec}(M, m, r) = c_r c_{m+r} c_{2m+r} c_{3m+r} \dots$

Frequencies of letters in decimations



The relative frequencies of letters in English text do not vary much from one piece of text to another.

Most frequent letters: E (≈ 0.12), T (≈ 0.09), A (≈ 0.085), O, N, I, S, R, H (all about 0.06 to 0.075), D (≈ 0.05), L (≈ 0.04).

It turns out any decimation of typical English text will also exhibit these same frequencies.

Performing an alphabetic shift on such a decimation of course changes the letter frequencies. It is easy to determine the amount of the shift by examining the letter frequencies in the shifted text.

E.g., if we have a sequence of letters that we believe is an alphabetic shift of a decimation of some English text, and if we see that the frequency of L is 0.128, it is highly likely that a shift of 7 places was used: A \rightarrow H, B \rightarrow I, C \rightarrow J, D \rightarrow K, E \rightarrow L,

Frequencies in decimations (continued)



Of course, it is not *always* the case that *all* letters occur with their typical frequencies.

For example, you can't be sure that the most frequent letter is E.

In our imagined example on the last page, we guessed an alphabetic shift of 7 since L had a high frequency. We would have to check the frequencies of other common letters as well to confirm it.

An alphabetic shift of 7 would mean T becomes A, A becomes H, and O, N, I, S, R, H become V, U, I, P, Z, Y. If these don't have roughly the right frequencies, then as a 2nd guess we should try an alphabetic shift of 18 (T \rightarrow L) rather than 7 (E \rightarrow L).

Finding the key if the period is known



Suppose that the period is m . Then

$$\text{Dec}(M', m, 1) = c'_1 c'_{m+1} c'_{2m+1} c'_{3m+1} \dots$$

is the same as

$$\text{Dec}(M, m, 1) = c_1 c_{m+1} c_{2m+1} c_{3m+1} \dots$$

alphabetically shifted by k_1 , where k_1 is the first term of the key.

We can find k_1 by examining letter frequencies in $\text{Dec}(M', m, 1)$.

Similarly $\text{Dec}(M', m, 2) = c'_2 c'_{m+2} c'_{2m+2} c'_{3m+2} \dots$ is the same as $\text{Dec}(M, m, 2) = c_2 c_{m+2} c_{2m+2} \dots$ alphabetically shifted by k_2 , where k_2 is the 2nd term of the key.

We can find k_2 by examining letter frequencies in $\text{Dec}(M', m, 2)$.

And we can similarly find k_3, k_4, \dots, k_m by examining letter frequencies in $\text{Dec}(M', m, 3), \text{Dec}(M', m, 4), \dots, \text{Dec}(M', m, m)$.

Finding the period



We have seen that if we know the period we can find the key.

So try period 1, then period 2, then 3, etc. until we find the one that works.

i.e., for each potential period m see if the frequency distributions of letters in the decimations $\text{Dec}(M', m, i)$ are alphabetic shifts of frequency distributions that one might get from ordinary text.

If they are not, then try the next value of m .

Coincidence index



The *coincidence index* of a piece of text is the probability that two randomly chosen letters are the same.

If the relative frequencies of the 26 letters are p_0, p_1, \dots, p_{25} then the coincidence index is $\sum_{i=0}^{25} p_i^2$.

For English text the coincidence index is usually about 0.065.

Alphabetic shifts do not change the coincidence index.

So if a Vigenère cipher has period m , the decimations $\text{Dec}(M', m, i)$ will have coincidence index about 0.065.

This provides a convenient way to find m : compute the coincidence index of $\text{Dec}(M', m, 1)$ for $m = 1, 2, 3, \dots$ until we find an m that gives a value greater than 0.06.

How well does this work?



For the above methods to work one needs a piece of ciphertext of length many times the length of the period m . Otherwise the decimations $\text{Dec}(M', m, i)$ will not be long enough to give meaningful frequency distributions.

In the extreme case that m is greater than the length of the ciphertext it is impossible to decrypt the message without knowing the key. The decimations would all have length 1 and so would give no meaningful statistical information.

One-time pads



A one-time pad system is simply a Vigenère system with a key that is used only once and only for a message that is shorter than the key itself. The key should be generated by some random process, and of course should only be known to the person who intends to send the message and intended recipient. Any statistical properties of the plaintext will be completely destroyed by the randomness of the key. There is no way to decrypt such a message without access to the key.

A problem with the one-time pad is that the key has to be long enough to cope with any message that might be sent, and may have to be kept secret for a long time before it is used.

But note that a key for a one-time pad could be encrypted using a simple substitution cipher. An enemy who intercepts this will not be able to use statistical analysis to decrypt it, because the message is just a random sequence of letters.

Block transposition ciphers



The plaintext $M = c_1 c_2 c_3 \dots c_N$ is split into blocks of length m ; the letters in each block are rearranged using the same permutation of $\{1, 2, \dots, m\}$. The key is the permutation.

For example, if the key is $[4, 2, 1, 3]$ then the cipher text will be

$$M' = c_4 c_2 c_1 c_3 c_8 c_6 c_5 c_7 c_{12} c_{10} c_9 c_{11} c_{16} c_{14} c_{13} c_{15} \dots$$

This process does not change the frequency distribution of the letters, but it does change the frequency distribution of the digraphs (pairs of adjacent letters).

A cryptanalyst who has intercepted some ciphertext that is known to have been created by a block transposition system will first have to find m , the block length.

The larger m is the harder this will be.

Finding the block length



Try the possibilities (2, then 3, then 4, and so on), doing various statistical tests, until we find one that looks plausible.

To test if m is plausible, examine the $[k, \ell]$ -decimation of M' of period m for various values of k and ℓ in the range $\{1, 2, \dots, m\}$. This is the following sequence of pairs:

$$\text{Dec}([k, \ell], m) = [c'_k c'_\ell, c'_{k+m} c'_{\ell+m}, c'_{k+2m} c'_{\ell+2m}, c'_{k+3m} c'_{\ell+3m}, \dots].$$

The idea is that if m is the period, and if c'_k and c'_ℓ were adjacent letters in the plaintext – say $c'_k = c_n$ and $c'_\ell = c_{n-1}$ – then c'_{k+qm} and $c'_{\ell+qm}$ would also have been adjacent in the plaintext, for all values of q .

Then the frequency distribution of pairs in $\text{Dec}([k, \ell], m)$ should match the typical frequency distribution of digraphs in normal unencrypted text.

Finding the block length (continued)



A quick test is to see if the distribution might be right is to compute the coincidence index for $\text{Dec}([k, \ell], m)$.

For each pair of letters ij we let p_{ij} be the relative frequency of ij in $\text{Dec}([k, \ell], m)$. That is, p_{ij} is the number of occurrences of the pair ij in the sequence divided by the length of the sequence. Then the CI (coincidence index) is $\sum_{ij} p_{ij}^2$.

If m is the period and c'_k and c'_ℓ were adjacent in the plaintext, this coincidence index should be similar to the digraph coincidence index for ordinary text, usually ≈ 0.007 or more.

A random sequence of letters will have digraph coincidence index $\approx (1/26)^2 \approx 0.0015$.

So, starting at $m = 2$, compute the CI of $\text{Dec}([1, \ell], m)$ for all ℓ from 2 to m . Reject m if no ℓ gives $\text{CI} \geq 0.007$.

Otherwise m is a good candidate for the block length.

Coincidence discriminant



Another statistic one can use instead of or as well as the CI is the *coincidence discriminant (CD)*.

For each letter i define p_{i-} to be $\sum_h p_{ih}$, the relative frequency in $\text{Dec}([k, \ell], m)$ of pairs that begin with i .

And define p_{-j} to be $\sum_h p_{hj}$, the relative frequency in $\text{Dec}([k, \ell], m)$ of pairs that end with j .

The CD is defined to be $\sum_{ij} (p_{ij} - p_{i-} p_{-j})^2$.

If the event that a randomly chosen term of $\text{Dec}([k, \ell], m)$ starts with i and the event that it ends with j are independent of each other, then p_{ij} will equal $p_{i-} p_{-j}$.

This means that for random text the CD should be very small.

But if ij is very common or very uncommon then $(p_{ij} - p_{i-} p_{-j})^2$ will be large. Hence the CD is large for normal text.

If m is the period and c'_k and c'_ℓ were adjacent in the plaintext, the CD should probably be ≥ 0.003 , otherwise much smaller.

Finding the key knowing the block length



If m is the block length, compute the CI and/or the CD of $\text{Dec}([k, \ell], m)$, for all k and ℓ .

Large values of the CI (say ≥ 0.007) and large values of the CD (say ≥ 0.003) suggest that c'_k and c'_ℓ were adjacent in the plaintext.

By finding the most likely adjacencies one can figure out how to reorder the letters in each block to recover the plaintext.

But actually the CI and the CD tell you only if the two letters in question were (probably) adjacent, they don't tell you which is first and which is second.

So you are just as likely to come up with the reverse of the correct order as the correct order. But you will easily recognize if you have the wrong one – you will not see words but rather reverses of words when you perform the decryption.