Queen Mary
University of London

MAS 335 Cryptography

Notes 3: Stream ciphers — Spring 2008

Substitution ciphers have been used since time immemorial. As we have seen, they are vulnerable to frequency analysis based on the statistics of the language used. Although frequency analysis was first developed by Arab cryptographers in the tenth century, substitution ciphers continued to be used until quite recently. Simon Singh, in *The Code Book*, tells the dramatic story of how the breaking, by Elizabeth's cryptanalysts, of the cipher used by Mary Queen of Scots led to her trial and execution in 1587. Apparently Mary and her conspirators thought their cipher was secure.

Eventually, it was realised that better ciphers were needed. Many schemes were tried, but the essential idea was to use different substitutions for different letters of the plaintext. The general name of a cipher based on this principle is a *stream cipher*. In this chapter we discuss stream ciphers.

We begin with a general principle, known as *Kerckhoffs' Principle*:

> Alice and Bob must always assume that Eve knows the encryption system they are using, as well as having intercepted the ciphertext. All they can hope to keep secret is the key.

For, although cryptographers continually invent new systems, knowledge of these systems will soon spread in the intelligence community.

The Vigenère cipher

In 1562, Blaise de Vigenère invented a cipher in which a different Caesar shift is applied to each letter of the plaintext.

Suppose that we shift the first letter by 5, the second by 14, the third by 23, the fourth by 4, and the fifth by 18. Thus the word `enemy` would be encrypted as `JBBQQ`. Notice that the two occurrences of `e` in the original message are replaced by different letters (`J` and `B`). Conversely, different letters in the plaintext become the same in the ciphertext.

The key to this cipher is the sequence $(5, 14, 23, 4, 18)$. Vigenère's idea was that, instead of having to remember the sequence of numbers, it is enough to remember the letters obtained by shifting the letter `a` by these numbers. In this case, `aaaaa` would become `FOXES`; this is the key to the cipher.

We can represent the process by a *Vigenère square*, as shown in Table 1. Write down the plaintext with the key immediately under it:

e	n	e	m	y
F	O	X	E	S
J	B	B	Q	Q

Now look in row `e` and column `F` to find the first letter in the ciphertext to be `J`. Repeat for the remaining letters.

What if the message is longer than the key? Vigenère's idea here was to repeat the key as often as necessary:

e	n	e	m	y	p	a	t	r	o	l	s
F	O	X	E	S	F	O	X	E	S	F	O
J	B	B	Q	Q	U	O	Q	V	G	Q	G

So the ciphertext is `JBBQQ UOQVG QG`.

So the key is a simple word or phrase which can be easily memorised and can be changed frequently.

Breaking the Vigenère cipher

The Vigenère cipher is a great advance on the monoalphabetic substitution cipher, and was used for hundreds of years. However, it has two weaknesses, which eventually led to a system of cryptanalysis for it. These are that the cipher applied to each letter is a simple Caesar shift, which is very easy to break, and the fact that the key string repeats after a relatively short number of steps.

Suppose that we knew that the keyword contains five letters. Then we can divide the ciphertext into five strings, where the first string contains the first, sixth, eleventh, …, letter; the second string contains the second, seventh, twelfth, …, letter; and so on. Now each string is a Caesar cipher and can be attacked by the methods we have already discussed. (We cannot use digram or trigram frequencies here, since letters which are consecutive in one of the substrings were five steps apart in the original message. But the letter frequency analysis, and in particular the frequency patterns of consecutive letters in the alphabet, can be applied.) Once we have a conjectured decryption of each string, we can reassemble them to give the message.

a	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
b	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	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	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

Table 1: Vigenère square

How do we determine the length of the key? We could simply use trial and error. The frequency analysis is not likely to give sensible answers unless the assumed length is a small multiple of the true length.

A more systematic method uses repeats in the ciphertext. A common digram like `th` will probably occur many times in a reasonably long message. If the key length is 5, then the number of different encryptions of it is (at most) 5, and two occurrences will be encrypted in the same way if their positions in the plaintext differ by a multiple of 5. If the key is `FOXES`, then `th` will be encrypted as `YV`, `HE`, `QL`, `XZ`, or `LM`, according as its starting position is congruent to 1, 2, 3, 4 or 5 mod 5.

If we notice that the digram `YV` occurs in positions 1, 66, and 111 of the message, we might guess that it represents `th`, and that the length of the key is a common factor of 65 and 110. Since $\gcd(65, 110) = 5$, we would deduce that the key has length 5. We have more information too: if our guesses are correct, then the first two letters of the key are also revealed as `FO`. However, the deduction about the key length does not depend on `YV` actually being `th`; it could be any common digram.

Two digrams could agree by chance, so it is safer to apply the method to trigrams, if we have a reasonable amount of ciphertext.

The first person to propose this method was Charles Babbage, better known as the inventor of the "Difference Engine" and the "Analytical Engine" (two mechanical computers) in the nineteenth century. Babbage never published his decryption method, and Simon Singh speculates that it might have been used by British Intelligence (who would want the method kept secret!) A few years later, Friedrich Kasiski proposed a similar method which now carries his name.

To summarise, Kasiski's method consists of two parts:

- First, guess the keyword length m, by finding the greatest common divisor of the distance apart of the commonest digrams or trigrams.

- Then divide the message into substrings each consisting of the letters congruent to i mod m for $i = 1, 2, \ldots, m$, and apply frequency analysis to determine the shift associated with each substring.

Chi-squared

The method can be mechanised to some extent. We now describe a method for suggesting a solution to a Caesar cipher, which can be applied after we have found the length of the keyword. This uses the *chi-squared statistic*, which statisticians use for measuring the goodness of fit of data. Unlike statisticians, we make no assumptions about the distribution of our data, and draw no conclusions about the significance of the result; the method simply suggests a possible decryption.

It should be stressed that in simple cases, pattern matching by eye is perfectly satisfactory; but it is easier to tell the computer to optimize a complicated function than to do some pattern matching.

Suppose that n objects are put into q boxes, where the probability that each object is put into the ith box is p_i (with $\sum p_i = 1$). The expected number of objects in the ith box is $e_i = np_i$. Suppose that the actual number in the ith box is a_i. Then the chi-squared statistic is

$$X = \sum_{i=1}^{q} \frac{(a_i - e_i)^2}{e_i}.$$

The smaller the value of X, the better the data fit the prediction.

Now suppose we have a piece of text of length n encoded with a Caesar shift, which we want to find. We apply what we hope is the inverse shift to the text. If we are right, then the result should be plaintext, and the letter frequencies should approximate those in English text, that is, $e_i = np_i$, where p_i is the relative proportion of the occurrences of letter i in English. So we calculate the chi-squared statistic, where a_i is the actual number of occurrences of letter i in the shifted text. If we are right, its value will be small. So we try all 26 shifts; the most likely decryption is the one with the smallest value of X.

This method only uses letter frequencies and makes no use of digrams, trigrams, etc. So it can be applied separately to all the substrings of a Vigenère enciphered text, once we know the period.

Here is a worked example. The following is encrypted with a Vigenère cipher.

```
FZFGW BOPFW LWKRA SUQSY JHSIJ DHFVW ICCWA YHFRY GMEIJ
XWPXW WCKXZ JPXRC FBASX MOSMF LBLXZ NBDXG ICLRU JCOXO
NQBWZ JVXHH JSMIV NBQSL MSYSG PVBVK NGQIJ BOPVW FRFRY
GIQML MOARG UWZXM WSPSJ HCKZW WGXXA TBPMF NHXRV BVXXA
XHEIM XSLJS GCLOL MCRKZ YOIMU JKFXZ TIQTA HHRVW XCOGG
SJBVK FHFSF XGLWZ JKXWU TBPMV JFFRY NBEIJ TKKQA SRXWO
JZIEK XVBGG ZZAJG WHEIZ THAEQ ROAIZ JFCIW QJBVQ XZBIH
DOKHK YIMMV BVBXZ JFQLW UZBEK ZFBSX ROHMF LOAEA XMZLS
NBTSM QRYIO TFQLL MSQVG ZPIIG KUBXL NBDYH FBATA HYFRY
YVBHS NGFIK BVBRK ZRAIF QMXAZ NHBVS GPFXO NHETA SYBCW
XFXRU QCPIT DVBV
```

Step 1 asks us to guess the keyword length. We notice that the digram `HE` occurs at positions 182, 287 and 442 in the ciphertext, and in the first two of these it is part of the trigram `HEI`. So by the first step of Kasiski's algorithm, the keyword length should divide $\gcd(105, 155) = 5$. (I have anticipated this by writing the cipher in blocks of 5; usually Alice will not be so helpful!)

The first of the five substrings that we have to analyse is obtained by taking the first letter of each block; it is `FBLSJDIYGXWJFMLNIJNJJNMPNBFGMUWHWTNBXXGMY JTHXSFXJTJNTSJXZWTRJQXDYBJUZRLXNQTMZKNFHYNBZQNGNSXQD`. The letter frequencies in this substring are given in the third column of Table 2.

We calculate the chi-squared values using the frequency data from *Alice's Adventures in Wonderland*. Table 2 gives the calculation for shifts 0 and 5; it is easy to automate this to work out all values. The answers would not be very different if we had used different data for the frequencies.

We find that, for a shift of 5, the value of chi squared is 23.99. The smallest value for any other shift is 281.56, for a shift of 1. This strongly suggests that the shift is 5 and the first letter of the keyword is `F`.

By the same method (and the results are as clear-cut in all cases), we find the shifts for the other substrings to be $14, 23, 4, 18$, so that the keyword is `FOXES`. The decrypted text is

> Alice was beginning to get very tired of sitting by her sister on the bank and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, and "what is the use of a book," thought Alice, "without pictures or conversations?" So she was considering, in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

In fact, finding the period can also be mechanised to some extent, using a method due to William Friedman. See Garrett's book for a description of this.

Stream ciphers

The cryptographers now had two tasks. First, they had to find a way of producing a non-repeating key; second, to make the frequency analysis more difficult, they had to use an arbitrary permutation of the alphabet in each position, rather than just a shift. The two tasks require completely different ideas.

These complications also make it much more difficult to use the ciphers, especially in situations such as a battlefield signal unit. Thus it was necessary to move from hand to machine for the encryption and decryption.

One more thing to remember is that we are not restricted to using the Roman alphabet for our ciphers. We can translate our message into a string in any alphabet at all, and use this as the plaintext. In particular, the plaintext could be a string of digits

Letter	Frequency %	Observed	Expected Shift 0	Expected Shift 5
A	8.15	0	7.58	0.73
B	1.37	5	1.27	2.32
C	2.21	0	2.06	0.12
D	4.58	3	4.26	1.96
E	12.61	0	11.73	0.65
F	1.86	5	1.73	7.58
G	2.36	4	2.20	1.27
H	6.85	3	6.37	2.06
I	6.97	2	6.48	4.26
J	0.14	11	0.13	11.73
K	1.07	1	1.00	1.73
L	4.37	3	4.06	2.20
M	1.96	5	1.82	6.37
N	6.52	11	6.06	6.48
O	7.58	0	7.05	0.13
P	1.40	1	1.30	1.00
Q	0.19	4	0.18	4.06
R	5.02	2	4.67	1.82
S	6.05	4	5.63	6.06
T	9.93	6	9.23	7.05
U	3.22	2	2.99	1.30
V	0.78	0	0.73	0.18
W	2.49	4	2.32	4.67
X	0.13	9	0.12	5.63
Y	2.11	4	1.96	9.23
Z	0.07	4	0.65	2.99
$\sum(o-e)^2/e$			1949.79	23.99

Table 2: A chi-squared calculation

(so that the alphabet is $\{0,1,2,3,4,5,6,7,8,9\}$, or a string of binary digits (so that the alpabet is just $\{0,1\}$.

In the 1930s, a standard International Telegraph Code was agreed (see Figure 3). This is based on a code invented by Baudot, whose name has given rise to the word *baud* for the rate of information transmission. The ITC translates the 26 letters and 6 control characters into sequences of length 5 from a two-letter alphabet. With hindsight and familiarity with computers, we regard the symbols of the alphabet as 0 and 1; but originally they were two voltage levels in international telegraphy ($+80$ and -80 volts), or “hole” and “no hole” in punched paper tape. The names of the symbols don’t matter, but the names 0 and 1 will be very convenient later.

Using the ITC, a message is encoded into a string of zeros and ones. We can regard this as a string of length $5n$ over the alphabet $\{0,1\}$, or as a string of length n over an alphabet of 32 symbols (the 26 letters and six control characters), whichever is more convenient.

Generating the key

The best key is a completely random sequence of letters from the alphabet. Such a sequence is called a “one-time pad”. As we will see later, the one-time pad provides an absolutely secure form of encryption; no possible deductions about the plaintext can be made from knowledge of the ciphertext if this system is used properly.

However, it is very difficult to generate a truly random sequence. (There are rumours that people were employed by the CIA to toss coins all day and write down the results to produce one-time pads for the two-letter alphabet (whose letters might be called “heads” and “tails” in this case). It seems very likely that one-time pads were produced and used by intelligence services. Peter Wright, in *Spycatcher*, records the finding of one-time pads in the personal possessions of suspected Soviet spies in London by MI5 during the Cold War.

The difficulties of producing a random key led to various types of mechanical or electronic devices for producing what are known as “pseudo-random” keys. These are sequences of letters which, although not random, behave in many ways like a random sequence, so that a short sequence of the key gives very little information about the rest of the key. In particular, we require that each letter occurs with the same frequency, and similarly for digrams, trigrams, etc. We also require that the sequence does not repeat during the transmission of a typical message.

Every deterministic finite machine which outputs a string of characters must eventually repeat; its output will be *ultimately periodic*. That is because the machine must be in one of a finite (possibly very large) number of states at any moment. If it operates continuously, it must eventually return to the same state that it was in at some

A	11000
B	10011
C	01110
D	10010
E	10000
F	10110
G	01011
H	00101
I	01100
J	11010
K	11110
L	01001
M	00111
N	00110
O	00011
P	01101
Q	11101
R	01010
S	10100
T	00001
U	11100
V	01111
W	11001
X	10111
Y	10101
Z	10001
Letters	11111
Figures	11011
Line feed	01000
Carriage return	00010
Word space	00100
All space	00000

Table 3: International teleprinter code

previous time. From that point on, its behaviour will be the same as on the previous occasion; so the output is periodic. (The period may be very large. For example, a computer with 128 megabytes of memory has 2^{30} transistors, each capable of being in two states; so the number of configurations is $2^{2^{30}}$. In principle, the period could be as large as this number, approximately $10^{300000000}$.)

We will look later at some of pseudo-random number generators which have been used in practice.

Combining key and plaintext

The Vigenère square gives a method of combining plaintext with key to give ciphertext. We can descibe it more simply by identifying the letters `A...Z` with the elements $0\ldots25$ of $\mathbb{Z}/(26)$. Then the combination of plaintext letter p and key letter k gives the ciphertext letter $z = p+k$, where the addition is mod 26. Then decrypting simply involves subtraction mod 26: $p = z-k$.

In the Second World War, the Japanese military ciphers often used the digits $0\cdots9$ as symbols. The ciphers would also often use a codebook where various commonly used terms were encoded as groups of four digits. Thus, for example, `0700` could refer to the *kōkū tokushi musentai* (Air Special Radio Unit), and `4698` to the *kōkū tokushu jōhōtai* (Air Special Intelligence Unit). The key was a string of pseudo-random digits, and the encryption was addition mod 10, or addition without carrying. Thus, encrypting `4698` with key `7251` would give `1849`. Once again, decryption is subtraction mod 10 (subtraction without borrowing).

The same principle can be used in the simpler case of the binary alphabet. The rules for addition without carry give the addition table of the integers mod 2 (the finite field with two elements, often called the *binary field*:

$+$	0	1
0	0	1
1	1	0

Then, if the plaintext and key are strings of zeros and ones, we just add the mod 2; for example:

Plaintext:	01001001010...
Key:	10100010011...
Ciphertext:	11101011001...

Latin squares

It is possible to generalise the way in which we combine the plaintext and key to form the ciphertext in a stream cipher.

For each character of the key we associate a function mapping plaintext characters to ciphertext characters. This mapping must be a permutation, so that the recipient can invert it to recover the plaintext. So the addition table must have the property that each character appears exactly once in each column.

A Latin square of order q is an $q \times q$ array whose entries are taken from an alphabet of q symbols such that each symbol occurs exactly once in each row and column. This is a stronger requirement than we need; we will see later why it is a good feature from a cryptographic point of view.

In particular, the Vigenère square, the addition table of $0, \ldots, 9$ mod 10, and the addition table of the binary field (with the borders removed) are Latin squares. However, there are many other Latin squares. The exact number is not known; it is known that there are upper and lower bounds for the number of Latin squares of order q of the form $(cq)^{q^2}$ for positive constants c.

For example, here is a Latin square of order 10, using the alphabet $\{0, \ldots, 9\}$. I have bordered it with row and column indices for ease of use in enciphering.

	0	1	2	3	4	5	6	7	8	9
0	8	6	3	1	2	5	9	7	0	4
1	1	8	4	3	7	0	6	5	9	2
2	4	1	6	2	3	8	0	9	7	5
3	9	3	2	4	0	7	5	1	6	8
4	6	2	5	7	4	1	3	0	8	9
5	0	9	7	6	8	4	1	2	5	3
6	2	7	0	5	6	9	8	3	4	1
7	5	4	9	8	1	2	7	6	3	0
8	7	5	8	0	9	3	2	4	1	6
9	3	0	1	9	5	6	4	8	2	7

(This random Latin square was produced by a Markov chain algorithm due to Jacobson and Matthews.)

Thus, encrypting the plaintext 4698 with key 7251 using this square gives the ciphertext 0065. (For example, the entry in row 4 and column 7 is 0.) A Latin square used in this way is called a *substitution table*. The *columns* of the substitution table are the permutations of the alphabet associated with the key symbols. In the above, the key symbol 0 corresponds to the permutation

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 8 & 1 & 4 & 9 & 6 & 0 & 2 & 5 & 7 & 3 \end{pmatrix},$$

or in “cycle notation” $(0,8,7,5)(1)(2,4,6)(3,9)$.

We summarise a stream cipher in the following definition.

Definition: A *stream cipher* over an alphabet of q symbols $a_1, \dots, a_q$ requires a *key*, a random or pseudo-random string of symbols from the alphabet with the same length as the plaintext, and a *substitution table*, a Latin square of order q (whose entries are symbols from the alphabet, and whose rows and columns are indexed by these symbols). If the plaintext is $p_1 p_2 \dots p_n$ and the key is $k_1 k_2 \dots k_n$, then the ciphertext is $z_1 z_2 \dots z_n$, where $z_t = p_t \oplus k_t$ for $t = 1, \dots, n$; the operation $\oplus$ is defined as follows:

> $a_i \oplus a_j = a_k$ if and only if the symbol in the row labelled a_i and the column labelled a_j of the substitution table is a_k.

In the next section we will analyse the security of stream ciphers.